DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

REINFORCEMENT LEARNING BASED SOLUTION APPROACHES TO STATIC AND DYNAMIC MACHINE SCHEDULING PROBLEMS

by
Behice Meltem KAYHAN

October, 2022
İZMİR

REINFORCEMENT LEARNING BASED SOLUTION APPROACHES TO STATIC AND DYNAMIC MACHINE SCHEDULING PROBLEMS

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Industrial Engineering, Industrial Engineering Program

by
Behice Meltem KAYHAN

October, 2022
İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"REINFORCEMENT LEARNING BASED SOLUTION APPROACHES TO STATIC AND DYNAMIC MACHINE SCHEDULING PROBLEMS"** completed by **BEHİCE MELTEM KAYHAN** under supervision of **ASSOC. PROF. DR. GÖKALP YILDIZ** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. Gökalp YILDIZ

Supervisor

Prof. Dr. Adil BAYKASOĞLU

Thesis Committee Member

Assoc. Prof. Dr. Umay Zeynep UZUNOĞLU KOÇER

Thesis Committee Member

Assoc. Prof. Dr. Zeynel Abidin ÇİL

Examining Committee Member

Assoc. Prof. Dr. Leyla DEMİR

Examining Committee Member

Prof. Dr. Okan FISTIKOĞLU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENT

Firstly, I would like to express my deep gratitude to my supervisor, Assoc. Prof. Dr. Gökalp YILDIZ for his continuous support, patience, and guidance in completing my thesis and having time for me whenever I need it. He was not only a dedicated advisor, but also he improved and challenged me throughout my academic life.

Besides my supervisor, I would like to thank the members of my thesis committee, Prof. Dr. Adil BAYKASOĞLU, Assoc. Prof. Dr. Umay UZUNOĞLU KOÇER and the former member of my committee Dr. Özgür YALÇINKAYA for their insightful comments and suggestions to improve the quality of this dissertation.

In addition, I would like to express my thanks once more time to Prof. Dr. Adil BAYKASOĞLU as our department chair for giving me motivation and leading me with his academic passion.

Also, I would like to give very special thanks to my co-workers, Mümin Emre ŞENOL and Nurhan DUDAKLI. They are beyond being just co-workers; they helped me whenever I need, always improved my motivation, and encouraged me in the writing process of the dissertation. This journey would be very boring and hard without them.

Last but the most, I would like to express my deepest love and gratitude to my parents, Zöhre KAYHAN and Şahin KAYHAN, my brother Deniz KAYHAN, his wife Meltem ÇARPAR KAYHAN, my lovely niece Defne KAYHAN, my aunt Gülsem ARSLAN and my cousin Cansu ARSLAN who is a little sister to me. Their neverending support helped me overcome all troubles I faced throughout my life, and their unconditional love made life better. They always supported me in each decision of my life and stood by me. It is the greatest chance for me to have such a precious and full of love family. This dissertation would not have been possible without their love and trust.

Behice Meltem KAYHAN

REINFORCEMENT LEARNING BASED SOLUTION APPROACHES TO STATIC AND DYNAMIC MACHINE SCHEDULING PROBLEMS

ABSTRACT

This dissertation proposes Reinforcement Learning (RL) based solution approaches to static/deterministic and dynamic/stochastic machine scheduling problems, and it consists of three phases. In the first phase, a comprehensive literature review about the RL applications to machine scheduling problems was conducted to examine the essential aspects of the RL method and to reveal the deficiencies and promising areas in the literature. Then, the related literature was analyzed in detail according to applied algorithms, machine environments, job and machine characteristics, objective functions, and benchmark methods, and the essential findings of the review and the gaps in the related literature were discussed. In the second phase, the RL method was applied to static/deterministic unrelated parallel machine and hybrid flow shop scheduling problems. The performances of the proposed methods were compared with benchmark problems from the literature. In the third phase, the RL method was applied to the dynamic/stochastic hybrid flow shop scheduling problems, and a generic simulation model was developed. In this phase, first, the performances of the selected dispatching rules according to various objective functions were analyzed, and the best dispatching rules under each objective function were summarized. Later, the RL algorithm was applied to dynamic/stochastic hybrid flow shop scheduling problems to minimize mean flow time and tardiness. Finally, the performance of the proposed method was evaluated and compared with the selected dispatching rules.

Keywords: Reinforcement learning, machine scheduling, unrelated parallel machine scheduling problem, hybrid flow shop scheduling problem, dispatching rules, simulation

STATİK VE DİNAMİK MAKİNE ÇİZELGELEME PROBLEMLERİNE PEKİŞTİRMELİ ÖĞRENME TABANLI ÇÖZÜM YAKLAŞIMLARI

ÖZ

Bu tez, statik/deterministik ve dinamik/stokastik makine çizelgeleme problemlerine pekiştirmeli öğrenme tabanlı çözüm yaklaşımları önerir ve üç aşamadan oluşur. İlk aşamada, pekiştirmeli öğrenme yönteminin temel yönlerini incelemek ve literatürdeki eksiklikleri ve gelecek vaat eden alanları ortaya çıkarmak için makine çizelgeleme problemlerine pekiştirmeli öğrenme uygulamaları hakkında kapsamlı bir literatür taraması yapılmıştır. Ardından, ilgili literatür, uygulanan algoritmalar, makine ortamları, iş ve makine özellikleri, amaç fonksiyonları ve kıyaslama yöntemlerine göre detaylı bir şekilde incelenmiş ve incelemenin temel bulguları ve ilgili literatürdeki boşluklar tartışılmıştır. İkinci aşamada, pekiştirmeli öğrenme yöntemi statik/deterministik ilişkisiz paralel makine ve hibrit akış tipi atölye çizelgeleme problemlerine uygulanmıştır. Önerilen yöntemlerin performansları, literatürdeki kıyaslama problemleriyle karşılaştırılmıştır. Üçüncü aşamada, dinamik/stokastik hibrit akış tipi atölye çizelgeleme problemlerine pekiştirmeli öğrenme yöntemi uygulanmış ve genel bir simülasyon modeli geliştirilmiştir. Bu aşamada öncelikle, seçilen dağıtım kurallarının çeşitli amaç fonksiyonlarına göre performansları analiz edilmiş ve her bir amaç fonksiyonu altında en iyi dağıtım kuralları özetlenmiştir. Daha sonra, ortalama akış süresini ve gecikmeyi en aza indirmek için dinamik/stokastik hibrit akış tipi atölye çizelgeleme problemlerine pekiştirmeli öğrenme algoritması uygulandı. Son olarak, önerilen yöntemin performansı değerlendirilmiş ve seçilen dağıtım kuralları ile karşılaştırılmıştır.

Anahtar kelimeler: Pekiştirmeli öğrenme, makine çizelgeleme, ilişkisiz paralel makine çizelgeleme problemi, hibrit akış tipi atölye çizelgeleme problemi, gönderim kuralları, simülasyon

CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Motivation of the Study

Machine learning (ML) can be described as the science of developing models and algorithms, giving machines learning the ability to execute tasks without being explicitly programmed to do so (Russel & Norvig, 2009). Over the last decades, ML has become a prevalent technology with immense developments in computer science and wide usage of the internet. It has widespread practical applications in various areas, such as speech recognition, autonomous driving, energy efficiency, personalized healthcare treatment, and statistical learning (Gambella et al., 2020; Gao et al., 2020).

ML is divided into three main categories: (1) Supervised Learning (SL), (2) Unsupervised Learning (UL), and (3) Reinforcement Learning (RL). In SL, learning is provided by the data set that includes labels for each data. SL aims to extract information from the training data set and generalize this information for other situations not included in the training data set. UL aims to discover patterns hidden in the data set without needing labels. Unlike SL and UL, RL does not need the data set but learns via trial-and-error search (Russel & Norvig, 2009). It interacts with the environment, observes status changes, and learns to make decisions without having a supervisor. RL's distinctive features give it the advantage of being one of the most popular branches of machine learning, especially in sequential decision-making problems (Lee et al., 2018).

The history of RL goes back to the early days of neuroscience, computer science, psychology, and cybernetics. In the most general sense, RL mimics the simple principles of animal or human cognition. Animal or human cognition tends to perform the actions with positive consequences and learns these consequences by trial and error (Sigaud & Buffet, 2013). RL is generally formulated as a decision problem to find optimal policies in which actions are mapped to states. The learning system does not tell which action is the best; instead, it only evaluates the consequences of the action for the existing state (Barto & Mahadevan, 2003).

The RL approach is widely used in production and operations management problems. Particularly in decision problems with dynamic environments, it is shown that the RL approach can model a wide range of problems and get better results than conventional methods. For example, machine scheduling problems are one of these decision problems where the RL algorithms are frequently studied, and satisfactory results are obtained (Lihu & Holban, 2009).

Machine scheduling is defined as the sequencing of n jobs that have to be processed on m machines to minimize or maximize predefined objective functions under certain constraints (Graham et al., 1979). It is directly related to the operational activities of a company and has a crucial impact on competitiveness, efficiency, diminution of production cost, on-time delivery, and customer satisfaction (Fuchigami & Rangel, 2018). Therefore, machine scheduling problems have been studied in the literature broadly, and different algorithms have been developed to solve these problems efficiently (Lihu & Holban, 2009; Neto & Godinho Filho, 2013; Fuchigami & Rangel, 2018).

The RL methods have been actively applied to machine scheduling problems recently. Parente et al. (2020) showed that machine learning methods and Industry 4.0 concepts are getting to attract the attention of researchers and are frequently applied to production planning, control, and scheduling problems. Usuga Cadavid et al. (2020) indicated that RL methods had been actively applied to machine scheduling problems in recent years; because it provides autonomous decision-making depending on the system's state. Besides, it reduces decision-making delay and is more applicable in fully automated manufacturing systems. However, the studies that review the literature in depth considering the RL applications to the machine scheduling problems are very limited. Besides, there is a lack of research that handles the RL application to machine scheduling problems considering various aspects of the problem. Therefore, in this dissertation, we analyzed the related literature in depth, applied the RL algorithm to different machine scheduling problems that have not been studied before, and analyzed the algorithm's performance in deterministic and stochastic environments.

1.2 Research Objectives

Reinforcement learning is one of the most remarkable branches of machine learning and attracts the attention of researchers from numerous fields. Especially in recent years, the RL methods have been applied to machine scheduling problems. Therefore, in this dissertation, we have studied the RL applications to machine scheduling problems. The main research objectives of this dissertation are as follows;

- To conduct an in-depth analysis of the literature about the RL applications to machine scheduling problems to reveal the main concepts, fundamental aspects, and research gaps and to define the framework of the dissertation.

- To apply the RL algorithm to different machine scheduling problems that have not been studied before and analyze the algorithm's performance in both deterministic and stochastic environments.

Within this framework, we have divided the studies carried out in this dissertation into phases: Phase-1, Phase-2, and Phase-3. The outline of the dissertation is given in Figure 1.1.

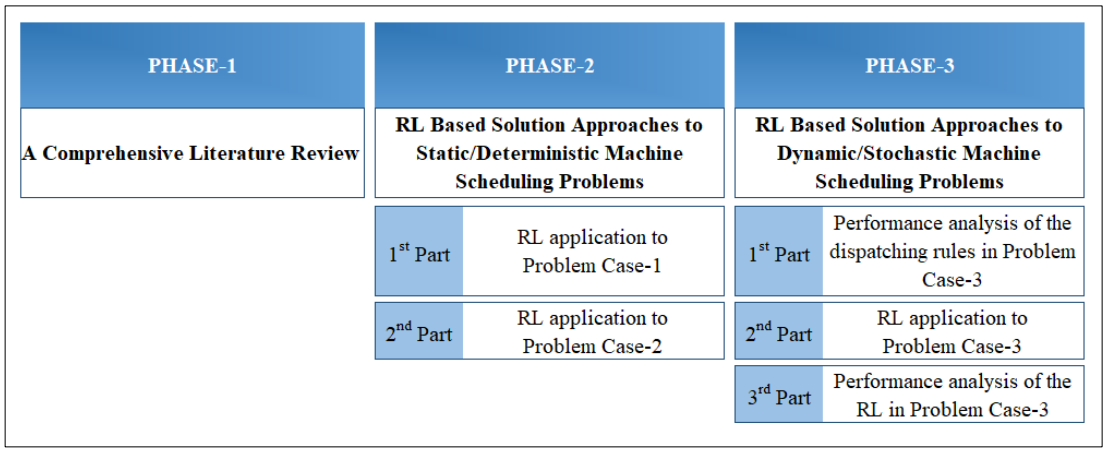

Figure 1.1 The outline of the dissertation

In Phase-1, we conducted a comprehensive literature review about the RL applications to machine scheduling problems. The studies done in this phase are as follows;

- The related literature was examined considering various aspects of the problem, such as applied algorithms, machine environments, job and machine characteristics, objectives, and benchmark methods.

- Classification and quantitative analysis of the literature were carried out.

- The related literature was analyzed in detail according to the machine environment, and the critical findings were discussed.

- Essential aspects of the RL method in machine scheduling problems were examined. The most frequently investigated problem types, objectives, and constraints were identified, and the deficiencies and promising areas were revealed. Finally, the contribution of this dissertation to existing literature was addressed.

In Phase-2, we applied the RL method to two types of static/deterministic machine scheduling problems. The studies carried out in this phase are as follows;

- In the first part of Phase-2, an unrelated parallel machine scheduling problem (UPMSP) with machine and sequence-dependent setup times, named Problem Case-1, was studied.

- In the second part of Phase-2, a hybrid flow shop scheduling problem named Problem Case-2 was handled.

- In both problems, the RL algorithm was applied to minimize the makespan, and the results were compared with benchmark problems from the literature.

In Phase-3, we analyzed the RL application to a dynamic/stochastic hybrid flow shop scheduling problem named Problem Case-3 and developed a generic simulation model. The studies carried out in this phase are as follows;

- In the first part of Phase-3, the performances of the selected dispatching rules according to various objective functions were analyzed, and the best dispatching rules according to each objective function were summarized.

- In the second part of Phase-3, the RL algorithm was applied to a stochastic hybrid flow shop scheduling problem to minimize mean flow time and tardiness.

- In the third part of Phase-3, the performance of the RL algorithm was evaluated and compared with dispatching rules.

The characteristics of Problem Case-1, Problem Case-2, and Problem Case-3 are summarized in Figure 1.2.

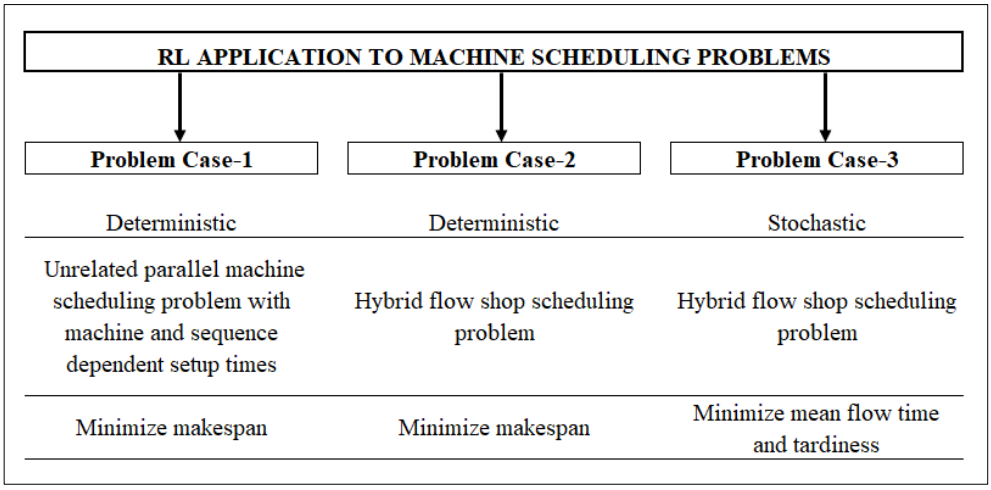

Figure 1.2 The characteristics of the problems

1.3 Main Contributions of This Dissertation

- In Problem Case-1, we applied the Q-learning algorithm to an unrelated parallel machine scheduling problem with the machine and sequence-dependent setup times to minimize makespan. Furthermore, we proposed a method to handle this problem in a deterministic and static manner.Problem Case-1 is the first study that addresses unrelated parallel machine scheduling problems with the machine and sequence-dependent setup time in a deterministic manner to minimize makespan.

- Besides these differences, the state definition of Problem Case-1 is independent of problem type, while the state definition is dependent on problem type in most of the studies in the related literature. Therefore, this proposed method is readily applicable to different machine scheduling problems.

- Another significant difference from the related literature can be stated in the design of the reward system. The reward system of the proposed method was wholly free of the parameters, and the occurrence of a particular decision-making problem was prevented.

- In the related literature there are two problems that resembles Problem Case-2. However, while these studies applied the RL algorithm to only one or two small-scale problems, Problem Case-2 consists of large-scale problems with various machines and jobs.

- In the related literature there are no study handles the hybrid flow shop scheduling problem considering the multi-objective and stochastic nature of the problem. Also, Problem Case-3 differs from other studies by addressing the objective function, which minimizes the mean tardiness and flow time.

1.4 Organization of the Dissertation

This dissertation involves five chapters, and the rest of the dissertation is organized as follows;

In Chapter 2, first, the RL method and machine scheduling problems are introduced, and essential aspects of the method and the problem are given. Afterward, a comprehensive literature review about the RL applications to machine scheduling problems is presented in detail. Finally, the essential findings of the review and the gaps in the related literature are summarized, and the contributions of this dissertation to the related literature are given.

In Chapter 3, firstly, the proposed method is explained in detail. Afterward, the application of the Q-learning algorithm to Problem Case-1 and Problem Case-2 is given. Finally, the results of the proposed method for these problems are presented.

In Chapter 4, first, the general structure of the problem is explained, and the selected objective functions and dispatching rules are introduced. Besides, the problem size and parameters are defined. Afterward, the performances of the dispatching rules accoring to objective functions are analyzed, and the application of the Q-learning algorithm to Problem Case-3 is given. Finally, the performance of the proposed method and the dispatching rules are compared in various aspects.

In Chapter 5, the dissertation is summarized, and conclusions are given. Finally, future research directions are highlighted.

CHAPTER 2
BACKGROUND AND LITERATURE REVIEW

Especially in recent years, the RL methods have been applied to machine scheduling problems and are among the top five most encouraging methods for scheduling literature (Lihu & Holban, 2009). However, there is no literature review in detail about the RL applications to machine scheduling problems. For this reason, a comprehensive review was conducted to analyze the related literature.

This chapter introduces the background of the RL method and the machine scheduling problem. First, the essential aspects of the RL method and machine scheduling problems are given. Afterward, the detail of the literature review is presented, and the trends and deficiencies in the literature, the most studied dimensions of the problem, and the potential research areas are summarized. In addition, the advantages, strengths, and challenging features of the RL are mentioned. Finally, the contribution of this dissertation to the related literature is given.

2.1 Reinforcement Learning

RL is one of the branches of machine learning. The primary purpose of RL is to maximize the rewards gained by an agent as a consequence of the action about the state of the environment. Unlike other machine learning algorithms, the agent is not told which action should be taken, but instead, the agent should learn which actions return the most reward by trial and error (Liu & Wang, 2009).

In the standard RL model, the agent refers to the decision-making unit which observes the environment and takes action. Each action will produce a result, which could be a reward or a punishment according to the state of the environment. Rewards or punishments received from the environment are used to evaluate actions. The agent decides which actions will be taken by itself, considering which actions yield the most rewards. The result of executing an action is evaluated via a reward function regarding

the current state and the current action. The framework of RL is given in Figure 2.1 (Kaelbling et al., 1996).

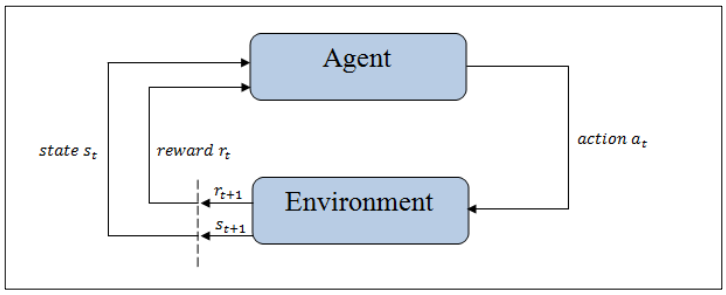

Figure 2.1 The framework of the RL

At each decision step t, the agent observes the system state s_t, executes an action a_t and receives a scalar reward r_t. After executing action a_t the current state of the environment changes, and the agent observes the next state s_{t+1} and takes the next action a_{t+1}. The algorithm keeps the values of the action and the current state of the environment at each decision step and uses this information to evaluate the next step (Kaelbling et al., 1996).

After numerous iterations with a sufficient learning level, the agent takes action to maximize the total reward in the long term. The primary purpose of the agent is to discover a policy that maximizes long-run objectives, which is defined as a value function. The value function evaluates the states in terms of the total amount of reward gained by an agent (Sutton & Barto, 1998).

RL primarily solves Markov Decision Problems (MDP), which consists of the following variables: a finite set of environment states S, a finite set of actions A, a reward function $R_a(s, s')$, and transition probabilities between states $T_a(s, s') \rightarrow [0,1]$. The solution of an MDP consists of a policy $\pi: S \rightarrow A$ that maps actions to states. An optimal policy (π^*) defines the best action in each state, providing a maximum reward in the long term. This long-term reward is named "value" and calculated by value functions. The decision-maker selects the value function depending on the type of problem (Sutton & Barto, 1998).

In the literature review, we analyzed the papers by considering the categories of learning algorithms, action selection methods, and state definition types. In the following subsections, we briefly introduce the methods frequently encountered under these categories in the related literature. Detailed information about the methods can be found in the references given in the relevant subsections.

2.1.1 Learning Algorithm

The learning algorithm defines the base algorithm of RL. These algorithms differ according to reward system, state representation, value update procedure, and policy adaptation. The most widely used learning algorithms in the literature are *Q-learning, SARSA, R-Learning, SMART, Deep RL, Approximate RL, Relational RL, Temporal Difference (λ),* and *Actor-Critic.*

In the *Q-learning* algorithm, a frequently used RL algorithm, action-value functions are stored in Q-values, $Q(s,a)$, expressing the utility of selecting action a in state s. The principle of the algorithm depends on the value iteration update. At each decision step t, the agent observes the state s_t, chooses the action a_t with the maximum Q-value, receives the reward *r*, and observes the new state s_{t+1}. Then Q-value $Q(s_t, a_t)$ is updated as in the following formula:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \qquad (2.1)$$

In this equation, r_{t+1} represents the immediate reward after taking action a, $max_a Q(s_{t+1}, a)$ denotes the maximum Q-value for all actions in the next state s_{t+1}, α is the learning rate of the algorithm, $0 < \alpha \leq 1$, and denotes the degree how which newly acquired information will affect old information, γ is the discount factor,$0 \leq \gamma \leq 1$, determines the importance of the future rewards, and the immediate rewards (Watkins & Dayan, 1992).

Unlike the Q-learning algorithm, the Q-values update procedure is based on the actions taken instead of the best possible action in the *SARSA* algorithm. The Q-

learning algorithm's Q-value is updated considering the action with the maximum Q-value in the next state (Sutton & Barto, 1998). In contrast, in the SARSA algorithm, the agent chooses an action with the same policy in the next state, and Q-value is updated according to the following formula;

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (2.2)$$

R-learning is an RL algorithm proposed by Schwartz (1993) that considers the average reward. R-learning considers the best possible action to update the Q-values same as in the Q-learning, but the average reward is substituted from the immediate reward as depicted in the following formula:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} - \rho + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (2.3)$$

$$\rho = \sum_{t=1}^{u} \frac{r_t}{u} \quad (2.4)$$

In this equation, ρ is the approximation of the expected reward per decision time step under policy π, and u is the number of the decision time step.

Another average reward RL algorithm *SMART* (Semi-Markov Average Reward Technique), is developed by Das et al. (1999). Differently from R-learning, it takes into account the T, which represents the total time till the decision step, instead of the number of the decision steps, and is calculated as follows;

$$\rho = \frac{\sum_{t=0}^{T} r}{T} \quad (2.5)$$

Deep RL combines the RL with a class of deep neural networks. In a deep neural network, multiple hidden layers are used to gradually construct more precise representations of the data (Mnih et al., 2015). This architecture is utilized to approximate the value functions in the RL problems.

Approximate RL is also based on the value function approximation instead of the precise Q-values. A weight factor is assigned to each state feature, and the algorithm

approximates a Q-value. After each decision step, the weight is updated, considering the difference between the prediction and target values (Qu et al., 2016b).

Relational RL utilizes the RL with the aid of logical and relational representations. In this algorithm, the states and the actions are depicted as logical facts, and the Q-values are stored in a regression tree instead of an explicit Q-table (De Raedt, 2008).

Temporal Difference (λ) pays regard to action history, which contributes to reaching the goal state, unlike the Q-learning and SARSA algorithm, which considers the last action before the goal state. When a reward is received for the current state, all states are updated according to their eligibility, which means the degree of the state has been visited recently. The algorithm aims to estimate the value function of the states instead of the state-action values (Sutton & Barto, 1998).

As mentioned before, the agent takes action in the state of the environment to reach a goal. In the *Actor-Critic* algorithm, this agent consists of two main interacting processes: actor and critic. The critic architecture evaluates the current policy to help the actor. The actor architecture tries to improve current policy considering the feedback that comes from the critic (Szepesvári, 2010).

2.1.2 Action Selection Methods

Another critical issue that directly affects the performance of algorithms is the action selection mechanism corresponding to the trade-off between exploration and exploitation. In this context, exploration defines finding more information about the environment, and exploitation defines taking actions with available information to maximize reward (Sutton & Barto, 1998). In the exploitation process, the agent may take advantage of known actions that yield better rewards, but the exploration process may allow the agent to find the maximum total reward in the long term.

This trade-off issue is generally managed with the *ε-greedy method*. This method involves selecting an action randomly with a small probability ε and selecting the

action with the maximum Q-value with probability $(1-\varepsilon)$ (Gosavi, 2015). Without information about the environment, the agent should discover the environment by exploration, which is crucial for learning. In this case, random selection probability ε is initialized with a high value to perform exploration, and it is decreased over time. *Boltzmann Probability* and *Simulated Annealing (SA) Temperature Procedure* approaches are also widely used to balance exploration and exploitation in the relevant literature.

In the *Boltzmann Probability*, the selection probability of each action is proportional to its Q-value. Unlike the ε-greedy method, which takes random action or the action with a maximum Q-value, this approach takes action with weighted probabilities (Sutton & Barto, 1998). The selection probability of action in a given state is calculated as in Equation 2.6, where τ_t is a temperature parameter that decreases over time.

$$p_t(a|s_t) = \frac{e^{\frac{Q(s_t,a)}{\tau_t}}}{\sum_a e^{\frac{Q(s_t,a)}{\tau_t}}} \tag{2.6}$$

Simulated Annealing (SA) Temperature Procedure is a hybrid approach to improve the Boltzmann Probability or the ε-greedy Method. The temperature parameter of Boltzmann probability or value of the ε is decreased over time as the agent gathers enough information. This way, the exploration decreases gradually and converges to an optimum value.

2.1.3 State Definition Types

As mentioned before, the agent observes the state and then takes action in RL. State definition is crucial since many problems have continuous states or state-space is exceptionally large. In such circumstances, it is hard to calculate and store the value function for each state-action pair. This problem is known as the curse of dimensionality (Gosavi, 2015). State aggregation methods or function approximators are used to approximate the value functions to deal with this problem.

As a result of the analysis of the related literature, the mainly employed state definition strategies were found, and these strategies were categorized as *All States, Aggregation, Clustering, Neural Network (NN), Case-Based Reasoning (CBR), Support Vector Regression (SVR), Regression Tree,* and *Gradient Descent Method*.

The category of *All States* indicates that each state of the environment is situated in the Q-table. This method is applied only to small-scale problems where a single feature defines states. In cases where the state space gets large, the *Aggregation* strategy can be applied. In this strategy, states are lumped together to reduce the state space. One or two features generally define states, and the ranges based on these features are determined in advance to categorize states. In *Clustering*, the states are defined by a set of features. The aim is to cluster these states based on these features. Clusters are constructed considering the similarity or dissimilarities of the states.

NN is a learning technique that is utilized to match input data to output data and is inspired by the neural structure of the human brain. It consists of many neurons that help link given input data to output data by calculating the weights of the linkages. It is trained on labeled data (i.e., inputs with output) to calculate the input weights and tunes the input weights until minimizing the network's error. Once the training process is completed, NN can generalize new, unseen inputs into accurate outputs (Russel & Norvig, 2009). In the related literature, it is used to predict the Q-value of action in a given state.

CBR relies on the idea that similar problems have similar solutions. This idea can be explained in the RL concept as similar states have the same Q-value for the same action. A case consists of a problem part which includes the state characteristics, and a solution part includes the Q-values of actions. Visited state-action pairs are stored in a case base. When an agent observes a new state, a similarity value is calculated between the new state and each state in the case base. Then, the state with the highest similarity value is selected, and the action that yields the highest Q-value is executed (Gabel & Riedmiller, 2006b).

SVR is a statistical learning technique that aims to find output values based on input values by finding a function with an acceptable error value (Vapnik, 2000). The features of the state and actions are stored in a vector form. The value of action depending on these features is found by modeling nonlinear relationships, and never-experienced states can be estimated (Csáji & Monostori, 2008).

A *Regression Tree* is a particular decision tree that handles real-valued variables (Van Otterlo, 2009). The regression tree creates different input space subsets by splitting input space based on specific limit values and aims to predict a target value. In the related literature, this approach was used to split state-action space into subsets, and the Q-values of the state-action pairs were stored in the regression tree (Palombarini & Martínez, 2012a).

In the *Gradient Descent Method*, states are represented by feature vectors, and a linear combination of these vectors constructs the value functions. The primary purpose is to find the weights of the linear functions minimizing the difference between real Q-values and predicted Q-values. In this regard, the value function is reorganized by updating the weights using the gradient of Q-values concerning the weights of functions (Zhang et al., 2012).

2.2 Machine Scheduling

Machine scheduling involves sequencing jobs that must be processed on a collection of machines under some constraints to optimize the predefined objective function (Nahmias & Olsen, 2015). The general restriction about the problem is that each machine can process at most one job, and each job can be processed on at most one machine simultaneously (Graham et al., 1979). Proper scheduling ensures meeting demands timely, minimizing work-in-process inventory, reducing setup times, increasing the utilization of machines, and minimizing worker and production costs (Nahmias & Olsen, 2015).

In the literature, scheduling problems are symbolized with three attributes using the notation $\alpha/\beta/\gamma$, where α represents the machine environment, β represents the job

characteristics, and γ represents the optimality criteria (Graham et al., 1979). This classification scheme is known as Graham's notation. Further details of these attributes are given in the following sections.

2.2.1 Machine Environment

In machine scheduling problems, the machine environment also represents the type of problem. Mainly studied machine environments are *Single Machine, Parallel Machine, Flow Shop, Job Shop,* and *Open Shop.*

(i) *Single Machine*: There is only one machine on which all jobs must be processed.

(ii) *Parallel Machine*: There are two or more machines, but all jobs must be processed on only one of these machines. Machines in this environment can be *Identical, Uniform, and Unrelated. Identical Parallel Machines* indicate that all machines can process all jobs at the same speed. *Uniform Parallel Machines* indicate that all machines can process all jobs, but the speed of the machines varies. Finally, *Unrelated Parallel Machine* indicates that processing time depends on both job and machine.

(iii) *Flow Shop*: A set of jobs must be processed by a set of machines in series, and each job follows the same order through the machines. A *Hybrid Flow Shop* integrates a flow shop and parallel machine in which there are *s* stages in series, and at least one of these stages contains multiple machines. This type of machine environment is also named *Flexible Flow Shop*.

(iv) *Job Shop*: Differently from the Flow Shop, each job has a specific route through the machines.

(v) *Open Shop*: The route of the jobs can be changed; in other words, there is no constraint on the processing order of each job.

2.2.2 Job Characteristics

The second attribute, symbolized as β, gives information about the constraints of the problem and processing characteristics. Common characteristics and constraints are *release date, preemption, precedence constraints, sequence-dependent setup times, limited buffer size,* and *machine breakdown*.

Release date implies that the release date of the job may vary. *Preemption* implies that the processing of a job may be abandoned and resumed later. *Precedence constraints* imply that a job can be started after its predecessor jobs have been completed. *Sequence-dependent setup times* imply a need for setup time between diverse types of jobs. Therefore, each machine's setup time for a particular job is determined by both that job and the previous job that has been processed on that machine. *Limited buffer size* implies that a blockage of the machine may occur due to the limited buffer area of the next machine. Finally, *Machine breakdown* indicates that machines cannot work for some periods due to damage, repair, etc.

2.2.3 Optimality Criteria

Let us assume that there are n jobs J_i, $(i = 1,2, \dots, n)$ with the attributes of t_i, r_i, p_i, d_i, w_i which refers to the start time, release date, processing time, due date, and weight of J_i, respectively and C_i refers to the completion time of J_i . The basic properties of job J_i are as follows:

$$\text{Lateness: } L_i = C_i - d_i \tag{2.7}$$

$$\text{Tardiness: } T_i = \max(C_i - d_i, 0) \tag{2.8}$$

$$\text{Earliness: } E_i = \max(d_i - C_i, 0) \tag{2.9}$$

$$\text{Flow Time: } F_i = C_i - r_i \tag{2.10}$$

$$\text{Waiting Time: } W_i = C_i - r_i - p_i \tag{2.11}$$

$$\text{Makespan: } C_{max} = \max\{C_1, C_2, C_3, \dots, C_n\} \tag{2.12}$$

$$\text{Tardy Job: } U_i = \begin{cases} 1, & if\ C_i > d_i \\ 0, & otherwise \end{cases} \tag{2.13}$$

An optimality criterion states the objective of the scheduling problem and is defined with one or more performance measures such as makespan, mean tardiness, and total flow time. Besides these mainly used performance measures, there are other performance measures in the analyzed literature such as *Work-in-Process*, *Cycle Time*, *Machine Downtime, Cost, Utilization, Due Date Deviation*, and *Throughput*. The list of all optimality criteria, machine environment, and job characteristics is given in Table 2.1.

Table 2.1 Graham notations

Machine Environment ($\boldsymbol{\alpha}$)		Optimality Criteria ($\boldsymbol{\gamma}$)	
Acronym	Meaning	Acronym	Meaning
1	Single machine	C_{max}	Makespan
P_m	Identical parallel machine	L_{max}	Maximum lateness
Q_m	Uniform parallel machine	T_{max}	Maximum tardiness
R_m	Unrelated parallel machine	$\sum F_i$	Total flow time
J	Job shop	$\sum T_i$	Total tardiness
F	Flow shop	$\sum L_i$	Total lateness
HF	Hybrid flow shop	$\sum E_i$	Total earliness
O	Open shop	$\sum W_i$	Total waiting time
Job Characteristics ($\boldsymbol{\beta}$)		$\sum U_i$	Total number of tardy jobs
Acronym	Meaning	$\sum w_i T_i$	Total weighted tardiness
r_j	Release date	$\sum w_i W_i$	Total weighted waiting time
$pmtn$	Preemption	$\frac{1}{n}\sum w_i T_i$	Mean weighted tardiness
$prec$	Precedence constraints	$\frac{1}{n}\sum L_i$	Mean lateness
S_{jk}	Sequence-dependent setup times	$\frac{1}{n}\sum T_i$	Mean tardiness
$bkdwn$	Machine breakdown	$\frac{1}{n}\sum W_i$	Mean waiting time
$block$	Limited buffer size	$\frac{1}{n}\sum F_i$	Mean flow time
		$\frac{1}{n}\sum E_i$	Mean earliness
		WIP	Work-in-process
		CT	Cycle time
		MDT	Machine down time
		TC	Total cost
		DD	Due-date deviation
		TP	Throughput
		U	Utilization
		SC	Setup cost

2.3 Literature Review of the RL Applications to Machine Scheduling Problems

The first application of the RL method to a scheduling problem was conducted by Zhang & Dietterich (1995) for the NASA space shuttle payload process to obtain a feasible schedule without any violation of resource constraints while minimizing the duration of the process. Afterward, many researchers applied RL algorithms to different machine scheduling problems (Kim & Lee, 1996; Riedmiller & Riedmiller, 1999; Aydin & Öztemel, 2000; Paternina-Arboleda & Das, 2001). Especially in machine scheduling problems that have a large set of data with dynamic environments, it is shown that the RL algorithms work well, and they are among the top five most encouraging methods for scheduling literature (Lihu & Holban, 2009).

The RL methods have been actively applied to machine scheduling problems recently. For example, Parente et al. (2020) showed that machine learning methods and Industry 4.0 concepts are getting to attract the attention of researchers and are frequently applied to production planning, control, and scheduling problems. Furthermore, Usuga Cadavid et al. (2020) indicated that RL methods have been actively applied to machine scheduling problems in recent years. Therefore, we aimed to conduct a comprehensive literature review about the RL method applications to machine scheduling problems and shed light on the related literature gap.

For this purpose, related databases were searched from 1995 to May 2022 exhaustively, and 94 papers were found in the scope of the dissertation. First, these papers were reviewed considering various aspects of the problem such as applied algorithm, machine environment, job and machine characteristics, objectives, benchmark method, and a detailed classification scheme was constructed based on these aspects. Afterward, the papers were analyzed and interpreted in the context of the machine environment, constraints, and objectives.

2.3.1 Methodology and Framework

In this phase, the papers discussing the application of RL algorithms to machine scheduling problems have been reviewed comprehensively. Primarily, Scopus (https://www.scopus.com) and Web of Science (webofknowledge.com/WOS) databases were searched using the following keywords in the whole articles:

"Reinforcement Learning" and "Scheduling"
"Q-Learning" and "Scheduling"
"Neuro-dynamic Programming" and "Scheduling"

In the beginning, the literature search was made very inclusively to prevent any paper in this field from being overlooked. All search results were scanned by reading the abstracts of the papers, and the papers that included scheduling problems in production systems were identified. In addition, both conference papers and articles were included. As a result of the comprehensive search, 124 papers were found. Firstly, these papers were examined in terms of problem type, and 16 papers dealt with other problems such as supply chain management, economic lot scheduling, load carrier scheduling, maintenance scheduling, gantry scheduling, job remaining time prediction, and transfer unit scheduling eliminated. Afterward, solution methods in these papers were analyzed, and 14 papers in which RL algorithms were applied as an auxiliary method were eliminated. After these eliminations, 94 papers have been found in the scope of this phase.

The papers are classified into eight categories for further analysis and are given in Table 2.2.

1. *The learning algorithm* defines the base algorithm of RL used in the paper.

2. *Problem type* gives information about the machine environment, job characteristics, and objective function of the problem.

3. *Objective* indicates whether the problem studied is a multi-objective (MO) problem or a single objective (SO) problem.

4. *Agent* indicates whether systems are modeled as Single-Agent (S) or Multi-Agent (M).

5. *The Benchmark method* indicates the methods used to compare the performance of RL.

6. *Action selection* indicates how the exploration and exploitation processes are balanced.

7. *State definition* indicates the state definition strategy to handle the curse of dimensionality.

8. *System type* indicates whether the analyzed problem is modeled as stochastic or deterministic.

2.3.2 Classification and Quantitative Analysis of the Literature

As a result of an exhaustive literature review, a total of 94 papers were found, including 64 journal articles and 30 conference papers, which handle the scheduling problems with the RL algorithms, and these papers are classified according to eight attributes. Table 2.2 presents the papers in the related literature review and the problem cases we studied in the dissertation. As stated in Chapter 1, while Problem Case-1 and Problem Case-2 address static/deterministic unrelated parallel machine and static/deterministic hybrid flow shop scheduling problems, Problem Case-3 considers dynamic/stochastic hybrid flow shop scheduling problems. A detailed comparison of the problem cases with the related literature is given in Section 2.3.4.5.

Table 2.2 Classification of the literature review

Author	Year	Learning Algorithm	Problem Type	Obj.	Agent	Benchmark Method	Action Selection	State Definition	System Type
Problem Case-1	**2022**	**Q-learning**	$\boldsymbol{R_m/S_{jk}/C_{max}}$	**SO**	**S**	**Benchmark Problems**	**ε-greedy**	**All States**	**Deterministic**
Problem Case-2	**2022**	**Q-learning**	$\boldsymbol{HF//C_{max}}$	**SO**	**S**	**Benchmark Problems**	**ε-greedy**	**Aggregation**	**Deterministic**
Problem Case-3	**2022**	**Q-learning**	$\boldsymbol{HF//\frac{1}{n}\sum T_i+\frac{1}{n}\sum F_i}$	**MO**	**S**	**Dispatching Rules**	**ε-greedy**	**Aggregation**	**Stochastic**
Lee & Lee	2022	Deep RL	$J\ /block/TP$	SO	S	Dispatching Rules	ε-greedy	Neural Network	Deterministic
Brammer et al.	2022	Q-learning	$F//C_{max}$	SO	S	Benchmark Problems	ε-greedy	Clustering	Deterministic
Chien & Lan	2021	Deep RL	$R_m/S_{jk}/C_{max}$	SO	S	Dispatching Rules	ε-greedy	Neural Network	Stochastic
Wang et al.	2021	Q-learning	$1//C_{max}+\sum T_i+\sum wC_i$	MO	S	Dispatching Rules	ε-greedy	Clustering	Stochastic
Samsonov et al.	2021	Deep RL	$J//C_{max}$	SO	S	Dispatching Rules	Not Mentioned	Neural Network	Deterministic
Paeng et al.	2021	Deep RL	$R_m/S_{jk}/\frac{1}{n}\sum T_i$	SO	S	Dispatching Rules	ε-greedy	Neural Network	Deterministic
Kuhnle et al.	2021	Deep RL	$J/block, bkdwn,\ S_{jk}$ $/\frac{1}{n}\sum w_iT_i+U+TP$	MO	S	Dispatching Rules	Not Mentioned	Neural Network	Stochastic
Park et al.	2021	Deep RL	$J//C_{max}$	SO	S	Dispatching Rules and RL Algorithms	Not Mentioned	Neural Network	Deterministic
Luo et al.	2021	Deep RL	$R_m/S_{jk}/\frac{1}{n}\sum T_i+TC$	MO	S	Dispatching Rules and RL Algorithms	Not Mentioned	Neural Network	Stochastic
Lee & Kim	2021	Q-learning	$F//C_{max}$	SO	S	Dispatching Rules	ε-greedy	Clustering	Stochastic
Yang et al.	2021	Deep RL	$F//\sum T_i$	SO	S	GA and RL Algorithms	Not Mentioned	Neural Network	Stochastic
Pan et al.	2021	Deep RL	$F//C_{max}$	SO	S	Benchmark Problems	Not Mentioned	Neural Network	Deterministic
Zhou et al.	2021a	Deep RL	$J//C_{max}+TC+U$	MO	S	Dispatching Rules and RL Algorithms	ε-greedy	Neural Network	Stochastic
Zhou et al.	2021b	Temporal Difference	$J//\frac{1}{n}\sum w_iT_i+U$		M	RL Algorithms	ε-greedy	Neural Network	Deterministic
Zhou et al.	2020	Q-learning	$R_m//C_{max}$	SO	S	No Benchmark	Not Mentioned	Neural Network	Stochastic
Park et al.	2020	Deep RL	$J/S_{jk}\ /C_{max}$	SO	M	GA and Dispatching Rules	ε-greedy	Neural Network	Stochastic

Table 2.2 Continues

Author	Year	Learning Algorithm	Problem Type	Obj.	Agent	Benchmark Method	Action Selection	State Definition	System Type
Guo et al.	2020	Q-learning	$Q_m//T_{max} + \sum U_i + \frac{1}{n}\sum W_i$	MO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
Wang et al.	2020	Q-learning	$J//\sum w_i E_i + TC$	MO	S	Dispatching Rules	ε-greedy	Clustering	Deterministic
Fang Guo et al.	2020	Q-learning	$HF//C_{max}$	SO	S	GA	ε-greedy	All States	Deterministic
Lee et al.	2020	Deep RL	$J//\sum wT_i$	SO	M	Dispatching Rules	ε-greedy	Neural Network	Deterministic
Luo	2020	Deep RL	$J//\sum T_i$	SO	S	Dispatching Rules	Boltzmann Probability	Neural Network	Stochastic
Liu et al.	2020	Deep RL	$J//C_{max}$	SO	M	Dispatching Rules	ε-greedy	Neural Network	Stochastic
Ren et al.	2020	Deep RL	$J//C_{max}$	SO	S	Benchmark Problems	Boltzmann Probability	Neural Network	Deterministic
Jiménez et al.	2020	Q-learning	$J//C_{max}$	SO	M	Benchmark Problems	ε-greedy	All States	Deterministic
Ábrahám et al.	2019	Q-learning	$R_m/prec/C_{max}$	SO	S	Benchmark Problems	Boltzmann Probability	All States	Deterministic
Reyna et al.	2019a	Q-learning	$F//C_{max}$	SO	S	Benchmark Problems	ε-greedy	All States	Deterministic
Reyna et al.	2019b	Q-learning	$HF/S_{jk}\ prec/C_{max}$	SO	M	Benchmark Problems	ε-greedy	All States	Deterministic
Han et al.	2019	Q-learning	$HF//C_{max}$	SO	S	AIS and GA	Boltzmann Probability	All States	Deterministic
Zhao et al.	2019	Q-learning	$J/bkdwn/C_{max}$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
Lin et al.	2019	Deep RL	$J//C_{max}$	SO	S	Dispatching Rules	ε-greedy	Neural Network	Deterministic
Waschneck et al.	2018a	Deep RL	$J/S_{jk}/U$	SO	M	Dispatching Rules	ε-greedy	Neural Network	Stochastic
Waschneck et al.	2018b	Deep RL	$J/S_{jk}/CT$	SO	M	Dispatching Rules	ε-greedy	Neural Network	Stochastic
Reyna et al.	2018	Q-learning	$F//C_{max}$	SO	S	Benchmark Problems	ε-greedy	Not Mentioned	Deterministic
Stricker et al.	2018	Q-learning	$J/bkdwn\ block/U$	SO	S	Dispatching Rules	ε-greedy	Neural Network	Stochastic
Shiue et al.	2018	Q-learning	$J//CT + \sum U_i + \mathrm{TP}$	MO	S	Dispatching Rules	ε-greedy	Neural Network	Stochastic
Wang	2018	Q-learning	$J//\sum E_i + \sum T_i$	MO	M	RL Algorithms	ε-greedy	Clustering	Stochastic

Table 2.2 Continues

Author	Year	Learning Algorithm	Problem Type	Obj.	Agent	Benchmark Method	Action Selection	State Definition	System Type
Thomas et al.	2018	Q-learning	$J//TP$	SO	S	Benchmark Problems	ε-greedy	Neural Network	Stochastic
Zhang et al.	2018	Q-learning	$J//CT$	SO	S	Dispatching Rules	ε-greedy	Neural Network	Stochastic
Bouazza et al.	2017	Q-learning	$J/S_{jk}\ prec/\sum w_i W_i$	SO	M	Dispatching Rules	Boltzmann Probability	Not Mentioned	Stochastic
Zhang et al.	2017	Q-learning	$J//CT$	SO	S	Dispatching Rules	ε-greedy	Neural Network	Stochastic
Xiao et al.	2017	Q-learning	$1//CT$	SO	S	Relative Value Iteration	SA Temperature Procedure	Aggregation	Stochastic
Kim & Shin	2017	Deep RL	$J/S_{jk}\ bkdwn/CT, WIP$	MO	S	Dispatching Rules	Not Mentioned	Not Mentioned	Stochastic
Arviv et al.	2016	Q-learning	$F//C_{max}$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
Qu et al.	2016a	Q-learning	$HF/S_{jk}\ block/TC + \sum U_i$	MO	M	No Benchmark	ε-greedy	Gradient Descent Method	Stochastic
Qu et al.	2016b	Approximate RL	$J/block/WIP + \sum U_i$	MO	S	Dispatching Rules	ε-greedy	Gradient Descent Method	Stochastic
Wang & Yan	2016	Q-learning	$J//\frac{1}{n}\sum T_i$	SO	M	RL Algorithms	ε-greedy	Clustering	Stochastic
Yuan et al.	2016	Q-learning	$P_m/bkdwn/L_{max}, \sum U_i$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
Qu et al.	2015	Q-learning	$HF/S_{jk}\ block/WIP + TC$	MO	S	Dispatching Rules	ε-greedy	Not Mentioned	Deterministic
Reyna et al.	2015	Q-learning	$J//C_{max}$	SO	S	Benchmark Problems	ε-greedy	Not Mentioned	Deterministic
Xanthopoulos et al.	2013	Q-learning	$1/S_{jk}/\frac{1}{n}\sum T_i + \frac{1}{n}\sum E_i$	MO	S	Fuzzy Logic MOEA	ε-greedy	Aggregation	Stochastic
Wang & Yan	2013a	Q-learning	$J/bkdwn\ block/\frac{1}{n}\sum T_i$	SO	M	RL Algorithms	ε-greedy	Clustering	Stochastic
Wang & Yan	2013b	Q-learning	$J//\frac{1}{n}\sum E_i$	SO	S	Dispatching Rules	ε-greedy	Gradient Descent Method	Stochastic
Atighehchian & Sepehri	2013	Q-learning	$1/S_{jk}/\sum w_i T_i$	SO	S	Dispatching Rules	ε-greedy	Neural Network	Stochastic
Yuan et al.	2013	Q-learning	$P_m//L_{max}, \sum U_i$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
Aissani et al.	2012	SARSA	$J/prec/C_{max}$	SO	M	Genetic Algorithm MILP	Boltzmann Probability	Not Mentioned	Not Mentioned
Zhang et al.	2012	R-learning	$R_m//\sum w_i T_i$	SO	S	Dispatching Rules	ε-greedy	Gradient Descent Method	Stochastic

Table 2.2 Continues

Author	Year	Learning Algorithm	Problem Type	Obj.	Agent	Benchmark Method	Action Selection	State Definition	System Type
Palombarini & Martínez	2012a	Relational RL	$R_m/S_{jk}\ bkdwn/\sum T_i$	SO	S	Benchmark Problems	ε-greedy	Relational Regression Tree	Stochastic
Palombarini & Martínez	2012b	Relational RL	$R_m/S_{jk}\ bkdwn/\sum T_i$	SO	S	No Benchmark	ε-greedy	Relational Regression Tree	Stochastic
Zhang et al.	2011	SARSA	$R_m/S_{jk}\ prec/TC$	SO	S	Benchmark Problems	ε-greedy	Gradient Descent Method	Stochastic
Gabel & Riedmiller	2011	Q-learning	$J//C_{max}$	SO	M	Benchmark Problems	ε-greedy	All States	Stochastic
Iwamura et al.	2010	Q-learning	$R_m//\sum F_i + TC$	MO	S	Average Improvement	ε-greedy	All States	Not Mentioned
Palombarini & Martínez	2010	Relational RL	$R_m//\sum T_i$	SO	S	Average Improvement	ε-greedy	Relational Regression Tree	Stochastic
Aissani et al.	2009	SARSA	$J//C_{max}$	SO	M	Genetic Algorithm	Boltzmann Probability	Not Mentioned	Stochastic
Yingzi et al.	2009	Q-learning	$J//C_{max}$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
Ramírez-Hernández & Fernandez	2009	Temporal Difference	$J/block\ S_{jk}\ bkdwn/WIP$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
Yang & Yan	2009	Q-learning	$J//\frac{1}{n}\sum T_i$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
Aissani & Trentesaux	2008	SARSA	$J/bkdwn/C_{max} + MDT$	MO	S	Benchmark Problems	Boltzmann Probability	Not Mentioned	Stochastic
Csáji & Monostori	2008	Q-learning	$J/S_{jk}\ pmtn\ /\sum U_i$	SO	S	Benchmark Problems	Boltzmann Probability	SVR	Stochastic
Gabel & Riedmiller	2008	Q-learning	$J//C_{max}$	SO	M	Dispatching Rules	ε-greedy	Neural Network	Stochastic
Gabel & Riedmiller	2007a	Q-learning	$J//C_{max}$	SO	M	Dispatching Rules	ε-greedy	Neural Network	Stochastic
Gabel &Riedmiller	2007b	Q-learning	$J//C_{max}$	SO	M	Dispatching Rules	ε-greedy	Neural Network	Stochastic
Yang & Yan	2007	Q-learning	$J//\frac{1}{n}\sum T_i$	SO	S	Dispatching Rules	ε-greedy	Clustering	Stochastic
Zhang et al.	2007	Q-learning	$R_m/S_{jk}/\frac{1}{n}\sum w_i T_i$	SO	S	Dispatching Rules	ε-greedy	Gradient Descent Method	Stochastic
Wang & Usher	2007	Q-learning	$J/S_{jk}/\frac{1}{n}\sum T_i$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
Gabel & Riedmiller	2006a	Q-learning	$J/S_{jk}/\frac{1}{n}\sum T_i$	SO	S	Dispatching Rules	ε-greedy	Neural Network	Deterministic

Table 2.2 Continues

Author	Year	Learning Algorithm	Problem Type	Obj.	Agent	Benchmark Method	Action Selection	State Definition	System Type
Gabel & Riedmiller	2006b	Q-learning	$J/prec\ bkdwn/\sum T_i$	SO	M	Dispatching Rules	ε-greedy	CBR	Stochastic
Csáji et al.	2006	Temporal Difference	$J/bkdwn/C_{max}$	SO	M	No Benchmark	SA Temperature Procedure	Neural Network	Stochastic
Monostori & Csáji	2006	Q-learning	$J/pmtn\ S_{jk}/\sum L_i$	SO	S	Benchmark Problems	Boltzmann Probability	SVR	Stochastic
Idrees et al.	2006	SMART	$1//\frac{1}{n}\sum T_i + TC$	MO	S	Dispatching Rules	ε-greedy	Not Mentioned	Stochastic
Wang & Usher	2005	Q-learning	$1//L_{max}, \sum U_i, \frac{1}{n}\sum L_i$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
Kong & Wu	2005	Q-learning	$1//\frac{1}{n}\sum T_i, T_{max}, \sum U_i$	SO	S	Dispatching Rules	SA Temperature Procedure	Aggregation	Stochastic
Csáji & Monostori	2005	Q-learning	$R_m/bkdwn\ prec/C_{max}$	SO	S	No Benchmark	SA Temperature Procedure	Gradient Descent Method	Stochastic
Ramírez-Hernández & Fernandez	2005	Q-learning	$J/block/TC$	SO	S	Modified Policy Iteration	ε-greedy	All States	Stochastic
Monostori et al.	2004	Temporal Difference	$J/prec/C_{max}$	SO	M	Branch and Bound	Boltzmann Probability	Neural Network	Stochastic
Hong & Prabhu	2004	Q-learning	$F/S_{jk}/DD + SC$	MO	S	Dispatching Rules	ε-greedy	All States	Deterministic
Wang & Usher	2004	Q-learning	$1//\frac{1}{n}\sum T_i$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
Wei & Zhao	2004	Q-learning	$J//\frac{1}{n}\sum T_i$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
Paternina-Arboleda & Das	2001	SMART	$F/bkdwn/WIP$	SO	S	Kanban CONWIP	ε-greedy	All States	Stochastic
Liu et al.	2001	Temporal Difference	$J/block/TP$	SO	S	Dispatching Rules	Not Mentioned	All States	Deterministic
Aydin & Öztemel	2000	Q-learning	$J//\frac{1}{n}\sum T_i$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
Miyashita	2000	Temporal Difference	$J//\sum w_i T_i + WIP$	MO	S	Random Operator	ε-greedy	CBR	Stochastic
Riedmiller & Riedmiller	1999	Q-learning	$J//\sum T_i$	SO	M S	Dispatching Rules	ε-greedy	Neural Network	Stochastic
Zhang & Dietterich	1996	Temporal Difference	$J/prec/C_{max}$	SO	S	Neural Network	SA Temperature Procedure	Neural Network	Stochastic
Zhang & Dietterich	1995	Temporal Difference	$J/prec/C_{max}$	SO	S	Iterative Repair	SA Temperature Procedure	Neural Network	Stochastic

The first analysis conducted on the papers is related to the publication date. The literature review has been conducted until May 2022. The first paper in this field was published in 1995. Therefore, the analysis includes the years between 1995 and 2022, as seen in Figure 2.2. Considering this analysis, studies in this field are increasing, especially in recent years.

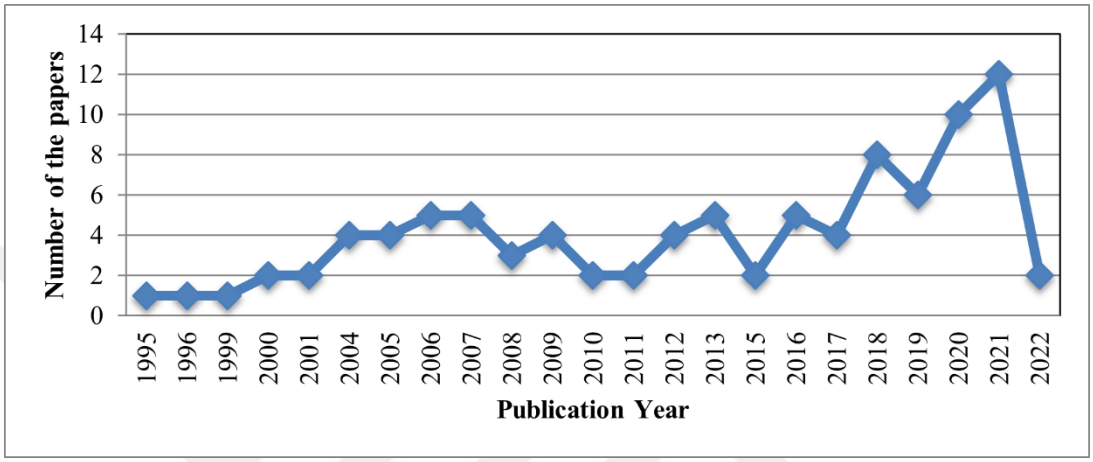

Figure 2.2 The yearly number of papers

The publication years of the papers were analyzed based on the manufacturing environment, given in Figure 2.3. Until 2000, the researchers addressed only the job shop scheduling problem. After 2001, they began to apply the RL algorithm to different manufacturing environments.

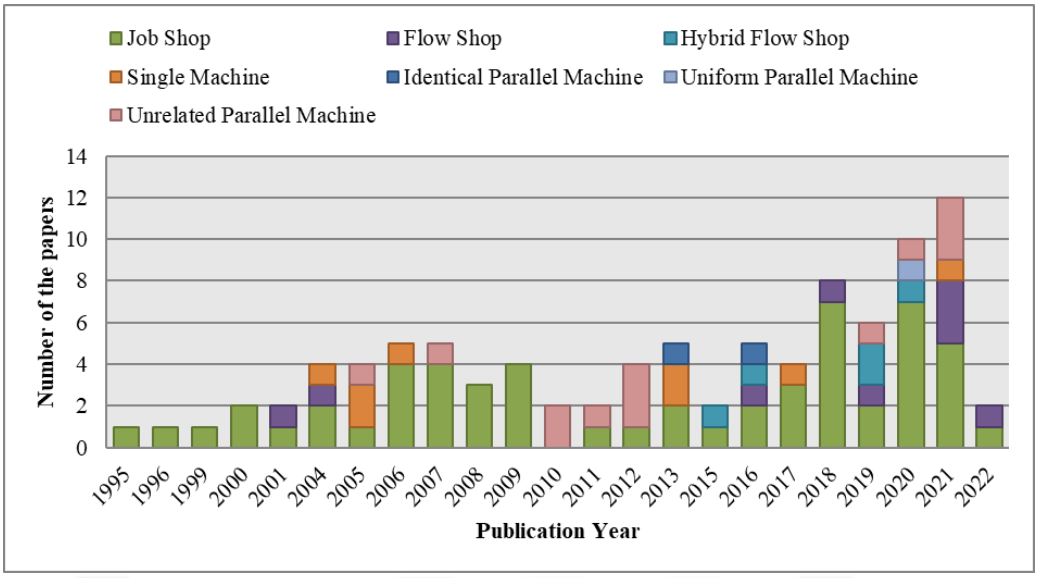

Figure 2.3 The yearly number of papers according to the manufacturing environment

Figure 2.4 demonstrates the distribution of manufacturing environments studied in the papers. The majority of the papers handle the job shop scheduling problem (56 out of 94). The unrelated parallel machine scheduling problem (13 out of 94) is the second most studied problem, followed by flow shop scheduling problems (9 out of 94). Identical parallel machines and uniform parallel machine scheduling problems were not discussed broadly.

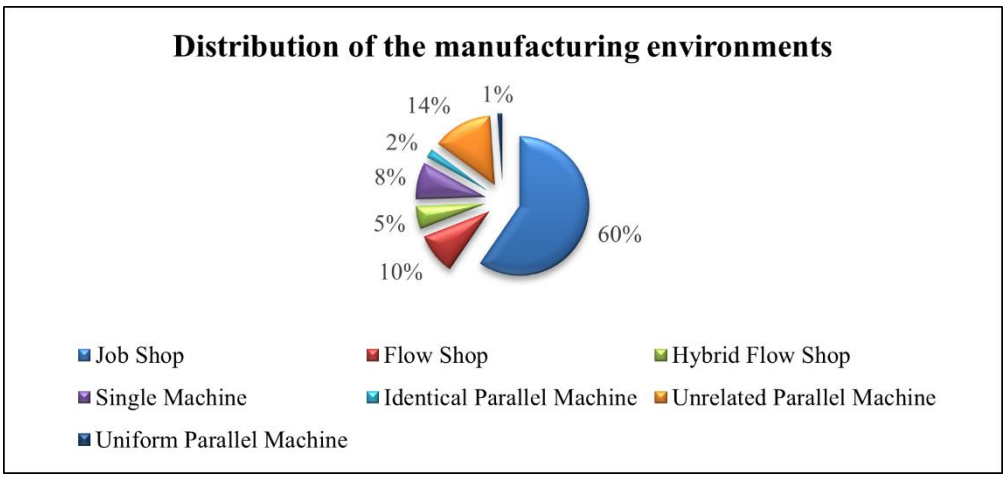

Figure 2.4 Distribution of the manufacturing environments

Figure 2.5 summarizes the base RL algorithms used in the papers. It is seen that the Q-learning algorithm is the most used, and it is followed by Temporal Difference and Deep RL algorithms. Especially in recent years, there has been a significant interest in Deep RL algorithms.

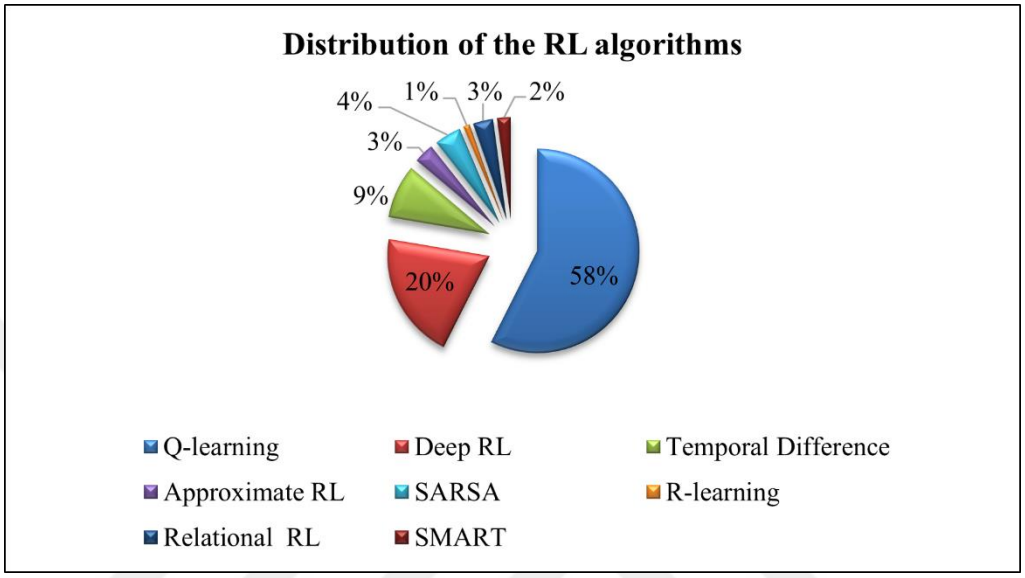

Figure 2.5 Distribution of the RL algorithms

Another analysis is related to the objective type of problem. Most of the problems were studied as single objective (SO) problems, as illustrated in Figure 2.6. However, there is a lack of studies considering multi-objective (MO) problems, especially in the studies that discuss single and identical parallel machines.

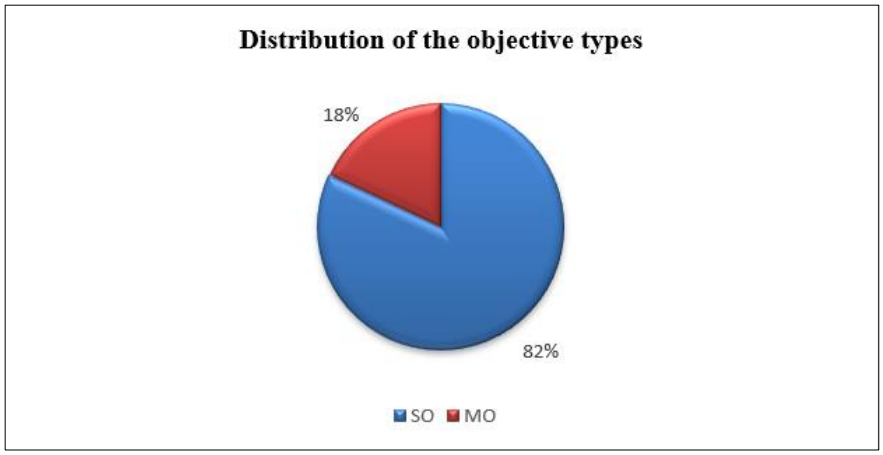

Figure 2.6 Distribution of objective types in the papers

In addition, the type of agent used in the papers was analyzed, as illustrated in Figure 2.7. Most papers applied the RL method with a single agent (71 out of 94).

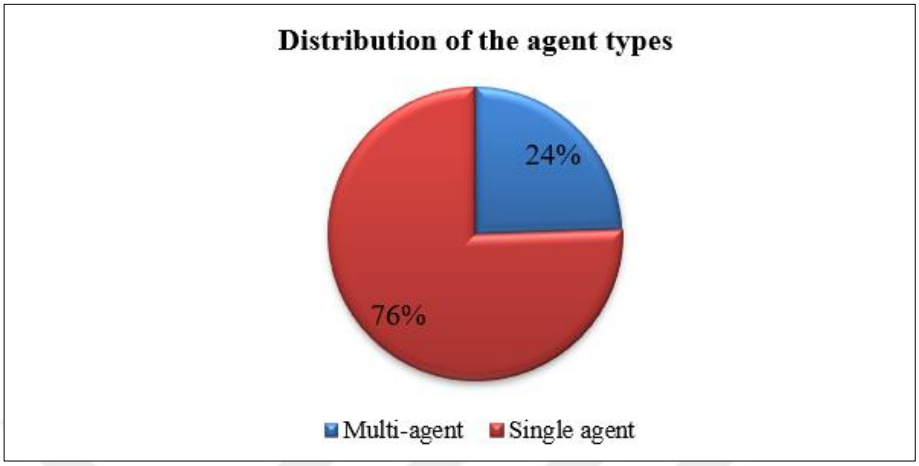

Figure 2.7 Distribution of the agent types

The action selection mechanism is a critical factor affecting exploration and exploitation trade-offs. In the related literature, due to the simplicity of the procedure, the $\varepsilon - greedy$ method was selected frequently (68 out of 94), followed by the Boltzmann probability method (11 out of 94) and SA temperature procedure (6 out of 94). The distribution of the action selection mechanism is given in Figure 2.8.

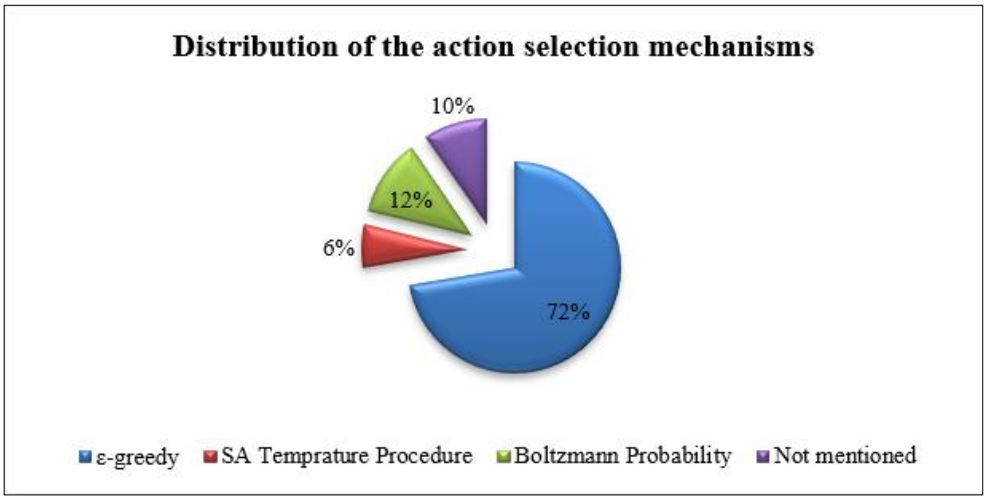

Figure 2.8 Distribution of the action selection mechanism

Especially in the problems with continuous state space or vast state space, another vital issue in RL is the state definition strategy. As seen in Figure 2.9, state aggregation and NNs methods are chiefly applied to deal with this problem.

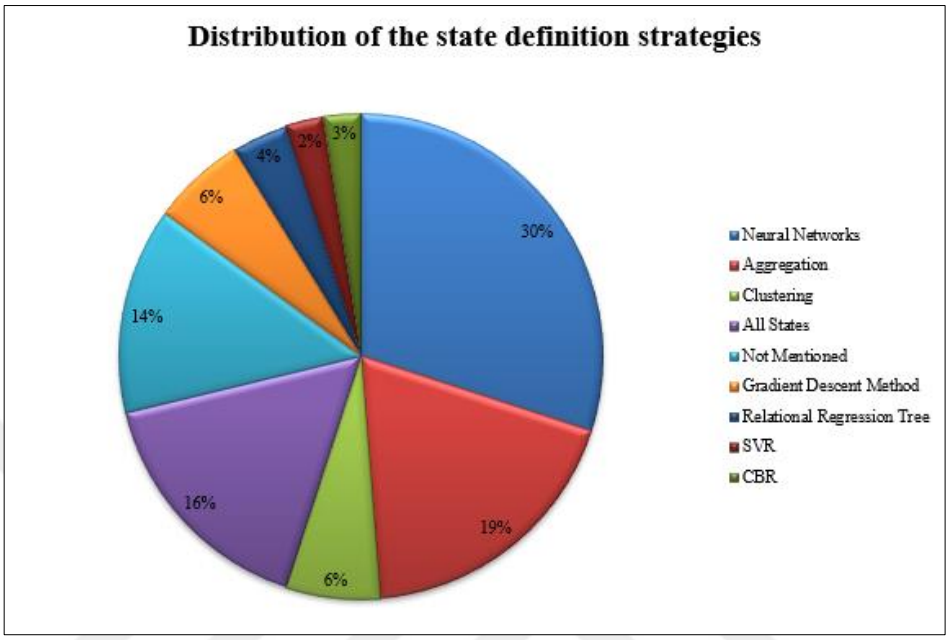

Figure 2.9 Distribution of the state definition strategies

Finally, the system types studied in these papers were analyzed, as depicted in Figure 2.10. The RL methods are frequently applied to problems in stochastic systems. In some papers, the RL algorithms were applied to deterministic problems to compare the effectiveness of the algorithm with benchmark problems.

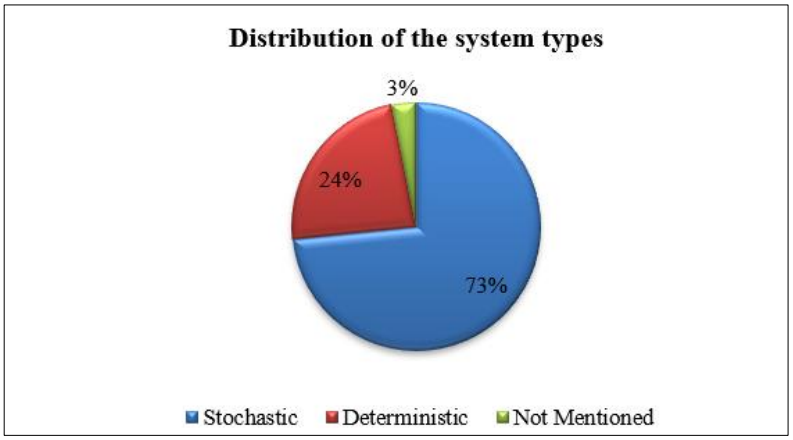

Figure 2.10 Distribution of the system types

2.3.3 Literature Analysis According to Machine Environment

In this section, machine scheduling problems are analyzed according to manufacturing environments. The abbreviations used in the following sections are given in Table 2.3.

Table 2.3 The explanations of abbreviations used in this section

Abbreviation	Explanation	Abbreviation	Explanation
MOEO	Multi-objective Evolutionary Optimization	WCOVERT	Weighted Cost Over Time
GA	Genetic Algorithm	AIS	Artificial Immune System
WSPT	Weighted Shortest Processing Time	WMDD	Weighted Modified Due Date
BBC	Behavior-Based Control	CONWIP	Constant Work in Process
MST	Minimum Slack Time	LPT	Longest Processing Time
EDD	Earliest Due Date	FIFO	First In First Out
ATC	Apparent Tardiness Cost	SPT	Shortest Processing Time
IG	Iterated Greedy	RVI	Relative Value Iteration

2.3.3.1 Single Machine Scheduling Problems

We found eight papers in the related literature that handle the single-machine environment. Wang & Usher (2004), Kong & Wu (2005), and Wang & Usher (2005) applied the Q-learning algorithm to the problem of $1//\frac{1}{n}\sum T_i$, $1//\frac{1}{n}\sum T_i$, T_{max}, $\sum U_i$, $1//L_{max}$, $\sum U_i$, $\frac{1}{n}\sum L_i$, respectively. The performances of the algorithm were compared with dispatching rules, and the rule selection percentages of the agent were investigated. The results of the studies show that the Q-learning algorithm learns to select the best dispatching rule previously known for each objective function. Wang & Usher (2004) also experimented to find essential factors that affect the performance of the Q-learning algorithm. They simulated the system under four conditions that considered the due date tightness and loading frequency of jobs and tried to determine the best factor level combinations for these conditions. The Q-learning obtained the best result in heavy loading and loose due date condition. Another important finding is that the agent's learning performance is improved when the policy table contains narrower ranges for the states and more ranges for the reward function. Idrees et al.

(2006) applied the SMART algorithm to the problem of $1//\frac{1}{n}\sum T_i + TC$. They aimed to minimize the mean tardiness and the cost of hiring extra workers. The agent decides on the number of workers and dispatching rules simultaneously in every state. This study is the first study that addresses the single machine scheduling problem as a multi-objective problem, and the study showed that the RL algorithm obtained better results than dispatching rules in the multi-objective case. Atighehchian & Sepehri (2013) developed a new function-based approach with Simulated Annealing and Neural Network (SANN) and applied the novel approach to the problem of $1/S_{jk}/\sum w_i T_i$. In this approach, contrary to previous studies, the job that will be processed next is selected rather than the dispatching rule at each decision point. The results of the algorithm were compared with dispatching rules and the Q-learning algorithm. It is concluded that the novel approach outperforms both the Q-learning algorithm and the dispatching rules. Xanthopoulos et al. (2013) studied the problem of $1/S_{jk}/\frac{1}{n}\sum T_i + \frac{1}{n}\sum E_i$ as a multi-objective optimization problem, they aimed to develop two adaptive schedulers based on RL, and the other is based on Fuzzy Logic. They compared the results of the schedulers with 15 dispatching rules. The results revealed that the RL and the fuzzy schedulers outperform all dispatching rules when the mean tardiness and earliness objectives are minimized simultaneously. Xiao et al. (2017) studied the problem of $1//CT$ in a batch processing system using the Q-learning algorithm and compared the method with the RVI method. This study showed that the Q-learning algorithm obtained only 0.141%-1.118% above the optimal value obtained by RVI; besides, it got results 100 times faster than RVI. Wang et al. (2021) applied the Q-learning algorithm to the problem of $1//\sum T_i + C_{max} + \sum w_i C_i$. Differently from other studies, they handled the integrated optimization of machine maintenance and scheduling problem and aimed to balance them. The result of the study indicated that Q-learning outperformed the selected dispatching rules and learned the proper strategy.

Generally, in the related literature, single machine scheduling problems have been handled without constraints, or only sequence-dependent setup time has been considered in a few papers. The effectiveness of the algorithm has been compared with

dispatching rules. These studies showed that the Q-learning algorithm tends to select the best dispatching rule previously known for each objective function. However, it was concluded that it did not outperform the dispatching rules. Also, the application of the dispatching rule is effortless in comparison with the RL algorithms. Therefore, RL does not provide an advantage for this scheduling problem.

On the other hand, although a limited number of studies examine the problem as a multi-objective problem, these studies showed that the RL algorithm gets better results than dispatching rules. There is no single dispatching rule for multi-objective problems that yields optimal results. Besides, dispatching rules are myopic. However, in these cases, the RL algorithm combines dispatching rules properly and benefits from the combination of dispatching rules in balancing the objectives. Since real-life scheduling problems usually include multiple objectives, applying the RL algorithms to multi-objective problems should be analyzed deeply.

2.3.3.2 Parallel Machine Scheduling Problems

In the related literature, two studies handle identical parallel machine problems. Yuan et al. (2013) and Yuan et al. (2016) applied the Q-learning algorithm to the problems of $P_m//L_{max}, \sum U_i$ and $P_m/bkdwn/L_{max}, \sum U_i$, respectively, for the selection of dispatching rules SPT, EDD, and FIFO. The studies show that the learning agent tends to choose the proper dispatching rule among SPT, EDD, and FIFO, which gives better results for each objective function.

In the context of a uniform parallel machine environment, Guo et al. (2020) handled the problem of $Q_m//T_{max} + \sum U_i + \frac{1}{n}\sum W_i$ and proposed a novel multi-stage Q-learning method to balance the different objectives. They compared the results of the proposed method with dispatching rules and the single-objective Q-learning method. The results of the study showed that the proposed method chooses the proper dispatching rules consistent with the objective more frequently and that the proposed method is more successful in balancing the different objectives.

The application of RL to the unrelated parallel machine scheduling problem was examined from different perspectives by researchers. Csáji & Monostori (2005) applied variants of the Q-learning algorithm to the problem of $R_m/bkdwn\ prec/C_{max}$ in various scheduling environments (deterministic-stochastic, static-dynamic). There is no benchmark method; instead, the performance was evaluated by the adaptivity of the algorithm to unexpected events such as new job arrivals and machine breakdowns. As a result, it is shown that this system easily adapts to unexpected events using previously obtained information. Zhang et al. (2007) and Zhang et al. (2012) handled the problem of $R_m/\ S_{jk}/\frac{1}{n}\sum w_iT_i$ and $R_m//\sum w_iT_i$, respectively, and compared the results of the algorithm with the results of selected heuristic rules. The results of the study indicated that the Q-learning algorithm outperforms heuristic rules in all test problems remarkably. They indicated that a heuristic is myopic; however, Q-learning can combine the advantages of different heuristics. Paeng et al. (2021) also studied the problem of $R_m/\ S_{jk}/\frac{1}{n}\sum T_i$ but applied a deep reinforcement learning-based solution approach, and they proposed a Neural Network design to solve the different scheduling problems data not involved in the training set. The experimental results demonstrated that the performance of the deep RL outperforms other dispatching rules and the computation time of the deep RL is shorter than considered metaheuristics. Iwamura et al. (2010) studied the problem of $R_m//\sum F_i + TC$ as a multi-objective problem and applied the Q-learning algorithm. They only analyzed the average improvement ratio of the objective function values, and they showed that the proposed methods improve the objective function values. Luo et al. (2021) also handled the problem of $R_m/\ S_{jk}/\frac{1}{n}\sum T_i + TC$ as a multi-objective problem under multi-resource constraints, they compared the performance of the proposed method with dispatching rules and indicated that the proposed method gets better results than dispatching rules. Palombarini & Martínez (2010), Palombarini & Martínez (2012a) and Palombarini & Martínez (2012b) applied the Relational RL algorithm to the problem of $R_m//\sum T_i$, $R_m/S_{jk}\ bkdwn/\sum T_i$, and $R_m/S_{jk}\ bkdwn/\sum T_i$ for repairing schedules, respectively. Unlike other studies, the Relational RL algorithm was applied to choose predetermined repair operators for repairing schedules instead of scheduling all jobs. They analyzed the average improvement as a performance measure, indicating

that the RL algorithm could learn how to repair schedules against unexpected events. Palombarini & Martínez (2012b), differently from these studies, only analyzed the learning ability of the agent under different objectives and implied that the agent achieved the goal after 450 episodes, and only between 5 and 8 repair steps were sufficient to repair the schedule. Zhang et al. (2011) studied the problem of $R_m/\ S_{jk}\ prec/TC$ in a semiconductor test scheduling problem which includes re-entrance of jobs. They applied the SARSA algorithm to industrial data sets and compared the results with individual dispatching rules and the industrial method the company had used for scheduling up to that time. The results of the study demonstrated that the SARSA algorithm outperforms all the investigated methods in each test problem. Ábrahám et al. (2019) studied the problem of $R_m/prec/C_{max}$ and they defined the states as the order of the tasks and calculated Q-values for task sequencing. They compared their method with benchmark problems and showed that the method could find the optimal solutions. The benchmark problems were solved by CPLEX. The results showed that RL obtained the same or better results much faster than CPLEX. Zhou et al. (2020) addressed the problem of $R_m//C_{max}$ in smart manufacturing and applied a deep RL-based method to two case studies to analyze the usefulness of the proposed method. Chien & Lan (2021) studied the problem of $R_m/S_{jk}/C_{max}$ with product families and aimed to integrate the GA into deep RL to generate more suitable training data. The results of the study showed that the deep RL agent outperforms dispatching rules and is more stable in various production scenarios.

The results of this section reveal that in comparison to identical and uniform parallel machine scheduling problems, RL mostly applied to the unrelated parallel machine scheduling problem. The study in uniform parallel machine environments demonstrates that Q-learning is more successful than dispatching rules in balancing objective functions simultaneously. In unrelated parallel machine environments, the performance of the RL was examined in various aspects. In three of these studies, different from others, RL algorithms were applied to repair the production schedules instead of constructing the schedule. As a result, it is demonstrated that the RL easily adapts to unexpected events. Studies in which the RL algorithm was compared with dispatching rules in unrelated parallel machine environments present that the RL

compounds the advantages of dispatching rules and learns how to combine them appropriately considering the objective of the problem; in this way, RL outperforms the dispatching rules.

Only two studies in unrelated parallel machine environments discuss the problem as multi-objective. However, the algorithm's results have not been compared with other dispatching rules or methods in one of them. The studies in the single machine and uniform parallel machine environments demonstrate the effectiveness of the RL algorithm in multi-objective cases. This case also can be investigated for unrelated parallel machine scheduling problems. The use of metaheuristics as a solution approach occupies a significant place in machine scheduling problems. Hence, the performance of the RL algorithms can be compared with metaheuristics, and the algorithm's superior and weak properties can be revealed.

2.3.3.3 Flow Shop Scheduling Problems

In the related literature, nine studies were found using the RL algorithms to flow shop scheduling problems. Paternina-Arboleda & Das (2001) applied the SMART algorithm to the problem of $F/bkdwn/WIP$ to develop intelligent dynamic control policies. They analyzed a serial line production system with four machines and a single product type. The performance of the algorithm was compared with Kanban, CONWIP, and BBC policy based on average WIP. The results of the study showed that the RL method and BBC policy yielded better results than other methods. Hong & Prabhu (2004) studied the problem of $F/S_{jk}/DD + SC$ as a multi-objective scheduling problem and applied the Q-learning algorithm for batch sequencing and sizing in a deterministic manner. They aimed to develop an approach suitable for Just-in-Time (JIT) production systems balancing the due-date deviation cost and setup cost. The performance of the approach was compared with four dispatching rules (EDD, MST, SPT, and LPT). The study showed that RL outperformed dispatching rules, especially for jobs with a tight due date in complex manufacturing systems. Arviv et al. (2016), Reyna et al. (2018), Reyna et al. (2019a), Lee & Kim (2021), Pan et al. (2021), and Brammer et al. (2022) applied the Q-learning algorithm to the problem

of $F//C_{max}$. Unlike classical flow shop problems, Arviv et al. (2016) considered scheduling the robots to minimize makespan for different collaboration levels. Lee & Kim (2021) also studied the robotic flow shop and applied the Q-learning algorithm to determine the robot task sequence. Experimental results showed that the RL algorithms get better results than dispatching rules and heuristics, which are widely used in scheduling robotic flow shop problems. Reyna et al. (2018) analyzed the effects of parameters on the learning process. The algorithm solved benchmark problems with the reported optimal results to validate the quality, and it was shown that the average relative error of the Q-learning algorithm is less than 1.00 %. Reyna et al. (2019a) incorporated the Nawaz-Enscore-Ham (NEH) heuristic into the Q-learning algorithm, and the results are compared with various metaheuristics from the literature. Experimental results show that Q-learning outperforms most of the metaheuristics. Pan et al. (2021) also proposed a model that integrates the NEH heuristic and deep RL to improve the performance of the algorithm. They compared the result of the proposed methods with different hybrid NEH heuristics from the literature and indicated that the deep RL improves the performance of the NEH significantly. Brammer et al. (2022) proposed a new flow shop problem that included multiple lines at the beginning of the production. They evaluated the performance of the algorithm on benchmark problems and demonstrated that the proposed method gets better results than the selected heuristics in a shorter computational time. Yang et al. (2021) studied the problem of $F//\sum T_i$ using deep RL. A dynamic system is designed to analyze the performance of the algorithm in real-time scheduling problems. The proposed method was compared with IG and GA, indicating that the proposed method gets better results regarding solution quality and computational time.

Applications of RL to hybrid flow shop scheduling problems have been studied in recent years. Qu et al. (2015) studied the problem of $HF/S_{jk}\ block/WIP + TC$ for proactive scheduling, they applied the approximate Q-learning algorithm and compared the performance of the algorithm with the FIFO method and heuristic greedy method. The results of the study verified that the approximate Q-learning method outperforms other methods and converges to optimal schedules faster. Qu et al. (2016a) studied the problem of $HF/S_{jk}\ block/TC + \sum U_i$ with a multi-skilled

workforce and multiple machine types. Different from the previous study, they combined machine scheduling with workforce scheduling. The Q-learning algorithm was applied to update the production schedules dynamically and enable real-time collaboration between the machines and the workforce. However, they did not compare the results of the algorithm with any other method. They only analyzed the average reward for diverse workforce skill sets. Han et al. (2019) and Fang Guo et al. (2020) applied the Q-learning algorithm to the problem of $HF//C_{max}$ and handled a small size case study to verify the algorithm. They compared the results of the Q-learning algorithm with the results of the GA. The results of the studies demonstrated that the Q-learning algorithm got better results than GA in these examples. Reyna et al. (2019b) studied the problem of $HF/S_{jk}\ prec/C_{max}$ in a deterministic environment. This study is essential due to considering the hybrid flow shop scheduling problem with additional constraints. They showed that the Q-learning algorithm obtains satisfactory results in complex scheduling problems.

To sum up, the RL algorithms were mainly applied to minimize makespan in flow shop scheduling problems. One of the studies tried to schedule robot transfers to minimize the makespan. Two others tried to find a suitable permutation of jobs. However, state definitions and actions were not defined clearly in these studies. Besides, the state space of the problem was too large, and it was too hard for the agent to visit each state. Finally, one study discusses the problem as a multi-objective problem, concluding that RL outperforms dispatching rules.

In the related literature, the minimization of makespan is the most studied optimality criterion in hybrid flow shop problems. However, these studies examined particular cases or small-scale problems. Further analysis is needed to investigate the effectiveness of the RL algorithm in large-scale problems. Another critical deficiency is that although real-life scheduling problems include dynamic properties, the studies in the related literature generally tackled the problem in a static environment. The studies on parallel machine scheduling problems demonstrated that the RL algorithm adapts to sudden changes. Therefore, this case can be a promising area for hybrid flow shop problems. For the multi-objective case, only one of the two studies compared the

performance of the algorithm with other methods and remarked that RL outperforms other methods. The other study only analyzed the average reward for different conditions. This issue also should be handled to compare the effectiveness of the RL algorithms with other methods.

2.3.3.4 Job Shop Scheduling Problems

Job shop scheduling problems (JSSP) have been frequently studied in the RL field. We have found 56 papers dealing with job shop scheduling problems and analyzed them based on the optimality criterion in the following sections.

2.3.3.4.1 JSSP with Makespan Criterion. Job shop scheduling problem with a makespan optimality criterion $J//C_{max}$ is addressed by Gabel & Riedmiller (2007a), Gabel & Riedmiller (2007b), Gabel & Riedmiller (2008), Yingzi et al. (2009), Aissani et al. (2009), Gabel & Riedmiller (2011), Reyna et al. (2015), Lin et al. (2019), Liu et al. (2020), Ren et al. (2020), Jiménez et al. (2020), Samsonov et al. (2021), and Park et al. (2021). Gabel & Riedmiller (2007a) applied the Q-learning algorithm to job shop benchmark problems. Gabel & Riedmiller (2007b) analyzed the performance of the algorithm on large-scale benchmark problems. Gabel & Riedmiller (2008) developed an approach in which each machine was assigned to an agent, and each agent decided which job to process next. These three problems were modeled as multi-agent cases and utilized a NN for value function approximation. Aissani et al. (2009) applied the SARSA algorithm utilizing Boltzmann probability in the action selection process. The study's results were compared with GA, showing that RL outperformed the GA. Gabel & Riedmiller (2011) applied the policy gradient RL algorithm and discussed the problems as sequential decision problems utilized by independent learning agents. Reyna et al. (2015) analyzed the effectiveness of the Q-learning algorithm by solving benchmark instances from the literature. They solved 15 instances that are the combination of 5 machines and 10, 15, and 20 jobs. The results showed that the algorithm finds the optimum value for 11 out of 15 problems and approximates the remaining instances very closely. Lin et al. (2019) applied the deep RL algorithm in a smart manufacturing factory. Liu et al. (2020) applied the actor-critic algorithm to

problems in static and dynamic job shop environments. Yingzi et al. (2009) applied the Q-learning algorithm for the dynamic scheduler. All of them compared the results of the algorithm with dispatching rules and demonstrated that the RL algorithm get better results in each case than a single dispatching rule strategy. Ren et al. (2020) applied the actor-critic algorithm and compared the effectiveness of the method with benchmark problems. Jiménez et al. (2020) developed a job shop scheduling tool using the multi-agent RL approach. The tool allows users to set the parameters of the RL method and adjust the schedule according to the shop floor conditions. Park et al. (2021) used a graph neural network to represent and determine the states and analyzed the generalizability capability of the algorithm. Samsonov et al. (2021) applied the deep RL algorithm and analyzed the performance of the algorithm in terms of solution quality, generalization capability, and solution speed and compared it with dispatching rules.

Park et al. (2020) addressed the problem with sequence-dependent setup times ($J/S_{jk}/C_{max}$). They applied deep RL algorithm and analyzed the efficiency of the proposed algorithm comparing with GA and dispatching rules. Numerical experiments showed that the proposed algorithm is superior to GA and dispatching rules, and besides, this proposed method ensures shorter computing time, approximately ten times that of GA.

Zhang & Dietterich (1995), Zhang & Dietterich (1996), Monostori et al. (2004), and Aissani et al. (2012) addressed the problem by considering precedence constraints ($J/prec/C_{max}$). Zhang & Dietterich (1995) applied a temporal difference algorithm to the problem to obtain a feasible schedule without any constraint violation and compared the results with the iterative repair method. Zhang & Dietterich (1996) also studied the same problem but compared the proposed algorithm with the NN method. Monostori et al. (2004) applied a temporal difference algorithm utilizing Boltzmann probability in the action selection process. They investigated how much computation is required to construct a schedule in case of new job arrivals. The flexibility of the algorithm was compared with the Branch and Bound algorithm. Aissani et al. (2012) applied the SARSA algorithm and compared the algorithm with a mixed-integer linear

program (MILP) and GA. The results of the study showed that the RL algorithm falls behind the MILP by an average of 3% deviations, but it reaches these results in less computational time. Besides, it is demonstrated that the RL algorithm is faster than GA in reacting to disturbances.

Csáji et al. (2006) and Zhao et al. (2019) addressed the problem with machine breakdowns ($J/bkdwn/C_{max}$). Csáji et al. (2006) aimed to investigate the reactivity of the algorithm to the machine and job disturbances. The results of the study showed that the proposed algorithm used previous information in unexpected events, thus accelerating the scheduling process. On the other hand, Zhao et al. (2019) compared the results of the method with dispatching rules. The results of the study showed that the Q-learning algorithm learns which dispatching rule is the right choice in specific states and thus gets better results than single dispatching rule selections in the case of machine failures. Aissani & Trentesaux (2008) considered the job shop scheduling problem as a multi-objective problem, and they aimed to minimize both makespan and machine downtime ($J/bkdwn/C_{max} + MDT$) and applied the SARSA algorithm to the benchmark problems in the literature. They determined the makespan as an effectiveness criterion and machine idle time as an efficiency criterion. The results of the study indicated that the proposed algorithm is effective and efficient, especially in scheduling problems that include machine disturbances. Zhou et al. (2021a) also considered a multi-objective job shop scheduling problem ($J//C_{max} + TC + U$) and applied Deep RL using composite rewards. They indicated that composite rewards help the agent balance multiple objectives and get better results.

In the relevant literature, most of the papers handle the problem without constraints or job characteristics. These studies generally compared the RL with dispatching rules in dynamic environments, showing that RL outperformed the dispatching rules. The problem instances with only up to 15 jobs were solved in static environments. Therefore, large-scale problem instances should also be solved to analyze the algorithm's performance in static environments. A significant part of the studies in this section analyzed the reactiveness of the algorithm to dynamic events such as new job arrival, machine breakdowns, and job cancellation. Studies demonstrated that the RL

algorithm easily compensates for the disturbances, uses previous information to reschedule, accelerates the scheduling process, and reacts more quickly than other methods. These findings are significant because they demonstrated that the RL algorithms are practical for real-life scheduling problems. Also, the sequence-dependent setup times are frequently encountered in many real-life scheduling problems; this constraint is handled only in one paper. It can be noted that this problem can be studied with different job characteristics and constraints, such as blockage, and sequence-dependent setup times, which are frequently encountered in real-life problems.

2.3.3.4.2 JSSP with Tardiness Related Criterion. Job shop scheduling problems with mean tardiness criterion ($J//\frac{1}{n}\sum T_i$) are studied by Aydin & Öztemel (2000), Wei & Zhao (2004), Yang & Yan (2007), Yang & Yan (2009), Wang & Yan (2016). Aydin & Öztemel (2000) developed a new version of the Q-learning algorithm called Q-III that uses past experiences of the agent with a strong predictor. The agent decides on which dispatching rule will be used. The behavior of the agent was compared with individual dispatching rules in different due date tightness conditions. The study showed that the agent obtained better results than dispatching rules except in one case. Wei & Zhao (2004) applied the Q-learning algorithm for composite rule selection, including job and machine selection. They indicated that the agent obtains better results than other rules by giving flexibility of switching rules in real-time. In the studies of Yang & Yan (2007) and Yang & Yan (2009), the dispatching rule was selected by a version of Q-learning named B-Q learning. Experiments showed that the agent gets better results than the single dispatching rule in each different due date urgency condition. Wang & Yan (2016) applied the Q-learning algorithm and incorporated a fuzzy reward into the Q-value updating process. They aimed to handle the large state space problem of the Q-learning and compared the results with different RL approaches.

Gabel & Riedmiller (2006a) and Wang & Usher (2007) addressed the problem of $J/S_{jk}/\frac{1}{n}\sum T_i$. Gabel & Riedmiller (2006a) developed a method that tracks the learning process, decides whether it is better to stop it, and utilizes a NN for value

function approximation. The proposed method prevents overfitting and gets better results than classical Q-learning. Wang & Usher (2007) investigated the impacts of the Q-learning algorithm factors such as state determination criterion, the number of reward ranges, the reward magnitude, and the range interval size. They tested these factors under different system conditions, considering the time between job arrivals and due date tightness and recommended factor levels for each system condition.

Lee et al. (2020) handled the problem of $J//\sum wT_i$ using deep RL and evaluated the performance of the proposed method with respect to the dispatching rules. The results of the study showed that the RL method outperforms the dispatching rules and also ensures better results in dynamic environments. They also indicated that RL has better performance when exact process times are unknown.

Wang & Yan (2013a) studied the problem of $J/bkdwn\ block/\frac{1}{n}\sum T_i$, and applied the Q-learning algorithm to select a scheduling strategy based on the multi-agent system in a dynamic environment. They constituted agent groups with different missions that interact with each other to find the optimal policy. Results of the study showed that the proposed method learns to select proper scheduling policy in changing situations through interactions and outperforms the B-Q learning proposed by Yang & Yan (2009).

Riedmiller & Riedmiller (1999) and Luo et al. (2020) studied the problem of $J//\sum T_i$ applying the Q-learning and deep RL methods, respectively. They compared the performance of the agent with dispatching rules and analyzed the generalization ability of the agent. Results of the study indicated that the RL algorithm outperforms dispatching rules, and the learning agent can generalize obtained information to an unknown case. Luo et al. (2020) also measured the performance of the deep RL method against the Q-learning method. The experiments demonstrated that the DQN agent outperforms the stand Q-learning agent in most test instances. Gabel & Riedmiller (2006b) handled the problem of $J/bkdwn\ prec/\sum T_i$ in multi-agent domains and embedded the algorithm in the CBR framework. The CBR framework is utilized to tackle the high dimension state-space problem and the complexity of the large-scale

problems. The results of the study demonstrated that the proposed method gets better results than dispatching rules.

Miyashita (2000), Wang (2018), Kuhnle et al. (2021), and Zhou et al. (2021b) studied the job shop scheduling problem as a multi-objective problem. Miyashita (2000) applied a temporal difference algorithm to the problem of $J//\sum w_i T_i + WIP$. The case-based function approximation was utilized to handle the high dimensional state space problem. The results of the study reported that the proposed method quickly improves the schedule quality. Wang (2018) handled the problem of $J//\sum E_i + \sum T_i$ and developed a new method that consists of multiple agents and a new state clustering method. The dynamic greedy strategy was utilized to improve the solution speed. The results of the proposed method were compared with other RL algorithms. The experimental results showed that the proposed method improved all objective values obtained by other algorithms. Kuhnle et al. (2021) studied the problem of $J/block, bkdwn, S_{jk}/\frac{1}{n}\sum w_i T_i + U + TP$ and applied deep RL. They compared the performance of the algorithm with dispatching rules and analyzed the effect of the parameters of the algorithm. Zhou et al. (2021b) handled the problem of $J//\frac{1}{n}\sum w_i T_i + U$ and applied multi-agent RL. They developed a new reward function to improve the learning abilities of the agents and applied the proposed method to real-world case studies. As a result of the study, they indicated that the proposed method significantly improves the learning ability of the agents and helps the agent overcome unexpected events.

In the job shop scheduling problems with the tardiness-related criterion, most papers aimed to minimize mean tardiness and did not consider different constraints and job characteristics. The studies handled the problem in a dynamic environment and compared the performances of the algorithms with dispatching rules. The results show that the RL algorithms outperform dispatching rules since the learning agent can switch dispatching rules based on the changing state of the systems. Besides, the studies indicated that the RL algorithm quickly reacts to unexpected events and generalizes its previous information to unknown situations. These features

demonstrate that the RL algorithm is a promising solution for real-life scheduling problems that include dynamic events and require fast decision-making.

On the other hand, it is challenging for the RL algorithms to have high dimensional state space for these problems. This is because many state features should be considered to appropriately handle job shop scheduling problems. Most of the time, it is hard to reduce state space by aggregating state features. Therefore, the researchers attempted to utilize a neural network or a CBR framework. These studies showed that the proposed methods improved the speed of convergence of the algorithm.

Considering real-life scheduling problems, they include more factors such as sequence-dependent setup times, machine breakdowns, blockages that affect the tardiness of the jobs, and multiple objectives conflicting with each other. These characteristics and constraints have rarely been studied in the related literature. Therefore, it can be beneficial to analyze these factors to make problems more realistic. In addition to these, sometimes, the minimization of the weighted tardiness criterion becomes a critical issue. Therefore, this problem can be researched to implement the RL algorithm in real-life scheduling problems.

2.3.3.4.3 JSSP with Other Optimality Criterion. Liu et al. (2001) and Ramírez-Hernández & Fernandez (2005) addressed the job shop problem with reentrant lines considering blockage constraints to maximize throughput and minimize total cost. Ramírez-Hernández & Fernandez (2009) applied a temporal difference algorithm to the problem of $J/block\ S_{jk}\ bkdwn/WIP$ and compared the results with dispatching rules. They suggest using the dispatching rules since they obtain better or similar performance than RL and are more straightforward to implement. Kim & Shin (2017) also handled the reentrant lines and applied the actor-critic algorithm to the problem of $J/S_{jk}\ bkdwn/WIP, CT$. They analyzed the problem under two objectives separately. The RL algorithm achieved a better performance than dispatching rules only in this study among the studies mentioned above.

Monostori & Csáji (2006) studied the problem of $J/pmtn\ S_{jk}/\sum L_i$, and utilized the support vector machine method for value function approximation. Csáji & Monostori (2008) also studied the same problem. However, different from the previous study, they aimed to minimize the number of tardy jobs.

Wang & Yan (2013b) addressed the problem of $J//\frac{1}{n}\sum E_i$ including the reprocessing of the jobs, and Qu et al. (2016b) applied an approximate RL algorithm to the problem of $J/block/WIP + \sum U_i$. Both of them compared the results of the algorithm with dispatching rules. Computational results indicated that the RL algorithm outperforms dispatching rules.

Shiue et al. (2018) studied the problem of $J//CT + \sum U_i +$ TP as a multi-objective problem and applied the Q-learning algorithm for real-time scheduling in an intelligent factory. They emphasized that the RL algorithm is suitable for the smart factory due to autonomous dynamic decision-making properties to changing shop-floor conditions. Wang et al. (2020) studied the problem of $J//\sum w_i E_i + TC$ and proposed a dual Q-learning algorithm to balance the localized targets and global targets and compared the results of the proposed method with the standard Q-learning method. Findings demonstrated that the proposed method is more adaptive than the standard Q-learning in situations with a high job arrival rate and an increasing number of products.

Zhang et al. (2017) and Zhang et al. (2018) studied the problem of $J//CT$ problem by applying the Q-learning algorithm. The second study also included the job batching process in the problem. Results of the study showed that the RL algorithm obtained a smaller cycle time value than dispatching rules. Bouazza et al. (2017) addressed the problem of $J/S_{jk}\ Prec/\sum w_i W_i$, and analyzed the distribution of machine and job selection rules of the agent. The RL algorithm improved the objective value between 19% and 46%, obtained by dispatching rules. Thomas et al. (2018) developed a method based on the Q-learning algorithm named MINERVA and solved the problem of $J//TP$ to detect bottleneck resources. They indicated that the proposed method could identify bottlenecks with high accuracy. Lee & Lee (2022) also studied the same

problem, but they considered the blockage of the machines ($J/block/TP$). They applied deep RL to obtain robust solutions in dynamic environments. Stricker et al. (2018) applied the Q-Learning algorithm to the problem of $J/bkdwn\ block/U$ for adaptive order dispatching and compared the results with the FIFO rule. Experimental results demonstrated that the RL algorithm improves the utilization more than the dispatching rule in a shorter time. They also indicated that the RL algorithm is more adaptive to changing conditions. Waschneck et al. (2018a) and Waschneck et al. (2018b) addressed the job shop problem with sequence-dependent setup times and applied the deep RL algorithm aiming to maximize machine utilization ($J/S_{jk}/U$) and minimize cycle time ($J/S_{jk}/CT$), respectively. They indicated that the proposed method could be a promising solution since the algorithm developed a good schedule without human intervention and obtained the same results reported in the literature without prior expert knowledge.

This section analyzes the studies that addressed different optimality criteria out of the makespan and tardiness-related criteria. It is seen that the minimization of WIP and Cycle Time were the frequently studied criteria. Besides, due date-related criteria such as earliness and lateness were also studied. The performance of the algorithm in dynamic environments is generally compared with dispatching rules. Most of the papers demonstrated that the RL algorithm outperforms dispatching rules. Unlike adopting single dispatching rules, the RL algorithm dynamically decides which dispatching rule will be used considering the changing conditions of the production system. The performance of the algorithm in the static environments was analyzed on small-scale problems. Results of the study showed that the RL algorithms obtained promising results on benchmark problems; however, it needs to be extended to large-scale problems. Some studies applied the RL algorithm to job shops with reentrant lines. Only one of the papers reported that the RL algorithm outperforms heuristic dispatching rules. Other studies said the algorithm needs to be improved to achieve better results than dispatching rules. Another issue that can be noted is that no study handled the flow time-related criteria. This criterion may be studied in future research.

2.3.4 Discussion

A comprehensive literature review concerning the applications of RL to machine scheduling problems was conducted. The scope of the literature review includes only machine scheduling problems. Therefore, other scheduling problems such as maintenance, carrier, gantry, and economic-lot scheduling were excluded. The detail of the findings and concluding remarks are introduced in the following subsections.

2.3.4.1 Important Findings

From the detailed analysis of the related papers, important findings can be summarized as follows:

- The efficiency of the RL algorithm in the single machine, parallel machine, and job shop environments was primarily compared with dispatching rules.

- The studies dealing with single and identical parallel machine problems have indicated that dispatching rules perform better than the RL algorithm in single-objective problems. In addition, dispatching rules provide superiority to the RL algorithms in terms of ease of implementation.

- The findings of the multi-objective problems in each machine environment have documented that the RL algorithms outperform dispatching rules since the RL algorithms learn to combine proper dispatching rules to achieve objectives simultaneously. Also, the RL algorithms learn to switch dispatching rules based on the changing state of the system.

- The minimization of the makespan is the most studied optimality criterion in the flow shop and job shop scheduling problems.

- The studies that consider the flow shop and job shop scheduling problems generally have analyzed the efficiency of the algorithm on small-scale

problems and compared the results with benchmark problems from the literature. In most papers, it has been reported that the RL algorithms achieved the best result.

- A significant part of the studies that handle the parallel machine and job shop scheduling problems have analyzed the adaptivity of the RL algorithms. The results have demonstrated that the RL algorithms quickly adapt to unexpected events using previously obtained information, accelerating the scheduling process. This feature makes the RL algorithm more applicable to real-life problems.

- Zhang (2011) and Jiménez et al. (2020) have applied the RL algorithm to industrial problems. The experimental results of the studies have indicated that the RL algorithm obtains better results in each case than a single dispatching rule strategy. Also, Shiue (2018), Lin et al. (2019), and Zhou (2020) investigated the RL application to dynamic scheduling in smart manufacturing. The studies have emphasized that the RL algorithm is suitable for smart factories because of the autonomous dynamic decision-making feature.

2.3.4.2 The Advantages and Strengths of the RL

The advantages and strengths of the RL, which have been reported in the related literature, can be identified as follows:

- Real manufacturing systems confront dynamic events such as machine breakdown, stochastic arrival of jobs, and order cancellation. The RL learns this dynamic system by getting feedback from the environment and can generalize the acquired information to new unseen situations. These features make the approach adaptive, flexible, and extensible. Therefore, it can be a suitable method for real manufacturing systems (Riedmiller & Riedmiller, 1999; Monostori et al., 2004; Csáji & Monostori, 2005; Gabel & Riedmiller,

2007a; Wang & Usher, 2007; Zhang et al., 2007; Zhang et al., 2011; Lee et al., 2020; Liu et al., 2020).

- Dispatching rules are useful and effortless in job sequencing problems. However, it needs expert knowledge to decide which dispatching rule will be applied in which state of the system. When the RL algorithms are applied to select dispatching rules, they provide autonomous decision-making depending on the system's state, reduce decision-making delay, and make it more applicable in fully automated manufacturing systems. Besides, the RL algorithms can consider many features of the system, switch between different dispatching rules and combine them appropriately. These characteristics of the algorithm give superiority to the algorithm against the single dispatching rule selection policy, especially in multi-objective problems (Riedmiller & Riedmiller, 1999; Aydin & Öztemel, 2000; Wei & Zhao, 2004; Zhang et al., 2007; Qu et al., 2016a; Zhao, 2019; Liu et al., 2020).

- The RL algorithms learn well by simulation without the need for complete information and quickly converge to near-optimal values (Zhang et al., 2011; Zhang et al., 2012; Lee et al., 2020).

- The metaheuristics such as Genetic Algorithm and Tabu Search obtains good results in static scheduling problems. However, they cannot adapt quickly to sudden changes in the system because they construct the whole schedule from scratch when the system changes. Unlike these algorithms, RL algorithms do not start scheduling from scratch, but instead, use previously obtained information. Therefore, this approach reacts to sudden changes easily and modifies the schedule in a shorter time (Paternina-Arboleda & Das, 2001; Csáji & Monostori, 2005; Aissani et al., 2009; Aissani et al., 2012; Atighehchian & Sepehri, 2013).

2.3.4.3 The Challenging Features of the RL

Despite the advantages and strengths of the RL mentioned above, the application of the RL in practice has some challenging features as follows:

- The main difficulty of the RL in applying to complex real-world problems lies in handling large and continuous state spaces. Therefore, the design of the state space and the accurate selection of state features are very crucial for the effectiveness and the scalability of the algorithm (Wang & Usher, 2004; Gabel & Riedmiller, 2006; Atighehchian & Sepehri, 2013; Wang & Yan, 2016; Xiao et al., 2017; Wang, 2018).

- The scalability of the RL is directly related to the design of the problem. The representation of all states of the environment in a Q-table can be applied only to small-scale problems and renders problems unscalable. Besides, the design of the states is specific to the number of jobs or machines in the problem, making it harder to scale to the larger problem. State aggregation or function approximation methods are utilized to handle this issue. The design of the problem and the selection of the proper function approximator or state aggregation directly affects the independence of the state-action space size from the problem instance size (Khadilkar, 2018; Qu et al., 2020).

- There is no restriction or strict rules in the reward system design. However, the design of the reward system can be complicated due to a large number of options (Xiao et al., 2017; Liu et al., 2020).

- Especially in multi-objective problems, it is crucial to design the reward system considering the importance level of the objectives. The reward function should be formulated to represent and balance each objective properly (Dulac-Arnold et al., 2019).

- The acts of the agent should be trustworthy and reasonably well in the exploration procedure when the learning occurs in the real system. Since the consequences of the actions will be observed in the real system, the actions of the agent in the exploration procedure may lead to high costs and low performance for these systems (Dulac-Arnold et al., 2019; Ding & Dong, 2020).

- The decision-making in scheduling can sometimes be overly critical; therefore, the explainability of the RL algorithm gains importance for the decision-makers. However, the RL algorithm lacks explainability as with many Machine Learning algorithms. The reason for an action of the agent cannot be clarified explicitly to decision-makers. Especially in the larger problems where the function approximator is used to estimate the value of the action, it gets complicated to explain why one action is chosen over another action in a particular state (Heuillet et al., 2021; Ding & Dong, 2020).

- The states should be visited a sufficient number of times to obtain good learning performance (Wang & Usher, 2004; Xiao et al., 2017).

- The RL algorithm does not guarantee to find optimal results (Xiao et al., 2017).

- In a single machine scheduling problem with a single objective, the RL method performs poorly than dispatching rules and is more complicated to implement.

- The parallel machine scheduling problem includes both job sequencing and machine selecting decisions. Considering these decisions, the main challenge here is how to design the scheduling problem and apply the RL algorithm. This characteristic of the problem type also causes a high-dimensional state space.

- The flow shop and job shop environment are constituted of multiple production stages, which means a scheduling problem arises at every stage. In the flow shop scheduling problems, this issue can be handled by applying the same

order of jobs or dispatching the jobs according to the FCFS rule at the next stages. While this approach may cause the algorithm to get a worse result, the application of the algorithm to each stage also results in large state space. Furthermore, job types have different routes in job shop scheduling problems, which means some jobs may not be processed in the first stage. In this situation, a decision-making problem arises at every stage, making it harder to implement the algorithm and causing the high dimensional state space problem.

2.3.4.4 Opportunities for Future Research

Having regard to the findings of the literature and the advantages and challenges of the RL discussed above, some of the opportunities for future research can be summarized as follows:

- In the related literature, all studies that handle problems in a multi-objective manner have reported that the RL outperforms dispatching rules. Since real-life scheduling problems usually include multiple objectives and the RL gets promising results in multi-objective problems, the application of the RL algorithms to these problems should be analyzed deeply.

- The metaheuristics occupy a significant place in machine scheduling problems and obtain superior results, especially in large-scale static problems. Hence, the performance of the RL algorithms can be compared with metaheuristics, and the superior and weak properties of the algorithm compared to metaheuristics can be revealed.

- The studies that analyze the algorithm's effectiveness on benchmark problems have solved small-scale static problems. However, further analysis is needed to investigate the effectiveness of the RL algorithm in large-scale problems.

- The vast majority of the papers have ignored the constraints such as machine breakdown, order cancellation, sequence-dependent setup time, and

precedence constraints. However, these constraints are frequently encountered in real-life scheduling problems. Hence, future research may intensify these issues to make problems more realistic.

- The learning algorithms affect the performance of the RL method considerably, and very few studies compare the performances of the different learning algorithms in scheduling problems. Therefore, it can contribute to literature to analyze the different learning algorithms by comparing each other.

- Uniform parallel machine scheduling problems can be encountered frequently in real-life situations where old and new machines exist simultaneously, but there are not enough studies that handle uniform parallel machine scheduling problems.

- Although hybrid flow shops are increasingly studied in the literature and frequently appear in the industry, there is not enough study considering them. Furthermore, the studies in the related literature generally have handled the problem in a static environment. Thus, developing and applying RL algorithms to this problem considering dynamic events can be promising.

- Job shop scheduling problems generally have been studied considering the makespan and tardiness-related criterion. However, no study has handled the flow time-related criteria. Therefore, this criterion may be studied for future research. In addition, minimizing mean weighted tardiness becomes a critical issue due to different customer priorities. Therefore, this problem can be studied to implement the RL algorithm in real-life scheduling problems.

2.3.4.5 Contribution of This Dissertation to Existing Literature

As stated in Chapter 1, we studied three problem cases in this dissertation. The first studied problem (Problem Case-1) addresses a parallel machine environment. Therefore, Table 2.4 summarizes studies handling parallel machine environments to compare Problem Case-1 with related studies. Also, the characteristics of Problem Case-1 are given in this table.

Table 2.4 Classification of the studies handling parallel machines in the literature review

Author	Year	Learning Algorithm	Problem Type	Obj.	Agent	Benchmark Method	Action Selection	State Definition	System Type
Problem Case-1	**2022**	**Q-learning**	$\boldsymbol{R_m/S_{jk}/C_{max}}$	**SO**	**S**	**Benchmark Problems**	**ε-greedy**	**All States**	**Deterministic**
Chien & Lan	2021	Deep RL	$R_m/S_{jk}/C_{max}$	SO	S	Dispatching Rules	ε-greedy	Neural Network	Stochastic
Paeng et al.	2021	Deep RL	$R_m/S_{jk}/\frac{1}{n}\sum T_i$	SO	S	Dispatching Rules	ε-greedy	Neural Network	Deterministic
Luo et al.	2021	Deep RL	$R_m/S_{jk}/\frac{1}{n}\sum T_i + TC$	MO	S	Dispatching Rules and RL Algorithms	Not Mentioned	Neural Network	Stochastic
Zhou et al.	2020	Q-learning	$R_m//C_{max}$	SO	S	No Benchmark	Not Mentioned	Neural Network	Stochastic
Guo et al.	2020	Q-learning	$Q_m//T_{max} + \sum U_i + \frac{1}{n}\sum W_i$	MO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
Ábrahám et al.	2019	Q-learning	$R_m/prec/C_{max}$	SO	S	Benchmark Problems	Boltzmann Probability	All States	Deterministic
Yuan et al.	2016	Q-learning	$P_m/bkdwn/L_{max}, \sum U_i$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
Yuan et al.	2013	Q-learning	$P_m//L_{max}, \sum U_i$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
Zhang et al.	2012	R-learning	$R_m//\sum w_i T_i$	SO	S	Dispatching Rules	ε-greedy	Gradient Descent Method	Stochastic
Palombarini & Martínez	2012a	Relational RL	$R_m/S_{jk}\ bkdwn/\sum T_i$	SO	S	Benchmark Problems	ε-greedy	Relational Regression Tree	Stochastic
Palombarini & Martínez	2012b	Relational RL	$R_m/S_{jk}\ bkdwn/\sum T_i$	SO	S	No Benchmark	ε-greedy	Relational Regression Tree	Stochastic
Zhang et al.	2011	SARSA	$R_m/S_{jk}\ prec/TC$	SO	S	Benchmark Problems	ε-greedy	Gradient Descent Method	Stochastic
Iwamura et al.	2010	Q-learning	$R_m//\sum F_i + TC$	MO	S	Average Improvement	ε-greedy	All States	Not Mentioned
Palombarini & Martínez	2010	Relational RL	$R_m//\sum T_i$	SO	S	Average Improvement	ε-greedy	Relational Regression Tree	Stochastic
Zhang et al.	2007	Q-learning	$R_m/S_{jk}/\frac{1}{n}\sum w_i T_i$	SO	S	Dispatching Rules	ε-greedy	Gradient Descent Method	Stochastic
Csáji & Monostori	2005	Q-learning	$R_m/bkdwn\ prec/C_{max}$	SO	S	No Benchmark	SA Temperature Procedure	Gradient Descent Method	Stochastic

In Problem Case-1, we applied the Q-learning algorithm to an unrelated parallel machine scheduling problem with the machine and sequence-dependent setup times to minimize makespan. Furthermore, we proposed a method to handle this problem in a deterministic and static manner. Although it seems like the study of Chien and Lan (2021) resembles Problem Case-1, they considered the problem stochastic and dynamic. Also, in their study, sequence-dependent setup times occur in product families, while in Problem Case-1, setup time is machine and sequence-dependent and occurs for each job in the problem. Therefore, it can be concluded that Problem Case-1 is the first study that addresses unrelated parallel machine scheduling problems with the machine and sequence-dependent setup time in a deterministic manner to minimize makespan.

Besides these differences, the state definition of Problem Case-1 is independent of problem type, while the state definition is dependent on problem type in most of the studies in the related literature. Therefore, this proposed method is readily applicable to different machine scheduling problems.

Another significant difference from the related literature can be stated in the design of the reward system. In the literature, predetermined fixed values are generally given as rewards or punishments. In this approach, rewards are determined according to a threshold value, and determining this value becomes a separate decision-making problem. In the proposed method, the agent receives a reward at the rate of improvement it makes in the objective function. Therefore, in this study, the reward system was wholly purged of parameters, and the occurrence of a particular decision-making problem was prevented.

In addition to Problem Case-1, in this dissertation, we studied problems that consider hybrid flow shop scheduling: Problem Case-2 and Problem Case-3. Therefore, Table 2.5 represents the studies that only address the hybrid flow shop scheduling problem to see the differences of Problem Case-2 and Problem Case-3 from the studies in the related literature explicitly. Also, the characteristics of Problem Case-2 and Problem Case-3 are given in Table 2.5.

Table 2.5 Classification of the studies handling hybrid flow shop scheduling in the literature review

Author	Year	Learning Algorithm	Problem Type	Obj.	Agent	Benchmark Method	Action Selection	State Definition	System Type
Problem Case-2	**2022**	**Q-learning**	$\boldsymbol{HF//C_{max}}$	**SO**	**S**	**Benchmark Problems**	**ε-greedy**	**All States**	**Deterministic**
Problem Case-3	**2022**	**Q-learning**	$\boldsymbol{HF//\frac{1}{n}\sum T_i+\frac{1}{n}\sum F_i}$	**MO**	**S**	**Dispatching Rules**	**ε-greedy**	**Aggregation**	**Stochastic**
Fang Guo et al.	2020	Q-learning	$HF//C_{max}$	SO	S	GA	ε-greedy	All States	Deterministic
Reyna et al.	2019b	Q-learning	$HF/S_{jk}\ prec/C_{max}$	SO	M	Benchmark Problems	ε-greedy	All States	Deterministic
Han et al.	2019	Q-learning	$HF//C_{max}$	SO	S	AIS and GA	Boltzmann Probability	All States	Deterministic
Qu et al.	2016a	Q-learning	$HF/S_{jk}\ block/TC+\sum U_i$	MO	M	No Benchmark	ε-greedy	Gradient Descent Method	Stochastic
Qu et al.	2015	Q-learning	$HF/S_{jk}\ block/WIP+TC$	MO	S	Dispatching Rules	ε-greedy	Not Mentioned	Deterministic

In the second phase of this dissertation, in addition to Problem Case-1, we applied the Q-learning algorithm to a hybrid flow shop scheduling problem named Problem Case-2 to minimize makespan. We used the same method which applied to Problem Case-1. The primary purpose of using the same method is to see the application of this method in a different machine environment and to analyze the performance of the proposed method in large-scale instances. Table 2.5 shows that two other studies consider the same problem type, like Problem Case-2. However, the study of Fang Guo et al. (2020) is a conference paper that analyzes the performance of the Q-learning algorithm only in one example that comprises 12 jobs. On the other hand, the study of Han et al. (2019) is only an extended version of the study of Fang Guo et al. (2020). In addition to the example in the previous study, they analyzed algorithm performance in another example comprising 20 jobs. While these studies applied the Q-learning algorithm to only one or two small-scale problems, Problem Case-2 consists of large-scale problems with various machines and jobs.

As concluded from the literature analysis, there is not enough study on hybrid flow shop scheduling problems considering multi-objective with dynamic properties. Thus, in the third phase of the dissertation, we applied the Q-learning algorithm to a stochastic and multi-objective hybrid flow shop scheduling problem named Problem Case-3. The objective of Problem Case-3 is to minimize the mean tardiness and mean flow time simultaneously. We compared the performance of the Q-learning algorithm with dispatching rules. As seen in Table 2.5, only the study of Qu et al. (2016a) handles the problem in a stochastic environment considering the dynamic arrivals of the jobs. However, in this study, the results of the algorithm have not been compared with any other method. For the multi-objective case, while Qu et al. (2016a) aim to minimize the number of tardy jobs and setup costs, Qu et al. (2015) aim to minimize the WIP and the cost of the tardy jobs. Only one (Qu et al. (2015)) of the two studies compared the performance of the algorithm with other methods. In conclusion, Table 2.5 shows that no study handles the hybrid flow shop scheduling problem considering the multi-objective and stochastic nature of the problem. Also, Problem Case-3 differs from other studies by addressing the objective function, which minimizes the mean tardiness and flow time.

CHAPTER 3
RL BASED SOLUTION APPROACHES TO STATIC/DETERMINISTIC PARALLEL MACHINE AND HYBRID FLOW SHOP SCHEDULING PROBLEMS

3.1 Introduction

In this chapter, we handled the application of the Q-learning algorithm to the unrelated parallel machine scheduling problem (UPMSP) with machine and sequence-dependent setup times and the hybrid flow shop scheduling problem (HFSSP) with identical machines. The primary purpose of this phase of the dissertation is to analyze the effectiveness of the Q-learning algorithm for the given problems and to simplify the modeling of the algorithm. A simple and easily applicable state representation was developed to simplify the modeling of the algorithm. In addition, a reward system was designed independently of the parameters. Finally, to assess the effectiveness of the proposed method and compare it with the other methods, we used the instances of benchmark problems from the literature.

3.2 Proposed Method

In this phase of the dissertation, we applied the Q-learning algorithm to the UPMSP with machine and sequence-dependent setup times and the HFSSP with identical machines. The objective of both problems is the minimization of the makespan. The common assumptions of the studied problems are as follows:

- There are n jobs expressed by $j_1, j_2, \dots, j_n$ that have to be processed on m machines.
- All jobs are available at time zero.
- Process times of the jobs are deterministic and known in advance.
- Machines are always available throughout the scheduling period.

- Preemption of the operation of a job is not allowed.

3.2.1 State and Action

The primary purpose of the proposed method is to acquire an order of jobs that will minimize the makespan. For this purpose, the state $(s = 1,2,3,\dots,n)$ is determined as the job order, and the action $(a = 1,2,3,\dots,n)$ is determined as the job number. Therefore, the number of states equals the number of jobs in the problem. In each state, the algorithm decides which job number will be assigned to which order. The environment starts with state 1, and the agent decides which job will be assigned to the first order, and the state turns to state 2, and the agent decides the next job that will be assigned to the second order, and this procedure goes on until all jobs are assigned to an order. Q-value $Q(i,j)$ denotes how advantageous it is to put job j to job order i. When the complete sequence is acquired, the jobs are assigned to the machine that will finish the job the earliest. An example of this procedure for a problem with six jobs is given in Table 3.1.

Table 3.1 An example of the state-action pairs

	j_1	j_2	j_3	j_4	j_5	j_6
1		$Q(1,2)$				
2			$Q(2,3)$			
3						$Q(3,6)$
4	$Q(4,1)$					
5					$Q(5,5)$	
6				$Q(6,4)$		

Table 3.2 shows the state-action pairs. In this table, the state i implies the job order, and the action j implies the job number. At the first state, job j_2 was selected, and the system state turned to *state 2*. At *state 2*, j_3 was selected, and the system state turned to *state 3*. At *state 3*, j_6 was selected, the system state changed to *state 4*, and this procedure continues until *state 6*. Finally, at *state 6*, j_4 was selected,

and a complete job sequence was constructed. The complete job sequence of this example is given in Table 3.2.

Table 3.2 An example of job sequence

Job Order	1	2	3	4	5	6
Job Number	2	3	6	1	5	4

3.2.2 Reward System

The reward system is one of the main components of the RL algorithm and is specially designed according to the problem type. There is no specific reward system that can be applied to all problems. The main idea behind the reward system is to give a reward or punishment to the agent to enable it to learn the consequences of the action in each state. In the proposed method, we would like the agent to learn to minimize the makespan, and this makespan value can be obtained only after all sequences is determined. Therefore, episodic Q-learning was used, where all Q-values are calculated after the schedule is finished, and an episode is completed when a job sequence is constructed. The following equation calculates the reward:

$$reward = \frac{worstC_{max} - newC_{max}}{worstC_{max} - bestC_{max}} \tag{3.1}$$

In Equation 3.1, $worstC_{max}$ denotes the maximum makespan value obtained so far, $bestC_{max}$ denotes the minimum makespan value obtained so far, and $newC_{max}$ represents the obtained makespan value in the current episode.

In the literature, predetermined fixed values are generally given as rewards or punishments. In this approach, rewards are determined according to a threshold value, and determining this value becomes a separate decision-making problem. Because the learning phase of the agent is affected by the reward values, this approach assumes that every action of the agent has the same reward value. However, sometimes the action of the agent can lead to a more remarkable improvement in the objective

function. When a fixed value reward system is applied, this improvement is disregarded. Therefore, in this phase, the reward system was wholly free of the parameters, and the occurrence of a particular decision-making problem was prevented. In the proposed system, the value of the reward is determined according to the obtained best and worst makespan values so far. The agent receives a reward at the rate of improvement it makes in the objective function. As a result, the agent enables choosing more profitable actions.

3.2.3 Action Selection

The trade-off between exploration and exploitation is handled with the decreasing *ε-greedy method.* In this method, an initial ε value is assigned, and the ε value is decreased as time passes to provide the agent to select the actions with the higher Q-value after getting enough information about the environment. In this method, $\varepsilon = 0.90$ is assigned in the beginning to allow the agent to select the most likely random actions (with probability ε). Then the ε value was updated by the following formula at the end of each episode:

$$\varepsilon = \varepsilon * (0.90\text{^}(eps/Teps)) \tag{3.2}$$

In Equation 3.2, eps denotes the current episode number, and $Teps$ denotes the predetermined total episode number. Through this equation, the ε value is decreased slowly and stopped when the ε value is equal to 0.10

3.3 Q-learning Application to Parallel Machine Scheduling Problem with Makespan Criterion

The classical Parallel Machine Scheduling Problem (PMSP) can be defined as a set of m machines and n jobs that have to be processed on one of the m machines. Each job can be processed on only one machine, and each machine can process only one job at the same time. The PMSP aims to comprise a schedule by sequencing given *n* jobs and to assign them to the machines to optimize a predetermined objective function

(Nahmias & Olsen, 2015). The mainly handled objective functions in the literature are the maximum completion time of all jobs, named as makespan, the total tardiness, the number of tardy jobs, and the total flow time (Lee et al., 2013).

During the last years, researchers and practitioners from numerous fields, including Computer Science, Operations Research, and Industrial Engineering, have studied various scheduling aspects extensively and made important contributions to the machine scheduling literature (Pinedo, 2012). The PMSP is one of the most considered problems in machine scheduling literature and still receives continuing interest from researchers. This rising interest can be linked to the wide usage of parallel machines (PM) in production systems (Edis et al., 2013; Ezugwu & Akutsah, 2018). Parallel machines are frequently encountered in industries such as textiles, electronics, machinery, and chemical. Besides the importance of PMSP from a practical point of view, it is also essential from a theoretical point of view because this problem is a special subtype of hybrid flow shop problems and is a generalization of single machine scheduling problems (Pinedo, 2012).

Although PMS problems are among the frequently handled problems in scheduling literature, the research that considered the Unrelated Parallel Machine Scheduling Problem (UPMSP) with the machine and sequence-dependent setup times is still insufficient (Edis et al., 2013; Ezugwu & Akutsah, 2018). Also, no study in the literature applies the RL algorithm to the problem mentioned above. Therefore, in this stage, we aimed to apply the Q-learning algorithm, one of the most popular RL algorithms, to UPMSP with machine and sequence-dependent setup times and to analyze the performance of the proposed algorithm.

3.3.1 The Details of Problem Case-1

In Problem Case-1, we handled static/deterministic UPMSP with machine and sequence-dependent setup times and applied the Q-learning algorithm. The main purpose of the algorithm is to acquire an order of jobs that will minimize the makespan.

The assumptions of Problem Case-1, in addition to assumptions given in Section 3.3, are as follows:

- There are m unrelated parallel machines, and each machine can process only one job at the same time.
- Setup times are both machine and sequence-dependent. It means that the setup time of a job depends on the machine to which it is assigned and the previous job, which is processed on the assigned machine.

3.3.2 Application of the Proposed Method to Problem Case-1

In Section 3.2, the state definition, action, reward system, and the action selection mechanism of the proposed method are explained in detail. Therefore, it is not explained again in this section. Instead, we explain how the proposed method is applied to Problem Case-1. The flowchart of the application of the proposed method to Problem Case-1 is given in Figure 3.1.

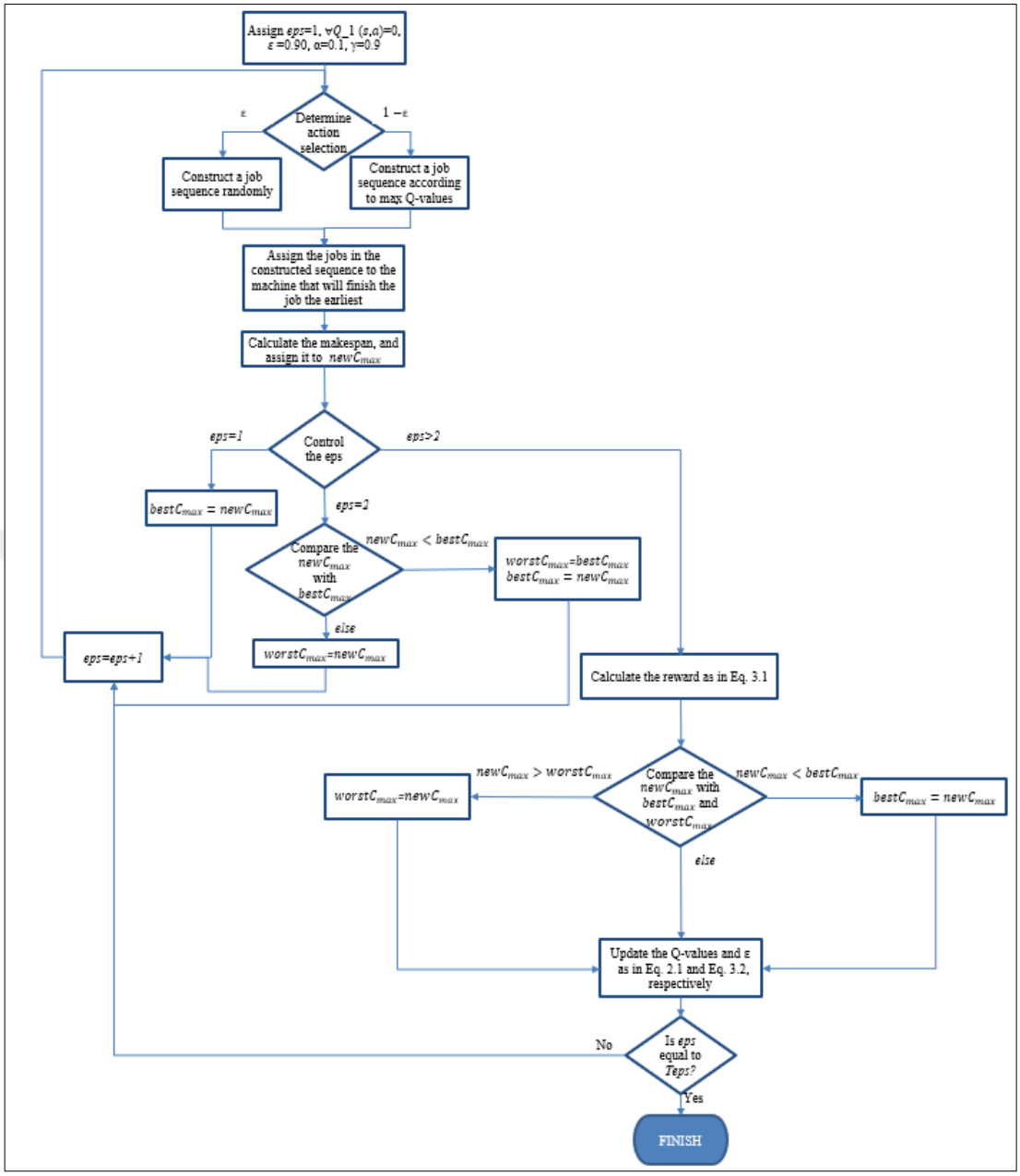

Figure 3.1 The flowchart of the application of the proposed method to Problem Case-1

In the proposed method, first of all, the episode number (*eps*) is assigned to 1, all Q-values are set to 0, and the value of ε is assigned as 0.90. The learning rate, α, is set to 0.1, and the discount factor, γ, is set to 0.9. Then, the action selection method of the current episode is selected. With probability ε, in each state, the action is selected randomly. That means the job number (action) is randomly assigned to the job order (state). After each selection, the selected job number is excluded from the action set to avoid re-selecting the same job. The state-action pairs are marked for updating at the

end of the episode. This selection continues until all the jobs are selected. With probability $1-\varepsilon$, in each state, the action with the maximum Q-value is selected, and the selected job number is excluded from the action set. This selection procedure goes on until all jobs are selected. Like random selection, the state-action pairs are marked for updating at the end of the episode. After the job sequence is constructed, the machine selection process starts. The machine finishing the job earliest is selected for the jobs in the sequence. The sum of the processing time and the setup time is considered in the selection. When all the jobs are assigned to a machine, makespan is calculated and assigned to $newC_{max}$. If the episode number is equal to 1, the $newC_{max}$ value is assigned to $bestC_{max}$; if the episode number is equal to 2, the $newC_{max}$ value is compared with $bestC_{max}$. If $newC_{max}$ is smaller than $bestC_{max}$, $bestC_{max}$ is assigned to $worstC_{max}$, and $newC_{max}$ is assigned to $bestC_{max}$; else, $newC_{max}$ is assigned to $worstC_{max}$. If the episode number exceeds 2, the reward is calculated as in Equation 3.1. After the calculation of the reward, $bestC_{max}$ and $worstC_{max}$ are updated according to $newC_{max}$. Finally, the Q-value update procedure starts. The Q-values of the remarked state-action pairs are updated as in Equation 2.1. At the end of the episode, the ε value is updated as in Equation 3.2, and the episode number is increased by one. This procedure repeats until the episode number reaches to the predetermined total episode number ($Teps$).

3.3.3 Experiments and Results

Problem Case-1 aims to analyze the effectiveness of the Q-learning algorithm applied to the UPMSP with machine and sequence-dependent setup times. The primary purpose of the proposed method is to obtain a suitable permutation of jobs that minimizes the makespan using the Q-learning algorithm in a deterministic environment. The algorithm of the proposed method was coded in Matlab R2018b. First of all, a small-scale problem was modeled step by step. Then, verification of each step was done carefully. After the verification process, the efficiency of the algorithm was analyzed. The full version of the code is given in Appendix A1.

Intending to test the effectiveness of the proposed method and to compare it with other methods, we used small-scale benchmark instances of the study of Vallada & Ruiz (2011). The data set includes the following combinations of the number of jobs and machines: $n = \{6,8,10,12\}$, $m = \{2,3,4,5\}$. Forty different instances exist for each combination, which equals 640 instances.

Vallada & Ruiz (2011) applied GA for this problem and developed two versions of GA symbolized with GA1 and GA2. They also constructed a Mixed Integer Programming (MIP) model and solved the small-scale instances. Finally, they compared the results of the proposed methods with the methods known from the literature: Multiple Insertion (MI) and the GA presented by Kurz & Askin (2001), denoted as GAK, and the heuristic proposed by Rabadi et al. (2006), denoted as Meta, and the calibrated version of the same heuristic denoted as MetaC.

In order to evaluate the performance of the proposed methods, the Relative Percentage Deviation is computed as follows:

$$RPD = \frac{Method_{sol} - Best_{sol}}{Best_{sol}} \times 100 \quad (3.3)$$

In Equation 3.3, $Method_{sol}$ refers to the solution of the proposed method, and $Best_{sol}$ refers to the best-known solution.

We applied the proposed Q-learning algorithm to 640 small instances and recorded the minimum makespan value obtained for each instance. In addition, the average RPD value was computed for each combination. Finally, the results of the proposed method were compared with the results that are given in the study of Vallada and Ruiz (2011). In this study, three different results were given for metaheuristics methods due to three different stopping criteria. However, we used only the best average RPD value for each instance to simplify the comparison. The results of the proposed method (RL) and mentioned methods are given in Table 3.3.

Table 3.3 The average RPD values for the proposed method and the mentioned methods

Instance		MI	MIP	Meta	MetaC	GAK	GA1	GA2	RL
6	2	13.62	**0**	5.07	5.39	1.28	**0**	**0**	**0**
6	3	19.24	**0**	5.91	6.33	0.19	0.06	0.07	**0**
6	4	18.84	**0**	9.74	11.24	0	0.3	0.2	**0**
6	5	15.49	**0**	10.22	10.78	0.36	0.11	0.1	**0**
8	2	14.15	**0**	3.77	4.77	1.58	**0**	**0**	**0**
8	3	18.94	**0**	4.5	5.6	1.23	0.2	0.24	**0**
8	4	19.1	**0**	8.67	9.55	2.65	0.66	0.39	**0**
8	5	22.58	**0**	11.28	13.25	8.78	0.49	0.11	0.33
10	2	16.56	0.4	2.72	3.34	2.61	0.13	**0.07**	0.3
10	3	19.89	0.09	3.86	4.82	2.71	0.26	**0.15**	0.33
10	4	21.17	**0**	5.59	6.97	8.77	0.45	0.26	0.32
10	5	24.38	**0**	9.63	1.53	19.45	1.16	1.03	0.78
12	2	17.26	1.63	1.97	3.22	2.94	0.16	**0.09**	1.78
12	3	23.35	3.19	2.33	3.72	7.77	0.2	**0.08**	0.57
12	4	25.99	2.57	5.82	7.16	19.61	1.29	0.75	**0.41**
12	5	21.35	0.24	6.17	8.44	35.85	1.95	1.49	**0.57**
Average		**19.49**	**0.5**	**6.08**	**6.63**	**7.24**	**0.464**	**0.3**	**0.34**

Table 3.3 shows the experimental results of the proposed algorithm. RPD implies how much the proposed algorithm deviates from the minimum makespan value. The best results for each combination are shown in boldface. The table shows that the proposed algorithm can obtain the minimum makespan value for all instances for 6 and 8 jobs with 2, 3, and 4 machines. Also, the proposed method outperforms other methods in instances with 12 jobs and 4 machines and 12 jobs, and 5 machines. It can be demonstrated that the proposed method can obtain approximately the same average value obtained by GA2.

In the proposed algorithm, the agent initially explores the environment and selects action randomly. As time progresses, it is expected to see that the agent learns the consequences of the actions. As a result of the learning phase, the obtained makespan value in each episode should converge to the optimum makespan value. Therefore, the obtained makespan value at the end of each episode was recorded to analyze whether the agent learns as expected in the algorithm. The simple moving average method was

used to see the trend of the data clearly, and the analysis was done for each instance. The results of the analysis for some of the combinations are given in Figure 3.2.

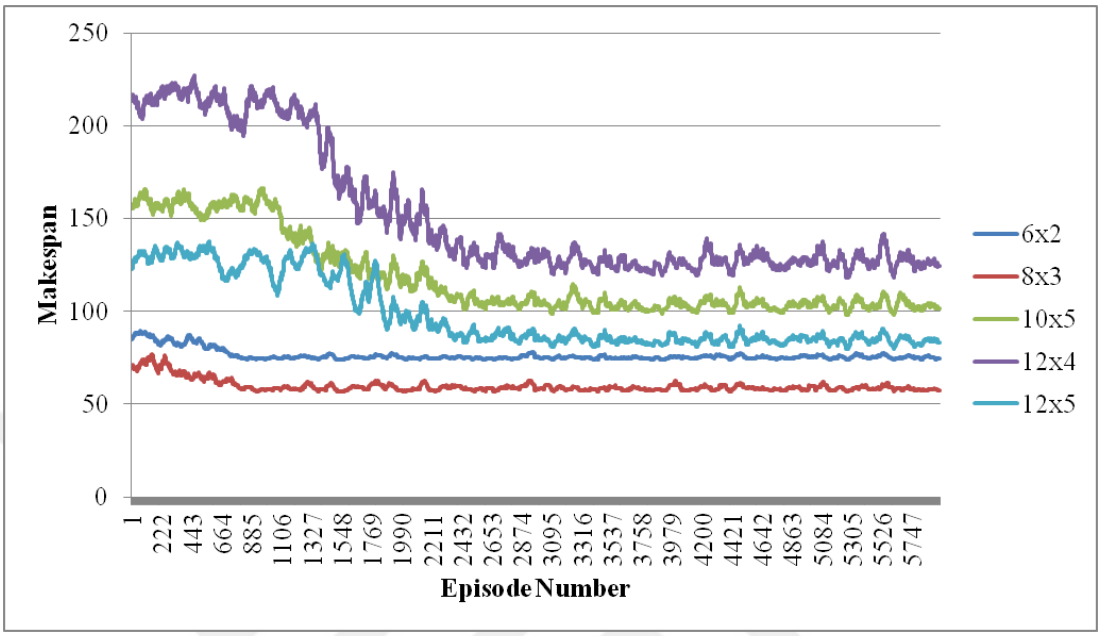

Figure 3.2 Learning phase of the algorithm for Problem Case-1

Figure 3.2 demonstrates the learning phase of the algorithm and shows that the algorithm learns to find the optimal order of the jobs in time. In smaller instances (6x2, 6x3, 6x4, 6x5), the algorithm converges to the optimum makespan value at approximately 800 episodes. For other instances, it takes roughly 3000 episodes to find the minimum makespan value.

3.4 Q-learning Application to Hybrid Flow Shop Scheduling Problem with Makespan Criterion

A hybrid flow shop can be described as a production system consisting of *k* stages, and at least one stage includes parallel machines. In this system, all jobs follow the same stage order. The operation of a job starts with the first stage, goes through the second stage, and goes on, and the job leaves the system when the operation of the job is completed in the final stage. Since the stages contain parallel machines, the HFS scheduling problems involve two decision problems in each stage: which machine the

job will be assigned to and which sequence the jobs will be processed on this machine. It is demonstrated that this problem is NP-hard even for a system that consists of two stages and involves parallel machines only in the first stage (Gupta, 1988).

Hybrid flow shop production systems have numerous advantages, such as balancing the workload of the stages, giving flexibility, and increasing production capacity. Moreover, sometimes, HFS is a requirement for some firms to meet customer demand. For example, in some production systems, the differences between the processing time of the stages are overmuch, and this situation causes bottlenecks. Therefore, duplicating machines in some stages is indispensable to prevent bottlenecks (Brah & Wheeler, 1998; Ribas et al., 2010).

Hybrid flow shop systems widely take parts in the manufacturing systems such as the packaging industry (Adler et al., 1993), automobile assembly plant (Agnetis et al., 1997), label sticker manufacturing (Lin & Liao, 2003), steel production (Li et al., 2011), personal computer manufacturing (Yu et al., 2017) and water-meter manufacturing (Cao et al., 2021). Because of its widespread applications, the HFS problem has attracted researchers, and it has been studied by handling the various aspects of the problem (Ruiz & Vázquez-Rodríguez, 2010).

In this section, we studied a hybrid flow shop scheduling problem considering the application of the Q-learning to minimize the makespan, named Problem Case-2. As stated in Section 2.3.4.5, there are not enough studies handling the application of the RL to hybrid flow shop scheduling problems to minimize makespan. Moreover, the studies in the related literature applied the Q-learning algorithm to only one or two small-scale problems. Therefore, we have analyzed the performance of the proposed method in large-scale problems with the various number of machines and jobs.

3.4.1 The Details of Problem Case-2

In Problem Case-2, we handled the hybrid flow shop scheduling problem aiming to minimize the makespan. The assumptions of Problem Case-2, in addition to assumptions given in Section 3.2, are as follows:

- The system has more than one stage, and at least one involves parallel machines.
- Each job is processed at each stage and follows the same stage order.
- The parallel machines in each stage are identical.
- There is a single buffer in front of each stage, and the capacity of the buffers is unlimited.
- The transportation time between the stages and the setup time is ignored.

3.4.2 Application of the Proposed Method to Problem Case-2

In Section 3.2, the state definition, action, reward system, and the action selection mechanism of the proposed method are explained in detail, and the application of the proposed method to Problem Case-1 is represented in Section 3.3.2. The application of the proposed method to Problem Case-1 and Problem Case-2 resembles substantially. Only the calculation of the makespan differs in these cases. Therefore, in this section, we only introduced the differences between them.

In Problem Case-1, the proposed method is applied to a parallel machine environment, and makespan is calculated at the end of one stage. However, Problem Case-2 comprises more than one stage, and makespan is calculated at the end of the last stage. Therefore, we named this process Procedure 1, and the flowchart of the application of the proposed method to Problem Case-2 is given in Figure 3.3.

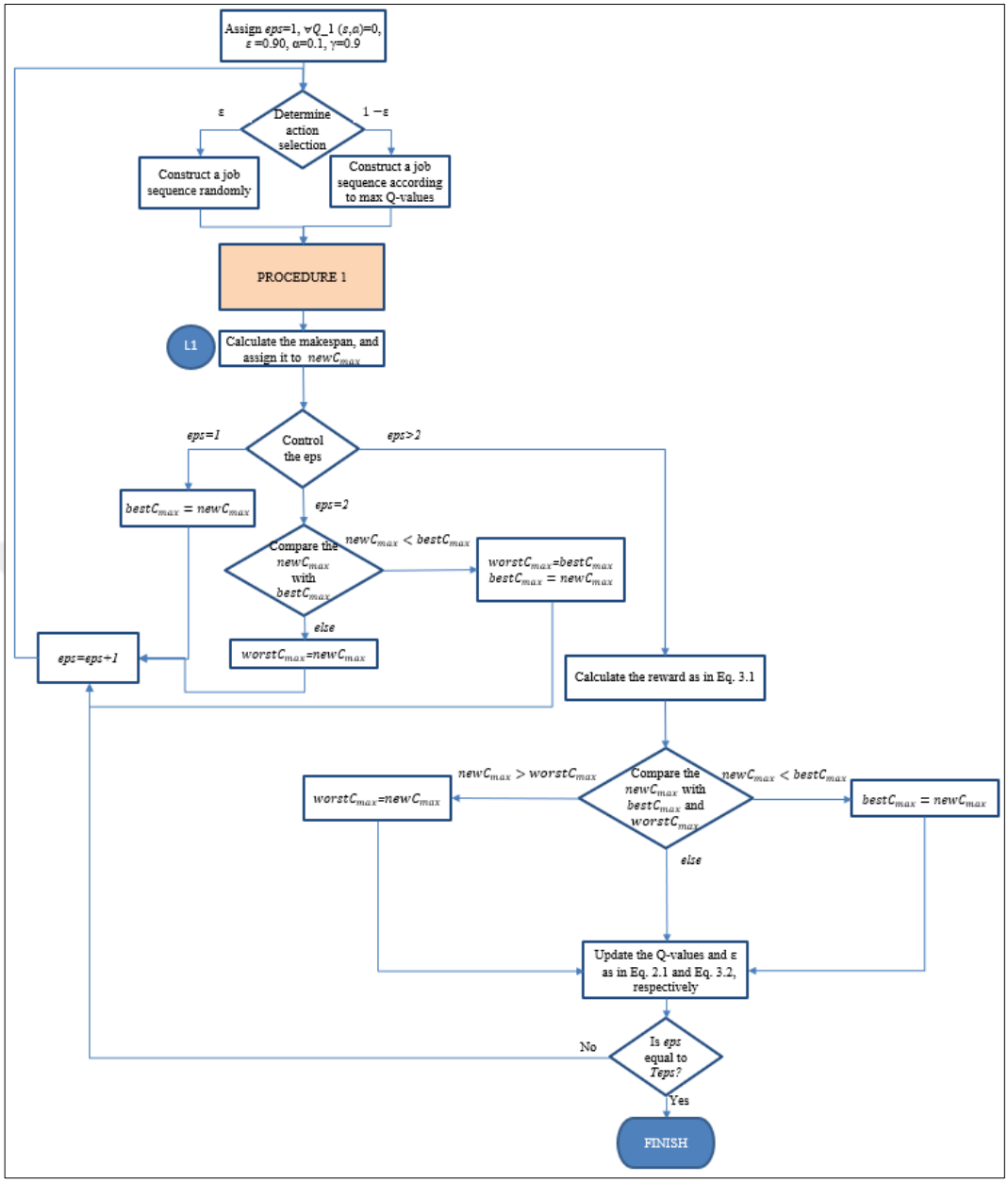

Figure 3.3 The flowchart of the application of the proposed method to Problem Case-2

As seen in Figure 3.3, the application of the proposed method to Problem Case-1 and Problem Case-2 is nearly identical. However, in Problem Case-2, differently in Problem Case-1, there is more than one stage, and all the jobs must be processed in each stage. Therefore, a new procedure (Procedure 1) occurs before the calculation of the makespan. The flowchart of Procedure 1 is given in Figure 3.4.

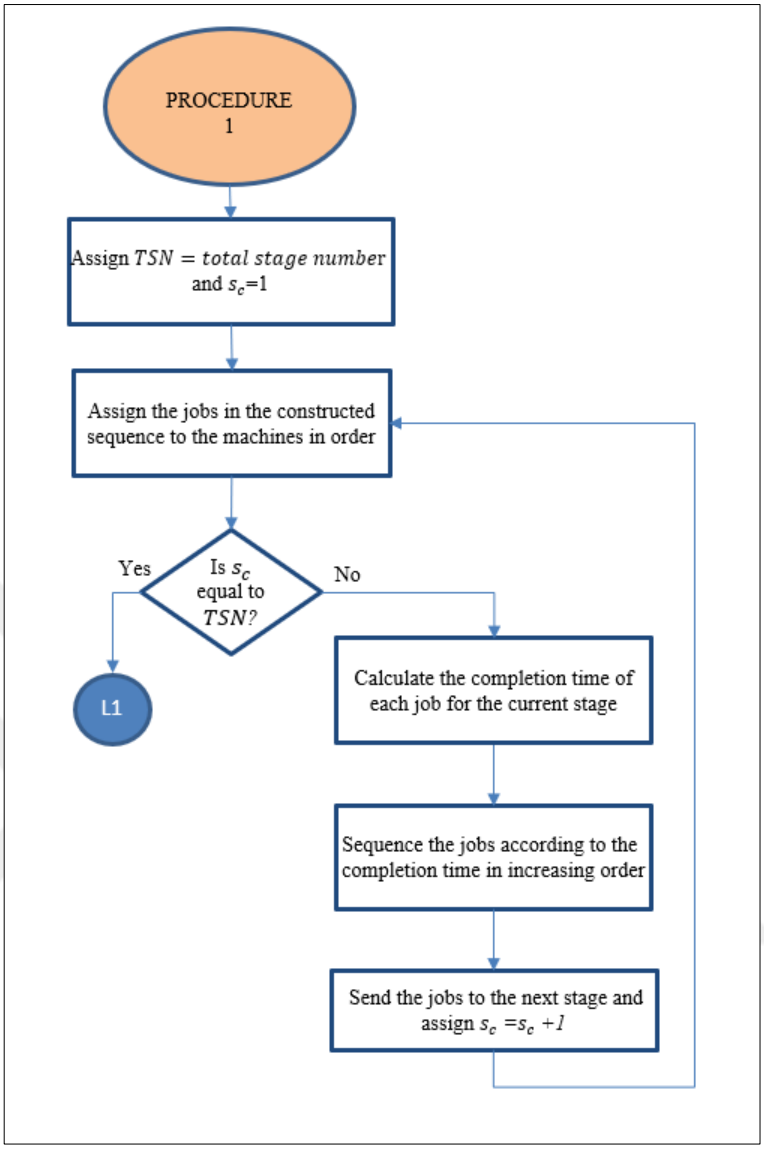

Figure 3.4 The flowchart of Procedure 1

At the beginning of Procedure 1, the total stage number in the problem is assigned to TSN, and the current stage number, s_c, is initialized to 1. Later, the jobs in the constructed sequence are assigned to the machine in order. Before calculating the makespan, the current stage number and the total stage number are controlled. If the current stage number equals the total stage number, the algorithm continues from the step labeled L1 in Figure 3.3. Else, the completion time of each job for the current stage is calculated, and the jobs are sequenced according to completion time in increasing order. Finally, the stage number is increased by one, and machine assignments for the next stage start, and this process continues until the current stage number reaches the total stage number.

3.4.3 Experiments and Results

In Problem Case-2, we applied the Q-learning algorithm to a deterministic hybrid flow shop scheduling problem. The algorithm of the proposed method was coded in Matlab R2018b. First, we developed the algorithm for Problem Case-1, and a new code was written for Procedure 1. Then, we embedded this code into the main algorithm, and verification of the algorithm was done carefully. Also, we coded the algorithm in generic form, which allows the algorithm to adapt to varied sizes of the problems. The full version of the code is given in Appendix A2.

The primary purpose of the proposed algorithm is to obtain a suitable permutation of jobs that minimizes the makespan using the Q-learning algorithm in a deterministic environment. In order to analyze the performance of the proposed method, we used the large-scale instances of the study of Öztop et al. (2019). In this study, they generated new instances with 40, 50, and 60 jobs, 5 stages in each instance, the number of machines in the stages is generated randomly between 3 and 5, and the processing time of the jobs generated using uniform distribution in the range [1, 100]. There are ten instances for each job size.

In order to evaluate the performance of the proposed method, we compared the results of the Q-learning algorithm with the best-known solution, and the RPD value

is calculated as represented in Equation 3.3. Finally, the result of the proposed method is given in Table 3.4.

Table 3.4 The result of the proposed method

Instance number	Best Solution	Q-learning Solution	RPD	Average RPD	Max-Min
40_5_1	688	728	5.81		
40_5_2	767	791	3.13		
40_5_3	801	837	4.49		
40_5_4	734	782	6.54		
40_5_5	713	746	4.63		
40_5_6	781	817	4.61		
40_5_7	735	770	4.76		
40_5_8	802	845	5.36		
40_5_9	840	884	5.24		
40_5_10	775	815	5.16	4.97	3.41
50_5_1	970	1015	4.64		
50_5_2	994	1025	3.12		
50_5_3	1029	1094	6.32		
50_5_4	909	939	3.30		
50_5_5	1026	1064	3.70		
50_5_6	840	901	7.26		
50_5_7	928	951	2.48		
50_5_8	857	879	2.57		
50_5_9	654	710	8.56		
50_5_10	969	990	2.17	4.41	6.40
60_5_1	1136	1163	2.38		
60_5_2	921	1012	9.88		
60_5_3	1158	1183	2.16		
60_5_4	788	850	7.87		
60_5_5	1062	1130	6.40		
60_5_6	1048	1074	2.48		
60_5_7	1225	1242	1.39		
60_5_8	1033	1050	1.65		
60_5_9	1156	1171	1.30		
60_5_10	921	986	7.06	4.26	8.58
Average RPD value			4.55		

In Table 3.4, the first, second, and last digits of the instance number indicate the number of jobs, stages, and problem index, respectively. The best solution indicates the reported result of the instance. Q-learning represents the minimum makespan value obtained at the end of the 10 replications by the Q-learning method. Average RPD implies the average of the RPD values in the same number of jobs, and Min-Max also shows the differences between the minimum and the maximum RPD values in the same number of jobs.

As seen in Table 3.4, the RPD value of the Q-learning algorithm changes between the value of 1.30 and 9.88. These results show that the Q-learning algorithm cannot obtain minimum values, but it approximates these results. When we compare the average RPD values of the instances with the same number of jobs, the proposed method obtains better results in instances with 60 jobs. However, the differences between the minimum and the maximum RPD values are larger, showing more variability in the instance. Another point can be concluded from the table, when the number of jobs in the instance increases, the average RPD value decreases; on the other hand, the variability of the RPD values increases.

In the Q-learning algorithm, the agent selects action randomly in the beginning to explore the environment. After getting enough information about the environment, the agent selects the actions with the higher Q-value. As a result of this process, it is expected that the value of the objective function converges to the optimum value over time. In Problem Case-2, the objective of the proposed method is to minimize the makespan value. Therefore, we expected to see that the agent learns the consequences of the actions, and as a result, this obtained makespan value decreases over time. In order to analyze whether the agent learns as expected in the algorithm, the obtained makespan value at the end of each episode was recorded. The simple moving average method was used to see the trend of the data clearly, and the analysis was done for each instance.

The result of the analysis for some instances with 40, 50, and 60 jobs is given in Figure 3.5.

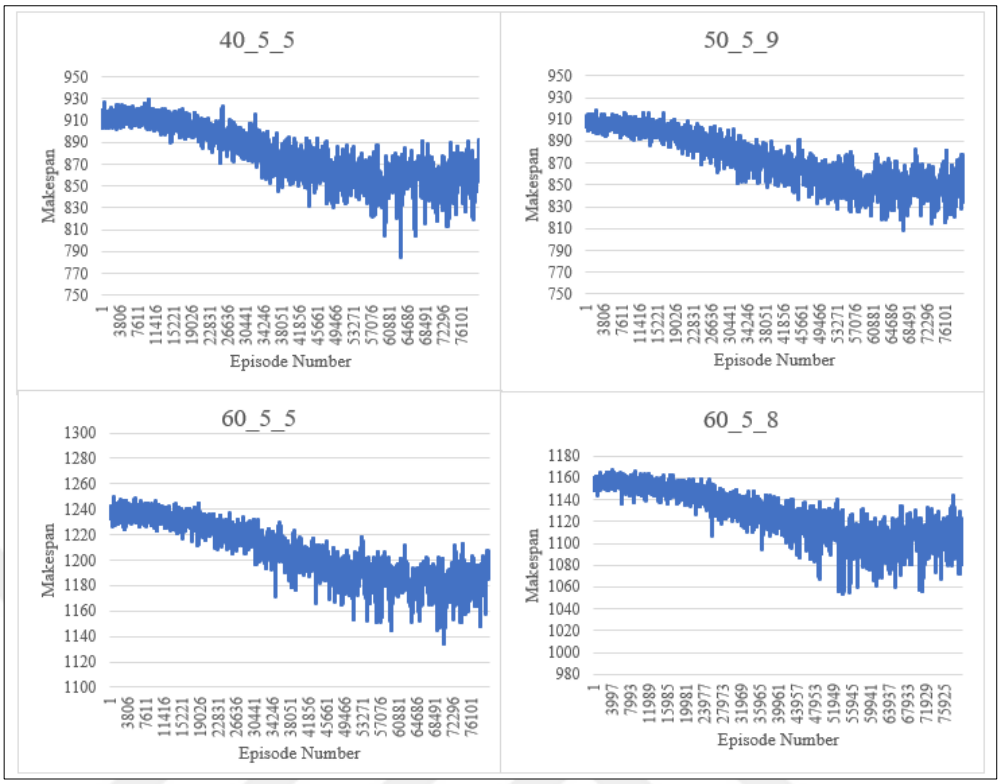

Figure 3.5 Learning phase of the algorithm for Problem Case-2

Figure 3.5 shows the obtained makespan value at the end of each episode in the instances with instance numbers 40_5_5, 50_5_9, 60_5_5, and 60_5_8. In this proposed method, the agent receives a reward at the rate of improvement it makes in the objective function. As seen in this figure, the obtained makespan value decreases over time in the proposed method, demonstrating that the agent learns the consequences of the actions and chooses more profitable actions.

CHAPTER 4
RL BASED SOLUTION APPROACHES TO DYNAMIC/STOCHASTIC HYBRID FLOW SHOP SCHEDULING PROBLEMS WITH MULTI-OBJECTIVES

4.1 Introduction

In addition to research articles, comprehensive literature reviews about the HFSSP were conducted by researchers. Ribas et al. (2010) and Ruiz & Vázquez-Rodríguez (2010) can be shown as state-of-the-art reviews on this problem. Ribas et al. (2010) stated that most of the studies address the problem considering a single objective; therefore, they suggested handling the problem in a multi-objective manner to fill the gap between the theory and real-world application. On the other hand, Ruiz & Vázquez-Rodríguez (2010) expressed that little research had been done considering the dynamic events and added that it could be promising to handle this issue. Also, as a result of the comprehensive review conducted in the scope of this dissertation, it is seen that no study handles the application of the RL to a hybrid flow shop scheduling problem considering the multi-objective and dynamic nature of the problem. Therefore, in this phase of the dissertation, we handle the application of the Q-learning algorithm to a dynamic HFSSP to minimize the mean tardiness and flow time. We used Q-learning in the selection of the proper dispatching rules to meet the predetermined objective.

4.2 General Structure of the Problem

In the standard form of the HFSSP, it is assumed that all jobs available at time zero or arrival times are known in advance, machines are always available, and the processing time of the jobs is deterministic. Therefore, based on these assumptions, most studies deal with this problem in a static environment (Ruiz & Vázquez-Rodríguez, 2010). However, in the real world, scheduling problems involve stochastic events that occur in a dynamic environment, such as dynamic job arrivals, order cancellation, due-date changes, and machine breakdowns (Tang et al., 2005 Ying &

Cheng, 2010; Qin et al., 2018). Since the characteristics of the static and dynamic scheduling problems are diverse, new solution approaches should be developed for dynamic problems (Tang et al., 2005).

In this phase of the dissertation, we handled the stochastic HFSSP with dynamic job arrivals. It is assumed that the time between job arrivals are exponentially ditributed and the arrival time of the jobs is unknown. However, the processing time of the jobs is generated using the uniform distribution and is known in advance. In order to imitate this system over time, the simulation tool was utilized, and the simulation model was developed in Arena 14.0 software to measure the performances of dispatching rules on predetermined objective functions. The following section gives the description of the problem and the flowchart of the model. Later, selected dispatching rules and objective functions are addressed, and the problem size and parameters of the generated models are presented in detail.

4.2.1 Description of the System

The production system consists of more than one stage and identical parallel machines at least in one stage. The following assumptions are considered for the system:

- Each machine can process only one job at a time.
- Jobs arrive at the system dynamically.
- The processing times of the jobs are generated using a uniform distribution.
- The parallel machines in each stage are identical.
- There is a single buffer in front of each stage.

- The selection of the job from the buffer for processing is based on a pre-defined dispatching rule.

This system was modeled in Arena 14.0 software. The simulation model was designed in a generic form that can be easily adapted to the system with a different number of machines and stages. The flowchart of the HFS is given in Figure 4.1, and the developed simulation model is given in Appendix A3.

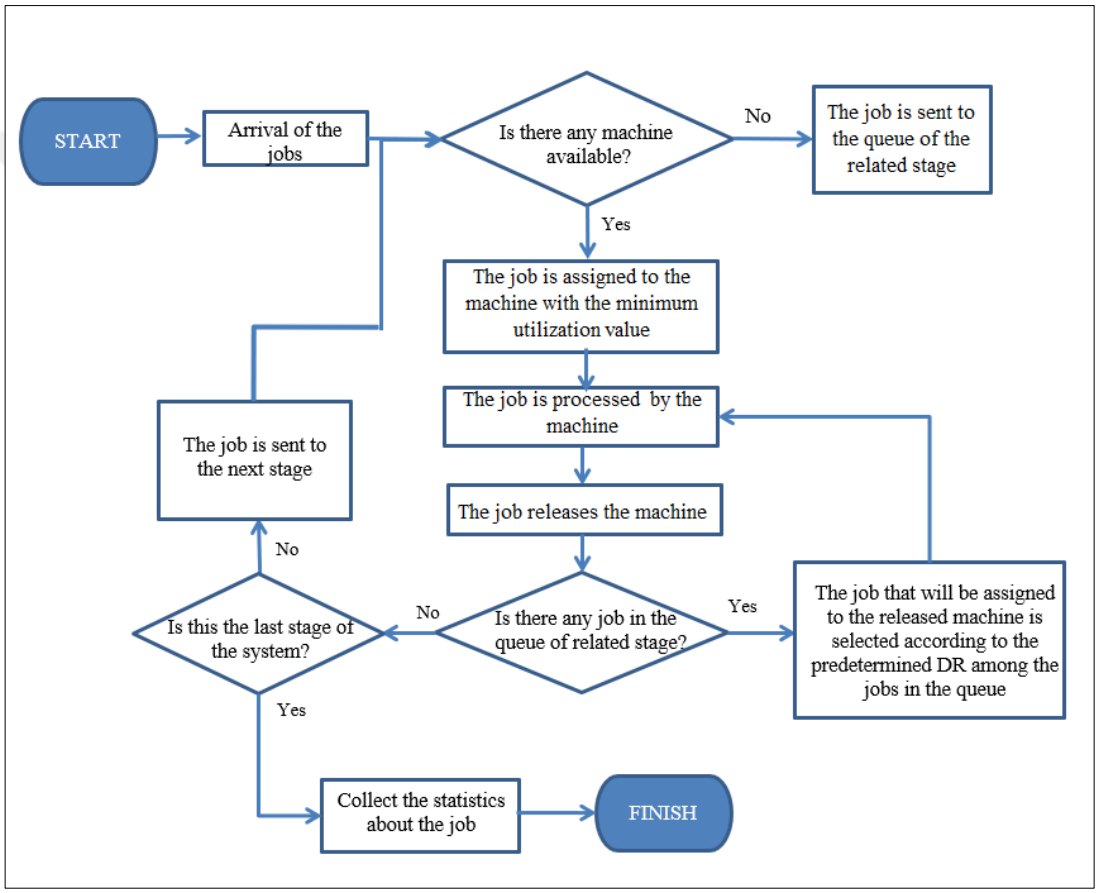

Figure 4.1 The flowchart of the HFS

In Problem Case-3, the job arrives system dynamically, and first, the machine availability is controlled. If all machines are busy, the job is sent to the queue of the related stage. If there is more than one machine available, the job is assigned to the machine with the minimum utilization value to balance the workload of the machine. Then, the job is processed with a predetermined process time. When the process is

completed, the queue of the current stage is controlled. If any job is in the queue, the next job is selected based on predetermined DR among the jobs in the queue. If no job is in the queue, the job is sent to the next stage. If this is the last stage of the system, the statistics about the job are collected, and the job leaves the system.

4.2.2 Selected Dispatching Rules

The dispatching rule is the simplest and easiest tool for machine scheduling problems to have a reasonably good schedule (Pinedo, 2012). It is used to determine the next job that will be processed from a set of jobs waiting in the queue (Korytkowski et al., 2013). It is popular, especially in complex and dynamic problems, due to its ease of applicability (Ruiz & Vázquez-Rodríguez, 2010). Dispatching rules may be categorized in several ways. For example, one may distinguish between *static* and *dynamic* rules. *Static* rules include just the job or machine data independent of the time. Therefore, the priority of the job does not vary by time. However, *dynamic* rules are time dependent. They include information like arrival times, remaining processing times, and remaining time to the due date. The priority value of the job is calculated when a dispatching decision is made since the priority of the job changes over time (Pinedo, 2012).

In this phase of the dissertation, we reviewed the literature and determined 23 static and dynamic dispatching rules. Before representing dispatching rules, we give the notations used in the dispatching rules in Table 4.1.

Table 4.1 The notations in dispatching rules

Notations	Description
t	The time at which the dispatching decision is made
p_j	Processing time of job j at the current stage
tp_j	The total processing time of job j
rp_j	The remaining processing time of job j
d_j	Due date of job j
r_j	Arrival time of job j
s_j	The slack time of job j
W_j	The total work content of jobs in the queue of the subsequent operation of job j
Z_j	Priority value assigned to the job j at the time of the decision of dispatching

Selected dispatching rules, short definitions, and the equations of the rules are as follows:

1. Shortest Processing Time (SPT): The job with the minimum processing time is chosen for loading.

$$Min \qquad Z_j = p_j \tag{4.1}$$

2. First Come First Served (FCFS): The job with the minimum arrival time is chosen for loading.

$$Min \qquad Z_j = r_j \tag{4.2}$$

3. Longest Processing Time (LPT): The job with the maximum processing time is chosen for loading.

$$Max \qquad Z_j = p_j \tag{4.3}$$

4. Most Work Remaining (MWKR): The job with the most remaining processing time is chosen for loading.

$$Max \qquad Z_j = rp_j \tag{4.4}$$

5. Least Work Remaining (LWKR): The job with the least remaining processing time is chosen for loading.

$$Min \qquad Z_j = rp_j \tag{4.5}$$

6. Most Total Work First (MTWF): The job with the maximum total processing time is chosen for loading.

$$Max \qquad Z_j = tp_j \tag{4.6}$$

7. Least Total Work First (LTWF): The job with the least total processing time is chosen for loading.

$$Min \qquad Z_j = tp_j \tag{4.7}$$

8. Earliest Due Date (EDD): The job with the earliest due date is chosen for loading.

$$Min \qquad Z_j = d_j \tag{4.8}$$

9. Processing Time to Remaining Work (PRW): The job with the smallest ratio of the processing time to remaining work is chosen for loading.

$$Min \qquad Z_j = p_j / rp_j \tag{4.9}$$

10. Slack: The job with the least slack value is chosen for loading.

$$Min \qquad Z_j = d_j - rp_j - t \tag{4.10}$$

11. Slack per Remaining Work (S/RW): The job with the smallest ratio of slack time to the remaining work is chosen for loading.

$$Min \qquad Z_j = (d_j - rp_j - t)/rp_j \tag{4.11}$$

12. Critical Ratio (CR): The job with the smallest ratio of the time until the due date to the processing time is chosen for loading.

$$Min \qquad Z_j = (d_j - t)/rp_j \tag{4.12}$$

13. Modified Due Date (MDD): The job with the minimum modified due date is chosen for loading.

$$Min \qquad Z_j = Max\{d_j, t + rp_j\} \tag{4.13}$$

14. High Response Ratio Next (HRN): The job with the maximum ratio of the waiting time plus processing time to processing time is chosen for loading.

$$Max \qquad Z_j = ((t - r_j) - (tp_j - rp_j) + p_j)/p_j \tag{4.14}$$

15. Combination of the slack per remaining work rule and the shortest processing time first rule ((S/RW) +SPT) (Anderson & Nyirenda, 1990): Compute two values for each job at each stage: slack per remaining processing time ratio multiplied with the current processing time for job *j*, and current processing time for job *j*. The maximum of these two values determines a job's priority. The job with the highest priority is selected.

$$Max \qquad Z_j = Max\{\left(\frac{d_j - rp_j - t}{rp_j}\right) * p_j, p_j\} \tag{4.15}$$

16. Combination of the critical ratio rule and the shortest processing time first rule (CR+SPT) (Anderson & Nyirenda, 1990): Compute two values for each job at each stage: critical ratio multiplied by the current processing time for job *j*, and the current processing time for job *j*. The maximum of these two values determines the priority. The job with the highest priority is selected.

$$Max \qquad Z_j = Max\{\left(\frac{d_j - t}{rp_j}\right) * p_j, p_j\} \tag{4.16}$$

17. Arrival time minus remaining processing time (AT-RPT) (Rajendran & Holthaus, 1999): The job with the minimum Z_j is chosen for loading.

$$Min \qquad Z_j = r_j - rp_j \tag{4.17}$$

18. Process Time Plus Work-in-Next Queue (PT+WINQ) (Holthaus & Rajendran, 1997): The job with the minimum Z_j is chosen for loading.

$$Min \qquad Z_j = p_j + W_j \tag{4.18}$$

19. Process Time Plus Work-in-Next Queue Plus Arrival Time (PT+WINQ+AT) (Holthaus & Rajendran, 1997): The job with the minimum Z_j is chosen for loading.

$$Min \qquad Z_j = p_j + W_j + r_j \tag{4.19}$$

20. Process Time Plus Work-in-Next Queue Plus Negative Slack (PT+WINQ+SL) (Holthaus & Rajendran, 1997): The job with the minimum Z_j is chosen for loading.

$$Min \qquad Z_j = p_j + W_j + \min\{slack, 0\} \tag{4.20}$$

21. Process Time Plus Work-in-Next Queue to Time in Shop ((PT + WINQ) /TIS) (Rajendran & Holthaus, 1999): The job with the minimum Z_j is chosen for loading.

$$Min \qquad Z_j = (p_j + W_j)/(t - r_j) \tag{4.21}$$

22. Process Time to Time in Shop (PT/TIS) (Rajendran & Holthaus, 1999): The job with the minimum Z_j is chosen for loading.

$$Min \qquad Z_j = p_j/(t - r_j) \tag{4.22}$$

23. Cost Over Time (COVERT): The *COVERT* rule calculates a penalty function, c_j, depending upon the slack of job j, s_j, and the sum of the expected waiting times for job's uncompleted operations, EWT_j and determines the priority index, Z_j, of the job. The job with the largest Z_j is chosen for loading. The priority index is given as follows:

$$Max \qquad Z_j = c_j/p_j \tag{4.23}$$

$$c_j = \begin{cases} \frac{EWT_j - s_j}{EWT_j} & 0 \le s_j < EWT_j \\ 0 & s_j \ge EWT_j \\ 1 & s_j < 0 \end{cases} \tag{4.24}$$

When we analyze the formulas of dispatching rules, we see that some pairs of dispatching rules will have the same results. One of them is the pair of SPT and PT+WINQ rules. PT+WINQ rule considers the processing time in the current stage and the total work content of jobs in the next operation queue. In Problem Case-3, there is one queue in each stage; therefore, work in the next queue is the same for all waiting jobs in the current queue and will not affect the priority of jobs.

Another pair of dispatching rules with the same results is S/RW and CR. We can show the relationship between S/RW, and CR as follows:

$$S/RW = (d_j - rp_j - t)/rp_j \tag{4.25}$$

$$S/RW = \frac{d_j - t}{rp_j} - \frac{rp_j}{rp_j} \tag{4.26}$$

$$S/RW = CR - 1 \tag{4.27}$$

As seen in Equations 4.25-4.27, the S/RW rule can be converted to the CR rule by adding the same integer value, prioritizing the jobs precisely as the CR rule gives. Therefore, we omitted the PT+WINQ and the S/RW rules and used 21 dispatching rules in all experiments.

4.2.3 Selected Objective Functions

In hybrid flow shop scheduling problems, the mostly handled objective is makespan, namely maximum completion time (Ruiz & Vázquez-Rodríguez, 2010). This objective function can be applied to static problems where all jobs are initially available. However, this study considers dynamic systems where jobs arrive at the system over time. Therefore, we do not take into consideration the makespan. The objective functions we will use to analyze the performance of dispatching rules are given below. Before representing objective functions, we give the notations for objective functions in Table 4.2.

Table 4.2 The notations for objective functions

Notations	Description
C_j	Completion time of job *j*
n	The number of completed jobs
F_j	Flow time of job *j*
T_j	Tardiness of job *j*
E_j	Earliness of job *j*
WT_j	Waiting time of job *j*

Selected objective functions and the equations of the objective functions as below:

1. Flow time related objective functions

- Maximum flow time:

$$max_j(F_j) \tag{4.28}$$

- Mean flow time:

$$\frac{1}{n}\sum_{j=1}^{n} F_j \tag{4.29}$$

2. Tardiness related objective functions

- Maximum tardiness:

$$max_j(T_j) \tag{4.30}$$

- Mean tardiness:

$$\frac{1}{n}\sum_{j=1}^{n} T_j \tag{4.31}$$

- Proportion of tardy jobs:

$$\frac{1}{n}\sum_{j=1}^{n} U_j, \qquad (U_j = \begin{cases} 1, & if\ C_j > d_j \\ 0, & otherwise \end{cases}) \tag{4.32}$$

3. Earliness related objective functions

- Maximum earliness:

$$max_j(E_j) \tag{4.33}$$

- Mean earliness:

$$\frac{1}{n}\sum_{j=1}^{n} E_j \tag{4.34}$$

4. Waiting time related objective functions

- Maximum waiting time:

$$max_j(WT_j) \tag{4.35}$$

- Mean waiting time:

$$\frac{1}{n}\sum_{j=1}^{n} WT_j \tag{4.36}$$

4.2.4 Problem Sizes and Parameters

A hybrid flow shop simulation model was developed in Arena 14.0 software to measure the performances of dispatching rules on predetermined objective functions. The simulation model was designed in a generic form to be easily adapted to various stage and machine settings. The problem size was determined as small, medium, and large considering the number of stages as 2, 5, and 9, respectively. Two problems with a different number of machines selected randomly between 1 and 5 were considered for each problem size. The experiments were carried out for 85% (85U), 90% (90U), and 95% (95U) of machine utilization levels. The processing times of jobs in each stage were assigned according to these utilization levels. The time between job arrivals to the system follows an exponential distribution with a mean of 5 time units. Two different due date tightness was selected: tight due dates (*TDD*) and loose due dates (*LDD*). The total work content (*TWK*) method was used for due date settings in all experiments as given in Equation 4.37.

$$d_j = r_j + k * tp_j \tag{4.37}$$

In Equation 4.37, d_j, r_j, and tp_j indicate the due date, the arrival time, and the total processing time of job j, respectively, and k is the due date allowance factor. The allowance factor for different problems was determined according to the proportion of tardy jobs in each utilization level. For example, the allowance factor for TDD is set so that the proportion of tardy jobs according to the selected dispatching rules varies between 70% and 90% at the 95% utilization level. For LDD, it is set so that the proportion of tardy jobs is between 10% and 30% at the 85% utilization level.

Each simulation experiment consists of 20 replications. The flow times of the jobs were observed to determine when the system reaches a steady state. Simulation run length was fixed as 165000-time units, and warm-up was fixed as 15000-time units for all simulation experiments. The instance set is given in Table 4.3 and the name of the problems and problem parameters are given in Table 4.4.

Table 4.3 The instance set

Problem size	small:{2 stages}, medium:{5 stages}, large:{9 stages}	3
Number of instances	For each size	2
Utilization	85%, 90%, 95%	3
Total work content	tight due dates, loose due dates	2
	Total	36

Table 4.4 Problem parameters

	The number of stages	The number of machines in each stage	Allowance factor	
			Tight	Loose
Problem 1.1	2	{3,5}	1.25	1.5
Problem 1.2	2	{5,3}	1.25	1.5
Problem 2.1	5	{2,3,1,4,2}	1.7	1.9
Problem 2.2	5	{5,2,5,3,1}	1.5	1.7
Problem 3.1	9	{4,3,1,2,5,1,5,3,5}	1.6	1.8
Problem 3.2	9	{3,1,2,2,3,5,3,4,5}	1.6	1.8

The processing time of jobs in each stage was generated using a uniform distribution. The parameters of the distributions were assigned according to utilization levels. The process time settings of Problem 1.1 are given in Table 4.5, and the rest of the process time settings is given in Appendix A4.

Table 4.5 The process time distributions of Problem 1.1

		Stage Number	
		1	2
Utilization Level	%95	UNIF(12,16.5)	UNIF(20,27.5)
	%90	UNIF(10,17)	UNIF(20,25)
	%85	UNIF(11,14)	UNIF(19,23.5)

4.3 Performance Analysis of the Dispatching Rules

This section analyzes the performances of the dispatching rules in varied sizes of hybrid flow shop scheduling problems mentioned in the previous section. First, the individual performance of dispatching rules under nine objective functions is given for the problems with 95U and TDD. Later, the similarity and the performance of dispatching rules are compared between the same size of problems. Finally, the best five dispatching rules for different utilization and due date settings are summarized under each objective function. We used some of the abbreviations of the dispatching rules as follows in the tables:

- PT+WINQ+AT: P+W+A
- PT+WINQ+SL: P+W+S
- (PT + WINQ)/TIS: (P+W)/T
- (S/RW)+SPT: (S/RW)+S

4.3.1 The Performances of the Dispatching Rules According to Objective Functions

The performances of the dispatching rules under pre-determined objective functions are analyzed for three utilization levels (95%, 90%, 85%) and two due date tightness (TDD, LDD) for each problem. In this section, the performances of the dispatching rules for 95% utilization level and tight due date (TDD) tightness of each problem are given in Table 4.6-Table 4.11, and the rest of the results are given in Appendix A5.

In these tables, the best and the worst values of the dispatching rules under pre-determined objective functions are remarked with green and red colors, respectively. Then, the values between them are graded according to the best and worst values. The performances of the dispatching rules for Problem 1.1 are given in Table 4.6.

Table 4.6 The performances of DRs in 95U and TDD for Problem 1.1

Dispatching Rules	Mean Flow Time	Max Flow Time	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting Time	Max Waiting Time	Proportion of Tardy Jobs
SPT	88.923	8676.2	43.329	8614.8	1.9266	13.868	50.907	8627	0.62825
FCFS	103.77	371.9	57.307	334.01	1.0494	13.861	65.762	340.94	0.84648
LWKR	87.923	8607.9	42.354	8543.5	1.9509	13.874	49.907	8556.4	0.6221
LTWF	89.361	9171.8	43.811	9103.2	1.9701	13.878	51.345	9116.9	0.60319
EDD	101.49	380.27	54.983	319.89	1.0078	13.873	63.48	331.06	0.84603
PRW	97.622	9110.1	52.15	9055.8	2.0485	13.873	59.606	9066.7	0.62392
SLACK	103.24	380.7	56.675	321.55	0.9516	13.867	65.228	332.2	0.85742
CR	100.08	436.39	53.531	377.16	0.97095	13.86	62.064	388.39	0.85559
MDD	88.279	8434.5	42.084	8370.3	1.3246	13.868	50.264	8383.1	0.7246
AT-RPT	105.83	372.57	59.423	335.21	1.1124	13.859	67.815	341.88	0.8404
P+W+A	101.79	372.18	55.351	329.44	1.0803	13.866	63.775	336.95	0.84076
P+W+S	101.37	379.92	54.926	320.03	1.0742	13.873	63.356	331.27	0.84072
(P+W)/T	99.194	428.53	52.745	378.33	1.0727	13.86	61.177	387.75	0.84171
PT/TIS	98.496	449.8	52.051	394.2	1.0761	13.859	60.479	404.99	0.84115
LPT	172.28	12112	126.83	12083	2.0804	13.822	134.26	12089	0.63729
MWKR	159.76	11201	114.47	11173	2.2361	13.833	121.74	11179	0.61374
MTWF	155.58	11989	110.59	11963	2.5367	13.84	117.56	11968	0.56831
HRN	99.059	492.25	52.555	426.54	1.0171	13.872	61.042	439.59	0.84672
(S/RW)+S	172.28	12112	126.83	12084	2.0811	13.822	134.26	12089	0.63721
CR+SPT	171.68	12279	126.75	12250	2.5938	13.807	133.66	12256	0.57309
COVERT	89.394	8166.2	43.191	8105.9	1.3165	13.863	51.379	8118	0.72595

Table 4.6 shows that SPT, LWKR, and LTWF rules obtain satisfactory results in the objectives of minimizing mean flow time, mean waiting time, and mean tardiness, and the results of these dispatching rules are close to each other in most of the objectives. Also, the performances of the COVERT and MDD rules are remarkably similar to these rules except for the objective of minimizing the proportion of tardy jobs. Likewise, FCFS, EDD, PT+WINQ+AT, PT+WINQ+SL, and AT-RPT rules have approximately the same performance in most objectives and obtain good results in the

objectives of minimizing maximum flow time, tardiness, and waiting times. Finally, MTWF and CR+SPT rules have superior results in the objectives of minimizing the proportion of tardy jobs and maximum earliness. In addition, they obtain approximately the same results in most of the objectives. The performances of the dispatching rules for Problem 1.2 are given in Table 4.7.

Table 4.7 The performances of DRs in 95U and TDD for Problem 1.2

Dispatching Rules	Mean Flow Time	Max Flow Time	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting Time	Max Waiting Time	Proportion of Tardy Jobs
SPT	85.578	8390.1	40.629	8340.2	2.5581	10.962	47.573	8350.2	0.501
FCFS	90.231	325.79	43.943	278.63	1.2181	10.963	52.226	287.8	0.81176
LWKR	85.475	8136.9	40.575	8082.5	2.6068	10.962	47.47	8093.4	0.49132
LTWF	85.531	8277.7	40.649	8223.3	2.6237	10.962	47.526	8234.2	0.49029
EDD	90.076	327.74	43.813	277.7	1.2435	10.963	52.071	287.53	0.80817
PRW	89.617	7993.4	44.687	7947.6	2.5768	10.963	51.612	7956.8	0.52615
SLACK	90.193	325.78	43.901	277.39	1.2148	10.962	52.187	286.76	0.81183
CR	89.902	355.94	43.612	304.1	1.2166	10.962	51.897	314.44	0.81176
MDD	86.811	8118.6	39.71	8064.5	1.4056	10.962	47.806	8075.3	0.72111
AT-RPT	90.366	325.21	44.116	279.7	1.2562	10.963	52.361	288.62	0.80847
P+W+A	89.994	326.53	43.715	277.51	1.2276	10.962	51.988	287.09	0.81037
P+W+S	89.923	327.03	43.742	277.42	1.3261	10.961	51.917	287.2	0.80435
(P+W)/T	89.532	349.45	43.255	298.49	1.23	10.962	51.526	308.65	0.80959
PT/TIS	89.605	352.76	43.327	301.52	1.2287	10.962	51.599	311.64	0.80984
LPT	98.773	8739.8	53.312	8694.3	2.0461	10.961	60.767	8703.4	0.62982
MWKR	95.391	8246.7	50.259	8206	2.3755	10.962	57.385	8214.1	0.57476
MTWF	95.171	8450.7	50.114	8410	2.4513	10.963	57.165	8418.1	0.55966
HRN	89.83	357.36	43.551	306.29	1.2283	10.962	51.824	316.45	0.8098
(S/RW)+S	98.773	8739.8	53.312	8694.3	2.0461	10.961	60.767	8703.4	0.62982
CR+SPT	97.85	8406.3	53.038	8361.7	2.6951	10.961	59.844	8370.6	0.54281
COVERT	86.054	7832.9	39.939	7783	1.3916	10.963	48.048	7793	0.72893

When we examine Table 4.7, it is clear that the results of Problem 1.2 are very similar to Problem 1.1. For example, like in Problem 1.1, SPT, LWKR, and LTWF rules get superior results in minimizing mean flow time, waiting time, and tardiness. Likewise, FCFS, EDD, PT+WINQ+AT, PT+WINQ+SL, and AT-RPT rules obtain satisfactory results in minimizing maximum flow time, tardiness, and waiting times. However, different from Problem 1.1, LWKR and LTWF rules obtain better results

than MTWF and CR+SPT in minimizing the proportion of tardy jobs. The performances of the dispatching rules for Problem 2.1 are given in Table 4.8.

Table 4.8 The performances of DRs in 95U and TDD for Problem 2.1

Dispatching Rules	Mean Flow Time	Max Flow Time	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting Time	Max Waiting Time	Proportion of Tardy Jobs
SPT	151.04	10111	64.718	10003	10.611	58.89	94.021	10047	0.40339
FCFS	190.65	530.85	96.23	455.71	2.5094	58.147	133.63	483.31	0.86448
LWKR	166.51	13043	80.133	12907	10.549	57.839	109.49	12963	0.37097
LTWF	171.39	14504	85.566	14362	11.107	57.227	114.37	14421	0.3456
EDD	189.07	559.96	94.379	435.73	2.2452	57.84	132.05	484.92	0.86336
PRW	174.02	10659	88.609	10584	11.526	58.816	117	10615	0.44433
SLACK	190.02	554.15	95.204	432	2.1123	57.722	133	479.22	0.8704
CR	184.22	684.66	89.348	557.29	2.0618	57.697	127.2	608.52	0.87643
MDD	168.94	12872	74.777	12736	2.7651	57.681	111.92	12792	0.72795
AT-RPT	191.64	528.68	97.412	458.43	2.7066	58.575	134.62	485.32	0.86655
P+W+A	185.91	529.05	91.699	446.24	2.7174	58.093	128.89	475.79	0.85125
P+W+S	182.85	547.6	89.173	426.22	3.2523	58.341	125.83	474.07	0.84788
(P+W)/T	168.53	590.72	75.016	493.65	3.4228	58.316	111.5	531.18	0.81056
PT/TIS	166.26	621.92	72.845	523.59	3.5229	58.427	109.23	561.21	0.80371
LPT	628.88	23606	539.08	23542	7.1392	56.808	571.86	23568	0.68227
MWKR	285.36	15815	201.36	15756	12.944	58.41	228.33	15780	0.47514
MTWF	249.16	16071	167.54	16020	15.323	58.461	192.13	16041	0.40023
HRN	176.05	672.69	81.969	550.41	2.854	57.989	119.03	599.6	0.83821
(S/RW)+S	622.91	24104	534.99	24041	9.0159	57.462	565.89	24067	0.63095
CR+SPT	568.88	23301	485.06	23240	13.116	57.297	511.85	23265	0.46859
COVERT	156.97	9581.9	63.078	9475	3.0377	58.546	99.955	9519	0.746

According to Table 4.8, it can be concluded that SPT and COVERT rules obtain reliable results in minimizing mean flow time, waiting time, and tardiness. On the other hand, FCFS, PT+WINQ+AT, and AT-RPT obtain reliable results in maximum flow time, tardiness, and waiting time objectives and the results of these dispatching rules are close to each other in most objectives. In addition, the pair of LWKR and LTWF rules have similar performance, and they obtain satisfactory results in minimizing the proportion of tardy jobs. Likewise, MTWF and LWKR rules obtain the same results in most objectives; however, the performance of these dispatching rules is weak in most of the objectives except the minimizing the proportion of tardy jobs. The performances of the dispatching rules for Problem 2.2 are given in Table 4.9.

Table 4.9 The performances of DRs in 95U and TDD for Problem 2.2

Dispatching Rules	Mean Flow Time	Max Flow Time	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting Time	Max Waiting Time	Proportion of Tardy Jobs
SPT	173.22	10119	67.35	9982.3	8.19	58.414	97.18	10028	0.49772
FCFS	217.26	571.77	105.69	495.75	2.4886	58.871	141.22	517.05	0.87381
LWKR	181.29	11965	75.505	11796	8.2744	58.357	105.25	11852	0.4621
LTWF	190.24	13688	84.846	13511	8.6618	57.883	114.2	13570	0.42475
EDD	211.97	603.61	99.873	451.41	1.9617	58.056	135.93	496.67	0.87682
PRW	215.35	10942	109.98	10838	8.6982	59.158	139.31	10873	0.53608
SLACK	215.13	605.94	102.84	456.04	1.7702	57.664	139.09	501.35	0.88974
CR	202.98	730.36	90.655	581.03	1.7313	58.03	126.94	625.43	0.89614
MDD	184.15	11515	72.43	11347	2.3396	58.318	108.11	11403	0.76925
AT-RPT	221.4	575.65	109.82	500.79	2.486	58.796	145.36	521.94	0.88379
P+W+A	209.37	565.02	98.027	484.5	2.7161	58.461	133.33	505.93	0.85753
P+W+S	205.73	596.54	94.456	444.91	2.7819	57.712	129.69	489.9	0.86387
(P+W)/T	191.02	669.39	80.319	556.3	3.3555	58.637	114.98	589.81	0.82087
PT/TIS	188.76	681.52	78.136	564.34	3.4328	58.565	112.72	599.03	0.81549
LPT	928.08	28128	819.73	28056	5.7298	57.26	852.03	28080	0.73226
MWKR	427.29	18067	322.93	18004	9.7064	58.734	351.24	18025	0.5777
MTWF	315.57	17769	215.16	17716	13.659	59.085	239.52	17734	0.4357
HRN	198.23	770.46	86.676	607.81	2.5075	58.424	122.19	661.15	0.85674
(S/RW)+S	889.76	28916	784.44	28844	8.7642	57.358	813.7	28868	0.63856
CR+SPT	802.66	28536	700.55	28468	11.973	58.098	726.6	28490	0.49758
COVERT	178.08	9428.3	66.636	9297.8	2.616	58.581	102.04	9341.3	0.76948

As seen in Table 4.9, the results of Problem 2.1 and Problem 2.2 are almost identical. For instance, while SPT and COVERT rules have better results in minimizing mean flow time, waiting time, and tardiness, LWKR, and LTWF rules obtain satisfactory results in minimizing the proportion of tardy jobs. Likewise, the rest of the dispatching rules have similar performance as in Problem 2.1. The performances of the dispatching rules for Problem 3.1 are given in Table 4.10.

Table 4.10 The performances of DRs in 95U and TDD for Problem 3.1

Dispatching Rules	Mean Flow Time	Max Flow Time	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting Time	Max Waiting Time	Proportion of Tardy Jobs
SPT	272.57	11104	81.226	10871	21.469	110.63	139.56	10959	0.36679
FCFS	335.06	761.66	127.07	588.03	4.8225	109.57	202.05	645.1	0.84529
LWKR	308.9	18214	119.53	17921	23.453	107.85	175.89	18031	0.31229
LTWF	313.56	19186	125.09	18884	24.348	107.05	180.55	18997	0.29486
EDD	332.33	818.98	123.95	552.26	4.4416	107.44	199.32	644.55	0.84446
PRW	308.67	13595	120.71	13424	24.86	112.73	175.65	13488	0.38268
SLACK	333.96	804.6	125.16	544.69	4.0203	107.19	200.95	631.8	0.85328
CR	326.44	1011.5	117.43	752.83	3.8087	107.76	193.43	844.7	0.86067
MDD	311.67	16402	105.38	16113	6.5327	107.03	178.65	16221	0.62652
AT-RPT	336.31	754.01	128.94	603.35	5.4389	111.15	203.3	655.25	0.84496
P+W+A	323.92	756.6	116.68	566.21	5.5774	109.74	190.91	628.85	0.82056
P+W+S	315.09	792.7	109.9	529.38	7.6301	110.16	182.08	620.55	0.80638
(P+W)/T	294.09	847.77	89.046	623.5	7.7707	110.19	161.08	702.73	0.75298
PT/TIS	290.82	862.16	86.134	640.65	8.1248	109.89	157.81	717.36	0.74169
LPT	1453.1	31682	1248.6	31526	8.4063	105.92	1320	31584	0.79122
MWKR	417.85	18738	233.3	18607	28.263	111.85	284.84	18656	0.41318
MTWF	382.65	18944	201.58	18819	31.742	112.66	249.64	18866	0.36044
HRN	309.6	932.72	102.76	674.89	5.9852	109.23	176.58	768.05	0.80013
(S/RW)+S	1439.5	32195	1237.4	32043	10.793	106.71	1306.5	32100	0.75539
CR+SPT	1306.1	30943	1115	30784	21.742	106.87	1173	30844	0.54492
COVERT	287.01	10077	79.616	9848.5	5.4215	110.67	154	9933.9	0.76118

When we look over Table 4.10, like in the previous problems, SPT and COVERT rules obtain superior results in minimizing mean flow time, waiting time, and tardiness. However, different from previous problems, the performance of the LWKR and LTWF decreased in minimizing mean flow time and tardiness. On the other hand, FCFS, PT+WINQ+AT, and AT-RPT obtain reliable results in maximum flow time, tardiness, and waiting time and the results of these dispatching rules are very close to each other in most objectives. Similarly, LTWF and LWKR rules obtain the same results in most objectives. These dispatching rules have satisfactory results in minimizing the proportion of tardy jobs. Finally, the performances of the dispatching rules for Problem 3.2 are given in Table 4.11.

Table 4.11 The performances of DRs in 95U and TDD for Problem 3.2

Dispatching Rules	Mean Flow Time	Max Flow Time	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting Time	Max Waiting Time	Proportion of Tardy Jobs
SPT	284.78	11746	88.698	11498	21.356	114.84	147	11591	0.37953
FCFS	360.06	805.57	143.83	630.61	4.2094	113.38	222.28	688.09	0.86923
LWKR	327.44	18760	130.08	18457	23.088	109.11	189.66	18570	0.33223
LTWF	333.73	20530	137.47	20208	24.183	108.86	195.95	20329	0.30986
EDD	355.38	865.77	138.61	582.1	3.6705	109.17	217.6	682.23	0.86964
PRW	324.02	13867	128.65	13688	25.075	117.9	186.25	13755	0.39306
SLACK	358.14	855.97	140.93	579.11	3.227	109.91	220.37	674.73	0.87939
CR	350.03	1076.2	132.59	793.83	3.0028	111.66	212.26	897.22	0.88811
MDD	331.56	17940	116.9	17639	5.7787	108.19	193.78	17752	0.64678
AT-RPT	362.21	802.72	146.39	647.13	4.6124	116.34	224.44	699.94	0.87054
P+W+A	346.29	798.12	130.89	609.55	5.0432	114.01	208.51	670.55	0.84286
P+W+S	336.25	837.89	122.55	557.32	6.7421	113.32	198.47	654.16	0.83105
(P+W)/T	309.35	872.66	96.497	644.18	7.5792	114.13	171.58	723.56	0.76771
PT/TIS	304.37	884.81	91.989	655.81	8.0514	114.69	166.6	736.75	0.75192
LPT	2552.3	53479	2337.9	53334	6.4681	109.71	2414.3	53388	0.84567
MWKR	468.44	20451	276.15	20311	28.147	117.37	330.67	20363	0.43613
MTWF	423.13	21209	235.26	21088	32.56	117.83	285.36	21134	0.37295
HRN	326.73	992.68	111.66	712.44	5.3695	113.34	188.95	811.88	0.82049
(S/RW)+S	2527.6	53455	2315.7	53310	8.8925	111.12	2389.6	53364	0.81036
CR+SPT	2128.1	49278	1927.6	49120	20.226	112.6	1990.1	49179	0.60136
COVERT	299.61	10459	83.984	10214	4.8028	114.72	161.84	10306	0.7768

Table 4.11 shows that Problem 3.1 and Problem 3.2 have equivalent results concerning the performance of the dispatching rules. For example, while the performance of LTWF and LWKR rules are good in minimizing the proportion of tardy jobs, SPT and COVERT rules obtain superior results in minimizing mean flow time, waiting time, and tardiness.

To sum up, the performance of the dispatching rules in the problems of the same size is approximately identical. However, it can be shown that the performance of the dispatching rules may differ based on the problem size. For instance, while the performance of LTWF and LWKR rules in minimizing mean flow time and tardiness gets worse when the stage number increases, the performance of the MWKR and MTWF rules in minimizing mean flow time and tardiness increases. On the other hand, some dispatching rules have equivalent results in each problem independently of

problem size. For example, while SPT and COVERT rules always obtain satisfactory results in minimizing mean flow time, waiting time, and tardiness, the performances of LTWF and LWKR rules are always adequate in minimizing the proportion of tardy jobs.

4.3.2 The Ranking of The Dispatching Rules According to Objective Functions

This section summarizes the best five dispatching rules under each objective function for all combinations of utilization levels and due date tightness settings for each problem. The results of each problem are given in Table 4.12, Table 4.13, …, and Table 4.17. The best five dispatching rules in the same utilization level are compared between due date tightness settings, and differences are remarked with yellow.

For example, Table 4.12 presents that the best five dispatching rules remain the same for 95% utilization level for each objective function except for minimizing the proportion of tardy jobs. In this objective function, while the MWKR rule is among the best five rules in TDD, the SPT rule is among the best in LDD. For a 90% utilization level, the best five dispatching rules are the same for LDD and TDD. On the other hand, the best five dispatching rules of the objective of minimizing mean flow time, earliness, tardiness, waiting time, and maximum earliness vary between TDD and LDD for a 85% utilization level.

In summary, the best five dispatching rules remain the same for a 95% utilization level to minimize mean flow time, waiting time, earliness, and maximum tardiness in all problems. For minimizing the proportion of tardy jobs, the best five dispatching rules are unchanged in most problems, except for Problem 1.1 and Problem 3.2, for all utilization levels and due date tightness settings. The best five dispatching rules vary mainly in minimizing mean tardiness.

Table 4.12 The best five dispatching rules for Problem 1.1

		Mean Flow Time	Max Flow Time	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting Time	Max Waiting Time	Proportion of Tardy Jobs
95%	TDD	LWKR	FCFS	MDD	EDD	SLACK	CR+SPT	LWKR	EDD	MTWF
		MDD	P+W+A	LWKR	P+W+S	CR	LPT	MDD	P+W+S	CR+SPT
		SPT	AT-RPT	COVERT	SLACK	EDD	(S/RW)+S	SPT	SLACK	LTWF
		LTWF	P+W+S	SPT	P+W+A	HRN	MWKR	LTWF	P+W+A	MWKR
		COVERT	EDD	LTWF	FCFS	FCFS	MTWF	COVERT	FCFS	LWKR
	LDD	LWKR	FCFS	MDD	P+W+S	SLACK	CR+SPT	LWKR	P+W+S	LTWF
		SPT	P+W+A	COVERT	EDD	CR	(S/RW)+S	SPT	EDD	CR+SPT
		MDD	AT-RPT	LWKR	SLACK	EDD	LPT	MDD	SLACK	LWKR
		LTWF	P+W+S	SPT	P+W+A	HRN	MWKR	LTWF	P+W+A	SPT
		COVERT	SLACK	LTWF	FCFS	FCFS	MTWF	COVERT	FCFS	MTWF
90%	TDD	LWKR	FCFS	MDD	SLACK	CR	P+W+A	LWKR	SLACK	LTWF
		SPT	AT-RPT	COVERT	P+W+S	SLACK	EDD	SPT	P+W+S	LWKR
		LTWF	P+W+A	LWKR	EDD	HRN	MDD	LTWF	EDD	SPT
		MDD	SLACK	SPT	P+W+A	EDD	P+W+S	MDD	P+W+A	CR+SPT
		COVERT	P+W+S	EDD	FCFS	P+W+A	PRW	COVERT	FCFS	PRW
	LDD	LWKR	FCFS	MDD	SLACK	CR	P+W+A	LWKR	SLACK	LTWF
		SPT	AT-RPT	COVERT	P+W+S	SLACK	EDD	SPT	P+W+S	LWKR
		LTWF	P+W+A	LWKR	EDD	HRN	MDD	LTWF	EDD	SPT
		MDD	SLACK	SPT	P+W+A	EDD	P+W+S	MDD	P+W+A	CR+SPT
		COVERT	P+W+S	EDD	FCFS	P+W+A	PRW	COVERT	FCFS	PRW
85%	TDD	LWKR	AT-RPT	MDD	SLACK	SLACK	PRW	LWKR	SLACK	CR+SPT
		SPT	FCFS	COVERT	P+W+S	CR	LPT	SPT	P+W+A	LTWF
		MDD	SLACK	LWKR	P+W+A	HRN	(S/RW)+S	MDD	P+W+S	MTWF
		LTWF	P+W+A	SPT	EDD	EDD	MTWF	LTWF	FCFS	LWKR
		COVERT	P+W+S	LTWF	FCFS	(P+W)/T	PT/TIS	COVERT	EDD	SPT
	LDD	LWKR	AT-RPT	MDD	SLACK	CR	(S/RW)+S	LWKR	SLACK	LTWF
		SPT	FCFS	COVERT	P+W+S	SLACK	PRW	SPT	P+W+A	LWKR
		LTWF	SLACK	EDD	EDD	HRN	LPT	LTWF	FCFS	SPT
		MDD	P+W+A	HRN	P+W+A	COVERT	SPT	MDD	P+W+S	PRW
		PRW	P+W+S	PT/TIS	FCFS	PT/TIS	MWKR	PRW	EDD	CR+SPT

Table 4.13 The best five dispatching rules for Problem 1.2

		Mean Flow Time	Max Flow Time	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting Time	Max Waiting Time	Proportion of Tardy Jobs
95%	TDD	LWKR	AT-RPT	MDD	SLACK	SLACK	P+W+S	LWKR	SLACK	LTWF
		LTWF	SLACK	COVERT	P+W+S	CR	CR+SPT	LTWF	P+W+A	LWKR
		SPT	FCFS	LWKR	P+W+A	FCFS	LPT	SPT	P+W+S	SPT
		COVERT	P+W+A	SPT	EDD	P+W+A	(S/RW)+S	MDD	EDD	PRW
		MDD	P+W+S	LTWF	FCFS	HRN	LWKR	COVERT	FCFS	CR+SPT
	LDD	LWKR	AT-RPT	MDD	P+W+S	CR	LPT	LWKR	P+W+A	LTWF
		LTWF	FCFS	COVERT	P+W+A	SLACK	(S/RW)+S	LTWF	SLACK	LWKR
		SPT	P+W+A	(P+W)/T	SLACK	FCFS	SLACK	SPT	P+W+S	SPT
		COVERT	SLACK	PT/TIS	EDD	P+W+A	P+W+A	MDD	FCFS	CR+SPT
		MDD	P+W+S	HRN	FCFS	HRN	PT/TIS	COVERT	EDD	PRW
90%	TDD	LWKR	FCFS	MDD	SLACK	SLACK	LPT	LWKR	SLACK	LTWF
		LTWF	SLACK	COVERT	P+W+S	CR	(S/RW)+S	LTWF	P+W+S	LWKR
		SPT	AT-RPT	LWKR	EDD	FCFS	MWKR	SPT	FCFS	SPT
		COVERT	P+W+A	SPT	FCFS	(P+W)/T	MTWF	MDD	P+W+A	PRW
		PRW	P+W+S	LTWF	P+W+A	PT/TIS	CR+SPT	COVERT	EDD	CR+SPT
	LDD	LWKR	FCFS	MDD	SLACK	CR	LPT	LWKR	SLACK	LWKR
		LTWF	AT-RPT	COVERT	EDD	FCFS	(S/RW)+S	LTWF	FCFS	LTWF
		SPT	P+W+A	PT/TIS	P+W+S	SLACK	MWKR	SPT	P+W+A	SPT
		PRW	SLACK	(P+W)/T	P+W+A	AT-RPT	MTWF	MDD	EDD	CR+SPT
		COVERT	EDD	HRN	FCFS	PT/TIS	CR+SPT	PRW	P+W+S	PRW
85%	TDD	LWKR	AT-RPT	MDD	SLACK	SLACK	HRN	LWKR	SLACK	LTWF
		LTWF	FCFS	COVERT	P+W+A	FCFS	SLACK	LTWF	P+W+A	LWKR
		SPT	P+W+A	(P+W)/T	FCFS	CR	CR	SPT	FCFS	SPT
		COVERT	SLACK	PT/TIS	P+W+S	PT/TIS	P+W+S	MDD	P+W+S	CR+SPT
		MDD	P+W+S	EDD	EDD	(P+W)/T	LPT	COVERT	AT-RPT	PRW
	LDD	LWKR	AT-RPT	MDD	P+W+A	FCFS	HRN	LWKR	P+W+A	LTWF
		LTWF	FCFS	COVERT	SLACK	CR	CR	LTWF	SLACK	LWKR
		SPT	P+W+A	(P+W)/T	FCFS	SLACK	FCFS	SPT	FCFS	SPT
		P+W+S	SLACK	PT/TIS	P+W+S	AT-RPT	SLACK	MDD	P+W+S	CR+SPT
		COVERT	P+W+S	EDD	EDD	PT/TIS	AT-RPT	P+W+S	AT-RPT	PRW

Table 4.14 The best five dispatching rules for Problem 2.1

		Mean Flow Time	Max Flow Time	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting Time	Max Waiting Time	Proportion of Tardy Jobs
95%	TDD	SPT	AT-RPT	COVERT	P+W+S	CR	LPT	SPT	P+W+S	LTWF
		COVERT	P+W+A	SPT	SLACK	SLACK	LTWF	COVERT	P+W+A	LWKR
		PT/TIS	FCFS	PT/TIS	EDD	EDD	CR+SPT	PT/TIS	SLACK	MTWF
		LWKR	P+W+S	MDD	P+W+A	FCFS	(S/RW)+S	LWKR	FCFS	SPT
		(P+W)/T	SLACK	(P+W)/T	FCFS	AT-RPT	MDD	(P+W)/T	EDD	PRW
	LDD	SPT	AT-RPT	COVERT	P+W+S	CR	LPT	SPT	P+W+A	LTWF
		COVERT	P+W+A	SPT	SLACK	SLACK	LTWF	COVERT	P+W+S	LWKR
		PT/TIS	FCFS	PT/TIS	EDD	EDD	MDD	PT/TIS	SLACK	SPT
		LWKR	P+W+S	(P+W)/T	P+W+A	FCFS	CR	LWKR	FCFS	MTWF
		(P+W)/T	SLACK	MDD	FCFS	MDD	EDD	(P+W)/T	AT-RPT	PRW
90%	TDD	SPT	AT-RPT	COVERT	P+W+S	CR	LPT	SPT	P+W+S	LTWF
		COVERT	FCFS	PT/TIS	SLACK	SLACK	(S/RW)+S	COVERT	P+W+A	LWKR
		PT/TIS	P+W+A	(P+W)/T	EDD	EDD	CR+SPT	PT/TIS	SLACK	SPT
		LWKR	P+W+S	SPT	P+W+A	COVERT	LWKR	LWKR	FCFS	MTWF
		(P+W)/T	SLACK	MDD	FCFS	FCFS	MDD	(P+W)/T	EDD	PRW
	LDD	SPT	AT-RPT	COVERT	P+W+S	CR	LPT	SPT	P+W+A	LTWF
		PT/TIS	FCFS	PT/TIS	SLACK	SLACK	(S/RW)+S	PT/TIS	P+W+S	LWKR
		LWKR	P+W+A	(P+W)/T	EDD	EDD	LWKR	LWKR	SLACK	SPT
		(P+W)/T	P+W+S	MDD	P+W+A	COVERT	CR+SPT	(P+W)/T	FCFS	PRW
		COVERT	SLACK	HRN	FCFS	FCFS	MDD	COVERT	AT-RPT	MTWF
85%	TDD	SPT	AT-RPT	COVERT	P+W+S	CR	CR+SPT	SPT	P+W+A	LTWF
		LWKR	FCFS	PT/TIS	SLACK	SLACK	LPT	LWKR	P+W+S	LWKR
		PT/TIS	P+W+A	MDD	EDD	COVERT	LTWF	PT/TIS	SLACK	SPT
		P+W+S	P+W+S	(P+W)/T	P+W+A	EDD	(S/RW)+S	P+W+S	FCFS	PRW
		(P+W)/T	SLACK	P+W+S	FCFS	FCFS	COVERT	(P+W)/T	EDD	MTWF
	LDD	SPT	AT-RPT	COVERT	P+W+S	CR	CR+SPT	SPT	P+W+A	LTWF
		P+W+S	FCFS	PT/TIS	SLACK	COVERT	(S/RW)+S	P+W+S	P+W+S	LWKR
		LWKR	P+W+A	MDD	EDD	SLACK	LTWF	LWKR	FCFS	SPT
		PT/TIS	P+W+S	(P+W)/T	P+W+A	FCFS	EDD	PT/TIS	SLACK	PRW
		(P+W)/T	SLACK	HRN	CR	AT-RPT	MDD	(P+W)/T	AT-RPT	MDD

Table 4.15 The best five dispatching rules for Problem 2.2

		Mean Flow Time	Max Flow Time	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting Time	Max Waiting Time	Proportion of Tardy Jobs
95%	TDD	SPT	P+W+A	COVERT	P+W+S	CR	LPT	SPT	P+W+S	LTWF
		COVERT	FCFS	SPT	EDD	SLACK	(S/RW)+S	COVERT	EDD	MTWF
		LWKR	AT-RPT	MDD	SLACK	EDD	SLACK	LWKR	SLACK	LWKR
		MDD	P+W+S	LWKR	P+W+A	MDD	P+W+S	MDD	P+W+A	CR+SPT
		PT/TIS	EDD	PT/TIS	FCFS	AT-RPT	LTWF	PT/TIS	FCFS	SPT
	LDD	SPT	P+W+A	COVERT	P+W+S	CR	(S/RW)+S	SPT	P+W+S	LTWF
		LWKR	FCFS	SPT	EDD	SLACK	LPT	LWKR	EDD	LWKR
		COVERT	AT-RPT	MDD	SLACK	EDD	EDD	COVERT	SLACK	SPT
		MDD	P+W+S	PT/TIS	P+W+A	MDD	SLACK	MDD	P+W+A	MTWF
		PT/TIS	EDD	(P+W)/T	FCFS	AT-RPT	P+W+S	PT/TIS	FCFS	CR+SPT
90%	TDD	SPT	P+W+A	COVERT	P+W+S	CR	LPT	SPT	P+W+S	LTWF
		LWKR	FCFS	MDD	EDD	SLACK	(S/RW)+S	LWKR	EDD	LWKR
		COVERT	AT-RPT	SPT	SLACK	EDD	CR+SPT	COVERT	SLACK	SPT
		MDD	P+W+S	PT/TIS	P+W+A	COVERT	LWKR	MDD	P+W+A	MTWF
		PT/TIS	SLACK	(P+W)/T	FCFS	MDD	SLACK	PT/TIS	FCFS	CR+SPT
	LDD	SPT	P+W+A	COVERT	P+W+S	CR	LPT	SPT	P+W+S	LTWF
		LWKR	FCFS	MDD	EDD	SLACK	CR+SPT	LWKR	EDD	LWKR
		PT/TIS	AT-RPT	PT/TIS	SLACK	COVERT	SLACK	PT/TIS	SLACK	SPT
		(P+W)/T	P+W+S	(P+W)/T	P+W+A	EDD	LWKR	(P+W)/T	P+W+A	PRW
		MDD	SLACK	P+W+S	CR	MDD	(S/RW)+S	MDD	FCFS	MTWF
85%	TDD	SPT	P+W+A	COVERT	P+W+S	CR	P+W+A	SPT	P+W+S	LTWF
		LWKR	FCFS	MDD	EDD	SLACK	LPT	LWKR	EDD	LWKR
		PT/TIS	AT-RPT	P+W+S	SLACK	COVERT	PRW	PT/TIS	SLACK	SPT
		MDD	P+W+S	PT/TIS	CR	EDD	(S/RW)+S	MDD	P+W+A	MTWF
		P+W+S	SLACK	SPT	P+W+A	MDD	LWKR	P+W+S	FCFS	PRW
	LDD	SPT	P+W+A	COVERT	P+W+S	CR	LWKR	SPT	P+W+S	LWKR
		LWKR	FCFS	EDD	EDD	COVERT	SLACK	LWKR	EDD	LTWF
		P+W+S	AT-RPT	MDD	SLACK	SLACK	P+W+A	P+W+S	SLACK	SPT
		PT/TIS	P+W+S	P+W+S	CR	AT-RPT	PRW	PT/TIS	P+W+A	MDD
		(P+W)/T	(P+W)/T	CR	P+W+A	EDD	COVERT	(P+W)/T	FCFS	PRW

Table 4.16 The best five dispatching rules for Problem 3.1

		Mean Flow Time	Max Flow Time	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting Time	Max Waiting Time	Proportion of Tardy Jobs
95%	TDD	SPT	AT-RPT	COVERT	P+W+S	CR	LPT	SPT	P+W+S	LTWF
		COVERT	P+W+A	SPT	SLACK	SLACK	(S/RW)+S	COVERT	P+W+A	LWKR
		PT/TIS	FCFS	PT/TIS	EDD	EDD	CR+SPT	PT/TIS	SLACK	MTWF
		(P+W)/T	P+W+S	(P+W)/T	P+W+A	FCFS	MDD	(P+W)/T	EDD	SPT
		PRW	SLACK	HRN	FCFS	COVERT	LTWF	PRW	FCFS	PRW
	LDD	SPT	AT-RPT	COVERT	P+W+S	CR	LTWF	SPT	P+W+S	LTWF
		PT/TIS	P+W+A	PT/TIS	SLACK	SLACK	MDD	PT/TIS	P+W+A	LWKR
		COVERT	FCFS	(P+W)/T	EDD	EDD	EDD	COVERT	SLACK	SPT
		(P+W)/T	P+W+S	SPT	P+W+A	FCFS	LPT	(P+W)/T	FCFS	PRW
		PRW	SLACK	HRN	FCFS	COVERT	SLACK	PRW	EDD	MTWF
90%	TDD	SPT	AT-RPT	PT/TIS	P+W+S	CR	EDD	SPT	P+W+S	LTWF
		PT/TIS	FCFS	(P+W)/T	SLACK	SLACK	MDD	PT/TIS	P+W+A	LWKR
		(P+W)/T	P+W+A	COVERT	EDD	COVERT	LTWF	(P+W)/T	SLACK	SPT
		P+W+S	P+W+S	P+W+S	P+W+A	EDD	LWKR	P+W+S	FCFS	PRW
		PRW	SLACK	HRN	FCFS	FCFS	LPT	PRW	EDD	MTWF
	LDD	SPT	AT-RPT	PT/TIS	P+W+S	CR	LWKR	SPT	P+W+S	SPT
		P+W+S	FCFS	(P+W)/T	SLACK	COVERT	LTWF	P+W+S	P+W+A	LTWF
		PT/TIS	P+W+A	COVERT	EDD	SLACK	EDD	PT/TIS	SLACK	LWKR
		(P+W)/T	P+W+S	P+W+S	P+W+A	EDD	MDD	(P+W)/T	FCFS	PRW
		PRW	SLACK	HRN	(P+W)/T	LPT	CR	PRW	AT-RPT	MDD
85%	TDD	SPT	AT-RPT	PT/TIS	P+W+S	COVERT	LPT	SPT	P+W+S	LTWF
		P+W+S	FCFS	(P+W)/T	SLACK	HRN	(S/RW)+S	P+W+S	SLACK	LWKR
		PT/TIS	P+W+A	CR	EDD	LPT	CR+SPT	PT/TIS	P+W+A	SPT
		(P+W)/T	P+W+S	P+W+S	P+W+A	(S/RW)+S	HRN	(P+W)/T	FCFS	MDD
		PRW	SLACK	MDD	CR	CR+SPT	MWKR	PRW	EDD	PRW
	LDD	SPT	AT-RPT	PT/TIS	P+W+S	COVERT	LPT	SPT	P+W+S	MDD
		P+W+S	FCFS	(P+W)/T	SLACK	LPT	(S/RW)+S	P+W+S	P+W+A	SPT
		PT/TIS	P+W+A	CR	EDD	(S/RW)+S	CR+SPT	PT/TIS	SLACK	PT/TIS
		(P+W)/T	(P+W)/T	SLACK	CR	HRN	HRN	(P+W)/T	FCFS	LWKR
		PRW	P+W+S	MDD	P+W+A	CR+SPT	COVERT	PRW	AT-RPT	(P+W)/T

Table 4.17 The best five dispatching rules for Problem 3.2

		Mean Flow Time	Max Flow Time	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting Time	Max Waiting Time	Proportion of Tardy Jobs
95%	**TDD**	SPT	P+W+A	COVERT	P+W+S	CR	MDD	SPT	P+W+S	LTWF
		COVERT	AT-RPT	SPT	SLACK	SLACK	LTWF	COVERT	P+W+A	LWKR
		PT/TIS	FCFS	PT/TIS	EDD	EDD	LWKR	PT/TIS	SLACK	MTWF
		(P+W)/T	P+W+S	(P+W)/T	P+W+A	FCFS	EDD	(P+W)/T	EDD	SPT
		PRW	SLACK	HRN	FCFS	AT-RPT	LPT	PRW	FCFS	PRW
	LDD	SPT	P+W+A	COVERT	P+W+S	CR	MDD	SPT	P+W+S	LTWF
		PT/TIS	AT-RPT	PT/TIS	SLACK	SLACK	LTWF	PT/TIS	P+W+A	LWKR
		COVERT	FCFS	SPT	EDD	EDD	EDD	COVERT	SLACK	SPT
		(P+W)/T	P+W+S	(P+W)/T	P+W+A	FCFS	SLACK	(P+W)/T	FCFS	PRW
		PRW	SLACK	HRN	FCFS	AT-RPT	LWKR	PRW	EDD	MTWF
90%	**TDD**	SPT	AT-RPT	PT/TIS	P+W+S	CR	(S/RW)+S	SPT	P+W+S	LTWF
		PT/TIS	P+W+A	COVERT	SLACK	SLACK	LPT	PT/TIS	SLACK	LWKR
		(P+W)/T	FCFS	(P+W)/T	EDD	COVERT	CR+SPT	(P+W)/T	P+W+A	SPT
		P+W+S	P+W+S	P+W+S	P+W+A	EDD	MDD	P+W+S	EDD	PRW
		PRW	(P+W)/T	SPT	(P+W)/T	LPT	SLACK	PRW	FCFS	MTWF
	LDD	SPT	AT-RPT	PT/TIS	P+W+S	LPT	(S/RW)+S	SPT	P+W+S	SPT
		P+W+S	P+W+A	(P+W)/T	SLACK	CR	LPT	P+W+S	P+W+A	LTWF
		PT/TIS	FCFS	COVERT	EDD	COVERT	MDD	PT/TIS	SLACK	LWKR
		(P+W)/T	P+W+S	P+W+S	P+W+A	SLACK	LWKR	(P+W)/T	FCFS	PRW
		PRW	(P+W)/T	HRN	(P+W)/T	EDD	SLACK	PRW	EDD	MDD
85%	**TDD**	SPT	AT-RPT	COVERT	P+W+S	CR	(S/RW)+S	SPT	P+W+S	SPT
		P+W+S	FCFS	PT/TIS	SLACK	COVERT	LPT	P+W+S	SLACK	LTWF
		PT/TIS	P+W+A	(P+W)/T	EDD	LPT	CR+SPT	PT/TIS	P+W+A	LWKR
		(P+W)/T	P+W+S	P+W+S	CR	SLACK	LWKR	(P+W)/T	FCFS	MDD
		PRW	(P+W)/T	HRN	P+W+A	EDD	MDD	PRW	EDD	PRW
	LDD	SPT	AT-RPT	COVERT	P+W+S	LPT	(S/RW)+S	SPT	P+W+S	MDD
		P+W+S	FCFS	PT/TIS	SLACK	COVERT	LPT	P+W+S	P+W+A	SPT
		PT/TIS	P+W+A	(P+W)/T	EDD	CR	HRN	PT/TIS	SLACK	LWKR
		(P+W)/T	(P+W)/T	HRN	CR	(S/RW)+S	LWKR	(P+W)/T	FCFS	LTWF
		PRW	P+W+S	CR	HRN	AT-RPT	MDD	PRW	EDD	PT/TIS

4.4 Application of the Q-learning Algorithm

This section analyzes the performance of the Q-learning algorithm in stochastic and multi-objective hybrid flow shop scheduling problems. In section 4.3, we analyzed the performances of the dispatching rules under various objective functions and determined the best five dispatching rules for each objective function. Based on these studies, we examined whether the agent learns to select the proper dispatching rule according to the objective function, namely minimizing the mean tardiness and flow time.

The flow time is the time that the job spents in the system. Minimizing the flow time leads to a decrease in the work-in-process inventory, increased capacity utilization, and rapid turnaround of the job (Framinan & Leisten, 2003). In the aspect of the service systems, minimizing flow time decreases the time spent by the customer in the system and optimizes customer service. Therefore, this objective is critical for firms with high inventory holding costs or service systems where customers leave the system because of long waiting times (Baker & Trietsch, 2013). On the other hand, tardiness is another critical optimality criterion that directly affects the customer. Tardiness calculates how much the job completion time exceeds the due date; namely it shows how much a firm complies with due dates. Therefore, minimizing tardiness is essential for firms to satisfy customer requests in time (Baker & Trietsch, 2013; Aqil & Allali, 2020).

Considering these issues, in Problem Case-3, minimizing the mean tardiness and flow time under different system conditions is aimed. For this purpose, a Q-learning method was applied, and the agent was expected to learn how to achieve this goal. The environment where the learning actualize was provided by simulation. The selected state representation of the environment, the actions and reward function are given following section in detail.

4.4.1 Determination of States, Actions and Reward Function

In the Q-learning algorithm, the agent perceives the current situation of the system (the system state) from the environment and takes action. State definition is crucial since many problems have continuous states or state-space is exceptionally large. In such circumstances, it is hard to calculate and store the value function for each state-action pair; therefore, the *Aggregation* strategy can be applied. In this strategy, states are lumped together to reduce the state space. One or two features generally define states, and the ranges based on these features are determined in advance to categorize states. The categorization of system states can be constructed concerning several measures such as the number of jobs in the buffer, the number of tardy jobs in the buffer, or the lateness or average slack time of those jobs. Also, the system state should be compatible with the objective of the problem. Therefore, in Problem Case-3, we selected the mean slack time (MST) of the queue as the system state because it gives information about both flow time and tardiness of the jobs. The formula of MST is given in Equation 4.38.

$$MST = \frac{1}{n}\sum_{j=1}^{n} d_j - rp_j - t \tag{4.38}$$

We used the aggregation method and determined ten intervals for MST. The intervals of the states were determined through trial-and-error experiments by balancing the number of visits to each state. The state intervals for 95% utilization and TDD setting of Problem 1.1 is given in Table 4.18, and the rest of the intervals are given in Appendix A6.

Table 4.18 The state intervals for 95% utilization and TDD setting of Problem 1.1

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ -0.32	1
NQ≥2 and -7.99 ≤ MST<-0.32	2
NQ≥2 and -20.32 ≤ MST<-7.99	3
NQ≥2 and -43.81 ≤ MST<-20.32	4
NQ≥2 and -84.75 ≤ MST<-43.81	5
NQ≥2 and -155.85 ≤ MST<-84.75	6
NQ≥2 and -286.96 ≤ MST<-155.85	7
NQ≥2 and -502.86 ≤ MST<-286.96	8
NQ≥2 and -882.9 ≤ MST<-502.86	9
NQ≥2 and MST<-882.9	10

In Table 4.18, NQ implies the number of jobs in the queue. If the number of jobs in the queue is less than two, there will be no selection decision, so this situation is determined as a dummy state. The detailed information about the dummy state is given in the following section.

After the determination of the system state, the agent takes action. In this study, the agent decides which dispatching rule will be used to select the job from the queue because selected dispatching rules directly affect the objective function. The best five dispatching rules for each objective function for all system conditions were obtained in the first part of this phase. In this part, the best dispatching rule for minimizing mean flow time and tardiness was determined as the action set of the Q-learning agent. An example of the action set for 95U and TDD setting of Problem 1.1 is given in Table 4.19.

Table 4.19 The action set for 95% and TDD setting of Problem 1.1

Objective	**Action**
Minimize mean flow time	Execute LWKR
Minimize mean tardiness	Execute MDD

The reward system is one of the main components of the RL algorithm and is specially designed according to the problem type. There is no specific reward system that can be applied to all problems. The main idea behind the reward system is to give a reward or punishment to the agent to enable it to learn the consequences of the action in each state. After executing the action, the agent receives a scalar reward r . In Problem Case-3, simultaneous minimization of mean flow time and mean tardiness is considered. Thus, two separate reward functions were developed for each action. For example, if the agent selects to execute LWKR, the agent gets a reward related to minimizing mean flow time, and if the agent selects to execute MDD, then the agent gets a reward related to minimizing mean tardiness. The reward function is given in Equation 4.39.

$$r = \begin{cases} \frac{worstF-newF}{worstF-bestF}, & if\ the\ objective\ is\ minimize\ mean\ flow\ time \\ \frac{worstT-newT}{worstT-bestT}, & if\ the\ objective\ is\ minimize\ mean\ tardiness \end{cases} \quad (4.39)$$

In this Equation, *worstF*, *newF*, *bestF*, *worstT*, *newT*, and *bestT* best indicates the maximum flow time value obtained so far, the obtained flow time value in the current stage, the minimum flow time value obtained so far, the maximum tardiness value obtained so far, the obtained tardiness value in the current stage, the minimum tardiness value obtained so far, respectively.

4.4.2 Simulation Model of the System

In order to analyze the performance of the selected dispatching rules and the proposed method, a hybrid flow shop production system with a Q-learning method was modeled in Arena 14.0 software. The system consists of various stages and identical parallel machines in each stage. It has been found that the system reaches a steady state after 15000 minutes, and simulation time was fixed as 165000 minutes. The next job from the buffer for processing was based on a predefined dispatching rule before 15000 minutes of simulation time. Finally, after 15000 minutes of simulation, the Q-learning method was applied to select the job. ARENA model of the developed simulation model is given in Appendix A7.

The main idea behind the proposed method is as follows: When a job is assigned to a machine, the agent observes the current state and takes action. First, the current state number and the selected action are recorded. Then, after processing the job, the agent observes the environment again and determines the next state. At the same time, the reward value for the selected action is calculated, and finally, the Q-value of the current state and selected action is updated. The detailed flow of the proposed method is given in Figure 4.2, Figure 4.3, and Figure 4.4. In these figures, some steps in the flow are labeled with L1, L2, L3, L4, and L5 in dark blue circles. If there is a label just after a step, the procedure goes to the step with that label.

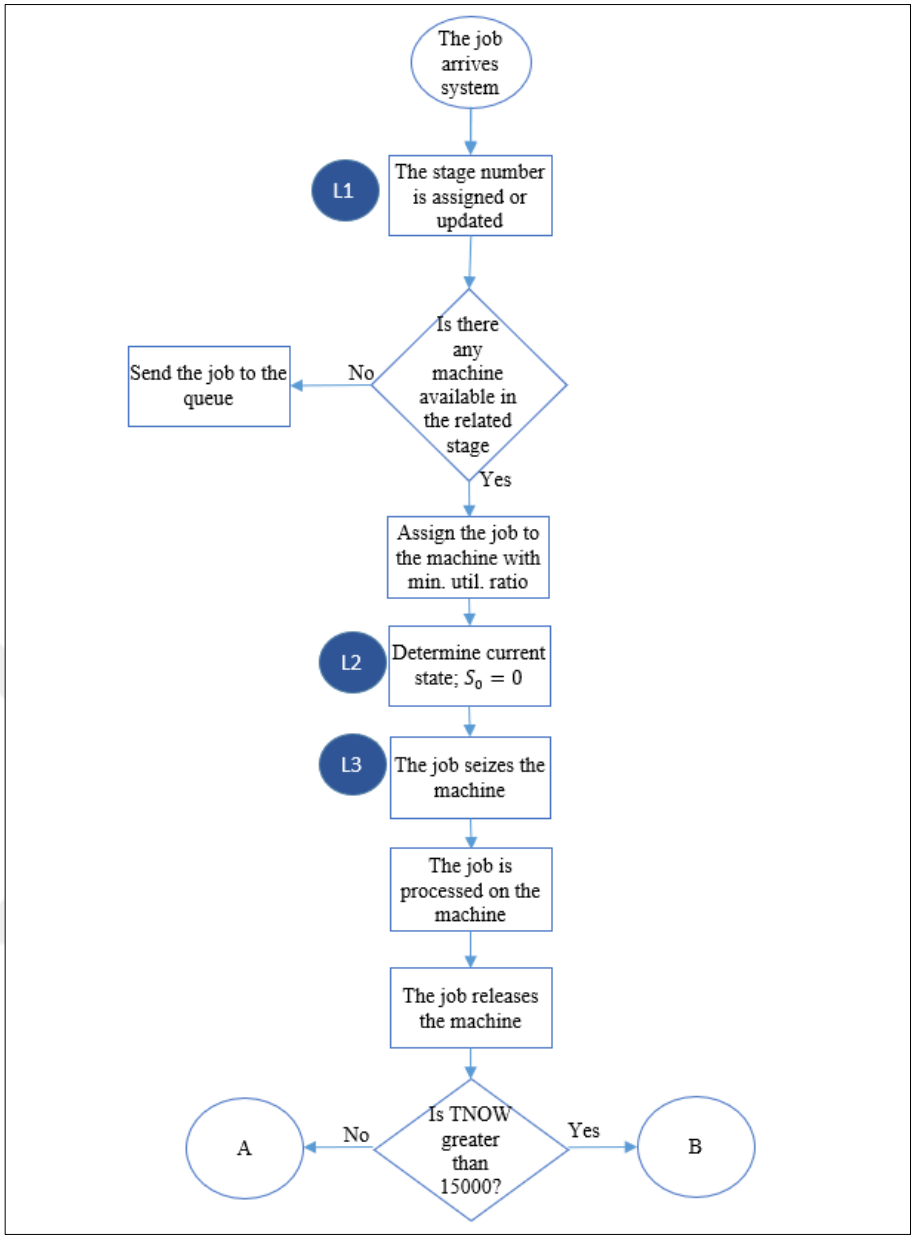

Figure 4.2 The flow chart of the main procedure

As seen in Figure 4.2, the jobs arrive at the system dynamically, and the stage number is assigned to the jobs. First of all, the queue of the related stage is controlled. If there is a job in the queue or all machines are busy, the job is sent to the queue of the related stage. If any machine is available, the job is sent to the machine with a minimum utilization ratio, and the current state is assigned as a dummy state. A dummy state indicates that the job is not selected using Q-learning, so the Q-value will not be updated. After these assignments, the job seizes the machine, is delayed by

process time, releases the machine, and controls the simulation time of the system. If the simulation time is greater than 15000 minutes, which shows the warm up period, it goes to procedure B, else it goes to procedure A. The flow chart of procedure A is given in Figure 4.3 and the flow chart of procedure B is given in Figure 4.4.

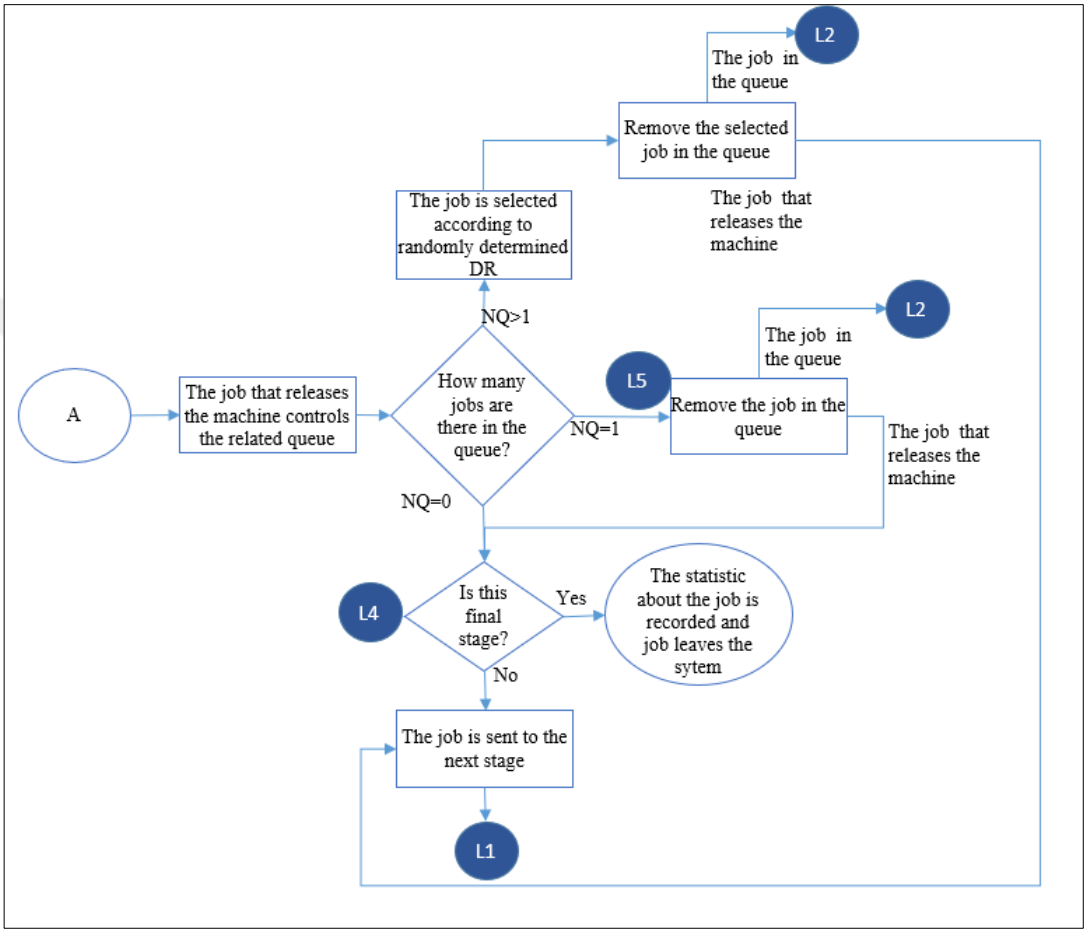

Figure 4.3 The flow chart of procedure A

In procedure A, the job that releases the machine controls how many jobs there are in the queue at the current stage. If there is no job in the related queue, the stage number of the job is controlled. If this is the final stage of the job, the statistic about the job is recorded, and the job leaves the system; otherwise, the job is sent to the next stage and goes to the step labeled with L1. If there is only one job in the queue, the job that releases the machine sends the job in the queue to the step labeled with L2 and goes to the next stage. If there is more than one job in the queue, the next job is selected according to predetermined DR.

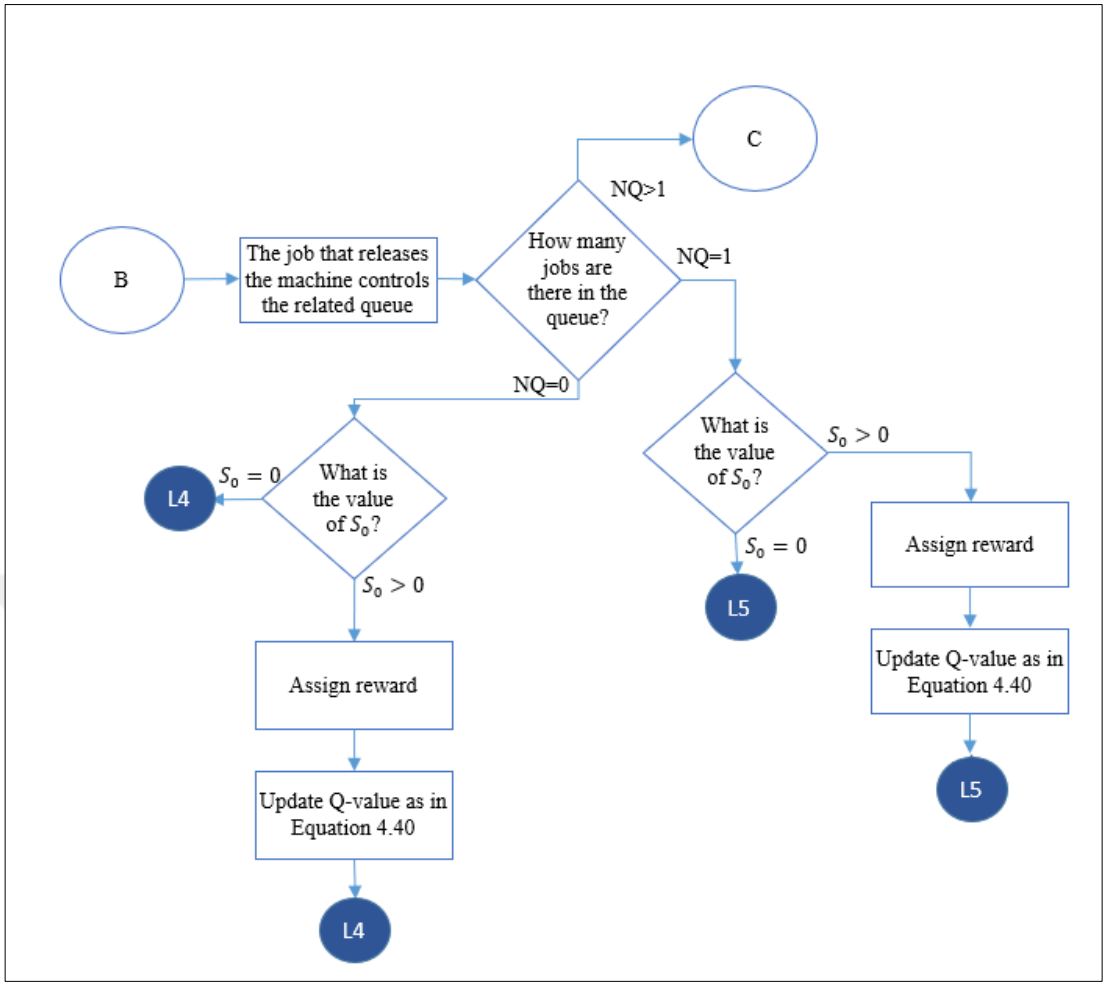

Figure 4.4 The flow chart of procedure B

In procedure B, the job that releases the machine controls how many jobs there are in the queue at the current stage. If there is no job in the related queue, the state value of the system at the time when the job is selected, namely the current state, is controlled. If the current state is a dummy state, the job goes to the step labeled with L4; else, the reward is calculated, and the Q-value of the state action pair is updated as in Equation 4.40, then the job is sent to the step labeled with L4.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r - Q(s,a)] \tag{4.40}$$

If there is only one job in the queue, the state value of the system at the time when the job is selected, namely the current state, is controlled. If the current state is a dummy state, the job goes to the step labeled with L5; else, the reward is calculated,

and the Q-value of the state action pair is updated as in Equation 4.40, then the job is sent to the step labeled with L5. If there is more than one job in the queue, the job goes to procedure C. The flow chart of procedure C is given in Figure 4.5.

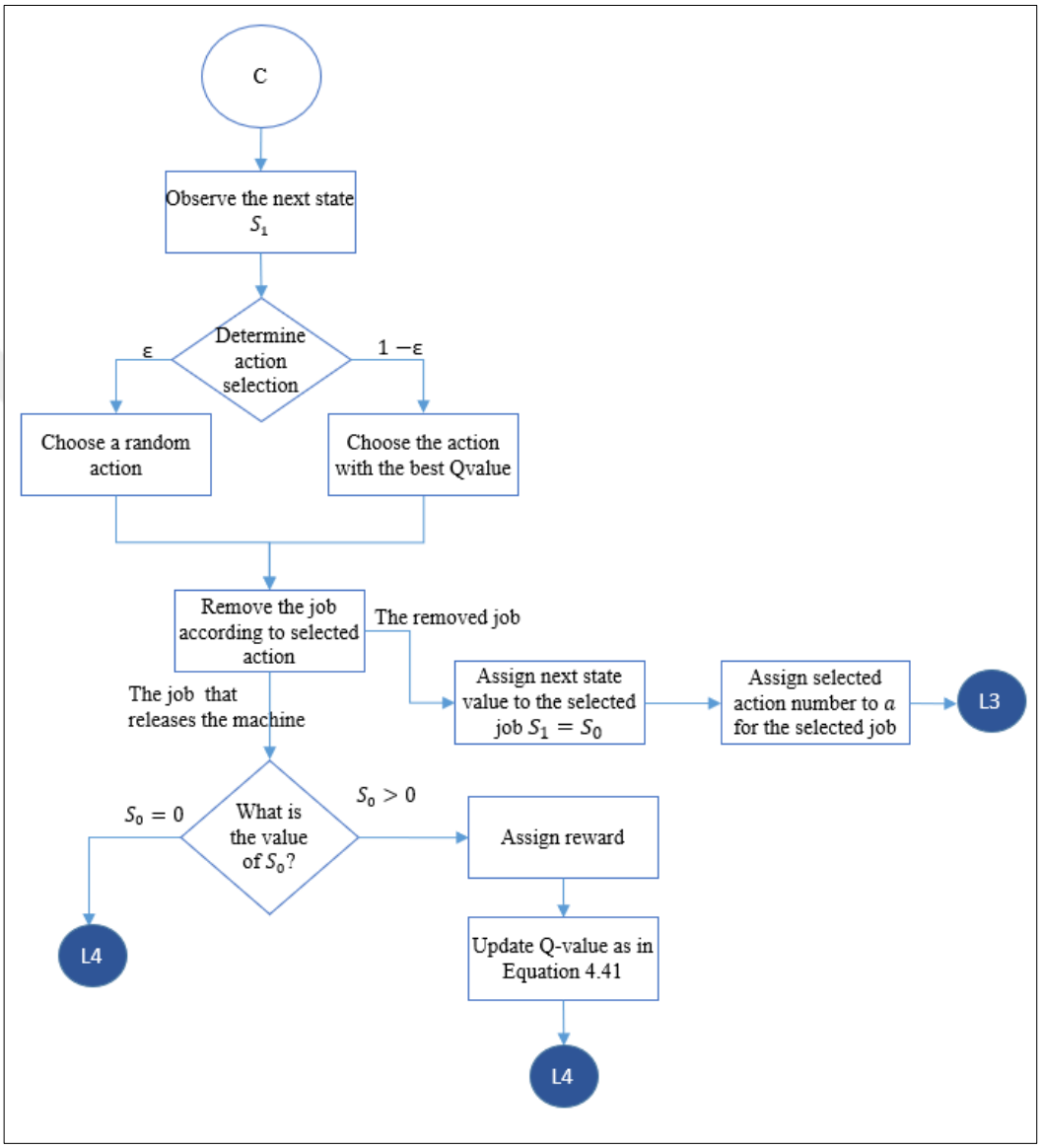

Figure 4.5 The flow chart of procedure C

In procedure C, the first step is observing the next state and determining action selection. Then, with probability ε, the agent selects a random action; with probability 1-ε, the agent chooses the action with the best Q-value in the current state. After the next job from the queue is removed according to the selected action, the next state

value and selected action number are assigned to the selected job. The removed job is sent to the step labeled with L3. Finally, the state value of the system is controlled. If the state value of the system at the time when the job is selected, namely the current state, is a dummy state, the job goes to the step labeled with L4; else, the reward of the job is calculated, and the Q-value of the state action pair is updated as in Equation 4.41, then the job is sent to the step labeled with L4.

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma max_{a'} Q(s',a') - Q(s,a)\right] \quad (4.41)$$

4.4.3 Experiments and Results

In this phase of the dissertation, we applied the Q-learning algorithm to HFSS problems involving a varied number of machines and stages and analyzed the performance of the proposed method in different utilization and due date settings. The details of the problems are given in Section 4.2.4.

In addition to 21 dispatching rules and the proposed method, we analyzed the performance of a dispatching rule named *Random*. While, in the Q-learning method, the agent decides which dispatching rule will be selected among the predetermined dispatching rules, in the *Random* rule, dispatching rule selection among the predetermined dispatching rules is executed randomly. This rule is included in the experiments to analyze the effect of the learning.

In Problem Case-3, minimizing the mean flow time and tardiness is aimed. Therefore, the mean flow time and tardiness value of the dispatching rules, Random, and the Q-learning method are recorded. However, the ranges of these objective functions differ from each other significantly. Therefore, in order to compare these values, a normalization procedure is conducted. Normalization ensures that the values of the objective functions are in the same scale. The normalization of the values is conducted based on the following equation:

$$Y_i = \frac{x_i - \frac{x_{max} - x_{min}}{2}}{\frac{x_{max} + x_{min}}{2}}, i = 1,2, \dots, n \quad (4.42)$$

where;

Y_i= The normalized value of the i^{th} alternative,

n= Total number of alternatives,

x_i= The value of i^{th} alternative,

x_{max}= maximum x_i, $i = 1,2, \dots, n$,

x_{min}= minimum x_i, , $i = 1,2, \dots, n$.

After normalizing the value of the objective functions, the next step is calculating a utility value to evaluate the performance of the selected alternatives. In the literature, there are different methods to calculate utility functions for multi-objective problems. Simple Additive Weighting (SAW) method is one of the most often applied methods because of its ease of application (Afshari et al., 2010). The main idea behind the SAW method is to find a utility value based on the sum of the performance of alternatives considering the weights of the objectives. A utility value is calculated for each alternative by multiplying the normalized values of the alternatives with the weights of the objectives. The formula of the SAW method is given in Equation 4.43

$$U_i = \sum_{j=1}^{m} w_j \times Y_{ji} \quad (4.43)$$

where;

U_i= The utiliy value of the i^{th} alternative, $i = 1,2, \dots, n$

m= The number of objectives,

w_j= The weight of j^{th} objective, $\sum_{j=1}^{m} w_j = 1$, $(w_j = w1, w2, \dots, wm)$

Y_{ji}= The normalized value of j^{th} objective for i^{th} alternative.

The abbreviations of the objective function weights in this problem are as follows;

$w1$= The weight of the mean tardiness,

$w2$= The weight of the mean flow time.

The performance analysis of the dispatching rules and the proposed method are constructed based on this utility value. Problem Case-3 is a minimization problem, so the alternative with a smaller utility value is better. The normalization and the utility value calculation are executed considering 21 dispatching rules, the Random rule, and the Q-learning method; in total, 23 alternatives.

As mentioned before, the action set of each problem was constructed based on the best dispatching rule of the objective function in the related utilization and due date settings. Although the ranks of the dispatching rules and the proposed method are calculated involving all 23 alternatives, the performance of the proposed method was compared with the best dispatching rules of each problem and the Random rule. The best rules regarding the objective functions in each problem for all utilization and due date settings are given in Table 4.20.

Table 4.20 The best dispatching rules in each problem for all utilization and due date settings

			Problem 1.1	Problem 1.2	Problem 2.1	Problem 2.2	Problem 3.1	Problem 3.2
Mean Flow Time	**95U**	**TDD**	LWKR	LWKR	SPT	SPT	SPT	SPT
		LDD	LWKR	LWKR	SPT	SPT	SPT	SPT
	90U	**TDD**	LWKR	LWKR	SPT	SPT	SPT	SPT
		LDD	LWKR	LWKR	SPT	SPT	SPT	SPT
	85U	**TDD**	LWKR	LWKR	SPT	SPT	SPT	SPT
		LDD	LWKR	LWKR	SPT	SPT	SPT	SPT
Mean Tardiness	**95U**	**TDD**	MDD	MDD	COVERT	COVERT	COVERT	COVERT
		LDD	MDD	MDD	COVERT	COVERT	COVERT	COVERT
	90U	**TDD**	MDD	MDD	COVERT	COVERT	PT/TIS	PT/TIS
		LDD	MDD	MDD	COVERT	COVERT	PT/TIS	PT/TIS
	85U	**TDD**	MDD	MDD	COVERT	COVERT	PT/TIS	COVERT
		LDD	MDD	MDD	COVERT	COVERT	PT/TIS	COVERT

4.4.3.1 Performance Analysis of the Dispatching Rules and the Proposed Method Under Equally Weighted Objectives

In the first part of the performance analysis, the ranks of the selected rules and the proposed method for different utilization levels and due date settings in each problem were compared when $w1 = w2$, and the results are given in Table 4.21.

Table 4.21 The ranks of the selected rules and the proposed method when $w1 = w2$

		95U TDD	95U LDD	90U TDD	90U LDD	85U TDD	85U LDD	Average Rank
Problem 1.1	**LWKR**	3	4	3	4	4	11	4.83
	MDD	2	1	1	1	1	1	1.17
	Q-learning	1	2	2	2	2	3	2.00
	Random	4	3	7	6	10	15	7.50
Problem 1.2	**LWKR**	4	5	4	15	15	15	9.67
	MDD	3	1	3	1	1	2	1.83
	Q-learning	1	3	2	3	3	1	2.17
	Random	7	4	5	5	13	14	8.00
Problem 2.1	**SPT**	1	1	3	6	7	12	5.00
	COVERT	3	3	2	2	2	3	2.50
	Q-learning	2	2	1	1	1	1	1.33
	Random	4	4	6	4	13	13	7.33
Problem 2.2	**SPT**	1	1	2	4	4	12	4.00
	COVERT	4	4	3	3	3	2	3.17
	Q-learning	2	2	1	1	2	1	1.50
	Random	3	3	5	5	12	9	6.17
Problem 3.1	**SPT**	1	1	5	8	11	15	6.83
	PT/TIS	5	5	2	1	1	1	2.50
	COVERT	4	4	6	6	12	11	7.17
	Q-learning	2	2	1	2	3	3	2.17
	Random	3	3	4	5	5	10	5.00
Problem 3.2	**SPT**	1	1	2	5	12	13	5.67
	PT/TIS	5	5	1	1	1	1	2.33
	COVERT	3	4	5	7	5	7	5.17
	Q-learning	2	2	4	3	4	4	3.17
	Random	4	3	7	6	9	12	6.83

According to Table 4.21, the average rank of the Q-learning algorithm is better than other selected rules in Problem 2.1, Problem 2.2 and Problem 3.1, and the second-best method for the rest of the problems. On the other hand, considering the average rank, while the MDD is the best method in Problem 1.1 and Problem 1.2, the best method in Problem 3.1 and Problem 3.2 is the PT/TIS.

In order to analyze the variation of the ranks, the rank of the selected methods in each problem for all utilization and due date settings are also shown by line chart and given in Figure 4.6.

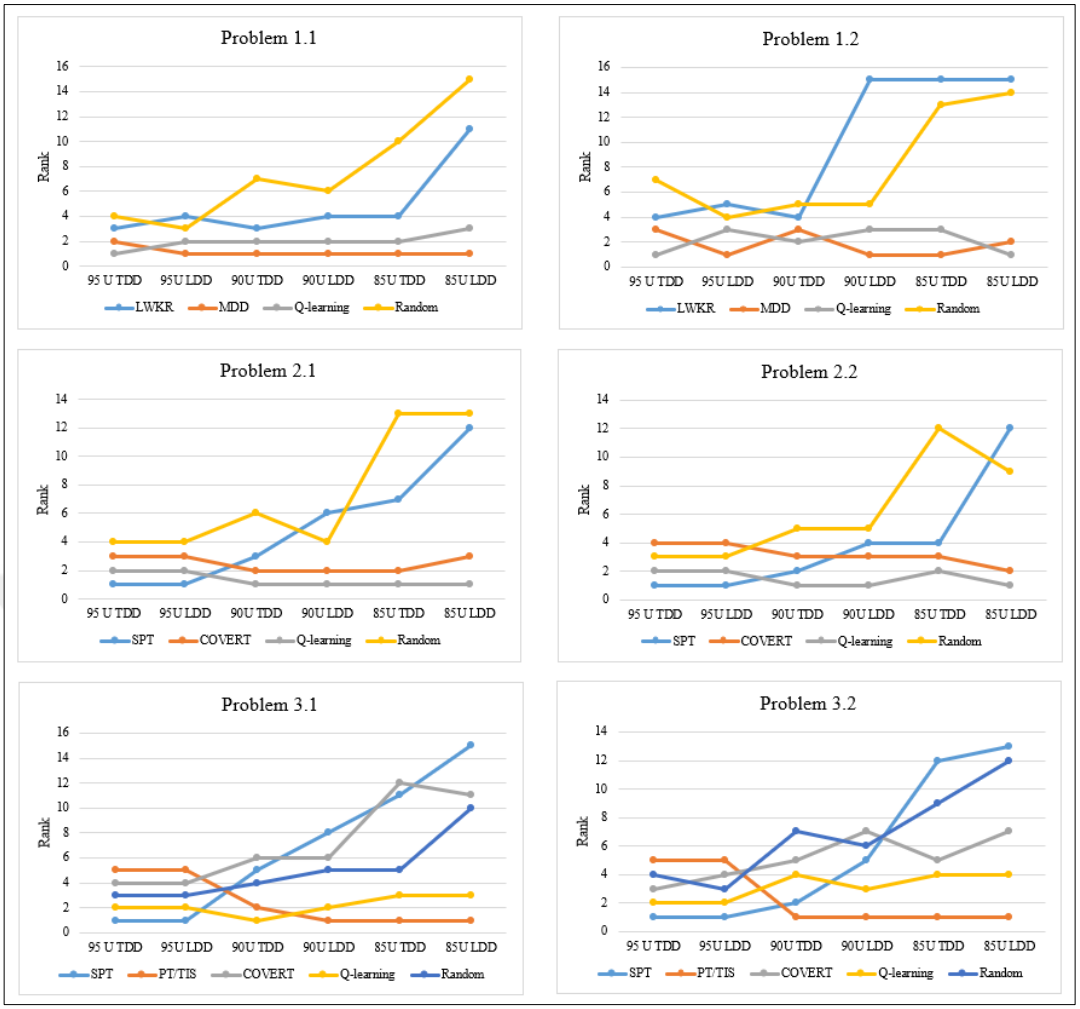

Figure 4.6 The ranks of the selected methods according to utilization and due date settings

Figure 4.6 shows that the ranks of the SPT, LWKR, and the Random vary significantly according to utilization and due date settings. On the other hand, the ranks of the MDD and the Q-learning algorithm changes little in Problem 1.1 and Problem 1.2. However, when we look at the rank of the COVERT, it is seen that while it is stable in Problem 2.1 and 2.2, it changes significantly in Problem 3.1 and Problem 3.2. In addition, although the average rank of the PT/TIS is better than the Q-learning algorithm in Problem 3.1 and Problem 3.2, it is more unstable than the rank of the Q-learning algorithm.

In addition to rank analysis, we examined the selection percentage of the dispatching rules in the Q-learning method in each problem for all system conditions

to analyze whether the agent learns to select the proper dispatching rule. The selection percentages of the dispatching rules for Problem 1.1 are given in Figure 4.7, and the selection percentages for the rest of the problems are given in Appendix A8.

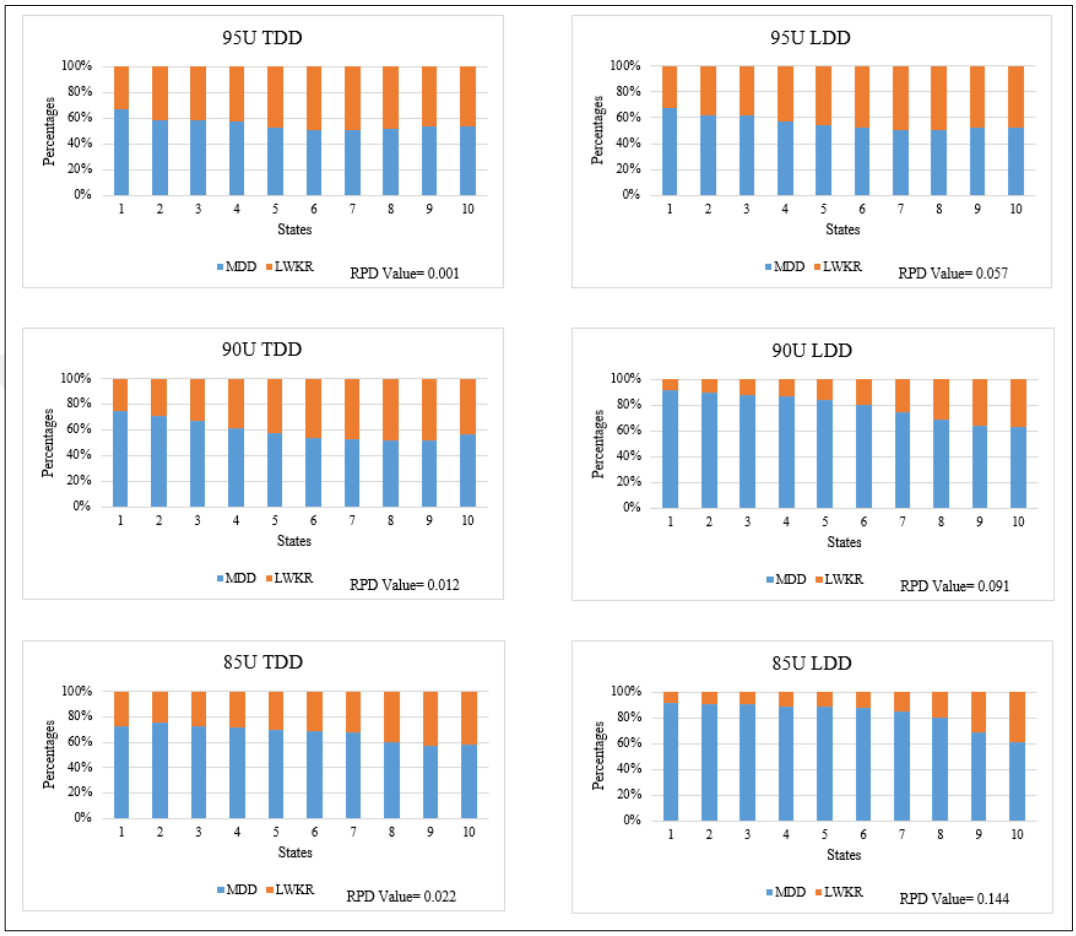

Figure 4.7 The selection percentages of the dispatching rules in Problem 1.1

In Problem 1.1, LWKR and MDD are the best rules for mean flow time and mean tardiness, respectively. Figure 4.7 shows that the selection percentages of the MDD rule decrease as the state number increases. This result can be explained as follows: In the proposed method, when the state number increases, the value of the MST decreases, indicating the tardiness level of the job increases. Therefore, if the agent selects the MDD rule when the MST is high, it receives more reward than in the state where the MST is low. In other words, it is more advantageous for the agent to select the MDD rule in the states with high MST.

Another thing that is seen in Figure 4.7, the selection percentages of the rules are remarkably close to each other in some problem settings. In order to examine this situation in detail, we compared the utility value of the dispatching rules by calculating the RPD values. In this problem, the utility value of the MDD is smaller in each problem setting. Therefore, the RPD value shows the relative percentage deviation of the LWKR from the MDD. It is clear that there is a relation between the RPD value and selection percentages of the dispatching rules. For example, while the RPD value is 0.001 in the problem setting of 95U and TDD, the selection percentages of the dispatching rules are extensively close to each other. On the other hand, the RPD value in the problem setting of 90U and LDD is extremely high in proportion to the RPD value in the problem setting of 95U and TDD, and it is undeniable that the selection percentage of the MDD outscores the selection percentage of the LWKR. As a result, based on these interpretations, it can be concluded that the agent learns to select the proper dispatching rule depending on changing situations.

4.4.3.2 Performance Analysis of the Dispatching Rules and the Proposed Method Under Differently Weighted Objectives

In this section, the performance of the proposed method was compared with dispatching rules by changing the value of objective weights. Moreover, it was aimed to analyze how the change in objective weights affects the performance of the selected methods. For this purpose, the value of $w1$ was assigned 0.01 and was increased by 0.01 from 0.01 to 1 to observe the variation in the ranks of the selected methods clearly. The ranks of the selected methods with respect to $w1$ in Problem 1.1 and Problem 1.2 is given in Figure 4.8 and Figure 4.9, respectively.

Figure 4.8 The ranks of the selected methods with respect to *w1* in Problem 1.1

Figure 4.9 The ranks of the selected methods with respect to *w1* in Problem 1.2

In Problem 1.1 and Problem 1.2, the best rule for mean flow time is LWKR, and the best rule for mean tardiness is MDD. Therefore, as expected, Figure 4.8 and 4.9 shows that while the weight of the mean tardiness increases, the rank of the LWKR improves; likewise, it shows that while the weight of the mean flow time increases, the rank of the MDD improves.

In Problem 1.1, for the 95U and TDD setting, while the value of $w1$ is between 0.33 and 0.65, the Q-learning method is the best and the second best for the rest of the weights. On the other hand, the rank of the MDD and LWKR varies between 1 and 4, and the rank of the Random varies between 3 and 4. For the same utilization level but LDD setting, the rank of the MDD is better than according to TDD setting because the due date allowance of the jobs is looser. However, the rank of the MDD varies between 1 and 5, while the rank of the Q-learning method varies between 1 and 2. For the rest of the utilization and due date settings, the rank of the MDD is 1 when the $w1$ is between 0.01 and 0.67 and varies between 1 and 5 for the rest of the weights. On the other hand, the rank of the LWKR is 1 only when the $w1$ is between 0.97 and 1 and varies between 1 and 13 for the rest of the weights. Likewise, the range of the rank of the Random is remarkably high. Therefore, it can be concluded that even if the importance of the objective functions change, the rank of the Q-learning method does not degenerate as much as the other methods.

In Problem 1.2, different from Problem 1.1, for the 95U and TDD settings, the ranks of the methods are more unstable. For example, the rank of the MDD varies from 1 to 7, and the rank of the Random varies from 4 to 7. Likewise, for the 90U and TDD settings, the rank of the MDD varies from 1 to 8. In addition, the rank of the LWKR and the Random is very variational in most of the utilization level and due date settings. Although the rank of the MDD is better than the rank of the Q-learning in loose due date settings, the rank of the Q-learning is more stable in most of the utilization level and due date settings.

The ranks of the selected methods with respect to $w1$ in Problem 2.1 and Problem 2.2 is given in Figure 4.10 and Figure 4.11, respectively.

Figure 4.10 The rank of the selected methods with respect to *w1* in Problem 2.1

Figure 4.11 The rank of the selected methods with respect to *w1* in Problem 2.2

In Problems 2.1 and 2.2, SPT and COVERT are the best rules for mean flow time and mean tardiness, respectively. Therefore, while the value of $w1$ increases, the rank of the SPT improves, and the rank of the COVERT gets worse in Figure 4.10 and Figure 4.11. In these problems, the ranks of the methods are more stable in high utilization levels, except for the Random. When the utilization level decreases, the variation of the SPT and the COVERT ranks increases. Again, these problems make it clear that the Q-learning rank is more stable than other methods. In addition, the rank of the Q-learning method is 1 for most of the changing values of weights in 85U and LDD settings.

The rank of the selected methods with respect to $w1$ in Problem 3.1 and Problem 3.2 are given in Figure 4.12 and Figure 4.13, respectively.

In Problems 3.1 and 3.2, while the best rule for mean flow time is SPT, the best rule for mean tardiness changes among PT/TIS and COVERT depending on different utilization and due date settings. In these problems, for 95U and TDD settings, the rank of the SPT is 1 for most of the changing values of weights and is more stable in high utilization levels. However, it may vary from 1 to 14 in low utilization levels. For problem settings where the COVERT is the best rule for mean tardiness, the rank of the Q-learning is better than the COVERT. However, for problem settings in which the PT/TIS is the best rule for mean tardiness, Q-learning does not outperform the PT/TIS. When we analyze this situation in detail, it is seen that although the PT/TIS is not the best rule for mean flow time, it obtains satisfying results in both mean flow time and mean tardiness. On the other hand, although Q-learning does not outperform PT/TIS, the variation in the rank of Q-learning is minor according to PT/TIS.

Figure 4.12 The rank of the selected methods with respect to *w1* in Problem 3.1

Figure 4.13 The rank of the selected methods with respect to *w1* in Problem 3.2

In the previous analyses, the ranks of the selected methods were examined under each combination of utilization and due date settings. In this analysis, we aimed to observe the rank of the selected methods for given weights independently from utilization and due date settings. Therefore, the ranks of the selected method in the all utilization and due date settings were averaged for given weights. The average ranks of the selected methods in Problem 1.1 and Problem 1.2 for the different values of the objective function weights are given in Figure 4.14 and Figure 4.15, respectively.

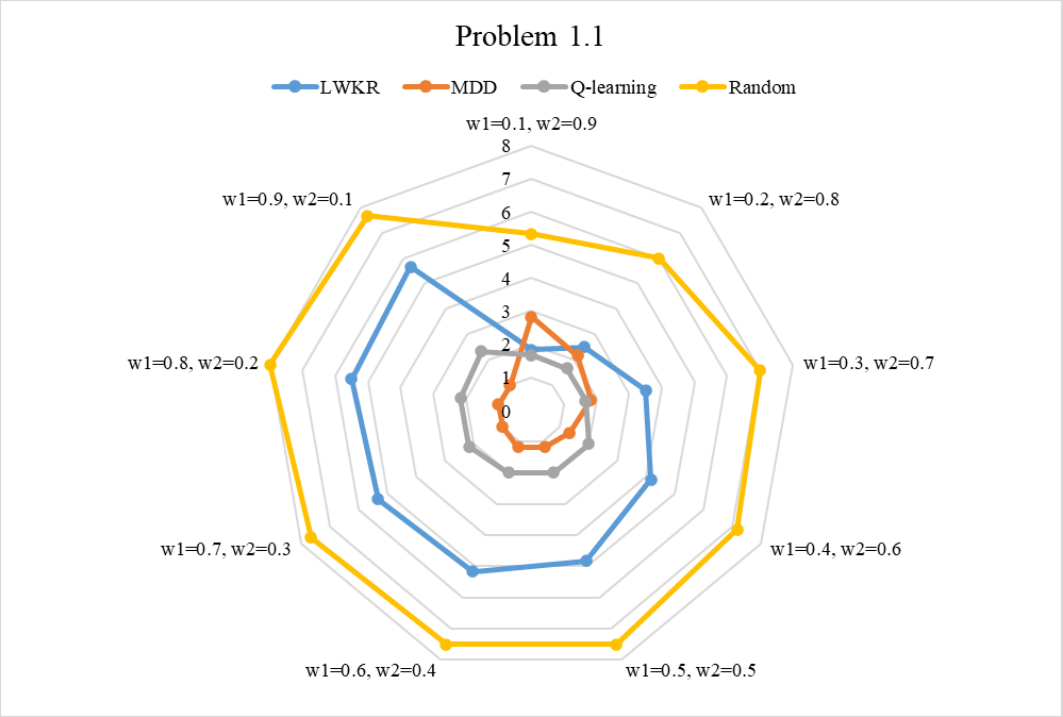

Figure 4.14 The average ranks of the selected methods in Problem 1.1

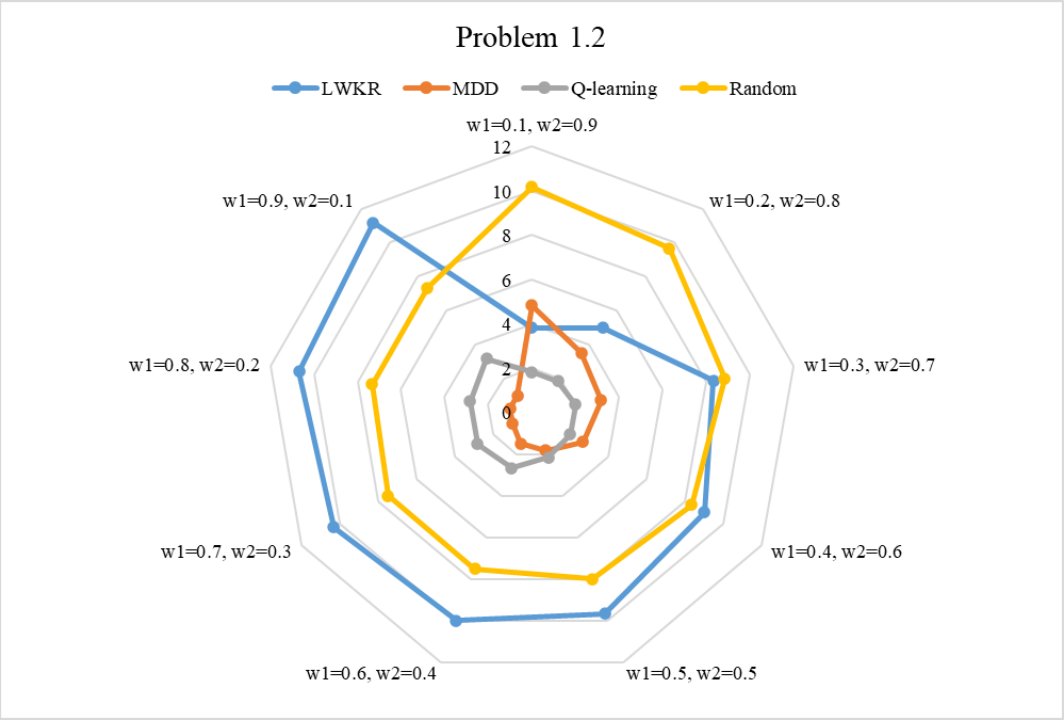

Figure 4.15 The average ranks of the selected methods in Problem 1.2

The average ranks of the Q-learning for all given weights are better than the LWKR and the Random rules in both Problem 1.1 and Problem 1.2. In addition, it outperforms the MDD when the $w1$ is 0.1, 0.2, and 0.3 in Problem 1.1 and when the $w1$ is 0.1, 0.2, 0.3, and 0.4 in Problem 1.2. Although the rank of the MDD is better than the Q-learning for the rest of the weights, the variation in the Q-learning is minor.

The average ranks of the selected methods in Problem 2.1 and Problem 2.2 for the different values of the objective function weights are given in Figure 4.16 and Figure 4.17, respectively.

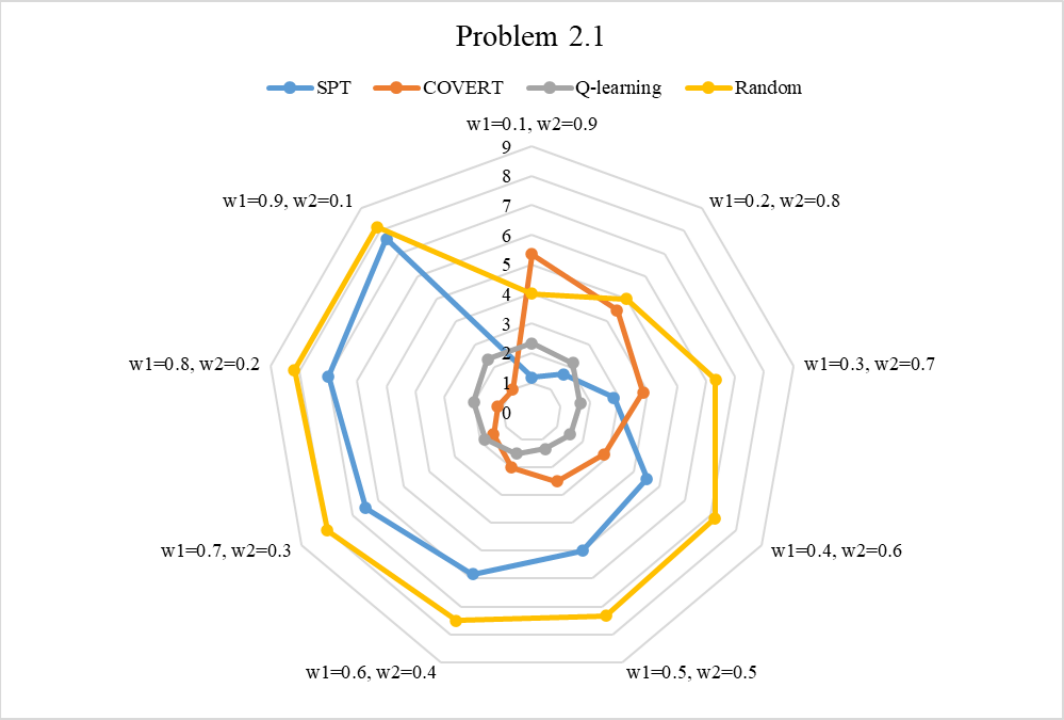

Figure 4.16 The average ranks of the selected methods in Problem 2.1

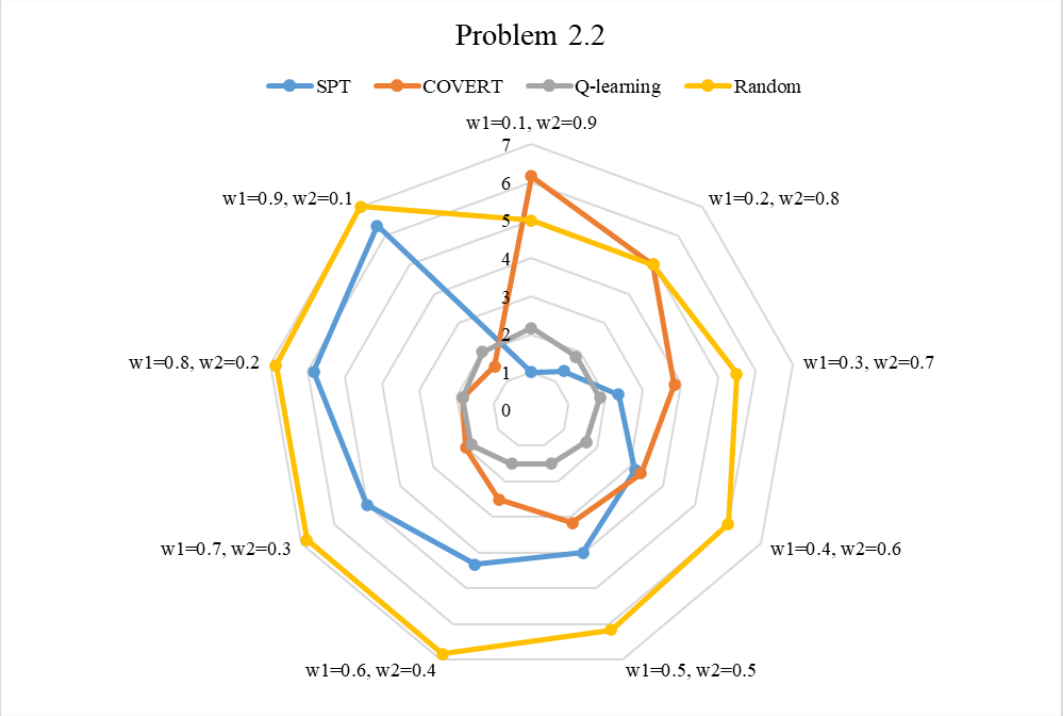

Figure 4.17 The average ranks of the selected methods in Problem 2.2

In Problem 2.1 and Problem 2.2, the Q-learning method obtains better results than all the selected rules when the w1 is 0.3, 0.4, and 0.5, demonstrating that the Q-learning better balances the objective functions when the importance of the weights is close to each other. Nevertheless, when the importance degree of an objective function is dramatically high against the other objective function, the best rule in the related objective function outperforms the Q-learning. On the other hand, in Problem 2.1 and Problem 2.2, the instability in the rank of the selected rules is much more.

The average ranks of the selected methods in Problem 3.1 and Problem 3.2 for the different values of the objective function weights are given in Figure 4.18 and Figure 4.19, respectively.

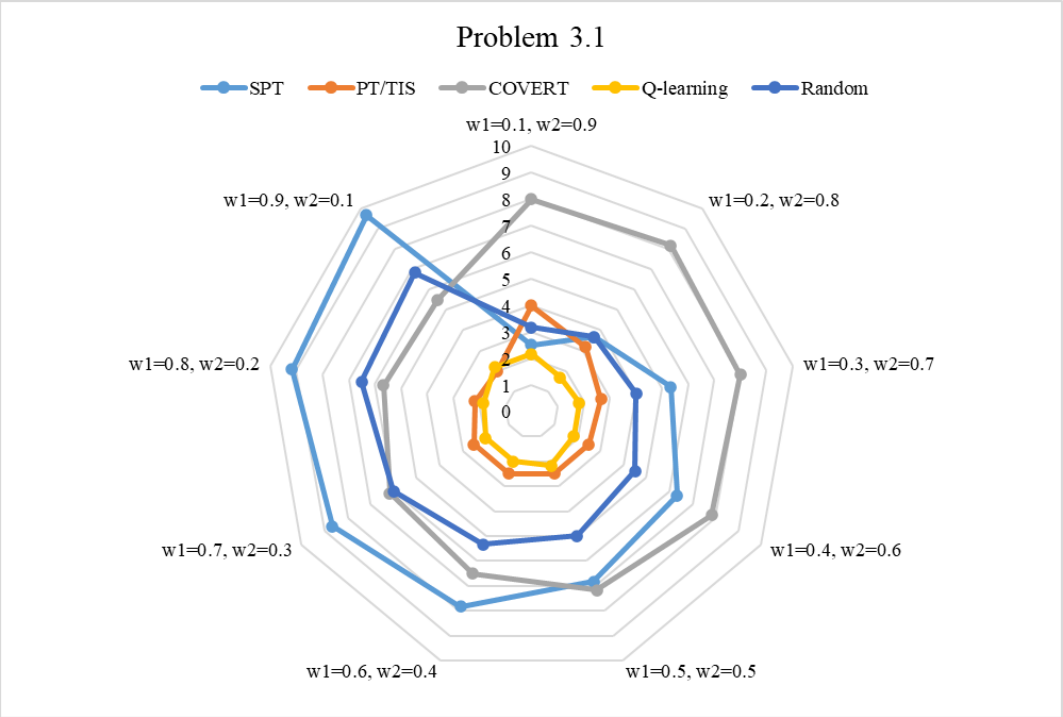

Figure 4.18 The average ranks of the selected methods in Problem 3.1

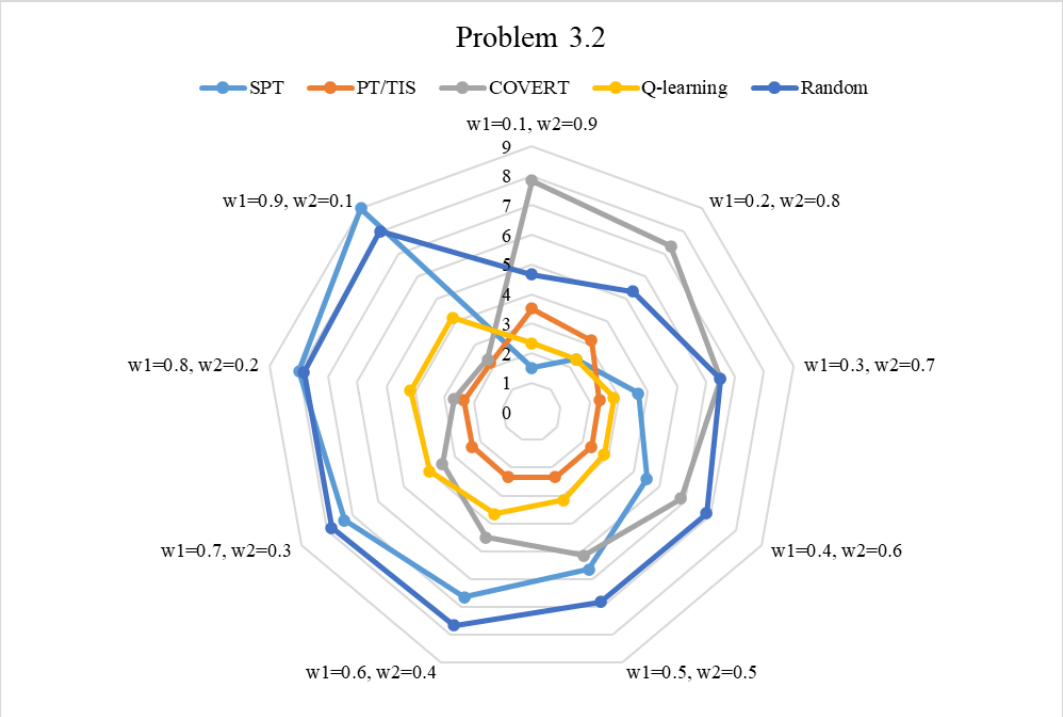

Figure 4.19 The average ranks of the selected methods in Problem 3.2

As mentioned before, the PT/TIS rule obtains satisfying results in both mean flow time and tardiness, especially in Problem 3.2. Therefore, it outscores the Q-learning method for most of the different weights of the objective functions in Problem 3.2. On the other hand, the average ranks of the Q-learning in Problem 3.1 are slightly better than the PT/TIS rule. In addition, the COVERT rule is the best rule in mean tardiness for some of the utilization and due date settings, it obtains satisfying result only in Problem 3.2 when the w1 is 0.9 and 0.8.

Another important finding that can be concluded from all the figures, the best rules in the mean tardiness objective obtain satisfactory results for different importance degrees of the objective functions. However, the best rules in the mean flow time objective obtain satisfying results only when the importance degree of the mean flow time is substantially high.

CHAPTER 5
CONCLUSIONS

5.1 Summary and Conclusions

In this dissertation, we proposed the RL based solution approaches to static and dynamic machine scheduling problems. The studies carried out in the scope of the dissertation consist of three phases named as Phase-1, Phase-2, and Phase-3.

In Phase-1, an in-depth literature analysis was carried out regarding the RL applications to machine scheduling problems. As a result of the analysis, it is seen that studies in this field are in an increasing trend, especially in recent years. However, most of the related studies have considered job shop scheduling problems and single objective cases. Thus, there is a lack of research on other machine environments and multi-objective cases. Therefore, we applied the RL algorithm to deterministic machine scheduling problems with the different types of machine environments. In addition, we studied a stochastic machine scheduling problem considering the multi-objective case.

In Phase-2, we aimed to apply RL to static/deterministic machine scheduling problems named Problem Case-1 and Problem Case-2. In Problem Case-1, we handled the application of the Q-learning algorithm to the particular case of a parallel machine scheduling problem: an unrelated parallel machine scheduling problem with the machine and sequence-dependent setup times. In Problem Case-2, we addressed the application of the Q-learning algorithm to a hybrid flow shop scheduling problem. In both of the problems, the objective is to minimize the makespan. The primary purpose in this phase is to analyze the effectiveness of the Q-learning algorithm in the given problems and to simplify the modeling of the algorithm. A simple and easily applicable state representation was developed to simplify the modeling of the algorithm. In addition, a reward system was designed independently of the parameters. These improvements help users to model different problems.

For Problem Case-1, the benchmark problems from the literature were selected to test the performance of the proposed method, and the results of the proposed method were compared with the results of the mentioned study. 640 different instances with various numbers of machine and job combinations were used. An RPD value that shows the deviation from the minimum makespan value was computed to evaluate the results. The proposed method found the minimum makespan value in 7 out of 16 instance combinations. For the remaining nine instance combinations, the RPD value has changed between 0.30 and 1.78. Problem Case-1 is the first one for this specific case, and the results show that the Q-learning algorithm can be applied to complicated machine scheduling problems even in deterministic environments.

For Problem Case-2, the large-scale instances of the study of Öztop et al. (2019) were selected to analyze the performance of the proposed method in large-scale problems. Instances comprise 40, 50, and 60 jobs, 5 stages in each instance, and ten instances for each job size. The results of the Q-learning algorithm were compared with the known best solutions of the instances, and the RPD values were computed to evaluate the results. The RPD values of the proposed method in these problems changes between the values of 1.30 and 9.88. These results show that the proposed method cannot obtain minimum values in large-scale problems, but it approximates to these results.

In addition, the learning phase of the proposed algorithm was analyzed in both problems. It was expected to see that the agent had learned to find the optimal order of the jobs in time. As a result of the analysis, it is seen that the obtained makespan value in each replication converged to the optimum makespan value in time.

The result of this phase demonstrates that the proposed Q-learning algorithm is applicable to complicated machine scheduling problems with small-scale instances. However, the performance of the prosed method decreases in large-scale problems.

In Phase-3, we handled a dynamic/stochastic hybrid flow shop scheduling problem with multi-objective named Problem Case-3. We developed a generic simulation

model to analyze the performance of the selected dispatching rules and the RL algorithm under various objective functions for different sizes of problems. First, we analyzed the performance of the selected dispatching rules under various objective functions. Then, the individual performance of dispatching rules under nine objective functions was given for the problems with 95U and TDD. Later, the similarity of the dispatching rules was compared between the same size of problems. As a result of detailed analysis, it was found that the performance of the dispatching rules in the problems of the same size is approximately identical. On the other hand, that the performance of the dispatching rules may differ based on the problem size. For instance, while the performance of some dispatching rules in minimizing mean flow time and tardiness gets worse when the stage number increases, the performance of some dispatching rules in minimizing mean flow time and tardiness improves. On the other hand, some dispatching rules have equivalent results in each problem independently of problem size. For example, while SPT and COVERT rules always obtain satisfactory results in minimizing mean flow time, waiting time, and tardiness, the performances of LTWF and LWKR rules are always adequate in minimizing the proportion of tardy jobs. Finally, the best five dispatching rules for different utilization and due date settings were summarized under each objective function.

In the second part of Phase-3, we handled the application of the RL method to a stochastic hybrid flow shop scheduling problem to minimize mean flow time and tardiness and examined whether the agent learns to select the proper dispatching rule according to the objective function. In order to analyze the performance of the selected dispatching rules and the proposed method, a hybrid flow shop production system with a Q-learning method was modeled in Arena 14.0 software. The system consists of various stages and identical parallel machines in each stage. The simulation model was designed in a generic form that can be easily adapted to the system with a different number of machines and stages.

In the third part of Phase-3, we evaluated the performance of the proposed method in different utilization and due date settings and compared it with dispatching rules. First, we considered the situation where the objectives have equal importance and

examined the ranks of the selected rules and the proposed method. The results showed that the average rank of the Q-learning is better than other selected rules in three of six problems and the second-best method for the rest of the problems. In addition, we analyzed the rank variation of the selected methods in each problem for all utilization and due date settings and found that the Q-learning is more stable than other methods in different utilization and due date settings. Besides, we examined the selection percentage of the dispatching rules in the Q-learning method to analyze whether the agent learns to select the proper dispatching rule and seen that the agent learns to select the proper dispatching rule depending on changing situations.

Afterward, we compared the performance of the proposed method with dispatching rules under each combination of utilization and due date settings by changing the value of objective weights to analyze how the change of objective weights affects the performance of the selected methods. As a result, it was found that even if the importance of the objective functions changes, the rank of the Q-learning method does not degenerate as much as the other methods.

Finally, we aimed to evaluate the rank of the selected methods for given weights, independently from utilization and due date settings. Therefore, the ranks of the selected methods in all utilization and due date settings were averaged for given weights. In general, the Q-learning method obtains satisfactory results when the importance of the weights is close to each other, demonstrating that Q-learning better balances the objective functions. Another important finding is that the best rules in the mean tardiness objective obtain satisfactory results for different importance degrees of the objective functions. However, the best rules in the mean flow time objective obtain satisfying results only when the importance degree of the mean flow time is substantially high.

5.2 Future Works

As a result of the comprehensive review and originated from the studies done in the scope of the dissertation, future research directions can be listed as follows;

- Especially in recent years, there has been a significant interest in Deep RL algorithms and studies shows that Deep RL outperforms the dispatching rules and also ensures better results in dynamic environments. In addition, in the Deep RL algorithms, the system state can be represented by several features giving more precise information about the environment. Therefore, this algorithm can be applied to real life machine scheduling problems.

- In this dissertation, we handled a multi-objective HFSSP. Since real-life scheduling problems usually include multiple objectives, different conflicting objectives should be analyzed deeply.

- The performance of the RL algorithms generally has been compared with dispatching rules or metaheuristics. Since the learning algorithm affects the performance of the RL considerably, analyzing the different learning algorithms by comparing each other can contribute to the literature.

REFERENCES

Ábrahám, G., Auer, P., Dósa, G., Dulai, T., & Werner-Stark, Á. (2019). A Reinforcement Learning Motivated Algorithm for Process Optimization. *Periodica Polytechnica Civil Engineering*, *63*(4), 961–970. https://doi.org/10.3311/ppci.14295

Adler, L., Fraiman, N., Kobacker, E., Pinedo, M., Plotnicoff, J. C., & Wu, T. P. (1993). BPSS: A Scheduling Support System for the Packaging Industry. *Operations Research*, *41*(4), 641–648. https://doi.org/10.1287/OPRE.41.4.641

Afshari, A., Mojahed, M., & Yusuff, R. M. (2010). Simple Additive Weighting approach to Personnel Selection problem. *International Journal of Innovation, Management and Technology*, *1*(5), 511–515.

Agnetis, A., Pacifici, A., Rossi, F., Lucertini, M., Nicoletti, S., Nicolò, F., Oriolo, G., Pacciarelli, D., & Pesaro, E. (1997). Scheduling of flexible flow lines in an automobile assembly plant. *European Journal of Operational Research*, *97*(2), 348–362. https://doi.org/10.1016/S0377-2217(96)00203-2

Aissani, N., Bekrar, A., Trentesaux, D., & Beldjilali, B. (2012). Dynamic scheduling for multi-site companies: A decisional approach based on reinforcement multi-agent learning. *Journal of Intelligent Manufacturing*, *23*(6), 2513–2529. https://doi.org/10.1007/s10845-011-0580-y

Aissani, N., & Trentesaux, D. (2008). Efficient and effective reactive scheduling of manufacturing system using Sarsa-multi-objective agents. *MOSIM'08: 7th Conference Internationale de Modelisation et Simulation*, *January*.

Aissani, N., Trentesaux, D., & Beldjilali, B. (2009). Multi-agent reinforcement learning for adaptive scheduling: Application to multi-site company. *IFAC*

Proceedings Volumes, *42*, 1102–1107. https://doi.org/10.3182/20090603-3-RU-2001.0280

Anderson, E. J., & Nyirenda, J. C. (1990). Two new rules to minimize tardiness in a job shop. *The International Journal of Production Research*, *28*(12), 2277–2292. https://doi.org/10.1080/00207549008942866

Aqil, S., & Allali, K. (2020). Local search metaheuristic for solving hybrid flow shop problem in slabs and beams manufacturing. *Expert Systems with Applications*, *162*. https://doi.org/10.1016/j.eswa.2020.113716

Arviv, K., Stern, H., & Edan, Y. (2016). Collaborative reinforcement learning for a two-robot job transfer flow-shop scheduling problem. *International Journal of Production Research*, *54*(4), 1196–1209. https://doi.org/10.1080/00207543.2015.1057297

Atighehchian, A., & Sepehri, M. M. (2013). An environment-driven, function-based approach to dynamic single-machine scheduling. *European Journal of Industrial Engineering*, *7*(1), 100–118. https://doi.org/10.1504/EJIE.2013.051594

Aydin, M. E., & Öztemel, E. (2000). Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems*, *33*(2), 169–178. https://doi.org/10.1016/S0921-8890(00)00087-7

Baker, K., & Trietsch, D. (2013). *Principles of sequencing and scheduling*. John Wiley & Sons.

Barto, A. G., & Mahadevan, S. (2003). Recent Advances in Hierarchical Reinforcement Learning. *Discrete Event Dynamic Systems*, *13*(1), 41–77. https://doi.org/10.1023/A:1022140919877

Bouazza, W., Sallez, Y., & Beldjilali, B. (2017). A distributed approach solving partially flexible job-shop scheduling problem with a Q-learning effect. *IFAC-PapersOnLine*, *50*(1), 15890–15895. https://doi.org/10.1016/j.ifacol.2017.08.2354

Brah, S. A., & Wheeler, G. E. (1998). Comparison of Scheduling Rules in a Flow Shop with Multiple Processors: A Simulation. *Simulation*, *71*(5), 302–311. https://doi.org/10.1177/003754979807100501

Brammer, J., Lutz, B., & Neumann, D. (2022). Permutation flow shop scheduling with multiple lines and demand plans using reinforcement learning. *European Journal of Operational Research*, *299*(1), 75–86. https://doi.org/10.1016/j.ejor.2021.08.007

Cao, C., Zhang, Y., Gu, X., Li, D., & Li, J. (2020). An improved gravitational search algorithm to the hybrid flowshop with unrelated parallel machines scheduling problem. *International Journal of Production Research*, *2021*(18), 1–17. https://doi.org/10.1080/00207543.2020.1788732

Chien, C. F., & Lan, Y. bin. (2021). Agent-based approach integrating deep reinforcement learning and hybrid genetic algorithm for dynamic scheduling for Industry 3.5 smart production. *Computers and Industrial Engineering*, *162*. https://doi.org/10.1016/j.cie.2021.107782

Csáji, B. C., & Monostori, L. (2005). Stochastic approximate scheduling by neurodynamic learning. *IFAC Proceedings Volumes*, *38*(1), 355–360. https://doi.org/10.3182/20050703-6-cz-1902.01481

Csáji, B. C., & Monostori, L. (2008). Adaptive stochastic resource control: A machine learning approach. *Journal of Artificial Intelligence Research*, *32*, 453–486. https://doi.org/10.1613/jair.2548

Csáji, B. C., Monostori, L., & Kádár, B. (2006). Reinforcement learning in a distributed market-based production control system. *Advanced Engineering Informatics*, *20*(3), 279–288. https://doi.org/10.1016/j.aei.2006.01.001

Das, T. K., Gosavi, A., Mahadevan, S., & Marchalleck, N. (1999). Solving semi-Markov decision problems using average reward reinforcement learning. *Management Science*, *45*(4), 560–574. https://doi.org/10.1287/MNSC.45.4.560

De Raedt, L. (2008). Logical and Relational Learning. In *Cognitive Technologies*. Springer Science & Business Media. https://doi.org/10.1007/978-3-540-68856-3

Ding, Z., & Dong, H. (2020). Challenges of reinforcement learning. *Deep Reinforcement Learning: Fundamentals, Research and Applications*, 249–272. https://doi.org/10.1007/978-981-15-4095-0_7/COVER

Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., & Hester, T. (2021). Challenges of real-world reinforcement learning: definitions, benchmarks, and analysis. *Machine Learning*, *110*(9), 2419–2468. https://doi.org/10.1007/S10994-021-05961-4/FIGURES/14

Edis, E. B., Oguz, C., & Ozkarahan, I. (2013). Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. *European Journal of Operational Research*, *230*(3), 449–463. https://doi.org/10.1016/j.ejor.2013.02.042

Ezugwu, A. E., & Akutsah, F. (2018). An improved firefly algorithm for the unrelated parallel machines scheduling problem with sequence-dependent setup times. *IEEE Access*, *6*, 54459–54478. https://doi.org/10.1109/ACCESS.2018.2872110

Fonseca Reyna, Y. C., Martínez Jiménez, Y., Bermúdez Cabrera, J. M., & Méndez Hernández, B. M. (2015). A reinforcement learning approach for scheduling problems. *Investigacion Operacional*, *36*(3), 225–231.

Framinan, J. M., & Leisten, R. (2003). An efficient constructive heuristic for flowtime minimisation in permutation flow shops. *Omega*, *31*(4), 311–317. https://doi.org/10.1016/S0305-0483(03)00047-1

Fuchigami, H. Y., & Rangel, S. (2018). A survey of case studies in production scheduling: Analysis and perspectives. *Journal of Computational Science*. https://doi.org/10.1016/j.jocs.2017.06.004

Gabel, T., & Riedmiller, M. (2006a). Reducing policy degradation in neuro-dynamic programming. *ESANN 2006 Proceedings - European Symposium on Artificial Neural Networks*, *April*, 653–658.

Gabel, T., & Riedmiller, M. (2006b). Multi-agent case-based reasoning for cooperative reinforcement learners. *European Conference on Case-Based Reasoning*, *4106 LNAI*, 32–46. https://doi.org/10.1007/11805816_5

Gabel, T., & Riedmiller, M. (2007a). On a successful application of multi-agent reinforcement learning to operations research benchmarks. *Proceedings of the 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning, ADPRL 2007*, *Adprl*, 68–75. https://doi.org/10.1109/ADPRL.2007.368171

Gabel, T., & Riedmiller, M. (2007b). Scaling adaptive agent-based reactive job-shop scheduling to large-scale problems. *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling, CI-Sched 2007*, 259–266. https://doi.org/10.1109/SCIS.2007.367699

Gabel, T., & Riedmiller, M. (2008). Adaptive Reactive Job-Shop Scheduling With Reinforcement Learning Agents. *International Journal of Information*

Technology and Intelligent Computing, *24*(4), 14–18. http://ml.informatik.uni-freiburg.de/_media/publications/gr07.pdf

Gabel, T., & Riedmiller, M. (2011). Distributed policy search reinforcement learning for job-shop scheduling tasks. *International Journal of Production Research*, *50*(1), 41–61. https://doi.org/10.1080/00207543.2011.571443

Gambella, C., Ghaddar, B., & Naoum-Sawaya, J. (2020). Optimization problems for machine learning: A survey. *European Journal of Operational Research*, *xxxx*. https://doi.org/10.1016/j.ejor.2020.08.045

Gao, K., Mei, G., Piccialli, F., Cuomo, S., Tu, J., & Huo, Z. (2020). Julia language in machine learning: Algorithms, applications, and open issues. *Computer Science Review*, *37*, 100254. https://doi.org/10.1016/j.cosrev.2020.100254

Gosavi, A. (2015). *Simulation-based optimization*. Springer.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. H. G. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*. https://doi.org/10.1016/S0167-5060(08)70356-X

Guo, F., Li, Y., Liu, A., & Liu, Z. (2020). A Reinforcement Learning Method to Scheduling Problem of Steel Production Process. *Journal of Physics: Conference Series*, *1486*(7), 13–15. https://doi.org/10.1088/1742-6596/1486/7/072035

Guo, L., Zhuang, Z., Huang, Z., & Qin, W. (2020, August). Optimization of dynamic multi-objective non-identical parallel machine scheduling with multi-stage reinforcement learning. In *2020 IEEE 16th international conference on automation science and engineering (CASE)* (pp. 1215-1219).

Gupta, J. N. D. (1988). Two-Stage, Hybrid Flowshop Scheduling Problem. *Journal of the Operational Research Society*, *39*(4), 359–3641. https://doi.org/10.1057/JORS.1988.63

Han, W., Guo, F., & Su, X. (2019). A reinforcement learning method for a hybrid flow-shop scheduling problem. *Algorithms*, *12*(11). https://doi.org/10.3390/a12110222

Heuillet, A., Couthouis, F., & Díaz-Rodríguez, N. (2021). Explainability in deep reinforcement learning. *Knowledge-Based Systems*, *214*, 106685. https://doi.org/10.1016/J.KNOSYS.2020.106685

Holthaus, O., & Rajendran, C. (1997). Efficient dispatching rules for scheduling in a job shop. *International Journal of Production Economics*, *48*(1), 87–105. https://doi.org/10.1016/S0925-5273(96)00068-0

Hong, J., & Prabhu, V. V. (2004). Distributed reinforcement learning control for batch sequencing and sizing in just-in-time manufacturing systems. *Applied Intelligence*, *20*(1), 71–87. https://doi.org/10.1023/B:APIN.0000011143.95085.74

Idrees, H. D., Sinnokrot, M. O., & Al-Shihabi, S. (2006). A reinforcement learning algorithm to minimize the mean tardiness of a single machine with controlled capacity. *Proceedings - Winter Simulation Conference*, *1*, 1765–1769. https://doi.org/10.1109/WSC.2006.322953

Iwamura, K., Mayumi, N., Tanimizu, Y., & Sugimura, N. (2010). A study on real-time scheduling for holonic manufacturing systems - Determination of utility values based on multi-agent reinforcement learning. *International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, *5696 LNAI*, 135–144. https://doi.org/10.1007/978-3-642-03668-2_13

Jiménez, Y. M., Palacio, J. C., & Nowé, A. (2020). Multi-Agent Reinforcement Learning Tool for Job Shop Scheduling Problems. *International Conference on Optimization and Learning*, 3–12. https://doi.org/10.1007/978-3-030-41913-4

Kaelbling, L., Littman, M. L., Moore, A. W., & Hall, S. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Khadilkar Harshad. (2018). A scalable reinforcement learning algorithm for scheduling railway lines. *IEEE Transactions on Intelligent Transportation*, *20*(2), 727–736. https://ieeexplore.ieee.org/abstract/document/8357909

Kim, G. H., & Lee, C. S. G. (1996). Genetic reinforcement learning for scheduling heterogeneous machines. *Proceedings - IEEE International Conference on Robotics and Automation*, *3*(April), 2798–2803. https://doi.org/10.1109/robot.1996.506586

Kim, N., & Shin, H. (2017). The application of actor-critic reinforcement learning for fab dispatching scheduling. *2017 Winter Simulation Conference (WSC). IEEE*, 4570–4571.

Kong, L. F., & Wu, J. (2005). Dynamic single machine scheduling using Q-learning agent. *2005 International Conference on Machine Learning and Cybernetics, ICMLC 2005, August*, 3237–3241. https://doi.org/10.1109/icmlc.2005.1527501

Korytkowski, P., Wiśniewski, T., & Rymaszewski, S. (2013). An evolutionary simulation-based optimization approach for dispatching scheduling. *Simulation Modelling Practice and Theory*, *35*, 69–85. https://doi.org/10.1016/j.simpat.2013.03.006

Kuhnle, A., Kaiser, J. P., Theiß, F., Stricker, N., & Lanza, G. (2021). Designing an adaptive production control system using reinforcement learning. *Journal of*

Intelligent Manufacturing, *32*(3), 855–876. https://doi.org/10.1007/s10845-020-01612-y

Kurz, M. E., & Askin, R. G. (2001). Heuristic scheduling of parallel machines with sequence-dependent set-up times. *International Journal of Production Research*, *39*(16), 3747–3769. https://doi.org/10.1080/00207540110064938

Lee, J. H., & Kim, H. J. (2021). Reinforcement learning for robotic flow shop scheduling with processing time variations. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2021.1887533

Lee, J. H., Shin, J., & Realff, M. J. (2018). Machine learning: Overview of the recent progresses and implications for the process systems engineering field. *Computers and Chemical Engineering*, *114*, 111–121. https://doi.org/10.1016/j.compchemeng.2017.10.008

Lee, K., Leung, J. Y. T., & Pinedo, M. L. (2013). Makespan minimization in online scheduling with machine eligibility. *Annals of Operations Research*, *204*(1), 189–222. https://doi.org/10.1007/s10479-012-1271-6

Lee, S., Cho, Y., & Lee, Y. H. (2020). Injection Mold Production Sustainable Scheduling Using Deep Reinforcement Learning. *Sustainability*, *12*(20), 1–17.

Lee, Y. H., & Lee, S. (2022). Deep reinforcement learning based scheduling within production plan in semiconductor fabrication. *Expert Systems with Applications*, *191*. https://doi.org/10.1016/j.eswa.2021.116222

Li, L., Huo, J., & Tang, O. (2011). A hybrid flowshop scheduling problem for a cold treating process in seamless steel tube production. *International Journal of Production Research*, *49*(15), 4679–4700. https://doi.org/10.1080/00207543.2010.495208

Lihu, A., & Holban, Ş. (2009). Top five most promising algorithms in scheduling. *Proceedings - 2009 5th International Symposium on Applied Computational Intelligence and Informatics, SACI 2009*, 397–404. https://doi.org/10.1109/SACI.2009.5136281

Lin, C. C., Deng, D. J., Chih, Y. L., & Chiu, H. T. (2019). Smart Manufacturing Scheduling with Edge Computing Using Multiclass Deep Q Network. *IEEE Transactions on Industrial Informatics*, *15*(7), 4276–4284. https://doi.org/10.1109/TII.2019.2908210

Lin, H. T., & Liao, C. J. (2003). A case study in a two-stage hybrid flow shop with setup time and dedicated machines. *International Journal of Production Economics*, *86*(2), 133–143. https://doi.org/10.1016/S0925-5273(03)00011-2

Liu, C. C., Jin, H. Y., Tian, Y., & Yu, H. Bin. (2001). Reinforcement learning approach to re-entrant manufacturing system scheduling. *2001 International Conferences on Info-Tech and Info-Net: A Key to Better Life, ICII 2001 - Proceedings*, *3*, 280–285. https://doi.org/10.1109/ICII.2001.983070

Liu, C. L., Chang, C. C., & Tseng, C. J. (2020). Actor-critic deep reinforcement learning for solving job shop scheduling problems. *IEEE Access*, *8*, 71752–71762. https://doi.org/10.1109/ACCESS.2020.2987820

Liu, W., & Wang, X. (2009). Dynamic Decision Model in Evolutionary Games Based on Reinforcement Learning. *Systems Engineering - Theory & Practice*, *29*(3), 28–33. https://doi.org/10.1016/s1874-8651(10)60008-7

Luo, P. C., Xiong, H. Q., Zhang, B. W., Peng, J. Y., & Xiong, Z. F. (2021). Multi-resource constrained dynamic workshop scheduling based on proximal policy optimisation. *International Journal of Production Research*, 1–19. https://doi.org/10.1080/00207543.2021.1975057

Luo, S. (2020). Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Applied Soft Computing Journal*, *91*, 106208. https://doi.org/10.1016/j.asoc.2020.106208

Miyashita, K. (2000). Learning scheduling control knowledge through reinforcements. *International Transactions in Operational Research*, *7*(2), 125–138. https://doi.org/10.1111/j.1475-3995.2000.tb00190.x

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236

Monostori, L., & Csáji, B. C. (2006). Stochastic dynamic production control by neurodynamic programming. *CIRP Annals - Manufacturing Technology*, *55*(1), 473–478. https://doi.org/10.1016/S0007-8506(07)60462-4

Monostori, L., Csáji, B. C., & Kádár, B. (2004). Adaptation and learning in distributed production control. *CIRP Annals - Manufacturing Technology*, *53*(1), 349–352. https://doi.org/10.1016/S0007-8506(07)60714-8

Nahmias, S., & Olsen, T. L. (2015). *Production and Operations Analysis* (Vol. 7). Waveland Press.

Öztop, H., Fatih Tasgetiren, M., Eliiyi, D. T., & Pan, Q. K. (2019). Metaheuristic algorithms for the hybrid flowshop scheduling problem. *Computers & Operations Research*, *111*, 177–196. https://doi.org/10.1016/J.COR.2019.06.009

Paeng, B., Park, I. B., & Park, J. (2021). Deep Reinforcement Learning for Minimizing Tardiness in Parallel Machine Scheduling with Sequence Dependent

Family Setups. *IEEE Access*, *9*, 101390–101401. https://doi.org/10.1109/ACCESS.2021.3097254

Palombarini, J., & Martínez, E. (2010). Learning to Repair Plans and Schedules Using a Relational (Deictic) Representation. *Computer Aided Chemical Engineering*, *27*(C), 1377–1382. https://doi.org/10.1016/S1570-7946(09)70620-0

Palombarini, J., & Martínez, E. (2012a). SmartGantt – An interactive system for generating and updating rescheduling knowledge using relational abstractions. *Computers and Chemical Engineering*, *47*, 202–216. https://doi.org/10.1016/j.compchemeng.2012.06.021

Palombarini, J., & Martínez, E. (2012b). SmartGantt – An intelligent system for real time rescheduling based on relational reinforcement learning. *Expert Systems With Applications*, *39*(11), 10251–10268. https://doi.org/10.1016/j.eswa.2012.02.176

Pan, Z., Wang, L., Wang, J., & Lu, J. (2021). Deep Reinforcement Learning Based Optimization Algorithm for Permutation Flow-Shop Scheduling. *IEEE Transactions on Emerging Topics in Computational Intelligence*. https://doi.org/10.1109/TETCI.2021.3098354

Parente, M., Figueira, G., Amorim, P., & Marques, A. (2020). Production scheduling in the context of Industry 4.0: review and trends. *International Journal of Production Research*, *58*(17), 5401–5431. https://doi.org/10.1080/00207543.2020.1718794

Park, I., Huh, J., Kim, J., & Park, J. (2020). *A Reinforcement Learning Approach to Robust Scheduling of Semiconductor Manufacturing Facilities*. *17*(3), 1420–1431.

Park, J., Chun, J., Kim, S. H., Kim, Y., & Park, J. (2021). Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning. *International Journal of Production Research*, *59*(11), 3360–3377. https://doi.org/10.1080/00207543.2020.1870013

Paternina-Arboleda, C. D., & Das, T. K. (2001). Intelligent dynamic control policies for serial production lines. *IIE Transactions (Institute of Industrial Engineers)*, *33*(1), 65–77. https://doi.org/10.1080/07408170108936807

Pinedo, M. L. (2012). *Scheduling*. Springer.

Qin, W., Zhang, J., & Song, D. (2018). An improved ant colony algorithm for dynamic hybrid flow shop scheduling with uncertain processing time. *Journal of Intelligent Manufacturing*, *29*(4), 891–904. https://doi.org/10.1007/S10845-015-1144-3/FIGURES/12

Qu, S., Chu, T., Wang, J., Leckie, J., & Jian, W. (2015). A centralized reinforcement learning approach for proactive scheduling in manufacturing. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, *2015-Octob*, 1–8. https://doi.org/10.1109/ETFA.2015.7301417

Qu, S., Wang, J., Govil, S., & Leckie, J. O. (2016a). Optimized Adaptive Scheduling of a Manufacturing Process System with Multi-skill Workforce and Multiple Machine Types: An Ontology-based, Multi-agent Reinforcement Learning Approach. *Procedia CIRP*, *57*, 55–60. https://doi.org/10.1016/j.procir.2016.11.011

Qu, S., Jie, W., & Shivani, G. (2016b). Learning adaptive dispatching rules for a manufacturing process system by using reinforcement learning approach. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, *2016-Novem*, 1–8. https://doi.org/10.1109/ETFA.2016.7733712

Rabadi, G., Moraga, R. J., & Al-Salem, A. (2006). Heuristics for the unrelated parallel machine scheduling problem with setup times. *Journal of Intelligent Manufacturing*, *17*(1), 85–97. https://doi.org/10.1007/s10845-005-5514-0

Ramírez-Hernández, J. A., & Fernandez, E. (2005). A case study in scheduling reentrant manufacturing lines: Optimal and simulation-based approaches. *Proceedings of the 44th IEEE Conference on Decision and Control, 2005*, 2158–2163. https://doi.org/10.1109/CDC.2005.1582481

Ramírez-Hernández, J. A., & Fernandez, E. (2009). A simulation-based Approximate Dynamic Programming approach for the control of the intel Mini-Fab benchmark model. *Proceedings - Winter Simulation Conference*, *December*, 1634–1645. https://doi.org/10.1109/WSC.2009.5429179

Ren, J., Ye, C., & Yang, F. (2020). A Novel Solution to JSPs Based on Long Short-Term Memory And Policy Gradient Algorithm. *International Journal of Simulation Modelling*, *19*(1), 157–168.

Reyna, Y. C. F., Cáceres, A. P., Jiménez, Y. M., & Reyes, Y. T. (2019a). An improvement of reinforcement learning approach for permutation of flow-shop scheduling problems. *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao*, *2019*(E18), 257–270.

Reyna, Y. C. F., Jiménez, Y. M., Cabrera, A. V., & Sánchez, E. A. (2019b). Optimization of heavily constrained hybrid-flexible flowshop problems using a multi-agent reinforcement learning approach. *Investigacion Operacional*, *40*(1), 100–111.

Reyna, Y. C. F., Jiménez, Y. M., & Nowé, A. (2018). Q-Learning algorithm performance for m-machine n-jobs flow shop scheduling to minimize makespan. *Investigación Operacional*, *38*(3), 281–290.

Ribas, I., Leisten, R., & Framiñan, J. M. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research*, *37*(8), 1439–1454. https://doi.org/10.1016/J.COR.2009.11.001

Riedmiller, S., & Riedmiller, M. (1999). A neural reinforcement learning approach to learn local dispatching policies in production scheduling. *IJCAI International Joint Conference on Artificial Intelligence*, *2*, 764–769.

Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research*, *205*(1), 1–18. https://doi.org/10.1016/j.ejor.2009.09.024

Russel, S. J., & Norvig, P. G. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Samsonov, V., Kemmerling, M., Paegert, M., Lütticke, D., Sauermann, F., Gützlaff, A., Schuh, G., & Meisen, T. (2021). Manufacturing control in job shop environments with reinforcement learning. *ICAART 2021 - Proceedings of the 13th International Conference on Agents and Artificial Intelligence*, *2*, 589–597. https://doi.org/10.5220/0010202405890597

Schwartz, A. (1993). A Reinforcement Learning Method for Maximizing Undiscounted Rewards. *Machine Learning Proceedings 1993*, *March*, 298–305. https://doi.org/10.1016/b978-1-55860-307-3.50045-9

Shiue, Y., Lee, K., & Su, C. (2018). Real-time scheduling for a smart factory using a reinforcement learning approach. *Computers & Industrial Engineering*, *125*(101), 604–614. https://doi.org/10.1016/j.cie.2018.03.039

Sigaud, O., & Buffet, O. (2013). Markov Decision Processes in Artificial Intelligence: MDPs, beyond MDPs and applications. In *John Wiley & Sons*. John Wiley and Sons. https://doi.org/10.1002/9781118557426

Stricker, N., Kuhnle, A., Sturm, R., & Friess, S. (2018). Reinforcement learning for adaptive order dispatching in the semiconductor industry. *CIRP Annals - Manufacturing Technology*, *67*(1), 511–514. https://doi.org/10.1016/j.cirp.2018.04.041

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning : an introduction*. MIT Press.

Szepesvári, C. (2010). Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *9*, 1–89. https://doi.org/10.2200/S00268ED1V01Y201005AIM009

Tang, L., Liu, W., & Liu, J. (2005). A neural network model and algorithm for the hybrid flow shop scheduling problem in a dynamic environment. *Journal of Intelligent Manufacturing 2005 16:3*, *16*(3), 361–370. https://doi.org/10.1007/S10845-005-7029-0

Tavares Neto, R. F., & Godinho Filho, M. (2013). Literature review regarding Ant Colony Optimization applied to scheduling problems: Guidelines for implementation and directions for future research. *Engineering Applications of Artificial Intelligence*, *26*(1), 150–161. https://doi.org/10.1016/j.engappai.2012.03.011

Thomas, T. E., Koo, J., Chaterji, S., & Bagchi, S. (2018). Minerva: A reinforcement learning-based technique for optimal scheduling and bottleneck detection in distributed factory operations. *2018 10th International Conference on Communication Systems and Networks, COMSNETS 2018*, *2018-Janua*, 129–136. https://doi.org/10.1109/COMSNETS.2018.8328189

Usuga Cadavid, J. P., Lamouri, S., Grabot, B., Pellerin, R., & Fortin, A. (2020). Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0. *Journal of Intelligent Manufacturing*, *31*(6), 1531–1558. https://doi.org/10.1007/s10845-019-01531-7

Vallada, E., & Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, *211*(3), 612–622. https://doi.org/10.1016/j.ejor.2011.01.011

van Otterlo, M. (2009). *The logic of adaptive behavior: knowledge representation and algorithms for adaptive sequential decision making under uncertainty in first-order and relational domains*. Ios Press.

Vapnik, V. (2000). *The nature of statistical learning theory* (1st ed.). Springer.

Wang, H., Sarker, B. R., Li, J., & Li, J. (2020). Adaptive scheduling for assembly job shop with uncertain assembly times based on dual Q- learning. *International Journal of Production Research*, *0*(0), 1–17. https://doi.org/10.1080/00207543.2020.1794075

Wang, H., & Yan, H. (2013a). An Adaptive Scheduling System in Knowledgeable Manufacturing Based on Multi-agent. *10th IEEE International Conference on Control and Automation (ICCA)*, 496–501. https://doi.org/10.1109/ICCA.2013.6564866

Wang, H. X., & Yan, H. Sen. (2013b). An adaptive assembly scheduling approach in knowledgeable manufacturing. *Applied Mechanics and Materials*, *433–435*, 2347–2350. https://doi.org/10.4028/www.scientific.net/AMM.433-435.2347

Wang, H. X., & Yan, H. sen. (2016). An interoperable adaptive scheduling strategy for knowledgeable manufacturing based on SMGWQ-learning. *Journal of*

Intelligent Manufacturing, *27*(5), 1085–1095. https://doi.org/10.1007/s10845-014-0936-1

Wang, H., Yan, Q., & Zhang, S. (2021). Integrated scheduling and flexible maintenance in deteriorating multi-state single machine system using a reinforcement learning approach. *Advanced Engineering Informatics*, *49*. https://doi.org/10.1016/j.aei.2021.101339

Wang, Y. C., & Usher, J. M. (2004). Learning policies for single machine job dispatching. *Robotics and Computer-Integrated Manufacturing*, *20*(6 SPEC. ISS.), 553–562. https://doi.org/10.1016/j.rcim.2004.07.003

Wang, Y. C., & Usher, J. M. (2005). Application of reinforcement learning for agent-based production scheduling. *Engineering Applications of Artificial Intelligence*, *18*(1), 73–82. https://doi.org/10.1016/j.engappai.2004.08.018

Wang, Y. C., & Usher, J. M. (2007). A reinforcement learning approach for developing routing policies in multi-agent production scheduling. *International Journal of Advanced Manufacturing Technology*, *33*(3–4), 323–333. https://doi.org/10.1007/s00170-006-0465-y

Wang, Y. F. (2018). Adaptive job shop scheduling strategy based on weighted Q-learning algorithm. *Journal of Intelligent Manufacturing*, *1997*. https://doi.org/10.1007/s10845-018-1454-3

Waschneck, B., Reichstaller, A., Belzner, L., Altenmüller, T., Bauernhansl, T., Knapp, A., & Kyek, A. (2018a). Optimization of global production scheduling with deep reinforcement learning. *Procedia CIRP*, *72*, 1264–1269. https://doi.org/10.1016/j.procir.2018.03.212

Waschneck, B., Reichstaller, A., Belzner, L., Altenmuller, T., Bauernhansl, T., Knapp, A., & Kyek, A. (2018). Deep reinforcement learning for semiconductor

production scheduling. *29th Annual SEMI Advanced Semiconductor Manufacturing Conference*, 301–306. https://doi.org/10.1109/ASMC.2018.8373191

Watkins, C. J., & Dayan, P. (1992). Q-Learning. *Machine Learning*, *8*(3–4), 279–292. https://doi.org/10.4018/978-1-59140-993-9.ch026

Wei, Y., & Zhao, M. (2004). Composite Rules Selection Using Reinforcement Learning for Dynamic Job-Shop Scheduling. *2004 IEEE Conference on Robotics, Automation and Mechatronics*, *2*, 1083–1088. https://doi.org/10.1109/ramech.2004.1438070

Xanthopoulos, A. S., Koulouriotis, D. E., Tourassis, V. D., & Emiris, D. M. (2013). Intelligent controllers for bi-objective dynamic scheduling on a single machine with sequence-dependent setups. *Applied Soft Computing Journal*, *13*(12), 4704–4717. https://doi.org/10.1016/j.asoc.2013.07.015

Xiao, Y., Tan, Q., Zhou, L., & Tang, H. (2017). Stochastic scheduling with compatible job families by an improved Q-learning algorithm. *Chinese Control Conference, CCC*, 2657–2662. https://doi.org/10.23919/ChiCC.2017.8027764

Yang, H. B., & Yan, H. S. (2009). An adaptive approach to dynamic scheduling in knowledgeable manufacturing cell. *International Journal of Advanced Manufacturing Technology*, *42*(3–4), 312–320. https://doi.org/10.1007/s00170-008-1588-0

Yang, H., & Yan, H. (2007). An adaptive policy of dynamic scheduling in knowledgeable manufacturing environment. *IEEE International Conference on Automation and Logistics7*, 835–840. https://doi.org/10.1109/ICAL.2007.4338680

Yang, S., Xu, Z., & Wang, J. (2021). Intelligent decision-making of scheduling for dynamic permutation flowshop via deep reinforcement learning. *Sensors (Switzerland)*, *21*(3), 1–21. https://doi.org/10.3390/s21031019

Ying, K. C., & Cheng, H. M. (2010). Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic. *Expert Systems with Applications*, *37*(4), 2848–2852. https://doi.org/10.1016/J.ESWA.2009.09.006

Yingzi, W. E. I., Xinli, J., & Pingbo, H. A. O. (2009). Pattern Driven Dynamic Scheduling Approach using Reinforcement Learning. *IEEE International Conference on Automation and Logistics*, *2002*, 514–519. https://doi.org/10.1109/ICAL.2009.5262867

Yu, J. M., Huang, R., & Lee, D. H. (2017). Iterative algorithms for batching and scheduling to minimise the total job tardiness in two-stage hybrid flow shops. *International Journal of Production Research*, *55*(11), 3266–3282. https://doi.org/10.1080/00207543.2017.1304661

Yuan, B., Jiang, Z., & Wang, L. (2016). Dynamic parallel machine scheduling with random breakdowns using the learning agent. *International Journal of Services Operations and Informatics*, *8*(2), 94–103. https://doi.org/10.1504/IJSOI.2016.080083

Yuan, B., Wang, L., & Jiang, Z. (2013). Dynamic Parallel Machine Scheduling Using the Learning Agent. *2013 IEEE International Conference on Industrial Engineering and Engineering Management*, 1565–1569. https://doi.org/10.1109/IEEM.2013.6962673

Zhang, T., Xie, S., & Rose, O. (2017). Real-time job shop scheduling based on simulation and Markov decision processes. *Proceedings - Winter Simulation Conference*, *5*, 3899–3907.

Zhang, T., Xie, S., & Rose, O. (2018). Real-time batching in job shops based on simulation and reinforcement learning. *Proceedings - Winter Simulation Conference*, 3331–3339. https://doi.org/10.1109/WSC.2018.8632524

Zhang, W., & Dietterich, T. G. (1995). A Reinforcement Learning Approach to Job-shop Scheduling. In *1995 International Joint Conference on Artificial Intelligence* (pp. 1114–1120).

Zhang, W., & Dietterich, T. G. (1996). High-performance job-shop scheduling with a time-delay TD (λ) network. *Advances in Neural Information Processing Systems*, *91*, 1024–1030.

Zhang, Z., Zheng, L., Hou, F., & Li, N. (2011). Semiconductor final test scheduling with Sarsa(λ, k) algorithm. *European Journal of Operational Research*, *215*(2), 446–458. https://doi.org/10.1016/j.ejor.2011.05.052

Zhang, Z., Zheng, L., Li, N., Wang, W., Zhong, S., & Hu, K. (2012). Minimizing mean weighted tardiness in unrelated parallel machine scheduling with reinforcement learning. *Computers and Operations Research*, *39*(7), 1315–1324. https://doi.org/10.1016/j.cor.2011.07.019

Zhang, Z., Zheng, L., & Weng, M. X. (2007). Dynamic parallel machine scheduling with mean weighted tardiness objective by Q-Learning. *International Journal of Advanced Manufacturing Technology*, *34*(9–10), 968–980. https://doi.org/10.1007/s00170-006-0662-8

Zhao, M., Li, X., Gao, L., Wang, L., & Xiao, M. (2019). An improved q-learning based rescheduling method for flexible job-shops with machine failures. *IEEE International Conference on Automation Science and Engineering*, *2019-Augus*, 331–337. https://doi.org/10.1109/COASE.2019.8843100

Zhou, L., Zhang, L., & Horn, B. K. P. (2020). Deep reinforcement learning-based dynamic scheduling in smart manufacturing. *Procedia CIRP*, *93*, 383–388. https://doi.org/10.1016/j.procir.2020.05.163

Zhou, T., Tang, D., Zhu, H., & Wang, L. (2021a). Reinforcement Learning with Composite Rewards for Production Scheduling in a Smart Factory. *IEEE Access*, *9*, 752–766. https://doi.org/10.1109/ACCESS.2020.3046784

Zhou, T., Tang, D., Zhu, H., & Zhang, Z. (2021b). Multi-agent reinforcement learning for online scheduling in smart factories. *Robotics and Computer-Integrated Manufacturing*, *72*. https://doi.org/10.1016/j.rcim.2021.102202

APPENDICES

Appendix A1. Matlab code for deterministic UPMSP

```
clearvars
clc

n=8; % number of jobs
m=2;  % number of machines
run_numb=2000;  %total episode number
alpha=0.1;
gamma=0.9;
run=5;
makespan=zeros(run_numb,1);
epsilon=0.9;
order=zeros(1,10);
for j1=1:run
clearvars -except pt st j1 makespan
n=8; % number of jobs
m=2;  % number of machines
run_numb=2000;  %total episode number
alpha=0.1;
gamma=0.9;
Qvalues=zeros(n);
dummy_Qvalues=zeros(n);
epsilon=0.9;
order=zeros(1,n);

for i1=1:run_numb

 if i1<=2
        random order
        assign_machine
        initialCmax

 else

    if epsilon>rand()
        random order

    else
        maxQvalue_order

    end
    assign_machine
    calculate_reward
    update_Qvalues
    epsilon=epsilon*(0.99^(i1/(run_numb*1)));
    if epsilon<=0.1
        epsilon=0.1;
    end
 end
   makespan(i1,j1)=Cmax;
end
end
bestmakespan=min(makespan);
```

random order

```
order=randperm(n);
```

assign machine

```
compTime=zeros(1,m);
dummy_compTime=zeros(1,m);
prev_jobM=zeros(1,m);
assigned_jobs=zeros(m,n);
for i3=1:n
for i4=1:m
    if prev_jobM(i4)==0
        dummy_compTime(i4)=pt(order(i3),i4);
    else
        dummy_compTime(i4)= compTime(i4)+ pt(order(i3),i4)...
        +st((prev_jobM(i4)+(n)*(i4-1)),order(i3));
    end

end
[A,B]=min(dummy_compTime);
    prev_jobM(B)=order(i3);
    compTime(B)=dummy_compTime(B);
    assigned_jobs(B,i3)=order(i3);

end
Cmax=max(compTime);
```

initialCmax

```
if i1==1
    bestCmax=Cmax;
    worstCmax=Cmax;
else
    if Cmax<bestCmax
        bestCmax=Cmax;
    end
    if Cmax>worstCmax
            worstCmax=Cmax;
    end
end
```

maxQvalue order

```
for i2=1:n
    A=dummy_Qvalues(i2,:);
    [M,I]=max(A) ;
    order(1,i2)=I  ;
    dummy_Qvalues(:,I)=-1000;

end
```

calculate reward

```
reward=(worstCmax-Cmax)/(worstCmax-bestCmax);
 if Cmax<bestCmax
        bestCmax=Cmax;

 end
    if Cmax>worstCmax
           worstCmax=Cmax;

    end
```

update Qvalues

```
for i5=1:n
    if i5<n
        Qvalues(i5,order(i5))=(1-
alpha)*Qvalues(i5,order(i5))+alpha*(reward+(gamma)*max(Qvalues((i5+1),:)))
;
    else
        Qvalues(i5,order(i5))=(1-
alpha)*Qvalues(i5,order(i5))+alpha*(reward);
    end
end
dummy_Qvalues=Qvalues;
```

Appendix A2. Matlab code for deterministic HFSSP

```
clear all
run=10;
n=50;
eps_numb=80000;
alpha=0.1;
gamma=0.9;
bestCmaxofEps=zeros(1,run);
Ran_Max=zeros(eps_numb,3);
load('50_5_2.mat');

for j1=1:run
Qvalues=zeros(n);
dummy_Qvalues=zeros(n);
Qselect=zeros(n);
epsilon=0.9;
bestorder=zeros(1,n);
ro=0; mo=0;
for i1=1:eps_numb

 if i1<=2
        random_order2
        AssignMachine_Identical
        initialCmax

 else

    if epsilon>rand()
        random_order2
        ro=ro+1;

    else
        maxQvalue_order
        mo=mo+1;

    end
    AssignMachine_Identical
    calculate_reward
    update_Qvalues
    epsilon=epsilon*(0.99^(i1/(eps_numb*100)));
    if epsilon<=0.1
        epsilon=0.1;
    end
 end
 Ran_Max(i1,1)=i1;
 Ran_Max(i1,2)=ro;
 Ran_Max(i1,3)=mo;
bestsofar(i1,j1)=bestCmax
makespan(i1,j1)=Cmax;
end
bestCmaxofEps(1,j1)=bestCmax;
end
```

random_order2

```
order=randperm(n);
order1=order.';
```

AssignMachine_Identical

```
s=5; m=[3,4,4,4,5];

order_CT=[order1 zeros(n,1)];
d_order_CT=[order1 zeros(n,1)];
for x=1:s
    CT=zeros(1,m(x));
    d_CT=zeros(1,m(x));
    for y=1:n
        [A,B]=min(d_CT);
        d_CT(B)=max(d_order_CT(y,2),CT(B))+pt(order1(y),x);
        CT(B)=d_CT(B);
        order_CT(y,2)=d_CT(B);
    end
    d_order_CT=sortrows(order_CT,2);
    order1=d_order_CT(:,1);
    order_CT(:,1)=order1;
end
Cmax=max(CT);
```

initialCmax

```
if i1==1
    bestCmax=Cmax;
    worstCmax=Cmax;
else
    if Cmax<bestCmax
        bestCmax=Cmax;
    elseif Cmax==worstCmax
           worstCmax=Cmax+1;

    elseif Cmax>worstCmax
            worstCmax=Cmax;
    end

end
```

maxQvalue_order

```
order=zeros(1,n);

for i2=1:n

    vectorA=dummy_Qvalues(i2,:);
    [M,I]=max(vectorA)
    order(1,i2)=I  ;
    dummy_Qvalues(:,I)=-1000;

end
order1=order.';
```

calculate_reward

```
reward=(worstCmax-Cmax)/(worstCmax-bestCmax);

 if Cmax<=bestCmax
        bestCmax=Cmax;
        bestorder=order;
 end
     if Cmax>worstCmax
           worstCmax=Cmax;

    end
```

update Qvalues

```
for i5=1:n
    if i5<n
        Qvalues(i5,order(i5))=(1-
alpha)*Qvalues(i5,order(i5))+alpha*(reward+(gamma)*max(Qvalues((i5+1),:)))
;
    else
        Qvalues(i5,order(i5))=(1-
alpha)*Qvalues(i5,order(i5))+alpha*(reward);
    end
    Qselect(i5,order(i5))=Qselect(i5,order(i5))+1;

end
dummy_Qvalues=Qvalues;
```

Appendix A3. ARENA model of the developed simulation model for Problem1.1

Mod file of the ARENA model

```
0$              CREATE,         1:Expo(Arrival,1):MARK(TimeIn):NEXT(39$);

39$             ASSIGN:         i3=1:
                                dd=TNOW:
                                ddQ(i3)=TNOW:
                                TW=0;
40$             WHILE:          i3<=totalstagenumber;
38$             ASSIGN:         ept(i3)=Util(i3):
                                dd=dd+TWK*ept(i3):
                                ddQ(i3)=dd:
                                TW=TW+ept(i3):
                                RW=TW:
                                i3=i3+1;
41$             ENDWHILE;
1$              ASSIGN:         s_number=s_number+1:
                                seizedmachine=0:
                                pt=ept(s_number):
                                PRW=pt/RW:
                                min_util=5:
                                NR_s=1:
                                entrytime_to_s(s_number)=TNOW;
2$              BRANCH,         1:
                                If,NQ(s_number)≥1,3$,Yes:
                                Else,9$,Yes;
3$              QUEUE,          QueuesHFS(s_number):DETACH;
9$              ASSIGN:         i2=control((2*s_number-1));
4$              WHILE:          i2<=control(2*s_number);
5$              BRANCH,         1:
                                If,(NR(i2)==0).AND.(DAVG(i2)<min_util),7$,Yes:
                                Else,8$,Yes;
7$              ASSIGN:         seizedmachine=i2:
                                min_util=DAVG(i2);
8$              ASSIGN:         i2=i2+1;
6$              ENDWHILE;
36$             BRANCH,         1:
                                If,seizedmachine==0,3$,Yes:
                                Else,10$,Yes;
10$             SEIZE,          1,Other:
                                seizedmachine,1:NEXT(11$);

11$             DELAY:          pt,,Other:NEXT(12$);

12$             RELEASE:        seizedmachine,1;
13$             ASSIGN:         RW=RW-pt;
14$             BRANCH,         1:
                                If,NQ(s_number)==1,16$,Yes:
                                If,NQ(s_number)≥2,32$,Yes:
                                Else,17$,Yes;
16$             ASSIGN:         sm_s=seizedmachine;
15$             REMOVE:         1,QueuesHFS(s_number),SeizeMachine;
17$             BRANCH,         1:
                                If,s_number<totalstagenumber,1$,Yes:
                                Else,18$,Yes;
18$             COUNT:          completedjobs,1;
24$             ASSIGN:         Tardiness=MX(TNOW-dd,0):
                                Earliness=MX(dd-TNOW,0):
                                WaitingTime=TNOW-TimeIn-TW;
22$             TALLY:          FlowTime,INT(TimeIn),1;
```

```
25$           TALLY:         Tardinessofjobs,Tardiness,1;
26$           TALLY:         Earlinessofjobs,Earliness,1;
27$           TALLY:         WaitingTimesofjobs,WaitingTime,1;
20$           BRANCH,        1:
                             If,TNOW>dd,21$,Yes:
                             Else,19$,Yes;
21$           COUNT:         TardyJobs,1;
23$           ASSIGN:        TotalTardiness=TotalTardiness+(TNOW-dd):
                             MeanTardiness=TotalTardiness/NC(1);
37$           WRITE,         Mean_tardiness_:
                             MeanTardiness;
19$           DISPOSE:       No;

SeizeMachine  ASSIGN:        seizedmachine=sm_s:NEXT(10$);

32$           ASSIGN:        JinQ=NQ(s_number);
28$           WHILE:         NQ(s_number)>0;
29$           REMOVE:        1,QueuesHFS(s_number),AssignDR;
30$           ENDWHILE;
31$           ASSIGN:        jQ=0;
33$           WAIT:          s_number;
53$           BRANCH,        1:
                             With,0.5,52$,Yes:
                             Else,55$,Yes;
52$           COUNT:         DR1,1;
51$           SEARCH,        QueuesHFS(s_number),1,NQ(s_number):MIN(A(3));
50$           ASSIGN:        sm_s=seizedmachine;
49$           REMOVE:        j,QueuesHFS(s_number),SeizeMachine:NEXT(17$);

55$           COUNT:         DR2,1;
54$           SEARCH,
QueuesHFS(s_number),1,NQ(s_number):MIN(A(10)):NEXT(50$);

AssignDR      ASSIGN:        slack=dd-RW-TNOW:
                             jQ=jQ+1:
                             S_RW=slack/RW:
                             CR=(dd-TNOW)/RW:
                             MDD=MX(dd,TNOW+RW):
                             WT=TNOW-entrytime_to_s(s_number):
                             HRN=(WT+pt)/pt:
                             S_RPT=MX(S_RW*pt, pt):
                             CR_RPT=MX(CR*pt, pt):
                             AT_RPT=TimeIn-RW:
                             SL=MIN(slack,0):
                             TIS=TNOW-TimeIn:
                             PT_TIS=pt/((TIS<>0)*TIS+(TIS==0)*d_WT):
                             s_Lateness=TNOW-ddQ(s_number):
                             s_tardiness=MX(TNOW-ddQ(s_number),0);
42$           BRANCH,        1:
                             If,s_number<totalstagenumber,43$,Yes:
                             Else,44$,Yes;
43$           ASSIGN:        WINQ=SAQUE(s_number+1,1):
                             PT_WINQ=pt+WINQ:
                             PT_WINQ_AT=pt+WINQ+TimeIn:
                             PT_WINQ_SL=pt+WINQ+SL:

PT_WINQ_TIS=PT_WINQ/((TIS<>0)*TIS+(TIS==0)*d_WT):
                             EWT=((JinQ*ept(s_number))/#ofMachS(s_number)):
                             i4=s_number;
46$           WHILE:         i4<totalstagenumber;
45$           ASSIGN:        EWT=EWT+((NQ(i4+1)*ept(i4+1))/#ofMachS(i4+1)):
                             i4=i4+1;
48$           ENDWHILE;
47$           ASSIGN:        COVERT=(((EWT-
slack)/EWT)*((0<=slack).AND.(slack<EWT))+0*(slack≥EWT)+1*(slack<0))/pt;
```

```
34$             BRANCH,         1:
                                If,jQ==JinQ,35$,Yes:
                                Else,3$,Yes;
35$             SIGNAL:         s_number:NEXT(3$);

44$             ASSIGN:         WINQ=0:
                                PT_WINQ=pt+WINQ:
                                PT_WINQ_AT=pt+WINQ+TimeIn:
                                PT_WINQ_SL=pt+WINQ+SL:

PT_WINQ_TIS=PT_WINQ/((TIS<>0)*TIS+(TIS==0)*d_WT):
                                EWT=((JinQ*ept(s_number))/#ofMachS(s_number)):
                                COVERT=(((EWT-
slack)/EWT)*((0<=slack).AND.(slack<EWT))+0*(slack≥EWT)+1*(slack<0))/pt:NEXT(
34$);
```

Exp file of the ARENA model

```
PROJECT,        "Unnamed Project","Rockwell
Automation",,,No,Yes,Yes,Yes,No,No,No,No,No,No;

ATTRIBUTES:     1,pt,DATATYPE(Real):
                2,TimeIn,DATATYPE(Real):
                3,RW,DATATYPE(Real):
                4,TW,DATATYPE(Real):
                5,dd,DATATYPE(Real):
                6,PRW,DATATYPE(Real):
                7,slack,DATATYPE(Real):
                8,S_RW,DATATYPE(Real):
                9,CR,DATATYPE(Real):
                10,MDD,DATATYPE(Real):
                11,AT_RPT,DATATYPE(Real):
                12,PT_WINQ,DATATYPE(Real):
                13,PT_WINQ_AT,DATATYPE(Real):
                14,PT_WINQ_SL,DATATYPE(Real):
                15,PT_WINQ_TIS,DATATYPE(Real):
                16,PT_TIS,DATATYPE(Real):
                17,HRN,DATATYPE(Real):
                18,S_RPT,DATATYPE(Real):
                19,CR_RPT,DATATYPE(Real):
                20,COVERT,DATATYPE(Real):
                21,s_number,DATATYPE(Real):
                22,ept(2),DATATYPE(Real):
                24,ddQ(2),DATATYPE(Real):
                26,i1,DATATYPE(Real):
                27,seizedmachine,DATATYPE(Real):
                28,Tardiness,DATATYPE(Real):
                29,Earliness,DATATYPE(Real):
                30,WaitingTime,DATATYPE(Real):
                31,entrytime_to_s(2),DATATYPE(Real):
                33,WT,DATATYPE(Real):
                34,SL,DATATYPE(Real):
                35,TIS,DATATYPE(Real):
                36,WINQ,DATATYPE(Real):
                37,EWT,DATATYPE(Real):
                38,s_Lateness,DATATYPE(Real):
                39,s_tardiness,DATATYPE(Real):
                42,i2,DATATYPE(Real):
                i3,DATATYPE(Real):
                i4,DATATYPE(Real);
```

```
FILES:
2,TotalLatenesofQ,"C:\Users\BMK\Dropbox\2StagesHFS_GM\latenes.txt",Sequentia
l,Free Format,Error,No,Close:

3,MeanTardofQ,"C:\Users\BMK\Dropbox\2StagesHFS_GM\meantardinessofQ.txt",Sequ
ential,Free Format,Error,No,Close:

8,epsilonvalue,"C:\Users\BMK\Dropbox\2StagesHFS_GM\epsilonvalue.txt",Sequent
ial,Free Format,Error,No,Close:

9,Qvalue_GM,"C:\Users\BMK\Dropbox\2StagesHFS_GM\Qvalue_controlGM.txt",Sequen
tial,Free Format,Error,No,Close:

12,reward_val,"C:\Users\BMK\Dropbox\2StagesHFS_GM\rewardvalue.txt",Sequentia
l,Free Format,Error,No,Close:

Mean_tardiness_,"C:\Users\BMK\Dropbox\2StagesHFS_GM\meantardiness.txt",Seque
ntial,Free Format,Error,No,Close:

MeanTardofQ2,"C:\Users\BMK\Dropbox\2StagesHFS_GM\meantardinessofQ2.txt",Sequ
ential,Free Format,Error,No,Close:

latenesQ2,"C:\Users\BMK\Dropbox\2StagesHFS_GM\latenes2.txt",Sequential,Free
Format,Error,No,Close;

VARIABLES:    1,Arrival,CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),5:
              2,TWK,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real),1.5:
              3,control(4),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),1,5,6,8:
              7,min_util,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
              8,NR_s,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
              9,sm_s,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
              10,totalstagenumber,CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),2:
              11,TotalTardiness,CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              12,MeanTardiness,CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              13,JinQ,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
              14,jQ,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
              15,#ofMachS(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),5,3:
              146,DR,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real),1:
              156,d_WT,CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),0.0000000001:
              159,i5,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real);

SEEDS:        1,,Common:
              2,,Common:
              3,,Common:
              4,,Common:
              5,,Common:
              6,,Common:
              7,,Common:
              8,,Common:
              9,,Common:
              10,,Common:
              11,35698572,Common:
              13,478570584,Common;

QUEUES:       1,queue1,FirstInFirstOut,,AUTOSTATS(No,,):
              2,queue2,FirstInFirstOut,,AUTOSTATS(No,,);
```

```
RESOURCES:
5,Machine,Capacity(1),,Stationary,COST(0.0,0.0,0.0),,AUTOSTATS(No,,):

8,Machine2,Capacity(1),,Stationary,COST(0.0,0.0,0.0),,AUTOSTATS(No,,);

COUNTERS:      1,completedjobs,,Replicate:
               2,TardyJobs,,Replicate:
               3,DR1,,Replicate:
               4,DR2,,Replicate;

TALLIES:       1,FlowTime,"flowtime.dat":
               2,Tardinessofjobs,"Tardiness.dat":
               3,Earlinessofjobs,"Earliness.dat":
               4,WaitingTimesofjobs,"WaitingTimes.dat";

DSTATS:        1,NR(1):
               2,NR(2):
               3,NR(3):
               4,NR(4):
               5,NR(5):
               6,NR(6):
               7,NR(7):
               8,NR(8):

9,(NR(1)+NR(2)+NR(3)+NR(4)+NR(5))/5,Stage1Utilization,"Stage1Util.dat":
               10,(NR(6)+NR(7)+NR(8))/3,Stage2Utilization,"Stage2Util.dat":
               11,NQ(1),,"WaitinginQ1.dat":
               12,NQ(2),,"WaitinginQ2.dat";

OUTPUTS:       1,TAVG(1),,MeanFlowTime:
               2,TMAX(1),,MaxFlowTime:
               3,TAVG(2),,MeanTardiness:
               4,TMAX(2),,MaxTardiness:
               5,TAVG(3),,MeanEarliness:
               6,TMAX(3),,MaxEarliness:
               7,TAVG(4),,MeanWaitingTime:
               8,TMAX(4),,MaxWaitingTime:
               9,NC(2),,NumberofTardyJobs:
               10,NC(2)/NC(1),,ProportionofTardyJobs;

REPLICATE,     20,0.0,165000,Yes,Yes,15000,,,24.0,Hours,No,No,,,No,No;

EXPRESSIONS:   1,Util(2),DATATYPE(Native),UNIF(10,32.5,4),UNIF(10,15,8);

SETS:          1,QueuesHFS,queue1,queue2;
```

Appendix A4. The process time distributions of the problems

Table A4.1 The process time distributions of Problem 1.2

		Stage Number	
		1	2
Utilization Level	%95	UNIF(20, 27.5)	UNIF(12, 16.5)
	%90	UNIF(20, 25)	UNIF(10, 17)
	%85	UNIF(19, 23.5)	UNIF(11, 14)

Table A4.2 The process time distributions of Problem 2.1

		Stage Number				
		1	2	3	4	5
Utilization Level	%95	UNIF(5,14)	UNIF(5,23.5)	UNIF(1,8.5)	UNIF(5,33)	UNIF(5,14)
	%90	UNIF(5,13)	UNIF(5,22)	UNIF(1,8)	UNIF(5,31)	UNIF(5,13)
	%85	UNIF(5,12)	UNIF(5,20.5)	UNIF(1,7.5)	UNIF(5,29)	UNIF(5,12)

Table A4.3 The process time distributions of Problem 2.2

		Stage Number				
		1	2	3	4	5
Utilization Level	%95	UNIF(5,42.5)	UNIF(5,14)	UNIF(5,42.5)	UNIF(5,23.5)	UNIF(1,8.5)
	%90	UNIF(5,40)	UNIF(5,13)	UNIF(5,40)	UNIF(5,22)	UNIF(1,8)
	%85	UNIF(5,37.5)	UNIF(5,12)	UNIF(5,37.5)	UNIF(5,20.5)	UNIF(1,7.5)

Table A4.4 The process time distributions of Problem 3.1

		Stage Number				
		1	2	3	4	5
Utilization Level	%95	UNIF(5,23.5)	UNIF(1,8.5)	UNIF(5,14)	UNIF(5,14)	UNIF(5,23.5)
	%90	UNIF(5,22)	UNIF(1,8)	UNIF(5,13)	UNIF(5,13)	UNIF(5,22)
	%85	UNIF(5,20.5)	UNIF(1,7.5)	UNIF(5,12)	UNIF(5,12)	UNIF(5,20.5)
		Stage Number				
		6	7	8	9	
Utilization Level	%95	UNIF(5,42.5)	UNIF(5,23.5)	UNIF(5,33)	UNIF(5,42.5)	
	%90	UNIF(5,40)	UNIF(5,22)	UNIF(5,31)	UNIF(5,40)	
	%85	UNIF(5,37.5)	UNIF(5,20.5)	UNIF(5,29)	UNIF(5,37.5)	

Table A4.5 The process time distributions of Problem 3.2

		Stage Number				
		1	2	3	4	5
Utilization Level	%95	UNIF(5,33)	UNIF(5,23.5)	UNIF(1,8.5)	UNIF(5,14)	UNIF(5,42.5)
	%90	UNIF(5,31)	UNIF(5,22)	UNIF(1,8)	UNIF(5,13)	UNIF(5,40)
	%85	UNIF(5,29)	UNIF(5,20.5)	UNIF(1,7.5)	UNIF(5,12)	UNIF(5,37.5)
		Stage Number				
		6	7	8	9	
Utilization Level	%95	UNIF(1,8.5)	UNIF(5,42.5)	UNIF(5,23.5)	UNIF(5,42.5)	
	%90	UNIF(1,8)	UNIF(5,40)	UNIF(5,22)	UNIF(5,40)	
	%85	UNIF(1,7.5)	UNIF(5,37.5)	UNIF(5,20.5)	UNIF(5,37.5)	

Appendix A5. The performances of the dispatching rules

The performances of the dispatching rules for problem 1.1

Table A5.1 The performances of DRs in 95U and LDD for Problem 1.1

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	88.923	8676.2	38.494	8602.5	6.5953	27.736	50.907	8627	11158
FCFS	103.77	371.9	49.851	327.19	3.0975	27.722	65.762	340.94	22489
LWKR	87.923	8607.9	37.577	8530.6	6.6778	27.749	49.907	8556.4	10961
LTWF	89.361	9171.8	39.149	9089.4	6.8118	27.756	51.345	9116.9	10202
EDD	101.12	382.18	47.093	310.35	2.9975	27.758	63.104	332.04	22110
PRW	97.622	9110.1	47.416	9044.9	6.8189	27.755	59.606	9066.7	11625
SLACK	102.8	381.82	48.617	310.67	2.8364	27.723	64.789	332.7	22571
CR	100.14	438.46	45.979	365.43	2.8612	27.724	62.126	388.63	22645
MDD	89.117	8181	35.668	8104.7	3.575	27.751	51.101	8130.2	17532
AT-RPT	105.83	372.57	52.023	328.88	3.2164	27.725	67.815	341.88	22687
P+W+A	101.79	372.18	47.87	322.15	3.1035	27.733	63.775	336.95	22320
P+W+S	100.77	381.53	47.092	309.68	3.3397	27.752	62.761	331.43	21981
(P+W)/T	99.194	428.53	45.319	368.94	3.1507	27.721	61.177	387.75	22183
PT/TIS	98.496	449.8	44.624	383.64	3.1528	27.718	60.479	404.99	22173
LPT	172.28	12112	122.4	12077	7.1535	27.683	134.26	12089	13066
MWKR	159.76	11201	110.26	11168	7.5284	27.702	121.74	11179	12420
MTWF	155.58	11989	106.87	11957	8.316	27.711	117.56	11968	11246
HRN	99.059	492.25	45.044	413.5	3.01	27.744	61.042	439.59	22360
(S/RW)+S	172.04	12246	122.58	12211	7.5653	27.678	134.02	12223	12839
CR+SPT	169.3	12160	121.03	12127	8.7611	27.632	131.28	12138	10490
COVERT	90.352	7958.9	36.988	7887.7	3.66	27.721	52.336	7911.4	17029

Table A5.2 The performances of DRs in 90U and TDD for Problem 1.1

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	57.978	2167.4	15.826	2111.5	2.85	12.923	21.974	2122.7	0.50405
FCFS	62.342	206.15	19.497	167.33	2.16	12.932	26.338	174.15	0.67684
LWKR	57.812	2149.5	15.67	2090	2.86	12.922	21.808	2101.9	0.49996
LTWF	58.346	2161.2	16.197	2097.9	2.86	12.922	22.343	2110.6	0.48825
EDD	61.022	216.69	18.108	159.99	2.09	12.92	25.018	170.54	0.67418
PRW	60.133	2301.4	18.04	2249	2.91	12.92	24.13	2259.5	0.50713
SLACK	62.032	213.6	19.037	159.52	2.01	12.922	26.028	169.16	0.69172
CR	61.286	240.43	18.306	180.96	2.03	12.927	25.282	192.67	0.69098
MDD	57.996	2122.3	15.311	2063.1	2.32	12.923	21.992	2074.9	0.58425
AT-RPT	63.743	206.31	20.995	171	2.26	12.929	27.739	177.42	0.66832
P+W+A	61.218	211.11	18.409	164.52	2.20	12.918	25.214	172.21	0.66927
P+W+S	60.908	216.32	18.098	159.37	2.19	12.92	24.904	170.03	0.66743
(P+W)/T	60.996	238.21	18.156	187.78	2.16	12.923	24.992	197.16	0.67388
PT/TIS	60.808	245.47	17.97	191.2	2.17	12.927	24.804	201.26	0.67347
LPT	78.871	2962.4	36.786	2934.3	2.92	12.925	42.866	2940	0.52087
MWKR	77.18	2991.3	35.195	2963.9	3.02	12.929	41.175	2969.3	0.50558
MTWF	76.132	3212.5	34.371	3186	3.24	12.928	40.127	3191.3	0.47338
HRN	60.972	263.01	18.054	203.88	2.09	12.929	24.968	215.58	0.67936
(S/RW)+S	78.871	2962.4	36.786	2934.3	2.92	12.925	42.866	2940	0.52085
CR+SPT	78.557	2981.9	36.881	2953.6	3.33	12.924	42.552	2959.3	0.47162
COVERT	58.339	2134.4	15.568	2075.7	2.23	12.927	22.336	2087.5	0.60292

Table A5.3 The performances of DRs in 90U and LDD for Problem 1.1

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	57.978	2167.4	12.349	2100.3	8.3767	25.847	21.974	2122.7	0.26886
FCFS	62.342	206.15	14.331	160.81	5.9946	25.864	26.338	174.15	0.51705
LWKR	57.812	2149.5	12.22	2078.1	8.4139	25.846	21.808	2101.9	0.26567
LTWF	58.346	2161.2	12.798	2085.3	8.4573	25.846	22.343	2110.6	0.25061
EDD	60.825	219.27	12.692	150.87	5.8731	25.841	24.821	171.85	0.49339
PRW	60.133	2301.4	14.563	2238.5	8.4354	25.841	24.13	2259.5	0.28587
SLACK	61.726	215.66	13.392	149.82	5.6719	25.846	25.722	169.73	0.51209
CR	61.317	240.36	12.978	170.11	5.6673	25.855	25.313	192.65	0.51915
MDD	58.451	1982.7	10.671	1911.9	6.2255	25.841	22.447	1935.5	0.40077
AT-RPT	63.743	206.31	15.882	165.01	6.1452	25.859	27.739	177.42	0.53065
P+W+A	61.218	211.11	13.199	157.72	5.9875	25.838	25.214	172.21	0.50825
P+W+S	60.561	218.58	12.89	150.58	6.3354	25.841	24.557	171.37	0.49309
(P+W)/T	60.996	238.21	12.99	178.65	6.0001	25.853	24.992	197.16	0.50702
PT/TIS	60.808	245.47	12.798	181.36	5.9965	25.861	24.804	201.26	0.50731
LPT	78.871	2962.4	33.503	2928.7	8.6397	25.852	42.866	2940	0.33003
MWKR	77.18	2991.3	32.033	2958.4	8.8601	25.859	41.175	2969.3	0.31825
MTWF	76.132	3212.5	31.522	3180.6	9.3965	25.859	40.127	3191.3	0.29425
HRN	60.972	263.01	12.803	192.19	5.8374	25.858	24.968	215.58	0.51097
(S/RW)+S	78.784	2999	33.688	2965.2	8.9106	25.849	42.779	2976.4	0.32776
CR+SPT	77.842	3080.8	33.634	3047.5	9.7986	25.856	41.837	3058.6	0.2717
COVERT	59.006	2016	11.022	1950.2	6.0216	25.842	23.002	1972	0.42142

Table A5.4 The performances of DRs in 85U and TDD for Problem 1.1

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	45.66	1119.4	7.1645	1062.8	3.7012	11.832	11.902	1074.1	0.37941
FCFS	47.575	150.23	8.5511	108.18	3.1726	11.833	13.817	115.09	0.51038
LWKR	45.649	1159.3	7.1596	1101.2	3.7075	11.832	11.891	1112.8	0.37741
LTWF	45.805	1166.1	7.2975	1108	3.6891	11.832	12.047	1119.6	0.37019
EDD	46.786	159.26	7.7221	105.73	3.1322	11.833	13.029	115.91	0.4999
PRW	46.203	1153.8	7.7293	1102	3.7225	11.831	12.446	1112.3	0.3846
SLACK	47.358	153.54	8.2096	104	3.0485	11.836	13.6	112.75	0.51902
CR	47.136	171.91	7.9952	116.47	3.0561	11.835	13.378	127.47	0.51974
MDD	45.766	1101.6	6.8664	1043.6	3.2974	11.832	12.008	1055.2	0.43766
AT-RPT	48.455	149.77	9.5526	114.43	3.2939	11.834	14.698	120.66	0.50022
P+W+A	46.902	156.2	7.9205	105.46	3.2144	11.833	13.145	114.83	0.50027
P+W+S	46.716	158.32	7.7637	105.02	3.2444	11.832	12.958	115.05	0.49324
(P+W)/T	47.008	174.36	7.9776	120.61	3.1656	11.835	13.251	131.26	0.50865
PT/TIS	46.927	182.6	7.8988	127.75	3.1686	11.832	13.169	138.68	0.50819
LPT	53.551	1288.5	15.086	1259.7	3.7304	11.831	19.795	1265.5	0.40157
MWKR	53.27	1347.1	14.854	1317.9	3.7801	11.832	19.513	1323.7	0.39395
MTWF	53.016	1418.7	14.742	1391.4	3.9224	11.831	19.259	1396.8	0.37419
HRN	46.959	185.65	7.8771	131.09	3.1147	11.834	13.201	141.99	0.50925
(S/RW)+S	53.551	1288.5	15.086	1259.7	3.7304	11.831	19.795	1265.5	0.40157
CR+SPT	53.414	1311.3	15.247	1282	4.0294	11.832	19.657	1287.8	0.36722
COVERT	45.934	1010.5	6.9277	953.83	3.1904	11.834	12.176	965.18	0.46062

Table A5.5 The performances of DRs in 85U and LDD for Problem 1.1

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	45.66	1119.4	4.8515	1051.4	9.8276	23.665	11.902	1074.1	0.17799
FCFS	47.575	150.23	5.2325	101.75	8.2933	23.668	13.817	115.09	0.32597
LWKR	45.649	1159.3	4.8554	1089.6	9.8428	23.666	11.891	1112.8	0.17693
LTWF	45.805	1166.1	5.0216	1096.4	9.8526	23.666	12.047	1119.6	0.16855
EDD	46.686	159.75	4.3378	95.541	8.288	23.668	12.928	116.18	0.29376
PRW	46.203	1153.8	5.3907	1091.6	9.8233	23.663	12.446	1112.3	0.18947
SLACK	47.178	156.1	4.6389	95.08	8.0963	23.671	13.421	114.16	0.30977
CR	47.146	172.18	4.5844	105.88	8.0737	23.671	13.389	127.76	0.31897
MDD	45.998	992.96	3.8695	923.54	8.5073	23.668	12.241	946.68	0.2431
AT-RPT	48.455	149.77	6.2839	108.25	8.4646	23.67	14.698	120.66	0.3437
P+W+A	46.902	156.2	4.5708	97.006	8.3042	23.668	13.145	114.83	0.31318
P+W+S	46.497	158.76	4.5779	95.488	8.7165	23.666	12.74	115.7	0.29645
(P+W)/T	47.008	174.36	4.646	110.09	8.2734	23.672	13.251	131.26	0.31751
PT/TIS	46.927	182.6	4.5632	116.85	8.2724	23.666	13.169	138.68	0.31733
LPT	53.551	1288.5	12.828	1254	9.9124	23.663	19.795	1265.5	0.23292
MWKR	53.27	1347.1	12.647	1312	10.013	23.665	19.513	1323.7	0.22785
MTWF	53.016	1418.7	12.698	1385.9	10.317	23.666	19.259	1396.8	0.21646
HRN	46.959	185.65	4.5169	120.19	8.1939	23.669	13.201	141.99	0.31555
(S/RW)+S	53.526	1304	12.947	1270.7	10.056	23.659	19.769	1281.8	0.23424
CR+SPT	53.145	1371.5	13.15	1337.4	10.64	23.665	19.388	1348.8	0.19947
COVERT	46.327	898.01	3.9477	831.71	8.2562	23.671	12.57	853.81	0.27088

The performances of the dispatching rules for problem 1.2

Table A5.6 The performances of DRs in 95U and LDD for Problem 1.2

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	85.578	8390.1	37.082	8330.2	8.5115	21.924	47.573	8350.2	0.27217
FCFS	90.231	325.79	36.841	269.62	3.6173	21.926	52.226	287.8	0.68745
LWKR	85.475	8136.9	37.046	8071.7	8.5789	21.925	47.47	8093.4	0.26624
LTWF	85.531	8277.7	37.118	8212.4	8.5948	21.925	47.526	8234.2	0.26569
EDD	90.012	328.8	36.669	268.4	3.6659	21.926	52.006	288.12	0.68409
PRW	89.617	7993.4	40.931	7938.4	8.3215	21.927	51.612	7956.8	0.30886
SLACK	90.169	326.85	36.775	268.32	3.6133	21.924	52.164	287.38	0.68631
CR	89.914	356.27	36.518	294.18	3.6126	21.925	51.908	314.84	0.68685
MDD	87.401	7396.5	33.198	7331.5	3.8042	21.927	48.396	7353.2	0.61106
AT-RPT	90.366	325.21	37.031	270.93	3.6727	21.926	52.361	288.62	0.68952
P+W+A	89.994	326.53	36.624	268.13	3.6388	21.924	51.988	287.09	0.68508
P+W+S	89.809	327.89	36.866	268.12	4.0658	21.926	51.803	287.67	0.69348
(P+W)/T	89.532	349.45	36.178	288.4	3.654	21.924	51.526	308.65	0.68329
PT/TIS	89.605	352.76	36.247	291.45	3.6497	21.924	51.599	311.64	0.68372
LPT	98.773	8739.8	48.729	8685.2	6.9643	21.922	60.767	8703.4	0.38009
MWKR	95.391	8246.7	46.188	8197.8	7.8057	21.925	57.385	8214.1	0.33972
MTWF	95.171	8450.7	46.19	8401.9	8.0284	21.927	57.165	8418.1	0.32807
HRN	89.83	357.36	36.47	296.3	3.6486	21.925	51.824	316.45	0.68421
(S/RW)+S	98.773	8739.8	48.729	8685.2	6.9643	21.922	60.767	8703.4	0.38009
CR+SPT	96.869	8587.1	48.713	8534.2	8.8525	21.925	58.863	8551.8	0.28203
COVERT	86.731	7316	33.631	7255.2	3.908	21.927	48.726	7275.5	0.59248

Table A5.7 The performances of DRs in 90U and TDD for Problem 1.2

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	56.269	2150.9	14.602	2102.2	3.3354	10.475	20.267	2111.9	0.40809
FCFS	58.411	188.62	15.667	144.03	2.2583	10.472	22.409	152.75	0.64546
LWKR	56.148	2219.4	14.525	2167.8	3.3782	10.475	20.147	2178.2	0.40067
LTWF	56.238	2236.3	14.647	2184.6	3.4113	10.475	20.236	2194.9	0.39635
EDD	58.141	192.19	15.441	144.02	2.302	10.473	22.139	153.57	0.63934
PRW	57.032	2182.6	15.378	2136	3.3481	10.475	21.03	2145.3	0.41926
SLACK	58.354	189	15.603	142.72	2.251	10.474	22.353	151.61	0.64574
CR	58.232	197.67	15.484	149.42	2.2543	10.474	22.23	158.95	0.64543
MDD	57.507	2141.9	13.955	2090.1	2.4501	10.475	20.505	2100.5	0.57244
AT-RPT	58.654	189.39	15.95	145.09	2.2982	10.473	22.653	153.74	0.64401
P+W+A	58.179	189.65	15.461	144.12	2.2844	10.474	22.177	152.9	0.64194
P+W+S	58.047	191.28	15.491	143.33	2.4462	10.475	22.045	152.71	0.62896
(P+W)/T	58.027	201.97	15.299	154.08	2.2746	10.475	22.025	163.55	0.6422
PT/TIS	58.009	206.65	15.282	158.21	2.275	10.474	22.007	167.81	0.6424
LPT	63.404	2433.9	21.473	2393	3.0712	10.47	27.402	2401.2	0.47623
MWKR	62.15	2435	20.396	2396.3	3.2476	10.47	26.148	2404.1	0.44824
MTWF	62.028	2449.3	20.363	2410.8	3.3369	10.47	26.026	2418.5	0.43044
HRN	58.051	212.32	15.325	163.25	2.2764	10.474	22.049	172.91	0.64175
(S/RW)+S	63.404	2433.9	21.473	2393	3.0712	10.47	27.402	2401.2	0.47623
CR+SPT	63.077	2443.1	21.602	2402.1	3.5274	10.47	27.075	2410.3	0.42047
COVERT	56.674	2037.6	14.08	1988.8	2.4076	10.475	20.672	1998.5	0.58226

Table A5.8 The performances of DRs in 90U and LDD for Problem 1.2

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	56.269	2150.9	11.972	2092.4	9.7063	20.951	20.267	2111.9	0.20008
FCFS	58.411	188.62	10.765	135.4	6.3567	20.944	22.409	152.75	0.4571
LWKR	56.148	2219.4	11.926	2157.5	9.78	20.951	20.147	2178.2	0.19692
LTWF	56.238	2236.3	12.053	2174.2	9.8173	20.951	20.236	2194.9	0.19693
EDD	58.089	192.21	10.54	134.21	6.453	20.947	22.088	153.26	0.44936
PRW	57.032	2182.6	12.652	2126.7	9.6225	20.951	21.03	2145.3	0.2158
SLACK	58.3	190.12	10.656	133.84	6.3588	20.948	22.298	152.25	0.45374
CR	58.236	197.38	10.583	139.9	6.3493	20.948	22.235	158.8	0.45522
MDD	57.365	1953.7	9.5771	1892.2	6.5795	20.95	20.998	1912.7	0.4032
AT-RPT	58.654	189.39	11.03	136.58	6.3785	20.946	22.653	153.74	0.46571
P+W+A	58.179	189.65	10.584	135.38	6.408	20.949	22.177	152.9	0.45128
P+W+S	57.838	192.35	10.867	134.35	7.0317	20.95	21.836	153.35	0.4599
(P+W)/T	58.027	201.97	10.429	144.66	6.4055	20.95	22.025	163.55	0.45052
PT/TIS	58.009	206.65	10.409	148.63	6.403	20.949	22.007	167.81	0.45062
LPT	63.404	2433.9	18.394	2384.8	8.9931	20.941	27.402	2401.2	0.26542
MWKR	62.15	2435	17.536	2388.6	9.3883	20.941	26.148	2404.1	0.24925
MTWF	62.028	2449.3	17.648	2403.1	9.6225	20.941	26.026	2418.5	0.23876
HRN	58.051	212.32	10.455	153.63	6.4064	20.949	22.049	172.91	0.45053
(S/RW)+S	63.404	2433.9	18.394	2384.8	8.9931	20.941	27.402	2401.2	0.26542
CR+SPT	62.581	2465.2	18.778	2416.8	10.199	20.941	26.579	2432.9	0.21027
COVERT	57.18	1920.2	9.7362	1862	6.5581	20.951	21.178	1881.4	0.39493

Table A5.9 The performances of DRs in 85U and TDD for Problem 1.2

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	44.79	974.26	6.7126	929.69	4.1142	9.363	11.036	938.61	0.29744
FCFS	45.366	135.45	6.4741	92.837	3.2993	9.3628	11.613	101.27	0.46874
LWKR	44.74	1036.9	6.6939	990.96	4.1456	9.3632	10.986	1000.1	0.29253
LTWF	44.758	1051.8	6.7204	1005.5	4.1534	9.3632	11.005	1014.8	0.29174
EDD	45.272	136.62	6.4125	93.528	3.3317	9.3635	11.519	102.09	0.46471
PRW	45.25	831.4	7.1739	789.71	4.1157	9.363	11.496	798.05	0.30934
SLACK	45.346	135.63	6.4525	92.521	3.2982	9.362	11.592	101	0.46838
CR	45.322	140.99	6.4299	96.232	3.2997	9.362	11.568	105.14	0.46857
MDD	45.211	944.48	6.0989	898.67	3.3793	9.3632	11.157	907.83	0.43407
AT-RPT	45.426	134.7	6.5642	93.695	3.3293	9.3628	11.673	101.85	0.46826
P+W+A	45.297	135.57	6.4156	92.663	3.3106	9.3632	11.543	101.13	0.46674
P+W+S	45.211	136.27	6.4999	93.205	3.4808	9.3623	11.457	101.71	0.46048
(P+W)/T	45.276	140.66	6.3925	96.538	3.3082	9.3628	11.522	105.28	0.46694
PT/TIS	45.281	140.13	6.3976	96.239	3.3076	9.3628	11.528	104.95	0.46716
LPT	46.222	848.65	7.9467	808.28	3.9161	9.3625	12.469	816.36	0.35162
MWKR	46.019	901.61	7.8598	863.21	4.0322	9.3628	12.266	870.89	0.33174
MTWF	45.997	947.13	7.8624	908.92	4.057	9.3628	12.243	916.56	0.32672
HRN	45.305	141.39	6.4232	97.509	3.3093	9.3619	11.552	106.22	0.46648
(S/RW)+S	46.222	848.65	7.9467	808.28	3.9161	9.3625	12.469	816.36	0.35162
CR+SPT	45.993	866.34	8.1405	826.48	4.3389	9.3632	12.239	834.45	0.29859
COVERT	45	864.91	6.1498	820.38	3.3415	9.3628	11.246	829.29	0.44859

Table A5.10 The performances of DRs in 85U and LDD for Problem 1.2

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	44.79	974.26	4.99	920.78	10.83	18.726	11.036	938.61	0.13895
FCFS	45.366	135.45	3.4962	84.456	8.7598	18.725	11.613	101.27	0.25809
LWKR	44.74	1036.9	4.379	981.77	10.869	18.726	10.986	1000.1	0.13631
LTWF	44.758	1051.8	5.0075	996.29	10.879	18.726	11.005	1014.8	0.13591
EDD	45.259	136.69	3.4436	84.761	8.8142	18.727	11.506	101.96	0.25517
PRW	45.25	831.4	5.3718	781.37	10.752	18.726	11.496	798.05	0.15197
SLACK	45.328	136.02	3.4659	84.415	8.7674	18.725	11.575	101.25	0.25627
CR	45.324	140.59	3.4551	87.149	8.7609	18.724	11.57	104.87	0.25702
MDD	45.27	823.33	3.2848	768.8	8.8452	18.726	11.316	786.98	0.23808
AT-RPT	45.426	134.7	3.5681	85.543	8.7716	18.725	11.673	101.85	0.2637
P+W+A	45.297	135.57	3.4508	84.239	8.7842	18.726	11.543	101.13	0.25604
P+W+S	45.117	136.58	3.7797	84.643	9.2926	18.726	11.363	101.82	0.28313
(P+W)/T	45.276	140.66	3.4297	87.79	8.7838	18.725	11.522	105.28	0.2556
PT/TIS	45.281	140.13	3.433	87.568	8.7814	18.725	11.528	104.95	0.25571
LPT	46.222	848.65	5.8861	800.21	10.293	18.725	12.469	816.36	0.1751
MWKR	46.019	901.61	5.9425	855.53	10.553	18.725	12.266	870.89	0.16702
MTWF	45.997	947.13	5.9822	901.28	10.615	18.725	12.243	916.56	0.16422
HRN	45.305	141.39	3.4594	88.859	8.7839	18.723	11.552	106.22	0.25613
(S/RW)+S	46.222	848.65	5.8861	800.21	10.293	18.725	12.469	816.36	0.1751
CR+SPT	45.926	893.76	6.3998	846.26	11.103	18.725	12.173	862.09	0.14504
COVERT	45.159	753.63	3.331	699.72	8.8017	18.727	11.406	717.69	0.24553

The performances of the dispatching rules for problem 2.1

Table A5.11 The performances of DRs in 95U and LDD for Problem 2.1

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	151.04	10111	60.66	9990.6	17.958	75.934	94.021	10047	0.29689
FCFS	190.65	530.85	86.848	448.08	4.5309	75.017	133.63	483.31	0.80736
LWKR	166.51	13043	76.034	12891	17.854	74.646	109.49	12963	0.27801
LTWF	171.39	14504	81.618	14345	18.562	73.965	114.37	14421	0.25987
EDD	188.81	563.49	84.625	423.32	4.1556	74.556	131.78	486.22	0.80259
PRW	174.02	10659	84.411	10575	18.733	75.853	117	10615	0.3578
SLACK	189.83	558.64	85.458	421.39	3.9627	74.577	132.81	482.69	0.80882
CR	184.59	684.95	80.076	543.48	3.8255	74.295	127.57	609.55	0.81631
MDD	170.32	12338	66.646	12188	4.6553	74.188	113.3	12259	0.66788
AT-RPT	191.64	528.68	88.023	450.98	4.722	75.536	134.62	485.32	0.81305
P+W+A	185.91	529.05	82.478	438.32	4.901	75.05	128.89	475.79	0.78984
P+W+S	181.7	553.28	79.41	414.33	6.0486	75.275	124.68	477.04	0.78731
(P+W)/T	168.53	590.72	66.303	483.18	6.1141	75.213	111.5	531.18	0.73665
PT/TIS	166.26	621.92	64.211	513.32	6.2926	75.347	109.23	561.21	0.72728
LPT	628.88	23606	532.05	23534	11.513	73.875	571.86	23568	0.63234
MWKR	285.36	15815	196.93	15749	19.914	75.563	228.33	15780	0.41835
MTWF	249.16	16071	164.01	16014	23.195	75.612	192.13	16041	0.3458
HRN	176.05	672.69	72.892	536.77	5.1812	74.765	119.03	599.6	0.76974
(S/RW)+S	614.6	24354	521.79	24284	15.539	74.565	557.57	24317	0.55209
CR+SPT	535.44	22961	447.93	22897	20.83	74.605	478.42	22927	0.39117
COVERT	159.48	9078.3	56.302	8953.6	5.1595	75.403	102.46	9012.7	0.6713

Table A5.12 The performances of DRs in 90U and TDD for Problem 2.1

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	97.946	3017.1	20.724	2907.3	14.581	56.432	43.944	2952.5	0.27593
FCFS	111.24	311.38	26.996	237.69	7.5562	56.309	57.241	263.72	0.61376
LWKR	103.24	3894.8	25.703	3764.7	14.257	55.978	49.247	3818.3	0.25563
LTWF	104.76	4149.6	27.451	4016.6	14.492	56.212	50.76	4071.3	0.24485
EDD	109.85	335.46	25.257	218.23	7.207	56.3	55.851	264.81	0.59494
PRW	105.61	3199.1	28.839	3123.9	15.027	56.593	51.613	3154.9	0.32448
SLACK	110.63	329.7	25.727	216.44	6.8993	56.424	56.629	259.59	0.60717
CR	109.41	396.94	24.342	275.65	6.734	56.247	55.41	324.69	0.62216
MDD	105.08	3494.4	21.012	3365.1	7.7308	56.16	51.082	3418.3	0.48057
AT-RPT	112.54	309.6	28.608	240.68	7.8693	56.518	58.54	266.52	0.62979
P+W+A	108.67	313.67	24.78	228.49	7.9095	56.288	54.672	256.78	0.59112
P+W+S	105.81	326.96	23.165	210.54	9.1569	56.413	51.81	255.95	0.58292
(P+W)/T	103.42	343.6	20.441	243.89	8.825	56.401	49.418	283.5	0.54274
PT/TIS	102.84	363.22	19.971	261.84	8.9264	56.401	48.846	300.66	0.53569
LPT	185.88	6086.4	105.43	6030.1	11.354	55.674	131.88	6053.3	0.50244
MWKR	137	4379.4	60.802	4317.6	15.6	56.398	83.004	4343	0.37318
MTWF	127.68	4541	53.251	4486.1	17.371	56.506	73.682	4508.7	0.32247
HRN	106.53	375.86	22.701	266.7	7.9676	56.386	52.535	309.31	0.57774
(S/RW)+S	184.46	6239.1	105.47	6182.9	12.817	55.735	130.45	6206.1	0.47084
CR+SPT	173.77	6097.7	97.884	6039.9	15.914	55.871	119.77	6063.7	0.35638
COVERT	102.67	2543.4	18.341	2444.1	7.4659	56.271	48.676	2484.9	0.53473

Table A5.13 The performances of DRs in 90U and LDD for Problem 2.1

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	97.946	3017.1	18.18	2894.4	22.837	72.632	43.944	2952.5	0.19292
FCFS	111.24	311.38	21.235	230.68	12.596	72.447	57.241	263.72	0.50487
LWKR	103.24	3894.8	23.054	3749.4	22.408	72.133	49.247	3818.3	0.18319
LTWF	104.76	4149.6	24.822	4000.9	22.663	72.379	50.76	4071.3	0.1775
EDD	109.6	339.56	19.262	207.19	12.26	72.376	55.603	267.76	0.4776
PRW	105.61	3199.1	26.038	3115.1	23.027	72.869	51.613	3154.9	0.2504
SLACK	110.42	333.05	19.683	204.85	11.858	72.487	56.426	261.61	0.48752
CR	109.66	395.05	18.57	259.37	11.511	72.457	55.66	322.68	0.50164
MDD	105.85	3318.4	15.962	3176.1	12.708	72.292	51.855	3243.5	0.3809
AT-RPT	112.54	309.6	22.703	233.51	12.764	72.8	58.54	266.52	0.52855
P+W+A	108.67	313.67	19.246	220.81	13.176	72.462	54.672	256.78	0.47875
P+W+S	104.75	329.84	17.587	198.61	15.434	72.613	50.754	258.8	0.47688
(P+W)/T	103.42	343.6	15.393	233.07	14.576	72.59	49.418	283.5	0.426
PT/TIS	102.84	363.22	14.986	251.16	14.742	72.59	48.846	300.66	0.4174
LPT	185.88	6086.4	100.74	6023.5	17.463	71.907	131.88	6053.3	0.44156
MWKR	137	4379.4	57.613	4310.3	23.211	72.714	83.004	4343	0.31772
MTWF	127.68	4541	50.611	4479.7	25.531	72.805	73.682	4508.7	0.27276
HRN	106.53	375.86	17.263	254.8	13.329	72.571	52.535	309.31	0.45995
(S/RW)+S	182.76	6372.7	100.49	6310.1	20.336	71.916	128.76	6339.8	0.39634
CR+SPT	167.5	5925	88.982	5853.9	24.081	72.141	113.5	5887.6	0.28966
COVERT	104.29	2548.9	13.97	2436.7	12.275	72.7	50.296	2489.6	0.42215

Table A5.14 The performances of DRs in 85U and TDD for Problem 2.1

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	78.059	1393.8	8.8241	1298	17.489	53.091	27.044	1337.5	0.18596
FCFS	84.188	218.65	9.8105	148.65	12.346	52.929	33.174	173.12	0.39268
LWKR	80.229	1693.6	10.63	1578.2	17.125	52.914	29.215	1625.7	0.17124
LTWF	81.009	1807	11.438	1685.8	17.152	52.856	29.995	1735.7	0.16957
EDD	82.817	243.33	8.3117	130.71	12.219	52.902	31.802	174.29	0.35326
PRW	81.908	1383.3	12.785	1303.5	17.602	53.053	30.893	1336.3	0.23247
SLACK	83.507	236.95	8.5853	127.44	11.802	52.9	32.492	168.72	0.36763
CR	83.451	263.99	8.2863	154.35	11.56	52.994	32.436	197.76	0.38535
MDD	81.476	1416	7.2926	1301.3	12.541	52.902	30.461	1348.3	0.28467
AT-RPT	85.219	217.02	11.106	151.05	12.611	53.067	34.205	174.89	0.42056
P+W+A	82.673	220.88	8.6841	140.88	12.736	53.001	31.658	168.13	0.36841
P+W+S	80.478	236.95	7.9754	125.35	14.221	53.086	29.463	168.63	0.36149
(P+W)/T	80.511	246.45	7.2979	154.34	13.511	53.024	29.496	189.42	0.33108
PT/TIS	80.275	255.76	7.142	161.23	13.591	52.991	29.26	198.34	0.3255
LPT	105.24	1967.6	33.408	1907.4	14.886	52.817	54.232	1932.1	0.35438
MWKR	94.923	1993.6	25.836	1934.6	17.638	53.281	43.908	1958.8	0.28665
MTWF	91.405	2047.1	23.612	1995.4	18.932	53.158	40.39	2016.7	0.25425
HRN	82.087	266.08	8.0732	161.03	12.71	52.979	31.072	201.46	0.35717
(S/RW)+S	104.73	2031.2	33.89	1971.3	15.882	52.882	53.719	1996	0.33939
CR+SPT	101.51	2127	32.857	2064.5	18.065	52.665	50.503	2090.2	0.26522
COVERT	81.469	1152.5	6.7076	1055.9	11.963	52.892	30.455	1095.7	0.34177

Table A5.15 The performances of DRs in 85U and LDD for Problem 2.1

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	78.059	1393.8	7.2544	1286.8	26.122	68.3	27.044	1337.5	0.1238
FCFS	84.188	218.65	6.6628	142.03	19.401	68.102	33.174	173.12	0.27907
LWKR	80.229	1693.6	8.9782	1564.6	25.676	68.089	29.215	1625.7	0.11746
LTWF	81.009	1807	9.7485	1671.5	25.665	67.976	29.995	1735.7	0.11738
EDD	82.731	246.34	5.2796	120.82	19.475	68.021	31.716	177.72	0.23409
PRW	81.908	1383.3	10.952	1294.2	25.972	68.299	30.893	1336.3	0.17204
SLACK	83.356	241.89	5.4443	118.3	19.015	68.024	32.341	173.54	0.24215
CR	83.62	263.81	5.2355	139.87	18.542	68.203	32.606	196.74	0.25694
MDD	81.824	1265.7	4.598	1138.4	19.701	68.021	30.809	1198.7	0.18681
AT-RPT	85.219	217.02	7.7316	144.54	19.439	68.316	34.205	174.89	0.31164
P+W+A	82.673	220.88	5.7481	133.78	20.002	68.201	31.658	168.13	0.25432
P+W+S	79.923	239.48	5.3085	113.84	22.312	68.3	28.909	170.46	0.25927
(P+W)/T	80.511	246.45	4.669	144.64	21.085	68.242	29.496	189.42	0.21967
PT/TIS	80.275	255.76	4.552	150.88	21.204	68.198	29.26	198.34	0.21446
LPT	105.24	1967.6	30.41	1900.3	22.092	68.024	54.232	1932.1	0.29386
MWKR	94.923	1993.6	23.591	1927.6	25.596	68.568	43.908	1958.8	0.23523
MTWF	91.405	2047.1	21.688	1989.3	27.212	68.414	40.39	2016.7	0.20846
HRN	82.087	266.08	5.1991	150.29	20.039	68.183	31.072	201.46	0.23983
(S/RW)+S	104.05	2121.7	31.064	2055.1	23.934	67.923	53.043	2086.6	0.27409
CR+SPT	100.08	2094.4	29.562	2022.5	26.41	67.902	49.065	2056.6	0.20975
COVERT	82.386	1038.4	4.2933	930.58	18.834	68.039	31.371	981.63	0.2284

The performances of the dispatching rules for problem 2.2

Table A5.16 The performances of DRs in 95U and LDD for Problem 2.2

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	173.22	10119	61.01	9964	17.059	82.33	97.18	10028	0.33073
FCFS	217.26	571.77	93.221	487.48	5.2252	82.764	141.22	517.05	0.80672
LWKR	181.29	11965	69.347	11773	17.324	82.222	105.25	11852	0.3049
LTWF	190.24	13688	78.803	13487	17.827	81.492	114.2	13570	0.28218
EDD	211.42	609.96	86.583	435.96	4.4264	81.269	135.38	501.55	0.79853
PRW	215.35	10942	103.29	10825	17.212	83.2	139.31	10873	0.40541
SLACK	214.4	611.73	89.205	441.43	4.0738	81.32	138.36	505.36	0.81294
CR	203.85	733.33	78.44	563.17	3.8566	81.718	127.81	627.51	0.82501
MDD	186.36	11170	61.973	10979	4.8793	81.491	110.32	11057	0.68294
AT-RPT	221.4	575.65	97.165	492.56	5.0335	82.687	145.36	521.94	0.82373
P+W+A	209.37	565.02	85.873	476.09	5.7708	82.386	133.33	505.93	0.77946
P+W+S	203.84	600.97	80.936	427.88	6.3608	81.486	127.8	493.64	0.78308
(P+W)/T	191.02	669.39	68.783	543.63	7.0286	82.464	114.98	589.81	0.72954
PT/TIS	188.76	681.52	66.684	551.21	7.1891	82.308	112.72	599.03	0.72092
LPT	928.08	28128	809.79	28046	10.995	81.213	852.03	28080	0.67063
MWKR	427.29	18067	315.63	17995	17.616	82.907	351.24	18025	0.50374
MTWF	315.57	17769	210.28	17709	23.98	83.5	239.52	17734	0.36229
HRN	198.23	770.46	74.431	586.57	5.4705	82.123	122.19	661.15	0.77129
(S/RW)+S	862.89	29252	750.93	29175	17.338	81.103	786.83	29207	0.53878
CR+SPT	721.99	27267	614.95	27195	22.239	82.325	645.94	27224	0.3944
COVERT	181.53	9314.1	57.54	9158.9	5.2758	82.836	105.49	9222.8	0.66478

Table A5.17 The performances of DRs in 90U and TDD for Problem 2.2

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	117.23	2987	21.517	2853.8	12.278	56.584	45.238	2898.2	0.34346
FCFS	132.39	329.5	31.775	255.79	7.3808	56.843	60.394	276.83	0.64258
LWKR	119.74	3949.7	23.9	3794	12.15	56.448	47.749	3845.9	0.32082
LTWF	123.34	4131.4	27.49	3966.6	12.144	56.501	51.345	4021.5	0.30596
EDD	128.54	362.26	27.001	215.52	6.4559	56.535	56.545	260.1	0.62228
PRW	130.12	3433.7	34.484	3331.9	12.358	56.685	58.126	3365.3	0.39649
SLACK	130.72	361.95	28.748	217.82	6.019	56.471	58.729	261.9	0.64839
CR	128.23	412.79	26.034	263.35	5.7955	56.652	56.238	308.65	0.67098
MDD	121.96	3643.3	20.817	3488	6.8554	56.557	49.961	3539.8	0.52588
AT-RPT	135.94	333.19	35.292	262.27	7.3482	56.807	63.944	282.95	0.66839
P+W+A	128.09	327.3	27.823	244.81	7.7251	56.808	56.097	266.11	0.61374
P+W+S	124.98	353.25	25.058	207.53	8.0716	56.584	52.986	252.35	0.61238
(P+W)/T	123.12	369.07	23.55	267.37	8.4303	56.51	51.119	297.61	0.57285
PT/TIS	122.5	388.28	22.989	275.41	8.4869	56.561	50.502	306.97	0.56751
LPT	239.27	7083.8	140.96	7016.8	9.6938	56.098	167.26	7039.1	0.55884
MWKR	185.5	5248.7	90.119	5173.3	12.62	56.658	113.49	5198.4	0.46034
MTWF	157.9	5313.6	65.705	5258.7	15.802	56.849	85.903	5276.9	0.35398
HRN	126.08	435.61	25.23	289.22	7.1445	56.849	54.086	336.17	0.61481
(S/RW)+S	232.24	7388	136.53	7326.3	12.288	56.2	160.24	7346.9	0.48926
CR+SPT	220.69	7326.2	127.42	7262.7	14.727	56.342	148.69	7283.5	0.38601
COVERT	121.59	2792.6	20.143	2662.3	6.5481	56.639	49.595	2705.5	0.57259

Table A5.18 The performances of DRs in 90U and LDD for Problem 2.2

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	117.23	2987	17.584	2836	22.745	79.372	45.238	2898.2	0.20797
FCFS	132.39	329.5	24.012	247.56	14.017	79.761	60.394	276.83	0.51574
LWKR	119.74	3949.7	20.011	3773.2	22.661	79.16	47.749	3845.9	0.1935
LTWF	123.34	4131.4	23.461	3944.6	22.515	79.26	51.345	4021.5	0.19023
EDD	128.23	369.72	18.909	200.68	13.068	79.242	56.24	265.4	0.46603
PRW	130.12	3433.7	29.998	3318.5	22.273	79.555	58.126	3365.3	0.28173
SLACK	130.25	367.12	20.241	203.16	12.384	79.143	58.256	265.47	0.48819
CR	129.08	412.84	18.335	244.34	11.653	79.51	57.081	306.83	0.51821
MDD	123.32	3177.6	14.389	3006.4	13.464	79.242	51.324	3076.9	0.38316
AT-RPT	135.94	333.19	27.116	254.27	13.572	79.673	63.944	282.95	0.55367
P+W+A	128.09	327.3	20.513	236.57	14.815	79.69	56.097	266.11	0.47418
P+W+S	123.71	361.25	17.496	194.58	16.177	79.372	51.719	259.65	0.46642
(P+W)/T	123.12	369.07	16.758	256.02	16.039	79.276	51.119	297.61	0.4287
PT/TIS	122.5	388.28	16.252	263.27	16.15	79.343	50.502	306.97	0.42145
LPT	239.27	7083.8	134.19	7007.9	17.331	78.84	167.26	7039.1	0.48071
MWKR	185.5	5248.7	84.854	5163.2	21.756	79.602	113.49	5198.4	0.38413
MTWF	157.9	5313.6	62.06	5251.4	26.558	79.795	85.903	5276.9	0.28676
HRN	126.08	435.61	17.731	270.85	14.045	79.682	54.086	336.17	0.46356
(S/RW)+S	226.94	7364.1	126.66	7294.1	22.121	79.185	154.94	7323	0.39434
CR+SPT	210.15	7360.7	113.3	7298.8	25.554	79.128	138.15	7324.3	0.29787
COVERT	123.91	2373.8	14.178	2227.8	12.663	79.529	51.915	2287.5	0.42029

Table A5.19 The performances of DRs in 85U and TDD for Problem 2.2

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	95.787	1412.4	9.1525	1289.8	15.386	53.564	27.772	1330.6	0.23366
FCFS	102.69	244.65	12.642	172.96	11.968	53.614	34.681	192.99	0.43249
LWKR	96.579	1546.5	9.8075	1406	15.25	53.41	28.564	1452.7	0.21538
LTWF	98.487	1786.3	11.487	1634.9	15.02	53.569	30.474	1685.4	0.2119
EDD	100.18	278.3	9.2283	138.94	11.067	53.437	32.167	181.13	0.38541
PRW	102.1	1536.3	15.266	1441.4	15.18	53.378	34.093	1472.9	0.28793
SLACK	101.52	277.57	10.067	141.03	10.563	53.535	33.511	181.59	0.41116
CR	101.28	293.04	9.436	164.95	10.171	53.641	33.271	203.79	0.4442
MDD	98.172	1346.1	7.4837	1207.6	11.332	53.437	30.158	1253.8	0.32407
AT-RPT	105.29	245.74	15.157	177.28	11.883	53.526	37.281	197.17	0.46867
P+W+A	100.37	244.17	10.624	165.17	12.266	53.366	32.365	185.81	0.4026
P+W+S	98.179	267.7	8.9356	130.63	12.777	53.476	30.165	171.63	0.39426
(P+W)/T	98.388	280.25	9.1772	186.71	12.81	53.678	30.374	213.21	0.37124
PT/TIS	98.166	281.17	8.9788	184.49	12.834	53.723	30.152	211.62	0.36783
LPT	133.18	2479.5	44.419	2413.5	13.259	53.376	65.167	2435.4	0.40533
MWKR	123.69	2273	36.762	2203.2	15.089	53.595	55.68	2226.4	0.35527
MTWF	113.15	2226.8	28.634	2174.7	17.497	53.77	45.144	2192	0.27956
HRN	99.917	314.75	9.4873	183.08	11.591	53.615	31.902	225.14	0.40045
(S/RW)+S	130.53	2538.8	43.723	2477.2	15.21	53.409	62.52	2497.7	0.36155
CR+SPT	127.32	2694.6	42.214	2631.8	16.914	53.597	59.307	2652.8	0.29178
COVERT	99.004	1258.3	7.7036	1139.7	10.72	53.682	30.989	1178.8	0.38295

Table A5.20 The performances of DRs in 85U and LDD for Problem 2.2

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	95.787	1412.4	6.7585	1273.4	26.595	75.092	27.772	1330.6	0.12897
FCFS	102.69	244.65	8.2693	165.34	21.198	75.233	34.681	192.99	0.29862
LWKR	96.579	1546.5	7.4486	1387.3	26.494	74.814	28.564	1452.7	0.11827
LTWF	98.487	1786.3	8.9405	1614.7	26.075	75.094	30.474	1685.4	0.12139
EDD	99.966	281.42	5.0187	124.1	20.676	74.869	31.952	182.98	0.22058
PRW	102.1	1536.3	12.326	1428.9	25.843	74.845	34.093	1472.9	0.19098
SLACK	101.17	280.86	5.5242	126.2	19.974	74.819	33.159	183.51	0.23829
CR	101.92	295.55	5.2479	147.52	18.944	75.273	33.913	203.13	0.2716
MDD	98.798	1224.6	5.0259	1070.6	20.851	74.869	30.784	1133.3	0.18196
AT-RPT	105.29	245.74	10.294	169.63	20.622	75.083	37.281	197.17	0.34415
P+W+A	100.37	244.17	6.6362	157.54	21.881	74.84	32.365	185.81	0.26152
P+W+S	97.507	270.23	5.2101	115.94	23.326	74.969	29.493	174.06	0.24917
(P+W)/T	98.388	280.25	5.4895	176.47	22.725	75.21	30.374	213.21	0.23206
PT/TIS	98.166	281.17	5.3195	174.4	22.777	75.256	30.152	211.62	0.22759
LPT	133.18	2479.5	40.048	2404.7	22.491	74.911	65.167	2435.4	0.32489
MWKR	123.69	2273	33.078	2193.9	25.008	75.189	55.68	2226.4	0.28262
MTWF	113.15	2226.8	25.998	2167.7	28.464	75.413	45.144	2192	0.21867
HRN	99.917	314.75	5.3974	167.5	21.104	75.128	31.902	225.14	0.24494
(S/RW)+S	129.02	2632.5	39.153	2563.8	25.753	74.874	61.01	2592.1	0.27728
CR+SPT	124.89	2595.5	37.279	2528.7	28.006	74.884	56.883	2556.1	0.21758
COVERT	100.32	1157.5	4.3087	1014.9	19.608	74.865	32.31	1072.9	0.22838

The performances of the dispatching rules for problem 3.1

Table A5.21 The performances of DRs in 95U and LDD for Problem 3.1

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	272.57	11104	73.02	10842	39.865	148.17	139.56	10959	0.24841
FCFS	335.06	761.66	106.2	569.88	10.555	146.93	202.05	645.1	0.74893
LWKR	308.9	18214	111.53	17884	42.053	144.63	175.89	18031	0.2267
LTWF	313.56	19186	117.39	18846	43.248	143.49	180.55	18997	0.21401
EDD	332.22	830.24	102.72	526.1	9.9163	144.05	199.21	653.91	0.74415
PRW	308.67	13595	112.65	13402	43.408	151.05	175.65	13488	0.28982
SLACK	333.71	814.6	103.59	517.56	9.3028	144.42	200.69	641.12	0.75178
CR	327.29	1008	96.636	715.55	8.7641	145.46	194.28	839.77	0.76182
MDD	313.97	15866	86.453	15543	11.894	143.97	180.96	15686	0.53149
AT-RPT	336.31	754.01	108.13	586.37	11.238	149.59	203.3	655.25	0.7528
P+W+A	323.92	756.6	96.581	546.74	12.071	147.05	190.91	628.85	0.71431
P+W+S	310.65	801.16	88.303	500.15	17.065	147.39	177.64	626.98	0.69757
(P+W)/T	294.09	847.77	70.957	598.07	16.283	147.88	161.08	702.73	0.62519
PT/TIS	290.82	862.16	68.343	615.65	16.936	147.4	157.81	717.36	0.61058
LPT	1453.1	31682	1228.9	31507	15.309	144.19	1320	31584	0.74158
MWKR	417.85	18738	224.52	18591	46.085	150.39	284.84	18656	0.34491
MTWF	382.65	18944	194.2	18803	50.965	151.61	249.64	18866	0.2959
HRN	309.6	932.72	83.247	644.58	13.067	146.35	176.58	768.05	0.67959
(S/RW)+S	1414.1	32155	1196.5	31978	21.898	145.99	1281	32057	0.67258
CR+SPT	1167.7	32076	967.79	31904	39.569	146.16	1034.6	31981	0.43431
COVERT	293.79	9905	65.463	9642.7	11.084	148	160.78	9759.3	0.64726

Table A5.22 The performances of DRs in 90U and TDD for Problem 3.1

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	190.67	2991.6	21.36	2775.1	32.318	108.62	64.655	2855.9	0.20357
FCFS	210.41	444.13	26.645	273.35	17.867	109.37	84.391	325.56	0.49973
LWKR	201.94	5122	32.626	4851.6	32.32	107.34	75.92	4953	0.19042
LTWF	203.46	5208	34.539	4929.4	32.705	107.33	77.447	5033.9	0.18523
EDD	208.31	496.62	24.189	240.95	17.506	107.26	82.296	332.39	0.47702
PRW	199.78	3420.1	31.921	3257.1	33.776	109.21	73.758	3318.1	0.2418
SLACK	209.5	479.19	24.38	230.44	16.509	107.85	83.485	318.65	0.48792
CR	208.69	550.73	23.005	303.81	15.942	107.74	82.676	392.54	0.50246
MDD	204.39	4303.5	21.885	4036.6	19.125	107.26	78.374	4136.6	0.32811
AT-RPT	211.84	433.8	29.086	284.3	18.873	109.14	85.827	333.66	0.52178
P+W+A	205.02	445.18	22.699	257.26	19.309	108.33	79.003	315.77	0.45989
P+W+S	197.8	470.37	19.826	218.25	23.658	108.58	71.782	307.83	0.44298
(P+W)/T	197.14	490.84	17.464	278.69	21.958	108.78	71.12	353.53	0.39606
PT/TIS	196.36	495.14	17.034	281.43	22.306	108.78	70.342	357.45	0.38665
LPT	350.97	7371.2	169.09	7213.3	19.754	107.4	224.95	7272.5	0.53753
MWKR	232.05	4815.7	65.255	4679.9	34.833	108.8	106.03	4730.8	0.30048
MTWF	224.07	4907	59.768	4783.3	37.33	109.31	98.051	4829.7	0.26764
HRN	202.61	524.57	20.479	288.52	19.501	108.14	76.591	373.13	0.44008
(S/RW)+S	348.1	7555.5	168.5	7403.9	22.035	107.63	222.08	7460.8	0.51288
CR+SPT	327.38	7819.9	156.26	7672.2	30.512	108.1	201.36	7727.6	0.3727
COVERT	202.17	2383.1	17.564	2164.8	17.025	108.96	76.152	2245.9	0.44736

Table A5.23 The performances of DRs in 90U and LDD for Problem 3.1

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	190.67	2991.6	17.239	2748.2	53.402	145.21	64.655	2855.9	0.12687
FCFS	210.41	444.13	16.685	256.73	33.111	146.39	84.391	325.56	0.34059
LWKR	201.94	5122	28.05	4817.8	52.948	143.39	75.92	4953	0.13066
LTWF	203.46	5208	30.002	4894.6	53.373	143.43	77.447	5033.9	0.12795
EDD	208.1	503.44	14.215	213.39	32.949	143.46	82.084	338.68	0.31024
PRW	199.78	3420.1	27.323	3236.8	54.382	145.99	73.758	3318.1	0.17159
SLACK	209.27	488.88	14.259	204.83	31.83	144.84	83.247	323.77	0.31394
CR	209.47	551.61	13.334	270.81	30.704	143.75	83.448	388.58	0.3255
MDD	205.53	3778.6	12.8	3476.3	34.109	143.46	79.51	3610.7	0.20537
AT-RPT	211.84	433.8	18.685	268.61	33.676	146.05	85.827	333.66	0.36905
P+W+A	205.02	445.18	13.679	239.35	35.493	145.17	79.003	315.77	0.30009
P+W+S	195.49	471.61	11.66	187.79	43.002	145.11	69.476	307.74	0.30045
(P+W)/T	197.14	490.84	9.8425	255.27	39.541	145.58	71.12	353.53	0.24017
PT/TIS	196.36	495.14	9.5807	257.35	40.057	145.58	70.342	357.45	0.23248
LPT	350.97	7371.2	157.22	7193.6	33.093	144.32	224.95	7272.5	0.46077
MWKR	232.05	4815.7	59.455	4663.1	54.238	145.6	106.03	4730.8	0.23953
MTWF	224.07	4907	54.747	4767.8	57.513	146.23	98.051	4829.7	0.21239
HRN	202.61	524.57	11.829	261.84	36.055	144.75	76.591	373.13	0.27451
(S/RW)+S	343.48	8081.5	154.95	7917.8	38.304	144.34	217.46	7990.4	0.42233
CR+SPT	309.57	7636.4	132.94	7468.1	50.217	144.91	183.55	7542.9	0.28458
COVERT	205.77	2210.8	10.434	1973.9	31.503	145.16	79.748	2078.1	0.28855

Table A5.24 The performances of DRs in 80U and TDD for Problem 3.1

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	158.3	1525.5	7.4988	1317.6	39.63	103.1	39.283	1395.2	0.1107
FCFS	166.75	333.07	6.7286	168.09	30.41	102.84	47.733	219.49	0.23369
LWKR	162.69	2136	11.356	1879.5	39.096	101.91	43.674	1975.6	0.11014
LTWF	163.39	2279.1	12.098	2016.8	39.145	101.92	44.367	2115.2	0.10997
EDD	165.13	383.55	5.3586	140.45	30.664	102.2	46.108	227.58	0.19956
PRW	162.46	1602.1	11.985	1449.1	39.965	103.67	43.435	1505.8	0.14881
SLACK	166.03	364.31	5.2301	129.23	29.633	102.39	47.011	211.3	0.20215
CR	166.48	396.49	4.9611	158.44	28.911	102.71	47.464	242.51	0.21413
MDD	164.27	1778.6	5.2095	1526.3	31.37	102.2	45.253	1620.9	0.13987
AT-RPT	168.14	321.59	8.6745	182.86	30.966	103.24	49.123	231.52	0.27556
P+W+A	164.03	333.97	5.4074	155.5	31.812	102.8	45.009	213.77	0.20267
P+W+S	159.74	357.39	5.0858	119.88	35.78	103.41	40.719	204.65	0.20415
(P+W)/T	160.91	364.94	4.2006	168.43	33.728	103.23	41.886	235.33	0.168
PT/TIS	160.68	368.96	4.1634	166.22	33.916	103.2	41.661	235.77	0.16443
LPT	201.83	2488.6	45.996	2337.9	22.695	84.4	82.813	2387.9	0.3688
MWKR	176.3	2248.2	27.922	2122.9	30.158	85.82	57.275	2164.5	0.24909
MTWF	173.04	2207.4	26.113	2087.2	31.608	86.238	54.017	2127.3	0.22587
HRN	163.5	390.35	7.5144	183.05	22.547	85.626	44.479	246.88	0.27889
(S/RW)+S	201.17	2527.1	46.197	2378.3	23.56	84.439	82.148	2427.6	0.36164
CR+SPT	197.39	2496.1	46.7	2348.4	27.837	84.853	78.374	2397.2	0.28774
COVERT	164.63	1180.6	6.6913	981.7	20.595	85.874	45.608	1047.4	0.2969

Table A5.25 The performances of DRs in 85U and TDD for Problem 3.1

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	158.3	1525.5	5.4698	1291.8	61.406	137.77	39.283	1395.2	0.06395
FCFS	166.75	333.07	3.0802	151.47	50.566	137.46	47.733	219.49	0.11553
LWKR	162.69	2136	8.9174	1847.7	60.462	136.1	43.674	1975.6	0.07041
LTWF	163.39	2279.1	9.6108	1984.1	60.462	136.07	44.367	2115.2	0.07098
EDD	165	392.71	2.0731	115.61	51.316	136.33	45.976	234.33	0.08472
PRW	162.46	1602.1	9.4287	1430.4	61.213	138.56	43.435	1505.8	0.09839
SLACK	165.82	372.01	1.9762	105.92	50.397	136.43	46.798	218.92	0.0824
CR	167.02	397.84	1.8572	130.45	49.078	137.05	47.997	243.21	0.08821
MDD	164.65	1458.5	2.0398	1175.7	51.624	136.33	45.635	1301.2	0.05874
AT-RPT	168.14	321.59	4.2923	166.72	50.388	138.05	49.123	231.52	0.15126
P+W+A	164.03	333.97	2.3179	137.95	52.528	137.43	45.009	213.77	0.09306
P+W+S	159.12	366.32	2.6522	97.96	57.766	137.79	40.104	210.99	0.1155
(P+W)/T	160.91	364.94	1.6644	147.02	54.997	138.01	41.886	235.33	0.07083
PT/TIS	160.68	368.96	1.659	144.63	55.217	138.01	41.661	235.77	0.06928
LPT	201.83	2488.6	38.866	2317.9	39.37	118.63	82.813	2387.9	0.27849
MWKR	176.3	2248.2	23.588	2106.2	49.629	120.55	57.275	2164.5	0.18167
MTWF	173.04	2207.4	22.269	2071.2	51.569	121.2	54.017	2127.3	0.16496
HRN	163.5	390.35	3.0998	159.66	41.937	120.09	44.479	246.88	0.12496
(S/RW)+S	200.07	2606.5	39.273	2442.8	41.535	118.75	81.055	2509.8	0.26807
CR+SPT	194.53	2601.8	39.729	2421.2	47.536	119.1	75.509	2495.2	0.20231
COVERT	166.31	979.4	2.7653	750.11	38.789	120.16	47.292	843.62	0.141

The performances of the dispatching rules for problem 3.2

Table A5.26 The performances of DRs in 95U and LDD for Problem 3.2

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	284.78	11746	76.909	11467	40.122	154.22	147	11591	0.25639
FCFS	360.06	805.57	121.46	612.66	9.3956	152.94	222.28	688.09	0.78161
LWKR	327.44	18760	121.29	18419	41.855	147.09	189.66	18570	0.24192
LTWF	333.73	20530	129.03	20168	43.305	146.46	195.95	20329	0.22511
EDD	354.93	876.09	115.51	555.21	8.574	146.57	217.15	689.71	0.77582
PRW	324.02	13867	120.08	13666	44.06	158.48	186.25	13755	0.29787
SLACK	357.43	862.48	117.35	548.23	7.9147	147.06	219.66	679.71	0.78455
CR	351.06	1076.9	110.34	757.46	7.2696	150.45	213.29	895.89	0.7978
MDD	334.8	17765	97.37	17425	10.558	146.35	197.03	17576	0.55643
AT-RPT	362.21	802.72	124.02	630.18	9.8021	157.06	224.44	699.94	0.78678
P+W+A	346.29	798.12	109.4	590.14	11.108	153.73	208.51	670.55	0.74391
P+W+S	330.76	846.29	98.519	527.74	15.746	153.45	192.99	660.8	0.72404
(P+W)/T	309.35	872.66	77.397	618.49	16.034	153.99	171.58	723.56	0.64368
PT/TIS	304.37	884.81	73.321	630.09	16.938	154.44	166.6	736.75	0.62326
LPT	2552.3	53479	2315.8	53316	11.888	150.69	2414.3	53388	0.80705
MWKR	468.44	20451	266.53	20294	46.082	158.43	330.67	20363	0.36638
MTWF	423.13	21209	227.38	21073	52.231	159.15	285.36	21134	0.30936
HRN	326.73	992.68	90.839	680.93	12.1	152.63	188.95	811.88	0.70338
(S/RW)+S	2508.6	54210	2279.5	54038	19.366	151.64	2370.5	54114	0.73168
CR+SPT	1949.1	48171	1738.2	48003	37.303	155.51	1811.2	48078	0.4948
COVERT	307.8	11166	69.735	10877	9.9199	153.97	170.03	11005	0.66606

Table A5.27 The performances of DRs in 90U and TDD for Problem 3.2

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	199.46	3110.2	22.778	2900.3	32.143	114.16	68.945	2978.7	0.21561
FCFS	224	466.84	31.914	297.08	16.738	114.02	93.487	350.03	0.54238
LWKR	213.49	5420.1	36.44	5138.8	31.775	111.95	82.975	5244.3	0.20901
LTWF	215.29	5470.7	38.726	5175	32.255	112.19	84.781	5285.9	0.20122
EDD	220.88	521.82	28.031	255.97	15.978	111.83	90.363	349.21	0.51552
PRW	209.46	3711.7	34.413	3559.2	33.772	114.38	78.952	3616	0.25528
SLACK	222.52	509.74	28.578	244.95	14.878	111.8	92.01	334.94	0.52819
CR	222.02	584.78	27.38	319.67	14.188	113.58	91.503	415.36	0.54658
MDD	216.55	4664.1	25.402	4388.8	17.676	111.77	86.038	4491.9	0.35375
AT-RPT	226.38	457.48	34.885	307.89	17.334	114.34	95.862	356.83	0.56547
P+W+A	217.38	463.02	26.961	277.73	18.403	113.9	86.868	336.59	0.49743
P+W+S	208.5	491.61	22.267	229.16	22.588	113.99	77.99	320.2	0.47227
(P+W)/T	206.77	501.1	19.727	288.23	21.778	114.15	76.259	362.08	0.41812
PT/TIS	205.7	509.17	19.065	295.09	22.186	114.08	75.189	365.77	0.40595
LPT	463.94	10747	271.33	10589	16.22	111.19	333.42	10648	0.62115
MWKR	250.95	5721.8	76.562	5580.3	34.436	114.09	120.43	5633.3	0.32615
MTWF	240.74	5740.4	69.726	5615.1	37.812	114.88	110.22	5662.1	0.28423
HRN	213.38	558	23.078	308.13	18.522	113.1	82.866	397.52	0.468
(S/RW)+S	460	10983	270.08	10826	18.902	111.03	329.49	10885	0.59126
CR+SPT	420.23	10682	239.88	10532	28.481	111.55	289.71	10588	0.43324
COVERT	212.38	2480	19.19	2264.1	15.628	114.32	81.872	2344.3	0.47441

Table A5.28 The performances of DRs in 90U and LDD for Problem 3.2

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	199.46	3110.2	18.27	2874.4	53.739	152.74	68.945	2978.7	0.13343
FCFS	224	466.84	20.62	280.1	31.547	152.97	93.487	350.03	0.38178
LWKR	213.49	5420.1	31.276	5103.6	52.716	150.27	82.975	5244.3	0.14274
LTWF	215.29	5470.7	33.593	5138	53.226	150.44	84.781	5285.9	0.1386
EDD	220.56	530.98	16.71	228.46	31.074	150.3	90.05	356.56	0.34097
PRW	209.46	3711.7	29.389	3540.6	54.851	153.02	78.952	3616	0.1809
SLACK	221.99	517.1	16.936	219.46	29.875	150.28	91.475	344.33	0.34724
CR	222.73	578.02	16.289	282.93	28.483	151.18	92.22	408.51	0.36233
MDD	217.92	3841.2	15.218	3527.4	32.226	150.18	87.407	3666.5	0.22649
AT-RPT	226.38	457.48	23.072	292.11	31.624	153.57	95.862	356.83	0.41188
P+W+A	217.38	463.02	16.785	260.79	34.331	152.5	86.868	336.59	0.33628
P+W+S	205.68	496.01	13.033	197.82	42.278	152.74	75.169	323.13	0.3205
(P+W)/T	206.77	501.1	11.41	265.98	39.565	152.72	76.259	362.08	0.26048
PT/TIS	205.7	509.17	10.98	273.33	40.205	152.55	75.189	365.77	0.24933
LPT	463.94	10747	256.87	10570	27.865	150.13	333.42	10648	0.5475
MWKR	250.95	5721.8	70.034	5562.6	54.013	153.05	120.43	5633.3	0.26136
MTWF	240.74	5740.4	64.232	5599.4	58.422	154.12	110.22	5662.1	0.22752
HRN	213.38	558	13.5	280.72	35.048	151.48	82.866	397.52	0.29704
(S/RW)+S	451.2	10741	250.91	10569	34.635	149.8	320.69	10646	0.49703
CR+SPT	392.58	10335	205.59	10167	47.948	151.26	262.06	10241	0.33711
COVERT	217.28	2331.7	11.724	2087.1	29.374	153.17	86.764	2194.9	0.31343

Table A5.29 The performances of DRs in 85U and TDD for Problem 3.2

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	165.65	1516.2	7.9353	1301.7	39.54	108.19	42.365	1381.5	0.11901
FCFS	176.66	347.59	8.7214	184.33	29.308	107.47	53.383	234.6	0.27359
LWKR	171.6	2281.9	12.93	2011.2	38.581	107.22	48.32	2112.7	0.12408
LTWF	172.51	2516.2	13.918	2242.1	38.653	107.74	49.234	2344.9	0.12312
EDD	174.28	401.99	6.3153	146.12	29.286	107.53	50.999	236.9	0.22465
PRW	170.43	1678.6	13.127	1518.7	39.944	108.71	47.153	1578.6	0.16081
SLACK	175.44	388.69	6.3114	137.16	28.117	107.58	52.163	224.22	0.22883
CR	176.22	423.27	6.1512	170.57	27.184	107.45	52.936	259.53	0.24489
MDD	173.33	1894.7	6.1846	1630.9	30.099	107.33	50.055	1729	0.1555
AT-RPT	178.8	337.75	11.059	200.04	29.507	108.1	55.522	246.96	0.31644
P+W+A	173.25	348.23	6.9417	171.93	30.946	107.86	49.966	226.35	0.23633
P+W+S	167.7	379.88	5.6514	126.71	35.197	108.09	44.424	215.93	0.22248
(P+W)/T	168.83	381.55	5.1219	179.71	33.546	107.84	45.546	247.06	0.18967
PT/TIS	168.45	387.93	4.993	187.54	33.79	107.88	45.173	252.37	0.1839
LPT	229.8	2852.4	60.18	2691.7	27.627	106.79	106.52	2751.5	0.3846
MWKR	188.7	2364.9	30.692	2225	39.239	108.16	65.424	2277.2	0.23514
MTWF	184.31	2349.1	28.755	2226.4	41.697	107.85	61.028	2272.3	0.20944
HRN	172.09	412.26	5.7164	178.99	30.88	107.55	48.806	258.91	0.21147
(S/RW)+S	228.34	2911.3	60.535	2752.5	29.443	106.74	105.06	2811.1	0.37323
CR+SPT	220.64	3105.1	59.203	2947.7	35.815	107.13	97.358	3006.2	0.28572
COVERT	174.61	1111.2	4.964	890.2	27.604	108.23	51.33	970.92	0.23113

Table A5.30 The performances of DRs in 85U and LDD for Problem 3.2

Dispatching Rules	Mean Flow	Max Flow	Mean Tardiness	Max Tardiness	Mean Earliness	Max Earliness	Mean Waiting	Max Waiting	Proportion of Tardy Jobs
SPT	165.65	1516.2	5.7002	1275.1	61.962	144.63	42.365	1381.5	0.06722
FCFS	176.66	347.59	4.2288	168.62	49.473	143.96	53.383	234.6	0.14428
LWKR	171.6	2281.9	10.113	1977.4	60.42	143.7	48.32	2112.7	0.07859
LTWF	172.51	2516.2	11.025	2207.8	60.417	144.3	49.234	2344.9	0.07904
EDD	174.17	412.83	2.4883	124.28	50.227	143.96	50.887	246.68	0.09661
PRW	170.43	1678.6	10.28	1498.8	61.754	145.51	47.153	1578.6	0.10577
SLACK	175.17	395.97	2.4284	113.03	49.165	144.17	51.89	231.34	0.09633
CR	176.73	424.08	2.3677	136.89	47.54	143.88	53.454	258.64	0.1047
MDD	173.75	1416.3	2.4232	1124.3	50.58	143.77	50.469	1253.8	0.06586
AT-RPT	178.8	337.75	5.7592	184.83	48.864	144.95	55.522	246.96	0.18268
P+W+A	173.25	348.23	3.1677	154.83	51.829	144.51	49.966	226.35	0.11651
P+W+S	166.85	384.98	2.8574	99.343	57.914	144.52	43.57	219.5	0.12359
(P+W)/T	168.83	381.55	2.1487	158.34	55.23	144.28	45.546	247.06	0.08467
PT/TIS	168.45	387.93	2.0964	167.76	55.551	144.34	45.173	252.37	0.08126
LPT	229.8	2852.4	52.314	2671.7	44.418	143.34	106.52	2751.5	0.30666
MWKR	188.7	2364.9	26.426	2207.8	59.63	145.01	65.424	2277.2	0.17896
MTWF	184.31	2349.1	25.063	2211.2	62.662	144.52	61.028	2272.3	0.16092
HRN	172.09	412.26	2.3299	154.69	52.151	143.68	48.806	258.91	0.09215
(S/RW)+S	226.83	2985.4	53.32	2809.4	48.391	143.12	103.55	2886	0.29147
CR+SPT	215.31	3072.1	49.762	2904.4	56.359	143.78	92.029	2979	0.21145
COVERT	176.58	964.13	2.0656	712.78	47.388	144.57	53.304	822.88	0.10849

Appendix A6. The state intervals of the problems

State intervals for Problem 1.1

Table A6.1 The state intervals for 95% utilization and LDD setting of Problem 1.1

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 10.21	1
NQ≥2 and 3.64 ≤ MST<10.21	2
NQ≥2 and -5 ≤ MST<3.64	3
NQ≥2 and -18.68 ≤ MST<-5	4
NQ≥2 and -46.05 ≤ MST<-18.68	5
NQ≥2 and -93.74 ≤ MST<-46.05	6
NQ≥2 and -189.48 ≤ MST<-93.74	7
NQ≥2 and -395.89 ≤ MST<-189.48	8
NQ≥2 and -732.66 ≤ MST<-395.89	9
NQ≥2 and MST<-732.66	10

Table A6.2 The state intervals for 90% utilization and TDD setting of Problem 1.1

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 3.62	1
NQ≥2 and 0 ≤ MST<3.62	2
NQ≥2 and -4.16 ≤ MST<0	3
NQ≥2 and -9.69 ≤ MST<-4.16	4
NQ≥2 and -17.97 ≤ MST<-9.69	5
NQ≥2 and -32.03 ≤ MST<-17.97	6
NQ≥2 and -55.37 ≤ MST<-32.03	7
NQ≥2 and -95.61 ≤ MST<-55.37	8
NQ≥2 and -173.78 ≤ MST<-95.61	9
NQ≥2 and MST<-173.78	10

Table A6.3 The state intervals for 90% utilization and LDD setting of Problem 1.1

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 12.7	1
NQ≥2 and 9.43 ≤ MST<12.7	2
NQ≥2 and 5.79 ≤ MST<9.43	3
NQ≥2 and 1.29 ≤ MST<5.79	4
NQ≥2 and -4.41 ≤ MST<1.29	5
NQ≥2 and -12.34 ≤ MST<-4.41	6
NQ≥2 and -26.5 ≤ MST<-12.34	7
NQ≥2 and -55.45 ≤ MST<-26.5	8
NQ≥2 and -121.39 ≤ MST<-55.45	9
NQ≥2 and MST<-121.39	10

Table A6.4 The state intervals for 85% utilization and TDD setting of Problem 1.1

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 4.78	1
NQ≥2 and 2.76 ≤ MST<4.78	2
NQ≥2 and 0.54 ≤ MST<2.76	3
NQ≥2 and -2.23 ≤ MST<0.54	4
NQ≥2 and -5.93 ≤ MST<-2.23	5
NQ≥2 and -10.93 ≤ MST<-5.93	6
NQ≥2 and -19.03 ≤ MST<-10.93	7
NQ≥2 and -34.87 ≤ MST<-19.03	8
NQ≥2 and -69.79 ≤ MST<-34.87	9
NQ≥2 and MST<-69.79	10

Table A6.5 The state intervals for 85% utilization and LDD setting of Problem 1.1

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 13.26	1
NQ≥2 and 11.36 ≤ MST<13.26	2
NQ≥2 and 9.44 ≤ MST<11.36	3
NQ≥2 and 7.24 ≤ MST<9.44	4
NQ≥2 and 4.53 ≤ MST<7.24	5
NQ≥2 and 0.98 ≤ MST<4.53	6
NQ≥2 and -4.22 ≤ MST<0.98	7
NQ≥2 and -13.08 ≤ MST<-4.22	8
NQ≥2 and -38.41 ≤ MST<-13.08	9
NQ≥2 and MST<-38.41	10

State intervals for Problem 1.2

Table A6.6 The state intervals for 95% utilization and TDD setting of Problem 1.2

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥-0.25	1
NQ≥2 and -9.12 ≤ MST<-0.25	2
NQ≥2 and -22.12 ≤ MST<-9.12	3
NQ≥2 and -45.67 ≤ MST<-22.12	4
NQ≥2 and -79.45 ≤ MST<-45.67	5
NQ≥2 and -150.73 ≤ MST<-79.45	6
NQ≥2 and -283.28 ≤ MST<-150.73	7
NQ≥2 and -505.06 ≤ MST<-283.28	8
NQ≥2 and -876.5 ≤ MST<-505.06	9
NQ≥2 and MST<-876.5	10

Table A6.7 The state intervals for 95% utilization and LDD setting of Problem 1.2

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 9.45	1
NQ≥2 and 3.88 ≤ MST<9.45	2
NQ≥2 and -7.12 ≤ MST<3.88	3
NQ≥2 and -21.15 ≤ MST<-7.12	4
NQ≥2 and -48.02 ≤ MST<-21.15	5
NQ≥2 and -96.77 ≤ MST<-48.02	6
NQ≥2 and -189.43 ≤ MST<-96.77	7
NQ≥2 and -398.75 ≤ MST<-189.43	8
NQ≥2 and -734.26 ≤ MST<-398.75	9
NQ≥2 and MST<-734.26	10

Table A6.8 The state intervals for 90% utilization and TDD setting of Problem 1.2

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 5.79	1
NQ≥2 and -1.29 ≤ MST<5.79	2
NQ≥2 and -5.68 ≤ MST<-1.29	3
NQ≥2 and -9.79 ≤ MST<-5.68	4
NQ≥2 and -19.27 ≤ MST<-9.79	5
NQ≥2 and -35.48 ≤ MST<-19.27	6
NQ≥2 and -58.93 ≤ MST<-35.48	7
NQ≥2 and -92.28 ≤ MST<-58.93	8
NQ≥2 and -175.77 ≤ MST<-92.28	9
NQ≥2 and MST<-175.77	10

Table A6.9 The state intervals for 90% utilization and LDD setting of Problem 1.2

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 15.6	1
NQ≥2 and 10.21 ≤ MST<15.6	2
NQ≥2 and 6.78 ≤ MST<10.21	3
NQ≥2 and 2.35 ≤ MST<6.78	4
NQ≥2 and -5.31 ≤ MST<2.35	5
NQ≥2 and -13.48 ≤ MST<-5.31	6
NQ≥2 and -25.79 ≤ MST<-13.48	7
NQ≥2 and -56.72 ≤ MST<-25.79	8
NQ≥2 and -124.25 ≤ MST<-56.72	9
NQ≥2 and MST<-124.25	10

Table A6.10 The state intervals for 85% utilization and TDD setting of Problem 1.2

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 5.47	1
NQ≥2 and 3.67 ≤ MST<5.47	2
NQ≥2 and 1.26 ≤ MST<3.67	3
NQ≥2 and -3.14 ≤ MST<1.26	4
NQ≥2 and -6.85 ≤ MST<-3.14	5
NQ≥2 and -11.87 ≤ MST<-6.85	6
NQ≥2 and -18.76 ≤ MST<-11.87	7
NQ≥2 and -35.32 ≤ MST<-18.76	8
NQ≥2 and -70.18 ≤ MST<-35.32	9
NQ≥2 and MST<-70.18	10

Table A6.11 The state intervals for 85% utilization and LDD setting of Problem 1.2

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 12.96	1
NQ≥2 and 10.54 ≤ MST <12.96	2
NQ≥2 and 8.73 ≤ MST <10.54	3
NQ≥2 and 7.24 ≤ MST <8.73	4
NQ≥2 and 3.19 ≤ MST <7.24	5
NQ≥2 and -1.12 ≤ MST <3.19	6
NQ≥2 and -3.27 ≤ MST <-1.12	7
NQ≥2 and -12.16 ≤ MST <-3.27	8
NQ≥2 and -38.41 ≤ MST <-12.16	9
NQ≥2 and MST <-38.41	10

State intervals for Problem 2.1

Table A6.12 The state intervals for 95% utilization and TDD setting of Problem 2.1

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 24.3	1
NQ≥2 and 12.78 ≤ MST<24.3	2
NQ≥2 and 1.37 ≤ MST<12.78	3
NQ≥2 and -12.15 ≤ MST<1.37	4
NQ≥2 and -33.51≤ MST<-12.15	5
NQ≥2 and -71.52 ≤ MST<-33.51	6
NQ≥2 and -142.69 ≤ MST<-71.52	7
NQ≥2 and -292.52 ≤ MST<-142.69	8
NQ≥2 and -708.8 ≤ MST<-292.52	9
NQ≥2 and MST<-708.8	10

Table A6.13 The state intervals for 95% utilization and LDD setting of Problem 2.1

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 35.74	1
NQ≥2 and 24.15 ≤ MST<35.74	2
NQ≥2 and 12.4 ≤ MST<24.15	3
NQ≥2 and -0.85≤ MST<12.4	4
NQ≥2 and -17.9≤ MST<-0.85	5
NQ≥2 and -47.52≤ MST<-17.9	6
NQ≥2 and -105.19≤ MST<-47.52	7
NQ≥2 and -243.54≤ MST<-105.19	8
NQ≥2 and -564.06≤ MST<-243.54	9
NQ≥2 and MST<-564.06	10

Table A6.14 The state intervals for 90% utilization and TDD setting of Problem 2.1

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 29.09	1
NQ≥2 and 23.46≤ MST<29.09	2
NQ≥2 and 17.7≤ MST<23.46	3
NQ≥2 and 11.38≤ MST<17.7	4
NQ≥2 and 4≤ MST<11.38	5
NQ≥2 and -5≤ MST<4	6
NQ≥2 and -18.52 ≤ MST<-5	7
NQ≥2 and -44.82 ≤ MST<-18.52	8
NQ≥2 and -115.88 ≤ MST<-44.82	9
NQ≥2 and MST<-115.88	10

Table A6.15 The state intervals for 90% utilization and LDD setting of Problem 2.1

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 39.98	1
NQ≥2 and 33.93 ≤ MST<39.98	2
NQ≥2 and 28.19 ≤ MST<33.93	3
NQ≥2 and 22.06 ≤ MST<28.19	4
NQ≥2 and 14.73≤ MST<22.06	5
NQ≥2 and 5.93 ≤ MST<14.73	6
NQ≥2 and -6.3≤ MST<5.93	7
NQ≥2 and -28.21≤ MST<-6.3	8
NQ≥2 and -92.12≤ MST<-28.21	9
NQ≥2 and MST<-92.12	10

Table A6.16 The state intervals for 85% utilization and TDD setting of Problem 2.1

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 30.05	1
NQ≥2 and 25.98 ≤ MST< 30.05	2
NQ≥2 and 22.3 ≤ MST< 25.98	3
NQ≥2 and 18.24 ≤ MST< 22.3	4
NQ≥2 and 13.72 ≤ MST< 18.24	5
NQ≥2 and 8.31 ≤ MST< 13.72	6
NQ≥2 and 1.62 ≤ MST< 8.31	7
NQ≥2 and -7.92 ≤ MST<1.62	8
NQ≥2 and -29.49 ≤ MST<-7.92	9
NQ≥2 and MST<-29.49	10

Table A6.17 The state intervals for 85% utilization and LDD setting of Problem 2.1

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 40.41	1
NQ≥2 and 35.87 ≤ MST< 40.41	2
NQ≥2 and 32.04 ≤ MST< 35.87	3
NQ≥2 and 27.95 ≤ MST< 32.04	4
NQ≥2 and 23.53 ≤ MST< 27.95	5
NQ≥2 and 18.68 ≤ MST< 23.53	6
NQ≥2 and 12.83 ≤ MST< 18.68	7
NQ≥2 and 4.54 ≤ MST< 12.83	8
NQ≥2 and -11.21 ≤ MST<4.54	9
NQ≥2 and MST<-11.21	10

State intervals for Problem 2.2

Table A6.18 The state intervals for 95% utilization and TDD setting of Problem 2.2

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 24.12	1
NQ≥2 and 14 ≤ MST< 24.12	2
NQ≥2 and 3.83 ≤ MST< 14	3
NQ≥2 and -7.74≤ MST< 3.83	4
NQ≥2 and -24.38≤ MST<-7.74	5
NQ≥2 and -52.47≤ MST<-24.38	6
NQ≥2 and -100.76≤ MST<-52.47	7
NQ≥2 and -190.89≤ MST<-100.76	8
NQ≥2 and -390.84≤ MST<-190.89	9
NQ≥2 and MST<-390.84	10

Table A6.19 The state intervals for 95% utilization and LDD setting of Problem 2.2

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 39.8	1
NQ≥2 and 29.68 ≤ MST<39.8	2
NQ≥2 and 20.37 ≤ MST<29.68	3
NQ≥2 and 10.14 ≤ MST<20.37	4
NQ≥2 and -2.94≤ MST<10.14	5
NQ≥2 and -23.38≤ MST<-2.94	6
NQ≥2 and -60.96≤ MST<-23.38	7
NQ≥2 and -141.8≤ MST<-60.96	8
NQ≥2 and -320.02≤ MST<-141.8	9
NQ≥2 and MST<-320.02	10

Table A6.20 The state intervals for 90% utilization and TDD setting of Problem 2.2

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 27.24	1
NQ≥2 and 21 ≤ MST<27.24	2
NQ≥2 and 15.25≤ MST<21	3
NQ≥2 and 9.08≤ MST<15.25	4
NQ≥2 and 1.86≤ MST<9.08	5
NQ≥2 and -7.52≤ MST<1.86	6
NQ≥2 and -22.57≤ MST<-7.52	7
NQ≥2 and -53.34 ≤ MST<-22.57	8
NQ≥2 and -135.89≤ MST<-53.34	9
NQ≥2 and MST<-135.89	10

Table A6.21 The state intervals for 90% utilization and LDD setting of Problem 2.2

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 42.16	1
NQ≥2 and 35.34 ≤ MST<42.16	2
NQ≥2 and 29.28 ≤ MST<33.93	3
NQ≥2 and 23.07 ≤ MST<28.19	4
NQ≥2 and 16.59 ≤ MST<22.06	5
NQ≥2 and 8.61 ≤ MST<14.73	6
NQ≥2 and -3.33 ≤ MST<5.93	7
NQ≥2 and -26.78 ≤ MST<-6.3	8
NQ≥2 and -88.37 ≤ MST<-28.21	9
NQ≥2 and MST<-88.37	10

Table A6.22 The state intervals for 85% utilization and TDD setting of Problem 2.2

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 28.35	1
NQ≥2 and 23.93 ≤ MST< 28.35	2
NQ≥2 and 19.92 ≤ MST< 23.93	3
NQ≥2 and 15.87 ≤ MST< 19.92	4
NQ≥2 and 11.59 ≤ MST< 15.87	5
NQ≥2 and 6.66 ≤ MST< 11.59	6
NQ≥2 and 0.14 ≤ MST< 6.66	7
NQ≥2 and -9.83≤ MST< 0.14	8
NQ≥2 and -33.57≤ MST<-9.83	9
NQ≥2 and MST<-33.57	10

Table A6.23 The state intervals for 85% utilization and LDD setting of Problem 2.2

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 42.63	1
NQ≥2 and 37.7 ≤ MST< 42.63	2
NQ≥2 and 33.36 ≤ MST< 37.7	3
NQ≥2 and 29.09 ≤ MST< 33.36	4
NQ≥2 and 24.6 ≤ MST< 29.09	5
NQ≥2 and 19.62 ≤ MST< 24.6	6
NQ≥2 and 13.71≤ MST< 19.62	7
NQ≥2 and 5.3 ≤ MST< 13.71	8
NQ≥2 and -11.62≤ MST<5.3	9
NQ≥2 and MST<-11.62	10

State intervals for Problem 3.1

Table A6.24 The state intervals for 95% utilization and TDD setting of Problem 3.1

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 56.91	1
NQ≥2 and 40.75 ≤ MST< 56.91	2
NQ≥2 and 24.87 ≤ MST< 40.75	3
NQ≥2 and 6.5 ≤ MST< 24.87	4
NQ≥2 and -16.49≤ MST< 6.5	5
NQ≥2 and -51.52≤ MST<-16.49	6
NQ≥2 and -117.53≤ MST<-51.52	7
NQ≥2 and -257.71 ≤ MST< -117.53	8
NQ≥2 and -587.31≤ MST<-257.71	9
NQ≥2 and MST<-587.31	10

Table A6.25 The state intervals for 95% utilization and LDD setting of Problem 3.1

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 83.05	1
NQ≥2 and 66.31≤ MST<83.05	2
NQ≥2 and 50.04≤ MST<66.31	3
NQ≥2 and 32.65≤ MST<50.04	4
NQ≥2 and 11.97≤ MST<32.65	5
NQ≥2 and -16.2≤ MST<11.97	6
NQ≥2 and -66.71≤ MST<-16.2	7
NQ≥2 and -174.27≤ MST<-66.71	8
NQ≥2 and -437.91≤ MST<-174.27	9
NQ≥2 and MST<-437.91	10

Table A6.26 The state intervals for 90% utilization and TDD setting of Problem 3.1

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 64.34	1
NQ≥2 and 55.63≤ MST<64.34	2
NQ≥2 and 47.71≤ MST<55.63	3
NQ≥2 and 39.41≤ MST<47.71	4
NQ≥2 and 30.04≤ MST<39.41	5
NQ≥2 and 19.08≤ MST<30.04	6
NQ≥2 and 5.96≤ MST<19.08	7
NQ≥2 and -11.56≤ MST<5.96	8
NQ≥2 and -39.66≤ MST<-11.56	9
NQ≥2 and MST<-39.66	10

Table A6.27 The state intervals for 90% utilization and LDD setting of Problem 3.1

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 90.25	1
NQ≥2 and 81.05 ≤ MST< 90.25	2
NQ≥2 and 72.73 ≤ MST< 81.05	3
NQ≥2 and 64.05 ≤ MST< 72.73	4
NQ≥2 and 54.97 ≤ MST< 64.05	5
NQ≥2 and 44.54 ≤ MST< 54.97	6
NQ≥2 and 32.11 ≤ MST< 44.54	7
NQ≥2 and 15.36 ≤ MST< 32.11	8
NQ≥2 and -11.54≤ MST< 15.36	9
NQ≥2 and MST< -11.54	10

Table A6.28 The state intervals for 85% utilization and TDD setting of Problem 3.1

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 64.57	1
NQ≥2 and 58.66 ≤ MST< 64.57	2
NQ≥2 and 53.21 ≤ MST< 58.66	3
NQ≥2 and 47.57 ≤ MST< 53.21	4
NQ≥2 and 41.40 ≤ MST< 47.57	5
NQ≥2 and 34.37 ≤ MST< 41.40	6
NQ≥2 and 26.03 ≤ MST< 34.37	7
NQ≥2 and 15.19 ≤ MST< 26.03	8
NQ≥2 and -2.38≤ MST< 15.19	9
NQ≥2 and MST<-2.38	10

Table A6.29 The state intervals for 85% utilization and LDD setting of Problem 3.1

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 89.43	1
NQ≥2 and 82.65 ≤ MST< 89.43	2
NQ≥2 and 76.89 ≤ MST< 82.65	3
NQ≥2 and 70.97 ≤ MST< 76.89	4
NQ≥2 and 64.6 ≤ MST< 70.97	5
NQ≥2 and 57.61 ≤ MST< 64.6	6
NQ≥2 and 49.29 ≤ MST< 57.61	7
NQ≥2 and 38.57≤ MST< 49.29	8
NQ≥2 and 21.67≤ MST< 38.57	9
NQ≥2 and MST<21.67	10

State intervals for Problem 3.2

Table A6.30 The state intervals for 95% utilization and TDD setting of Problem 3.2

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 58.98	1
NQ≥2 and 42.3 ≤ MST< 58.98	2
NQ≥2 and 26.03 ≤ MST< 42.3	3
NQ≥2 and 7.19 ≤ MST< 26.03	4
NQ≥2 and -15.63 ≤ MST< 7.19	5
NQ≥2 and -48.97 ≤ MST< -15.63	6
NQ≥2 and -113.79 ≤ MST< -48.97	7
NQ≥2 and -250.28 ≤ MST< -113.79	8
NQ≥2 and -572.44 ≤ MST< -250.28	9
NQ≥2 and MST< -572.44	10

Table A6.31 The state intervals for 95% utilization and LDD setting of Problem 3.2

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 86.25	1
NQ≥2 and 68.77 ≤ MST< 8 6.25	2
NQ≥2 and 52.44 ≤ MST< 68.77	3
NQ≥2 and 35.46 ≤ MST< 52.44	4
NQ≥2 and 15.68 ≤ MST< 35.46	5
NQ≥2 and -12.29 ≤ MST< 15.68	6
NQ≥2 and -56.43 ≤ MST< -12.29	7
NQ≥2 and -161.08 ≤ MST< -56.43	8
NQ≥2 and -408.57 ≤ MST< -161.08	9
NQ≥2 and MST< -408.57	10

Table A6.32 The state intervals for 90% utilization and TDD setting of Problem 3.2

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 65.78	1
NQ≥2 and 56.25 ≤ MST< 65.78	2
NQ≥2 and 48 ≤ MST< 56.25	3
NQ≥2 and 39.99 ≤ MST< 48	4
NQ≥2 and 32.14 ≤ MST< 39.99	5
NQ≥2 and 22.94 ≤ MST< 32.14	6
NQ≥2 and 10.86 ≤ MST< 22.94	7
NQ≥2 and -7.07 ≤ MST< 10.86	8
NQ≥2 and -40.36 ≤ MST< -7.07	9
NQ≥2 and MST<-40.36	10

Table A6.33 The state intervals for 90% utilization and LDD setting of Problem 3.2

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 92.95	1
NQ≥2 and 82.65 ≤ MST< 92.95	2
NQ≥2 and 73.71 ≤ MST< 82.65	3
NQ≥2 and 65.04 ≤ MST< 73.71	4
NQ≥2 and 56.37 ≤ MST< 65.04	5
NQ≥2 and 47.19 ≤ MST< 56.37	6
NQ≥2 and 36.36 ≤ MST< 47.19	7
NQ≥2 and 21.49 ≤ MST< 36.36	8
NQ≥2 and -9.2 ≤ MST< 21.49	9
NQ≥2 and MST<-9.2	10

Table A6.34 The state intervals for 85% utilization and TDD setting of Problem 3.2

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 66.59	1
NQ≥2 and 59.82 ≤ MST< 66.59	2
NQ≥2 and 53.87 ≤ MST< 59.82	3
NQ≥2 and 48.11 ≤ MST< 53.87	4
NQ≥2 and 42.29 ≤ MST< 48.11	5
NQ≥2 and 35.88 ≤ MST< 42.29	6
NQ≥2 and 28.34 ≤ MST< 35.88	7
NQ≥2 and 18.4 ≤ MST< 28.34	8
NQ≥2 and 1.1 ≤ MST< 18.4	9
NQ≥2 and MST< 1.1	10

Table A6.35 The state intervals for 85% utilization and LDD setting of Problem 3.2

Determination Criteria	State Number
NQ<2	Dummy State
NQ≥2 and MST ≥ 92.53	1
NQ≥2 and 84.84 ≤ MST< 92.53	2
NQ≥2 and 78.37 ≤ MST< 84.84	3
NQ≥2 and 72.22 ≤ MST< 78.37	4
NQ≥2 and 65.87 ≤ MST< 72.22	5
NQ≥2 and 58.93 ≤ MST< 65.87	6
NQ≥2 and 50.83 ≤ MST< 58.93	7
NQ≥2 and 41.04 ≤ MST< 50.83	8
NQ≥2 and 25.67 ≤ MST< 41.04	9
NQ≥2 and MST<25.67	10

Appendix A7. ARENA model of the developed Q-learning model for Problem1.1

Mod file of the ARENA model

```
0$              CREATE,         1:Expo(Arrival,1):MARK(TimeIn):NEXT(71$);

71$             ASSIGN:         i3=1:
                                dd=TNOW:
                                ddQ(i3)=TNOW:
                                TW=0:MARK(av_Wait);
72$             WHILE:          i3<=totalstagenumber;
2$              ASSIGN:         ept(i3)=Util(i3):
                                dd=dd+TWK*ept(i3):
                                ddQ(i3)=dd:
                                TW=TW+ept(i3):
                                RW=TW:
                                i3=i3+1;
73$             ENDWHILE;
1$              ASSIGN:         s_number=s_number+1:
                                seizedmachine=0:
                                pt=ept(s_number):
                                PRW=pt/RW:
                                min_util=5:
                                NR_s=1:
                                entrytime_to_s(s_number)=TNOW;
3$              BRANCH,         1:
                                If,NQ(s_number)>=1,4$,Yes:
                                Else,10$,Yes;
4$              QUEUE,          QueuesHFS(s_number):DETACH;
10$             ASSIGN:         i2=control((2*s_number-1));
5$              WHILE:          i2<=control(2*s_number);
6$              BRANCH,         1:
                                If,(NR(i2)==0).AND.(DAVG(i2)<min_util),8$,Yes:
                                Else,9$,Yes;
8$              ASSIGN:         seizedmachine=i2:
                                min_util=DAVG(i2);
9$              ASSIGN:         i2=i2+1;
7$              ENDWHILE;
58$             BRANCH,         1:
                                If,seizedmachine==0,4$,Yes:
                                Else,96$,Yes;
96$             ASSIGN:         sta0=0;
11$             SEIZE,          1,Other:
                                seizedmachine,1:NEXT(87$);

87$             BRANCH,         1:
                                If,TNOW>=15000,85$,Yes:
                                Else,12$,Yes;
85$             TALLY:          4+s_number,int(av_Wait),1;
86$             ASSIGN:         av_wait_in_Q(s_number)=TAVG(4+s_number);
120$            WRITE,          20+s_number,"%8.6f\n":
                                av_wait_in_Q(s_number);
12$             DELAY:          pt,,Other:NEXT(13$);

13$             RELEASE:        seizedmachine,1:MARK(av_Wait);
118$            WRITE,          TardinessofJ,"%8.6f\n":
                                est_tardiness;
14$             ASSIGN:         RW=RW-pt;
113$            BRANCH,         1:
                                If,TNOW<15000,59$,Yes:
                                Else,121$,Yes;
```

```
59$             BRANCH,         1:
                                If,NQ(s_number)>1,64$,Yes:
                                If,NQ(s_number)==1,116$,Yes:
                                Else,stagecontrol,Yes;
64$             ASSIGN:         JinQ=NQ(s_number);
60$             WHILE:          NQ(s_number)>0;
61$             REMOVE:         1,QueuesHFS(s_number),AssignDR;
62$             ENDWHILE;
63$             ASSIGN:         jQ=0;
65$             WAIT:           s_number;
68$             SEARCH,
QueuesHFS(s_number),1,NQ(s_number):MIN(A(DR(1)));
67$             ASSIGN:         sm_s=seizedmachine;
66$             REMOVE:         j,QueuesHFS(s_number),SeizeMachine;
stagecontrol    BRANCH,         1:
                                If,s_number<totalstagenumber,1$,Yes:
                                Else,17$,Yes;
17$             COUNT:          completedjobs,1;
23$             ASSIGN:         Tardiness=MX(TNOW-dd,0):
                                Earliness=MX(dd-TNOW,0):
                                WaitingTime=TNOW-TimeIn-TW;
21$             TALLY:          FlowTime,INT(TimeIn),1;
24$             TALLY:          Tardinessofjobs,Tardiness,1;
25$             TALLY:          Earlinessofjobs,Earliness,1;
26$             TALLY:          WaitingTimesofjobs,WaitingTime,1;
19$             BRANCH,         1:
                                If,TNOW>dd,20$,Yes:
                                Else,18$,Yes;
20$             COUNT:          TardyJobs,1;
22$             ASSIGN:         TotalTardiness=TotalTardiness+(TNOW-dd):
                                MeanTardiness=TotalTardiness/NC(1);
18$             DISPOSE:        No;

SeizeMachine    ASSIGN:         seizedmachine=sm_s:
                                act=v_act:
                                sta0=v_sta;
70$             SIGNAL:         111:NEXT(11$);

AssignDR        ASSIGN:         slack=dd-RW-TNOW:
                                jQ=jQ+1:
                                S_RW=slack/RW:
                                CR=(dd-TNOW)/RW:
                                MDD=MX(dd,TNOW+RW):
                                WT=TNOW-entrytime_to_s(s_number):
                                HRN=(WT+pt)/pt:
                                S_RPT=MX(S_RW*pt, pt):
                                CR_RPT=MX(CR*pt, pt):
                                AT_RPT=TimeIn-RW:
                                SL=MIN(slack,0):
                                TIS=TNOW-TimeIn:
                                PT_TIS=pt/((TIS<>0)*TIS+(TIS==0)*d_WT):
                                s_Lateness=TNOW-ddQ(s_number):
                                s_tardiness=MX(TNOW-ddQ(s_number),0);
78$             BRANCH,         1:
                                If,s_number<totalstagenumber,79$,Yes:
                                Else,80$,Yes;
79$             ASSIGN:         WINQ=SAQUE(s_number+1,1):
                                PT_WINQ=pt+WINQ:
                                PT_WINQ_AT=pt+WINQ+TimeIn:
                                PT_WINQ_SL=pt+WINQ+SL:

PT_WINQ_TIS=PT_WINQ/((TIS<>0)*TIS+(TIS==0)*d_WT):
                                EWT=((JinQ*ept(s_number))/#ofMachS(s_number)):
                                i4=s_number:
                                est_comp=TNOW+RW;
```

```
82$           WHILE:          i4<totalstagenumber;
81$           ASSIGN:         EWT=EWT+((NQ(i4+1)*ept(i4+1))/#ofMachS(i4+1)):
                              est_comp=est_comp+av_wait_in_Q(i4):
                              i4=i4+1;
84$           ENDWHILE;
83$           ASSIGN:         COVERT=(((EWT-
slack)/EWT)*((0<=slack).AND.(slack<EWT))+0*(slack>=EWT)+1*(slack<0))/pt:
                              est_tardiness=MX(est_comp-dd,0):
                              est_tardy=1*(est_tardiness>0):
                              est_lateness=est_comp-dd:
                              est_MEPT=est_comp-TNOW;
33$           BRANCH,         1:
                              If,jQ==JinQ,34$,Yes:
                              Else,4$,Yes;
34$           SIGNAL:         s_number:NEXT(4$);

80$           ASSIGN:         WINQ=0:
                              PT_WINQ=pt+WINQ:
                              PT_WINQ_AT=pt+WINQ+TimeIn:
                              PT_WINQ_SL=pt+WINQ+SL:

PT_WINQ_TIS=PT_WINQ/((TIS<>0)*TIS+(TIS==0)*d_WT):
                              EWT=((JinQ*ept(s_number))/#ofMachS(s_number)):
                              COVERT=(((EWT-
slack)/EWT)*((0<=slack).AND.(slack<EWT))+0*(slack>=EWT)+1*(slack<0))/pt:
                              est_comp=TNOW+RW+av_wait_in_Q(s_number):
                              est_tardiness=MX(est_comp-dd,0):
                              est_tardy=1*(est_tardiness>0):
                              est_lateness=est_comp-dd:
                              est_MEPT=est_comp-TNOW:NEXT(33$);

116$          ASSIGN:         sm_s=seizedmachine;
115$          REMOVE:
1,QueuesHFS(s_number),SeizeMachine1:NEXT(stagecontrol);

SeizeMachine1 ASSIGN:         seizedmachine=sm_s:
                              sta0=0:NEXT(11$);

121$          BRANCH,         1:
                              If,s_number==totalstagenumber,122$,Yes:
                              Else,123$,Yes;
122$          ASSIGN:         est_tardiness=MX(TNOW-dd,0):
                              est_flowtime=TNOW-TimeIn;
128$          WRITE,          est_comp_dif,"%8.6f\n":
                              est_comp-TNOW;
114$          BRANCH,         1:
                              If,NQ(s_number)>1,31$,Yes:
                              If,NQ(s_number)==1,16$,Yes:
                              Else,107$,Yes;
31$           ASSIGN:         JinQ=NQ(s_number);
27$           WHILE:          NQ(s_number)>0;
28$           REMOVE:         1,QueuesHFS(s_number),AssignDR;
29$           ENDWHILE;
30$           ASSIGN:         jQ=0;
32$           WAIT:           s_number;
74$           ASSIGN:
s_TL_before(s_number)=SAQUE(s_number,NSYM(s_Lateness)):

s_MT_before(s_number)=SAQUE(s_number,NSYM(s_tardiness))/NQ(s_number):

s_ML_before(s_number)=SAQUE(s_number,NSYM(s_Lateness))/NQ(s_number):

S_TP_before(s_number)=SAQUE(s_number,NSYM(est_tardy))/NQ(s_number):

est_TL_before(s_number)=SAQUE(s_number,NSYM(est_Lateness)):
```

```
est_MT_before(s_number)=SAQUE(s_number,NSYM(est_tardiness))/NQ(s_number):

est_MEPT_before(s_number)=SAQUE(s_number,NSYM(est_MEPT))/NQ(s_number):

est_ML_before(s_number)=SAQUE(s_number,NSYM(est_Lateness))/NQ(s_number):

mean_CR_before(s_number)=SAQUE(s_number,NSYM(CR))/NQ(s_number):
                         TRW_before(s_number)=SAQUE(s_number,NSYM(RW)):

MSL_before(s_number)=SAQUE(s_number,NSYM(slack))/NQ(s_number):

Pres_before(s_number)=est_TL_before(s_number)/TRW_before(s_number):
                         MSL=SAQUE(s_number,NSYM(slack))/NQ(s_number);
76$          SEARCH,     QueuesHFS(s_number),1,NQ(s_number):MAX(slack);
77$          ASSIGN:
MST_before(s_number)=AQUE(s_number,j,NSYM(slack));
75$          WRITE,      31,"%8.2f\n":
                         MSL;
35$          BRANCH,     1:
                         If,MSL>States(1),36$,Yes:
                         If,MSL<=States(1).and.MSL>States(2),37$,Yes:
                         If,MSL<=States(2).and.MSL>States(3),38$,Yes:
                         If,MSL<=States(3).and.MSL>States(4),39$,Yes:
                         If,MSL<=States(4).and.MSL>States(5),40$,Yes:
                         If,MSL<=States(5).and.MSL>States(6),41$,Yes:
                         If,MSL<=States(6).and.MSL>States(7),42$,Yes:
                         If,MSL<=States(7).and.MSL>States(8),43$,Yes:
                         If,MSL<=States(8).and.MSL>States(9),44$,Yes:
                         If,MSL<=States(9),45$,Yes;
36$          ASSIGN:     sta1=1;
140$         ASSIGN:     n_sta=n_sta+1;
141$         COUNT:      2+sta1,1;
46$          BRANCH,     1,10:
                         With,epsilon,51$,Yes:
                         With,1-epsilon,52$,Yes;
51$          COUNT:      randomselect,1;
47$          ASSIGN:     i5=AINT(1+UNIF(0,1,11)*totalaction#):
                         Qa=i5;
95$          BRANCH,     1:
                         If,i5==1,55$,Yes:
                         If,i5==2,55$,Yes;
55$          SEARCH,
QueuesHFS(s_number),1,NQ(s_number):MIN(A(DR(i5)));
54$          ASSIGN:     sm_s=seizedmachine:
                         v_act=i5:
                         v_sta=sta1;
53$          REMOVE:     j,QueuesHFS(s_number),SeizeMachine;
69$          WAIT:       111;
110$         BRANCH,     1:
                         If,sta0<>0,112$,Yes:
                         Else,111$,Yes;
112$         COUNT:      sta_c3,1;
56$          FINDJ,      1,totalaction#:max(Q(sta1,j));
assignreward BRANCH,     1:
                         If,act==1,131$,Yes:
                         Else,132$,Yes;
131$         ASSIGN:     reward=(worst_t-est_tardiness)/(worst_t-
best_t);
133$         BRANCH,     1:
                         If,est_tardiness>worst_t,134$,Yes:
                         If,est_tardiness<best_t,135$,Yes:
                         Else,129$,Yes;
134$         ASSIGN:     worst_t=est_tardiness;
129$         WRITE,      est_trd,"%8.6f\n":
```

```
                              est_tardiness;
130$          WRITE,          est_flw,"%8.6f\n":
                              est_flowtime;
103$          BRANCH,         1:
                              If,s_signal==4,104$,Yes:
                              Else,102$,Yes;
104$          SIGNAL:         4;
117$          ASSIGN:         s_signal=0:NEXT(assignQ);

assignQ       ASSIGN:         Q(sta0,act)=Q(sta0,act)+alpha1*(reward-
Q(sta0,act));
119$          SIGNAL:         21;
109$          BRANCH,         1:
                              If,NQ(s_number)==1,15$,Yes:
                              If,NQ(s_number)==0,stagecontrol,Yes;
15$           REMOVE:
1,QueuesHFS(s_number),SeizeMachine1:NEXT(stagecontrol);

102$          ASSIGN:         maxQvalue=Q(sta1,j):
                              Q(sta0,act)=(1-
alpha1)*Q(sta0,act)+alpha1*(reward+(gamma1)*maxQvalue);
142$          BRANCH,         1:
                              If,(TNOW<165000).AND.(TNOW>164900),143$,Yes:
                              Else,57$,Yes;
143$          WRITE,          Qvalue_GM,
                              "%8.2f  %8.2f\n %8.2f  %8.2f\n  %8.2f  %8.2f\n
%8.2f  %8.2f\n %8.2f  %8.2f\n %8.2f  %8.2f\n  %8.2f  %8.2f\n %8.2f  %8.2f\n
%8.2f  %8.2f\n %8.2f  %8.2f\n":
                              Q(1,1),
                              Q(1,2),
                              Q(2,1),
                              Q(2,2),
                              Q(3,1),
                              Q(3,2),
                              Q(4,1),
                              Q(4,2),
                              Q(5,1),
                              Q(5,2),
                              Q(6,1),
                              Q(6,2),
                              Q(7,1),
                              Q(7,2),
                              Q(8,1),
                              Q(8,2),
                              Q(9,1),
                              Q(9,2),
                              Q(10,1),
                              Q(10,2);
57$           SIGNAL:         21;
94$           WRITE,          51,"%8.6f\n":
                              epsilon:NEXT(stagecontrol);

135$          ASSIGN:         best_t=est_tardiness:NEXT(129$);

132$          ASSIGN:         reward=(worst_f-est_flowtime)/(worst_f-best_f);
136$          BRANCH,         1:
                              If,est_flowtime>worst_f,137$,Yes:
                              If,est_flowtime<best_f,138$,Yes:
                              Else,129$,Yes;
137$          ASSIGN:         worst_f=est_flowtime:NEXT(129$);

138$          ASSIGN:         best_f=est_flowtime:NEXT(129$);

111$          COUNT:          sta_c4,1:NEXT(stagecontrol);
```

```
52$          COUNT:          Qselect,1;
48$          FINDJ,          1,totalaction#:max(Q(sta1,j));
49$          ASSIGN:         Qa=j:
                             i5=Qa;
50$          COUNT:          22+(s_number-1)*totalaction#+Qa,1;
139$         COUNT:          40+(i5-1)*10+sta1,1:NEXT(95$);

37$          ASSIGN:         sta1=2:NEXT(140$);

38$          ASSIGN:         sta1=3:NEXT(140$);

39$          ASSIGN:         sta1=4:NEXT(140$);

40$          ASSIGN:         sta1=5:NEXT(140$);

41$          ASSIGN:         sta1=6:NEXT(140$);

42$          ASSIGN:         sta1=7:NEXT(140$);

43$          ASSIGN:         sta1=8:NEXT(140$);

44$          ASSIGN:         sta1=9:NEXT(140$);

45$          ASSIGN:         sta1=10:NEXT(140$);

16$          ASSIGN:         sm_s=seizedmachine:
                             sta1=0;
97$          BRANCH,         1:
                             If,sta0==0,105$,Yes:
                             Else,106$,Yes;
105$         COUNT:          sta_c1,1:NEXT(15$);

106$         COUNT:          sta_c2,1;
98$          DUPLICATE:      1,101$:NEXT(99$);

99$          WAIT:           4;
100$         DISPOSE:        No;

101$         ASSIGN:         s_signal=4:NEXT(assignreward);

107$         ASSIGN:         sta1=0;
108$         BRANCH,         1:
                             If,sta0<>0,106$,Yes:
                             Else,stagecontrol,Yes;
123$         ASSIGN:         i4=s_number:
                             est_comp=TNOW+RW;
125$         WHILE:          i4<totalstagenumber;
124$         ASSIGN:         est_comp=est_comp+av_wait_in_Q(i4+1):
                             i4=i4+1;
127$         ENDWHILE;
126$         ASSIGN:         est_tardiness=MX(est_comp-dd,0):
                             est_flowtime=est_comp-TimeIn:NEXT(114$);

88$          CREATE,         1,0:,1:NEXT(89$);

89$          WAIT:           21;
90$          ASSIGN:         epsilon=epsilon-0.000075;
91$          BRANCH,         1:
                             If,epsilon>0.10,89$,Yes:
                             Else,92$,Yes;
92$          ASSIGN:         epsilon=0.10;
93$          DISPOSE:        No;
```

Exp file of the ARENA model

```
PROJECT,       "Unnamed Project","Rockwell
Automation",,,No,Yes,Yes,Yes,No,No,No,No,No,No;

ATTRIBUTES:    1,pt,DATATYPE(Real):
               2,TimeIn,DATATYPE(Real):
               3,RW,DATATYPE(Real):
               4,TW,DATATYPE(Real):
               5,dd,DATATYPE(Real):
               6,PRW,DATATYPE(Real):
               7,slack,DATATYPE(Real):
               8,S_RW,DATATYPE(Real):
               9,CR,DATATYPE(Real):
               10,MDD,DATATYPE(Real):
               11,AT_RPT,DATATYPE(Real):
               12,PT_WINQ,DATATYPE(Real):
               13,PT_WINQ_AT,DATATYPE(Real):
               14,PT_WINQ_SL,DATATYPE(Real):
               15,PT_WINQ_TIS,DATATYPE(Real):
               16,PT_TIS,DATATYPE(Real):
               17,HRN,DATATYPE(Real):
               18,S_RPT,DATATYPE(Real):
               19,CR_RPT,DATATYPE(Real):
               20,COVERT,DATATYPE(Real):
               21,s_number,DATATYPE(Real):
               22,ept(2),DATATYPE(Real):
               24,ddQ(2),DATATYPE(Real):
               26,i1,DATATYPE(Real):
               27,seizedmachine,DATATYPE(Real):
               28,Tardiness,DATATYPE(Real):
               29,Earliness,DATATYPE(Real):
               30,WaitingTime,DATATYPE(Real):
               31,entrytime_to_s(2),DATATYPE(Real):
               33,WT,DATATYPE(Real):
               34,SL,DATATYPE(Real):
               35,TIS,DATATYPE(Real):
               36,WINQ,DATATYPE(Real):
               37,EWT,DATATYPE(Real):
               38,s_Lateness,DATATYPE(Real):
               39,s_tardiness,DATATYPE(Real):
               40,Qs,DATATYPE(Real):
               41,Qa,DATATYPE(Real):
               42,i2,DATATYPE(Real):
               44,reward,DATATYPE(Real):
               46,maxQvalue,DATATYPE(Real):
               est_flowtime,DATATYPE(Real):
               est_tardiness,DATATYPE(Real):
               ExitTime,DATATYPE(Real):
               est_comp2,DATATYPE(Real):
               est_MEPT,DATATYPE(Real):
               est_tardy,DATATYPE(Real):
               sta0,DATATYPE(Real),0:
               sta1,DATATYPE(Real):
               act,DATATYPE(Real),2:
               s_signal,DATATYPE(Real):
               est_comp,DATATYPE(Real):
               est_lateness,DATATYPE(Real):
               av_Wait,DATATYPE(Real):
               i3,DATATYPE(Real):
               i4,DATATYPE(Real);

FILES:
2,TotalLatenesofQ,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\late
nes.txt",Sequential,Free Format,Error,
```

```
              No,Close:

3,MeanTardofQ,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\meantard
inessofQ.txt",Sequential,Free Format,
              Error,No,Close:

9,Qvalue_GM,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\Qvalue_con
trolGM.txt",Sequential,Free Format,
              Error,No,Close:

12,reward_val,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\rewardva
lue.txt",Sequential,Free Format,Error,
              No,Close:

21,OrtBek1,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\ortbekQ1.tx
t",Sequential,Free Format,Error,No,
              Hold:

22,OrtBek2,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\ortbekQ2.tx
t",Sequential,Free Format,Error,No,
              Hold:

31,StatesQ1_bef,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\States
Q1_bef.txt",Sequential,Free Format,
              Error,No,Close:

32,StatesQ2_bef,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\States
Q2_bef.txt",Sequential,Free Format,
              Error,No,Close:

41,StatesQ1_after,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\Stat
esQ1_after.txt",Sequential,Free Format,
              Error,No,Close:

42,StatesQ2_after,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\Stat
esQ2_after.txt",Sequential,Free Format,
              Error,No,Close:

51,epsilonvalue1,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\epsil
onvalue1.txt",Sequential,Free Format,
              Error,No,Close:

52,epsilonvalue2,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\epsil
onvalue2.txt",Sequential,Free Format,
              Error,No,Close:

61,est_trd,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\esttard_.tx
t",Sequential,Free Format,Error,No,
              Close:

62,tardS2,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\tard_S2.txt"
,Sequential,Free Format,Error,No,Close:

Mean_tardiness_,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\meanta
rdiness.txt",Sequential,Free Format,
              Error,No,Close:

MeanTardofQ2,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\meantardi
nessofQ2.txt",Sequential,Free Format,
              Error,No,Close:

TardinessofJ,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\tardnssof
j.txt",Sequential,Free Format,Error,No,
              Hold:
```

```
est_flw,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\estflwtm_.txt"
,Sequential,Free Format,Error,No,Hold:

tard_dif,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\tard_differ.t
xt",Sequential,Free Format,Error,No,
             Hold:

remwork,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\remwork.txt",S
equential,Free Format,Error,No,Hold:

est_comp_dif,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\estcomp_d
if.txt",Sequential,Free Format,Error,
             No,Hold:

latenesQ2,"C:\Users\BMK\Dropbox\2stagesQlearnin_GM\MultiObj\4_2\latenes2.txt
",Sequential,Free Format,Error,No,
             Close;

VARIABLES:   1,Arrival,CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),5:
             2,TWK,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real),1.5:
             3,control(4),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),1,3,4,8:
             7,min_util,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
             8,NR_s,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
             9,sm_s,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
             10,totalstagenumber,CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),2:
             11,TotalTardiness,CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
             12,MeanTardiness,CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
             13,JinQ,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
             14,jQ,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
             15,#ofMachS(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),3,5:
             17,s_TL_before(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
             19,s_TL_after(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
             21,s_MT_before(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
             23,s_MT_after(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
             25,States(9),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),13.26,11.36,9.44,7.24,4.53,0.98,-4.22,-13.08,
             -38.41:
             146,DR(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),10,3:
             156,d_WT,CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),0.0000000001:
             157,alpha1,CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),0.1:
             158,gamma1,CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),0.9:
             159,i5,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
             161,s_ML_before(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
             163,s_ML_after(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
             MSL_before(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
             est_MEPT_after(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
```

```
              est_MT_before(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              av_wait_in_Q(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              totalstate#,CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),10:
              maxtardiness,CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),0:
              Pres_before(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              Q(10,2),CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
              est_MT_after(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              s_TP_before(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              MST_after(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              s_TP_after(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              v_act,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
              est_ML_before(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              mean_CR_before(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              TRW_after(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              totalaction#,CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),2:
              obj1(9),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),0,1.642844,3.087166,4.738524,6.837675,9.898444,
              15.389819,28.260539,79.827373:
              obj2(9),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),41.937674,45.281407,47.50237,49.632258,51.997074,
              55.23923,60.715347,72.638703,117.379753:
              c_tardiness,CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              s_DR,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
              est_TL_before(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              Pres_after(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              r1,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
              r2,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
              est_ML_after(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              MST_before(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              best_f,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real),38:
              best_t,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real),3:
              TRW_before(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              MSL,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
              worst_f,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real),40:
              mean_CR_after(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              worst_t,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real),5:
              v_sta,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
              MSL_after(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              est_TL_after(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              epsilon,CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real),0.9:
              cs1,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
              cs2,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
```

```
              ssssss,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real):
              est_MEPT_before(2),CLEAR(System),CATEGORY("None-
None"),DATATYPE(Real):
              n_sta,CLEAR(System),CATEGORY("None-None"),DATATYPE(Real),0;

SEEDS:        1,,Common:
              2,,Common:
              3,,Common:
              4,,Common:
              5,,Common:
              6,,Common:
              7,,Common:
              8,,Common:
              9,,Common:
              10,,Common:
              11,35698572,Common:
              12,96587423,Common:
              13,478570584,No;

QUEUES:       1,queue1,FirstInFirstOut,,AUTOSTATS(No,,):
              2,queue2,FirstInFirstOut,,AUTOSTATS(No,,);

RESOURCES:
3,Machine,Capacity(1),,Stationary,COST(0.0,0.0,0.0),,AUTOSTATS(No,,):

8,Machine2,Capacity(1),,Stationary,COST(0.0,0.0,0.0),,AUTOSTATS(No,,);

COUNTERS:     1,completedjobs,,Replicate:
              2,TardyJobs,,Replicate:
              3,S1,,Replicate:
              4,S2,,Replicate:
              5,S3,,Replicate:
              6,S4,,Replicate:
              7,S5,,Replicate:
              8,S6,,Replicate:
              9,S7,,Replicate:
              10,S8,,Replicate:
              11,S9,,Replicate:
              12,S10,,Replicate:
              23,a1_1,,Replicate:
              24,a2_1,,Replicate:
              25,a1_2,,Replicate:
              26,a2_2,,Replicate:
              33,Qselect,,Replicate:
              34,randomselect,,Replicate:
              41,1S1,,Replicate:
              42,1S2,,Replicate:
              43,1S3,,Replicate:
              44,1S4,,Replicate:
              45,1S5,,Replicate:
              46,1S6,,Replicate:
              47,1S7,,Replicate:
              48,1S8,,Replicate:
              49,1S9,,Replicate:
              50,1S10,,Replicate:
              51,2S1,,Replicate:
              52,2S2,,Replicate:
              53,2S3,,Replicate:
              54,2S4,,Replicate:
              55,2S5,,Replicate:
              56,2S6,,Replicate:
              57,2S7,,Replicate:
              58,2S8,,Replicate:
              59,2S9,,Replicate:
              60,2S10,,Replicate:
```

```
                sta_c1,,Replicate:
                sta_c2,,Replicate:
                sta_c3,,Replicate:
                sta_c4,,Replicate;

TALLIES:        1,FlowTime,"flowtime.dat":
                2,Tardinessofjobs,"Tardiness.dat":
                3,Earlinessofjobs,"Earliness.dat":
                4,WaitingTimesofjobs,"WaitingTimes.dat":
                5,WTinQ1:
                6,WTinQ2;

DSTATS:         1,NR(1):
                2,NR(2):
                3,NR(3):
                4,NR(4):
                5,NR(5):
                6,NR(6):
                7,NR(7):
                8,NR(8):
                9,(NR(1)+NR(2)+NR(3))/3,Stage1Utilization,"Stage1Util.dat":

10,(NR(4)+NR(5)+NR(6)+NR(7)+NR(8))/5,Stage2Utilization,"Stage2Util.dat":
                11,NQ(1),,"WaitinginQ1.dat":
                12,NQ(2),,"WaitinginQ2.dat";

OUTPUTS:        1,TAVG(1),,MeanFlowTime:
                2,TMAX(1),,MaxFlowTime:
                3,TAVG(2),,MeanTardiness:
                4,TMAX(2),,MaxTardiness:
                5,TAVG(3),,MeanEarliness:
                6,TMAX(3),,MaxEarliness:
                7,TAVG(4),,MeanWaitingTime:
                8,TMAX(4),,MaxWaitingTime:
                9,NC(2),,NumberofTardyJobs:
                10,NC(2)/NC(1),,ProportionofTardyJobs:
                11,NC(41):
                12,NC(42):
                13,NC(43):
                14,NC(44):
                15,NC(45):
                16,NC(46):
                17,NC(47):
                18,NC(48):
                19,NC(49):
                20,NC(50):
                21,NC(51):
                22,NC(52):
                23,NC(53):
                24,NC(54):
                25,NC(55):
                26,NC(56):
                27,NC(57):
                28,NC(58):
                29,NC(59):
                30,NC(60);

REPLICATE,      20,0.0,165000,Yes,Yes,15000,,,24.0,Hours,No,No,,,No,No;

EXPRESSIONS:    1,Util(2),DATATYPE(Native),UNIF(11,14,4),UNIF(19,23.5,7);

TRACE,          0,0;

SETS:           1,QueuesHFS,queue1,queue2;
```

Appendix A8. The selection percentages of the dispatching rules

Figure A8.1 The selection percentages of the dispatching rules in Problem 1.2

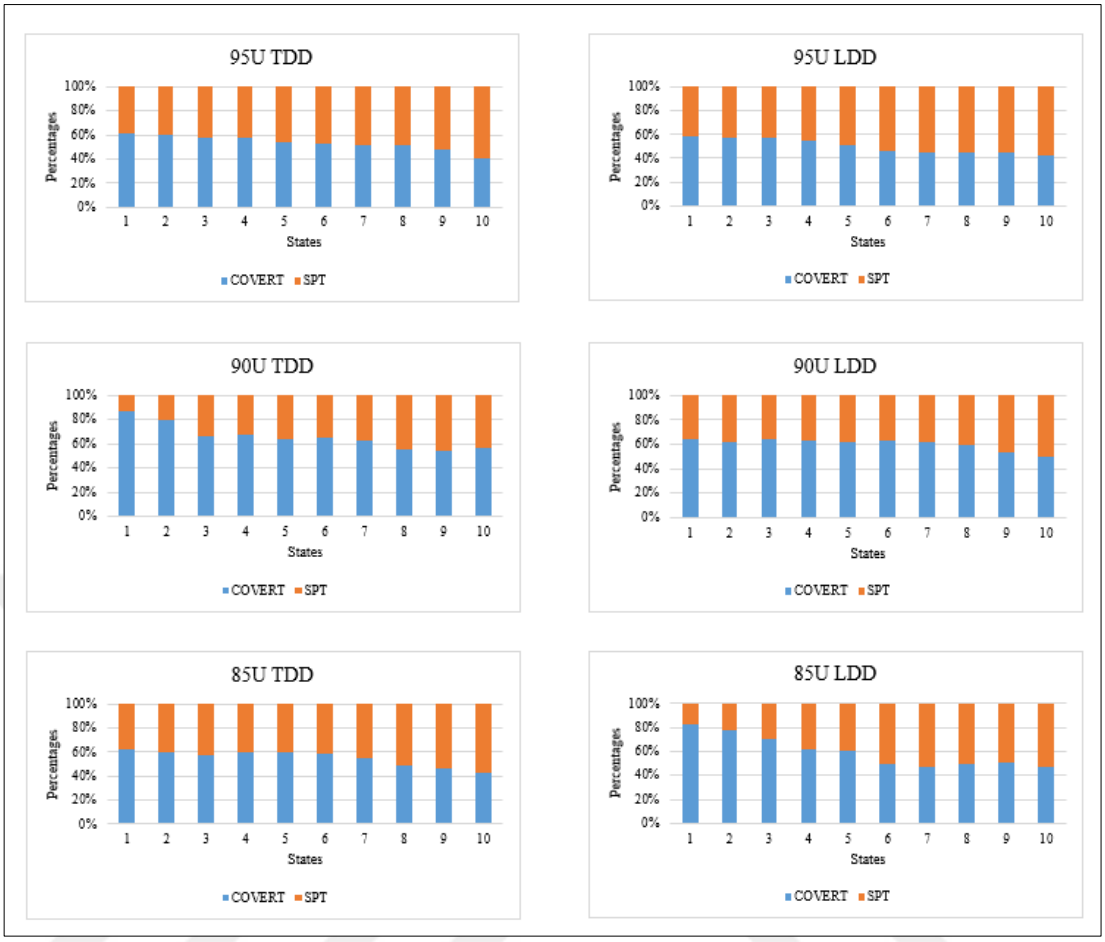

Figure A8.2 The selection percentages of the dispatching rules in Problem 2.1

Figure A8.3 The selection percentages of the dispatching rules in Problem 2.2

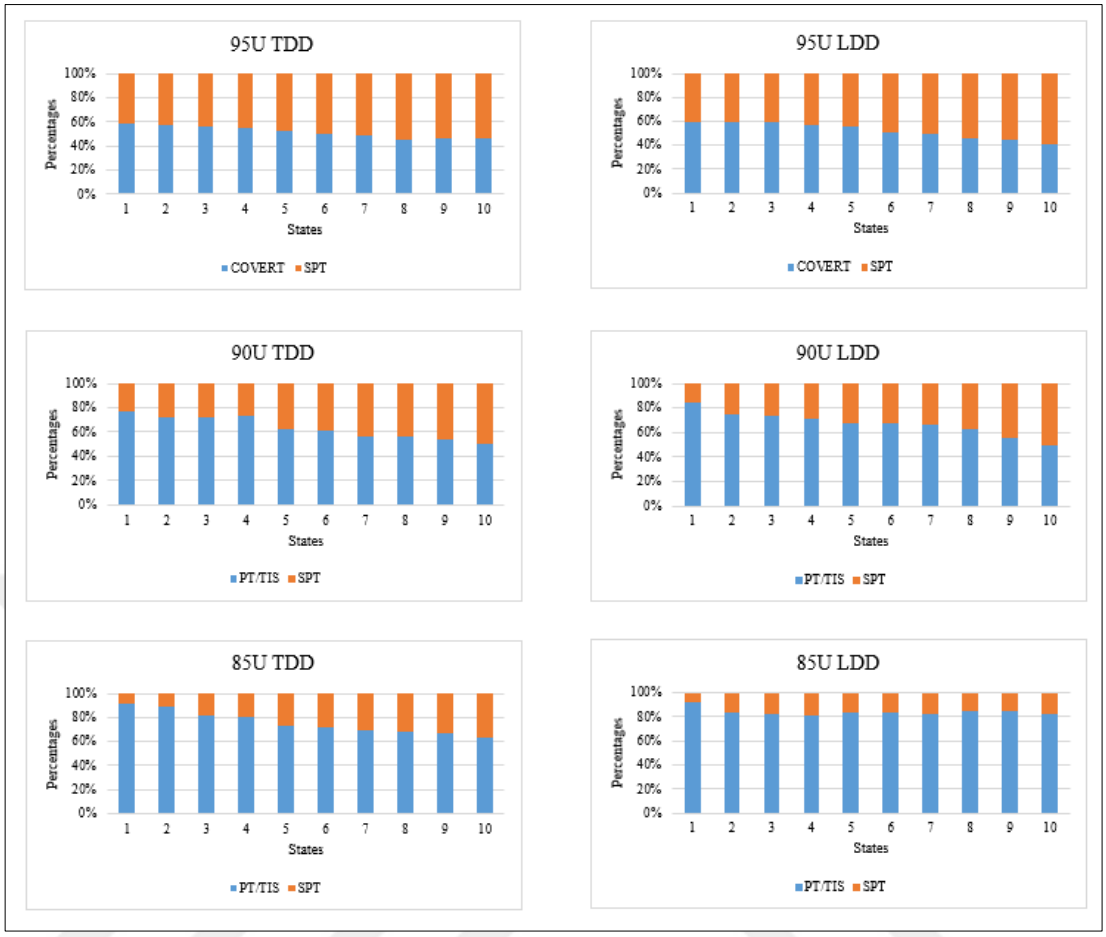

Figure A8.4 The selection percentages of the dispatching rules in Problem 3.1

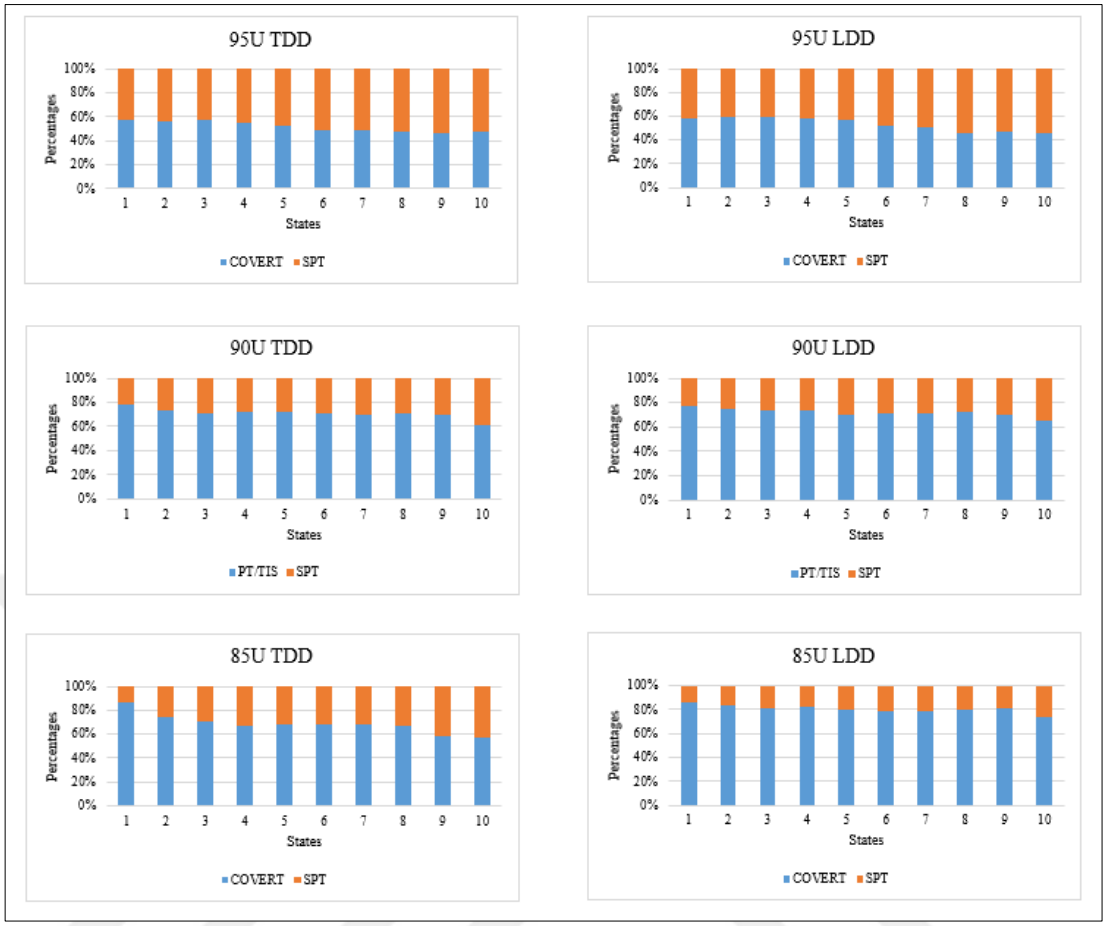

Figure A8.5 The selection percentages of the dispatching rules in Problem 3.2