





Article

A Multi-Agent Reinforcement Learning Approach to the Dynamic Job Shop Scheduling Problem

Ali Fırat İnal ^{1,*}, Çağrı Sel ², Adnan Aktepe ¹, Ahmet Kürşad Türker ¹ and Süleyman Ersöz ¹

¹ Department of Industrial Engineering, Kirikkale University, Kirikkale 71450, Turkey; aaktepe@kku.edu.tr (A.A.); kturker@kku.edu.tr (A.K.T.); serso@kku.edu.tr (S.E.)

² Department of Industrial Engineering, Karabük University, Karabük 78050, Turkey; cagrisel@karabuk.edu.tr

* Correspondence: afinal@kku.edu.tr; Tel.: +90-535-7141992

Abstract: In a production environment, scheduling decides job and machine allocations and the operation sequence. In a job shop production system, the wide variety of jobs, complex routes, and real-life events becomes challenging for scheduling activities. New, unexpected events disrupt the production schedule and require dynamic scheduling updates to the production schedule on an event-based basis. To solve the dynamic scheduling problem, we propose a multi-agent system with reinforcement learning aimed at the minimization of tardiness and flow time to improve the dynamic scheduling techniques. The performance of the proposed multi-agent system is compared with the first-in–first-out, shortest processing time, and earliest due date dispatching rules in terms of the minimization of tardy jobs, mean tardiness, maximum tardiness, mean earliness, maximum earliness, mean flow time, maximum flow time, work in process, and makespan. Five scenarios are generated with different arrival intervals of the jobs to the job shop production system. The results of the experiments, performed for the 3×3 , 5×5 , and 10×10 problem sizes, show that our multi-agent system overperforms compared to the dispatching rules as the workload of the job shop increases. Under a heavy workload, the proposed multi-agent system gives the best results for five performance criteria, which are the proportion of tardy jobs, mean tardiness, maximum tardiness, mean flow time, and maximum flow time.



Citation: İnal, A.F.; Sel, Ç.; Aktepe, A.; Türker, A.K.; Ersöz, S. A Multi-Agent Reinforcement Learning Approach to the Dynamic Job Shop Scheduling Problem. *Sustainability* **2023**, *15*, 8262. <https://doi.org/10.3390/su15108262>

Academic Editors: Samir Lamouri, Robert Pellerin and Pascal Forget

Received: 19 March 2023

Revised: 10 May 2023

Accepted: 16 May 2023

Published: 18 May 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: dynamic job shop scheduling problem; multi-agent system; reinforcement learning; Industry 4.0; dispatching rules

1. Introduction

Scheduling is one of the critical activities in production management to enhance a production system's performance. Scheduling determines the jobs produced on a machine and the production sequence [1–6]. In case the arrival times of the jobs are pre-known, all of the jobs in a process can be organized once by static scheduling. However, the arrival time of each job can barely be foreseen in practice, so it is a necessity to dynamically update the production schedule while the system is running.

In practice, many dynamic events such as arrival times, processing times, machine breakdowns, order cancellations, and due date changes can occur. The actual times of the events are not precise. Random events continuously corrupt the current schedule, so a revised schedule is needed every time a new event occurs. In this study, we propose a dynamic scheduling method based on an event-based simulation to model the rescheduling issue.

In dynamic scheduling problems, production systems are classified as job shop, flow shop, mixed shop, open shop, and group shop [7–14]. In a job shop production system, the variety of products is high, and the batch volume is low because of the varying customers' orders. In a dynamic job shop, new orders constantly arrive at the system to be produced, and completed orders leave the system. The continuous arrivals of jobs that require different

machine routes complicate the scheduling of job-shop-type production environments. In the literature, this problem is named the dynamic job shop scheduling problem (DJSP) and is in the NP-hard class [15–20].

In recent years, agent-based approaches have started to be used on many different problems and have been seen to give competing results [21–25]. In order to solve DJSP, a multi-agent system with reinforcement learning (MAS-RL) is proposed in this study. The MAS-RL is compared with the first-in–first-out (FIFO), shortest processing time (SPT), and earliest due date (EDD) dispatching rules. The number of tardy jobs, mean tardiness, maximum tardiness, mean earliness, maximum earliness, mean flow time, maximum flow time, work in process, and makespan are used as performance criteria. To examine how the scheduling techniques perform under different workloads, FIFO, SPT, EDD, and the MAS-RL are tested using five scenarios with changing job arrival rates. Experiments are carried out for the 3×3 , 5×5 , and 10×10 problem sizes. The results show that the MAS-RL is practical for the DJSP and yields better results than FIFO, SPT, and EDD under heavy workloads.

Another contribution of this study to the literature is that each job type can have different priority values on each machine in our problem. The job priorities on machines make flexible changes possible in the production schedule. We propose a unique technique to the job types' priorities on the machines in a job shop production system and use the MAS-RL to recalculate the priorities in each iteration.

This paper is organized as follows: the literature research is presented in Section 2, the problem statement is given in Section 3, the dispatching rules used to compare the proposed system are given in Section 4, the proposed MAS-RL structure is described in Section 5, the simulation model and parameters are presented in Section 6, the experimental tests and results are provided in Section 7, and, finally, the major conclusions and future research directions are discussed in Section 8.

2. Literature Summary

In this section, we review the relevant studies published during 2010–2022. For the literature review, we use “job shop scheduling”, “dynamic job shop scheduling”, “agent”, “multi-agent system”, and “reinforcement learning” as the keywords. We include the research papers indexed in Science Citation Index (SCI) and Science Citation Index Expanded (SCIE). We examine the DJSP characteristics and solution approach in the field. The interested readers are referred to recent review papers [26–35] in the field. The relevant studies in the literature are summarized under three main categories as the static problem, the dynamic problem, and DJSP.

2.1. Static Problem

In a static scheduling problem, it is assumed that all jobs are ready at the beginning of the scheduling period. The production is scheduled once, so no unexpected events can interfere with the schedule. Studies on the static problem using MAS are summarized in Table 1 in terms of problem/environment and solution method.

Table 1. Literature summarized by static problem.

Paper	Problem/Environment	Solution Method
[36]	JSP	MAS
[37]	Single machine	MAS
[38]	IMS	MAS
[39]	Two identical parallel machines	MAS
[40]	JSP	MAS with ACO
[41]	Flow shop	MAS with RL
[42]	Personalized manufacturing	MAS with RL

JSP: job shop scheduling problem; IMS: intelligent manufacturing systems; MAS: multi-agent system; ACO: ant colony optimization; RL: reinforcement learning.

Komma et al. (2011) [36] is the pioneering research in the field. They prepared a guide on designing agent architecture in different production systems using the Java Agent Development Framework. They consider a discrete event simulation by modeling the components of a production system. Owliya et al. (2012) [37] designed a MAS for general use. They tested the MAS structure on a single machine scheduling problem. They used cost and resource utilization rates as the performance criteria. Leitao et al. (2015) [38] designed a MAS and agents' communication with each other as block diagrams. These show the general behavior of the agents. Yu et al. (2018) [39] developed MAS-based scheduling on two identical parallel machines. They defined the operations and machines as agents. They took the makespan and total tardiness as the criteria. Wong et al. (2012) [40] designed a MAS using ACO for the process planning and integrated scheduling problem. They took the makespan, average flow time, and resource utilization rates as the criteria. Wang et al. (2019) [41] developed a MAS in which the agents communicate with each other by using the game theory method. They tested this MAS architecture on a simulation model of a smart workshop. They took the makespan, machine workloads, and energy consumption as the criteria. They showed that the MAS architecture yielded better results than the FIFO-based and SPT-based approaches. Kim et al. (2020) [42] designed a MAS for personalized manufacturing. They used the makespan and maximum tardiness as the performance criteria. They compared the designed MAS with the frequently used dispatching rules in the literature. They used the RL algorithm for the development of the decision mechanism.

While static scheduling problems are ideal for testing new solution methods, real-world scheduling problems are dynamic. For this reason, a technique that offers feasible solutions to the static problem may not provide a feasible solution to dynamic real-world problems.

2.2. Dynamic Problem

In a dynamic scheduling problem, the literature research focuses on the flow shop and job shop production systems. If there are dynamic events in a flow shop production system, the problem is called "D-Flow shop". Likewise, in others, if there are dynamic events in the job-shop-type production, the problem is called "DJSP"; and, if there are dynamic events in the flexible job-shop-type production, the problem is called the dynamic flexible job shop scheduling problem (DFJSP) in the literature. In this study, a solution method is proposed for DJSP, but there are different MAS and AI approaches designed for other problem types such as CPPS, DFJSP, D-Flow Shop, and SHFS in the literature. The studies on the dynamic problem are summarized in Table 2 in terms of the problem/environment, the solution method, and dynamic factors.

Table 2. Literature summarized by dynamic problem.

Paper	Problem/Environment	Solution Method
[43]	D-Parallel Machines	Two rival agents, GA
[44]	DFJSP	NRGA, NSGA-II
[45]	DFJSP	Dispatching rules, RL
[46]	DFJSP	MAS
[47]	DFJSP	MAS
[48]	D-Parallel Machines	MAS
[49]	Cloud manufacturing	MAS
[50]	CPPS	MAS with GA
[51]	DFJSP	MAS with ACO
[52]	D-Flow Shop	MAS with DSS
[53]	SHFS	MAS with GA
[54]	DFJSP	MAS with RL

DFJSP: dynamic flexible job shop scheduling problem; CPPS: cyber-physical production systems; SHFS: sustainable hybrid flow shop; MAS: multi-agent system; GA: genetic algorithm; NRGA: non-dominated ranking genetic algorithm; NSGA: non-dominated sorting genetic algorithm; ACO: ant colony optimization; RL: reinforcement learning; DSS: decision support system; PT: processing time; MB: machine breakdown; OA: order arrival; DD: due date; ST: setup time; OC: order cancellation; UO: urgent order.

Lee et al. (2017) [43] designed a structure with two rival agents in a system with two parallel machines. The agents' goal was to minimize the makespan. The structure was compared with the GA. Ahmadi et al. (2016) [44] used NSGA-II and NPGA in the DFJSP, considering machine breakdowns. Shiue et al. (2018) [45] designed a structure that changes the dispatching rules by using RL for the DFJSP. They chose the average flow time and number of tardy jobs as the criteria. Sahin et al. (2017) [46] designed a MAS for the DFJSP. Each agent tried to achieve its own goal. They made both dynamic and static scheduling and achieved satisfactory results. Maoudj et al. (2019) [47] designed a MAS architecture for the robotic flexible assembly cell, which is considered as the DFJSP. The MAS architecture created the schedule by switching between the dispatching rules. They used the makespan as a criterion. They showed that the agent architecture they designed could yield better results than the metaheuristics they compared it with. Huang and Liao (2012) [48] designed a MAS architecture for the dynamic parallel machine scheduling problem. In the MAS structure, which consists of work, machine, and management agent, the communication between agents is examined in detail. As the criteria, they considered total tardiness, flow time, resource utilization rates, and revenue value. Y. Liu et al. (2018) [49] studied cloud manufacturing. They created a MAS-based scheduling mechanism and tested it in a sample study using the simulation method. They explained the communication between the agents in detail. Jiang et al. (2017) [50] worked on dynamic scheduling in CPPS. They established a double-layered decision-making mechanism. This decision-making mechanism performed the rescheduling activity with a GA. The agents both collected information and took actions from the decision-making mechanism. S. Zhang and Wong (2017) [51] simulated different dynamic factors in different scenarios in the DFJSP. They hybridized the MAS-based approach they developed with the ACO. The makespan was considered as a criterion. Barenji et al. (2017) [52] worked on MAS-based DSS for solving the D-Flow Shop problem. They tested the MAS-based DSS by modeling a small- and medium-sized real-life system in a simulation environment. They proposed a system that can perform both static and dynamic scheduling. They used the makespan as a criterion. Shi et al. (2021) [53] designed a MAS that updates the priorities of the jobs with different types of GA. They tested the MAS structure in sustainable hybrid-flow-type production. They took into account the makespan, energy consumption, and carbon emissions as the criteria. The proposed MAS structure increased the computation time as the problem size increases but gave better results than the compared algorithms. Luo (2020) [54] designed a MAS with the RL approach for the DFJSP. The makespan was used as a criterion. The designed MAS was compared with the dispatching rules frequently used in the literature.

Since dynamic scheduling problems reflect real production systems, they are divided into many categories. It would not be correct to say that a method that gives feasible solutions for one category necessarily gives feasible solutions in other categories.

In this study, we propose a MAS-RL for the DJSP. The studies conducted on the DJSP are summarized in Table 3 according to the problem/environment, the solution method, and dynamic factors.

Table 3. Literature summarized by dynamic problem (via DJSP).

Paper	Problem/Environment	Solution Method
[55]	DJSP	GRASP
[56]	DJSP	SA
[57]	DJSP	Dispatching rules
[58]	DJSP	DSS, dispatching rules
[59]	DJSP	Single agent with RL
[60]	DJSP	Single agent with RL
[61]	DJSP	MAS
[62]	DJSP	MAS
[63]	DJSP	MAS
[64]	DJSP	MAS

DJSP: dynamic job shop scheduling problem; GRASP: greedy randomized adaptive search procedure; SA: simulated annealing; MAS: multi-agent system; RL: reinforcement learning; DSS: decision support system; PT: processing time; MB: machine breakdown; OA: order arrival; DD: due date; ST: setup time; OC: order cancellation; UO: urgent order.

Baykasoğlu and Karaslan (2017) [55] developed a GRASP-based approach to the DJSP problem. They used goal programming logic to reach better performance criteria. They showed that their proposed GRASP-based approach yields viable solutions. Sel and Hamzadayı (2018) [56] proposed a simulation optimization study based on the SA carried out in the DJSP. They considered average flow time and average tardiness as the criteria. The proposed simulation optimization approach yielded better results than EDD and FIFO. C. L. Zhang et al. (2019) [57] designed a two-agent structure in the DJSP to minimize the makespan. They showed that the proposed two-agent structure gives better results as the problem scale becomes larger. Turker et al. (2019) [58] proposed a DSS for the DJSP. The proposed DSS was designed to increase the performance of dispatching rules for dynamic scheduling by using real-time data. They used average machine utilization, average waiting time, work in process, number of tardy jobs, average tardiness, and average earliness as the performance criteria. They conducted the experiments in different scenarios with different job arrival rates. The proposed DSS decided about a job by considering not only the workload of the machine it is currently in but also the workloads of the machines it goes to in the following steps. Aydin and Öztemel (2000) [59] designed a single agent with RL for the DJSP. The agent was trained with RL and was able to carry out scheduling activities. The designed system consisted of two parts: agent and simulator. The agent determined the most appropriate dispatching rule by reading the data from the shop floor. The simulator, on the other hand, applied the dispatching rule determined by the agent. Kardos et al. (2020) [60] designed a MAS structure using RL for the DJSP. OA was taken as the dynamic factor in the problem. They compared this structure with the SPT and other dispatching rules in the literature. Average lead-time was considered as the objective function. They determined that the complexity of the production environment was important because, as it increases, using RL for dynamic scheduling becomes more effective. Erol et al. (2012) [61] developed a MAS-based scheduling approach for AGVs and machines in a dynamic production system. Jana et al. (2013) [62] designed a MAS using fuzzy multi-criteria decision making and multi-objective optimization techniques based on ratios. They tested the designed MAS in different scenarios. M. Leusin et al. (2018) [63] studied dynamic scheduling in smart workshop environments within the scope of Industry 4.0. They aimed to collect real-time data with the help of the IoT and RFID and to make decisions using the MAS structure based on these data. M. E. Leusin et al. (2018) [64] studied the data they received from a real production system. They aimed to reduce the workload of the job shop with their designed MAS. The proposed MAS yielded better results than the job shop's current scheduling strategy.

As can be seen from the literature summary, there are studies using MAS for the DJSP, but only one study was found that provided training for a MAS with RL on the DJSP.

We examined both the method and the problem characteristics of the studies in the literature and summarized them in the following list. We described the points of our study's similarities and differences from the literature. As a result of the literature review, the following list of improvements to the literature were reached.

1. In our study, each job type can have different priority values on each machine as a unique scheduling method. In the literature, there is no other study using this scheduling method exactly as it is in this study. With this method, we aim to give flexibility to the production schedule. This method, which has a unique scheduling way, is explained in detail in the following sections. It is thought that researchers can adapt this scheduling method to their own studies and maybe improve this method by making some changes.
2. No other studies using MAS with RL for DJSP were found in the literature. However, there is one study using a single agent with RL, which is Kardos et al. (2020) [60]. Since there are insufficient studies on this specific mixture of the problem and solution method, our study can be considered as a novel study conducted on this area.

The aspects of our study that differ from Kardos et al. (2020) [60] and how they are extended are mentioned in the following list.

- i. While a single agent with RL was proposed for the DJSP in Kardos et al. (2020) [60], we extend this approach by proposing a MAS with RL. In other words, a structure is designed in which there are multiple agents with different purposes. In this way, instead of trying to optimize the entire system with a single agent, it tries to be optimized with multiple agents in parts.
- ii. While Kardos et al. (2020) [60] considered only the OA as a dynamic factor, we extend dynamic factors such as OA, PT, and DD. Since using more dynamic events together means that the problem becomes more difficult to solve, we improve the literature in this aspect.
- iii. While Kardos et al. (2020) [60] took into account the average lead time as a performance criterion, we extend the number of performance criteria to nine, which is the proportion of the tardy jobs, mean tardiness, maximum tardiness, mean earliness, maximum earliness, mean flow time, maximum flow time, work in process, and makespan in this study. With the expansion of the performance criteria, the results in this specific area can be examined in a wider range, making it possible to reach conclusions from various aspects.

In addition to all these improvements to the literature, we aimed to make it easier for researchers who are not experts in MAS to understand the MAS easily and develop their own studies on this subject. From this aspect, the MAS-RL in this study was designed to be as understandable as possible, and each agent's working principle was explained in detail. In this way, we tried to encourage that MAS studies be carried out in the future.

3. Problem Statement

The main frame of the DJSP is to use a limited number of machines (or service providers), to process a specified number of jobs (or tasks), while trying to optimize the specified objectives such as the makespan or tardiness. Each of these jobs has a specified operation sequence or route through the machines, with a specified processing time at the corresponding machine. When the job passes through the last operation sequence, it is considered as a finished job.

The DJSP also has other constraints that needs to be taken care of. In some of the studies in the literature, the problem is attempted to be solved by the mathematical programming method, while, in other studies, simulation programs specially designed for scheduling problems are used. The advantages of using a simulation program are that the production schedule can be stopped and examined at any time, the workflows can be followed visually, and there is no need for a mathematical model. In our study, the Arena[®] package program was used as a simulation program. Within the modules of the program, the constraints

of DJSP are already present. For this reason, these constraints are given by linguistic expressions as follows.

1. Different operations are performed on different machines;
2. The machines operate only one job at the same time;
3. The jobs are operated on only one machine at the same time;
4. Operations that have started cannot be interrupted or paused;
5. The jobs must follow their routes in the specific order;
6. The queue capacity is unlimited for any machine.

Job shop scheduling problems can be of different sizes. The problem size is expressed as the number of job types and the number of machines ($j \times m$). The machine and job-type thresholds of a job shop that determine the complexity of a job shop-instance are not generally agreed upon in the literature [65]. In addition, there are studies in the literature that mention that the problem size does not make a difference for the performance of the dispatching rules [66,67].

In this study, we performed experiments for the 3×3 , 5×5 , and 10×10 problem sizes to show that our proposed approach works well for different problem sizes. The routes that jobs follow in a job-shop environment are complex and difficult to follow. To illustrate this, a visual representation of the 3×3 problem size dynamic job-shop environment is given in Figure 1.

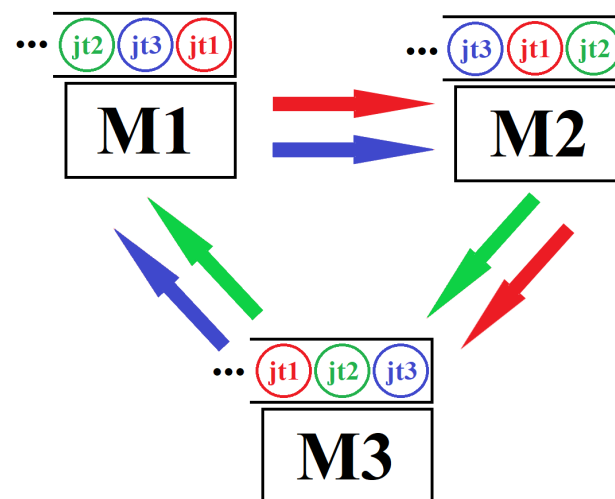


Figure 1. Visual representative of the 3×3 problem size dynamic job-shop environment.

“M” means machine, and “jt” means job type. Job type 1 is marked in red, job type 2 is marked in green, and job type 3 is marked in blue. Each job type visits each machine according to its route. For example, jt1’s route (red) is M1–M2–M3, jt2’s route (green) is M2–M3–M1, and jt3’s route (blue) is M3–M1–M2. A machine can have more than one job of the same job type in the queue.

The DJSP is an NP-hard class problem due to its complexity. The increase in the diversity of machines and job types and the increase in the complexity of the jobs’ routes make it almost impossible to reach the optimum solution of the problem in the polynomial time. Due to the stochastic and dynamic nature of the job arrivals to the system, it may cause a computational burden to produce reliable solutions even with the 3×3 problem size.

In Table 4, the model notations are described.

Table 4. Notations.

Notation	Description
j	j th job or job index
m	m th machine or machine index
t	Current time
A_j	Arrival time of j th job to the system
$A_{j,m}$	Arrival time of j th job to m th machine
C_j	Completion time of j th job
D_j	Due date of j th job
P_j	Total processing time of j th job
$P_{j,m}$	Processing time of j th job on m th machine
Z_m	Priority index of m th machine

In the job-shop-type production system, jobs (j) randomly arrive at the shop floor according to exponential distribution. The arrival time of each job is recorded as (A_j). The processing times of the job on each separate machine (m) are determined according to the normal distribution and are recorded as ($P_{j,m}$). The due dates of the jobs are assigned as (D_j). After the assignments, the jobs are directed to the first machines on their routes. If a machine is in idle state and the machine's queue is empty, the job's processing starts immediately, otherwise the job is directed to the machine's queue and waits for the machine to become idle. When the machine becomes idle, and the job is chosen to be next, it enters the machine and is processed as the time ($P_{j,m}$). The job is then routed to the next machine according to its route, and this sequence repeats until the job's route complete. Here, we assume that the transportation times between the machines can be neglected. The completion time of each job is recorded as (C_j) and is used to calculate the flow time (F_j) and deviation from the due date (Dev_j). These formulations are presented below. The flow time is calculated by Equation (1). Equation (2) calculates the deviation from the due date. A positive deviation indicates that the job is tardy (or late), and the outcome of the equation is considered tardiness (T_j). If the deviation is negative, it corresponds to earliness (E_j). It is undesirable for a job leaving the system to be early or tardy. A job that is early indirectly causes other jobs in the system to be tardy. This is a situation where every job is requested to be completed exactly on its due date [68].

$$F_j = C_j - A_j, \quad (1)$$

$$Dev_j = C_j - D_j, \quad (2)$$

4. Dispatching Rules Used

Dispatching rules are used for arranging jobs in the machines' queues according to a specific rule. Dispatching rules are known as priority rules. Dispatching rules are used to determine the job that starts processing on the machine as soon as the machine becomes idle. In this study, FIFO, SPT, and EDD rules, which are the most frequently used rules in the literature, were compared to the MAS-RL.

4.1. FIFO Rule

The way FIFO works is based on giving priority to the job that arrives to the machine first. The FIFO rule is one of the most frequently used rules in the literature. In addition, it is easy to apply in theory as well as in practice. FIFO fits for systems where the entities are food with a short expiration date or people. In fact, we unknowingly use FIFO in most of the public zones where we encounter queues such as supermarkets, restaurants, banks, and counters in our daily life. In order to mathematically apply FIFO, it is necessary to know the arrival times of the jobs in the queue of the machine ($A_{j,m}$). The job with the

minimum arrival time should be selected and processed on the machine as a priority. FIFO is formulated in Equation (3).

$$\text{Min}(Z_m) = A_{j,m} \quad \forall j, m \quad (3)$$

4.2. SPT Rule

SPT is based on giving priority to the job with the shortest processing time. As example of a real-life use of SPT is when there is a long queue in front of a cash register in a supermarket, but customers with a few items take the lead. SPT minimizes mean flow time. In order to mathematically apply SPT, it is necessary to know the processing times of the jobs in the queue of the machine ($P_{j,m}$). The job with the minimum processing time should be selected and processed on the machine as a priority. SPT is formulated in Equation (4).

$$\text{Min}(Z_m) = P_{j,m} \quad \forall j, m \quad (4)$$

4.3. EDD Rule

EDD is based on prioritizing the job with the shortest due date. For example, a real-life use of EDD is when a customer with very urgent business takes the lead while there is a long queue in front of a cash register in a supermarket. EDD is generally used for minimizing tardiness. In order to mathematically apply EDD, it is necessary to know the due dates of the jobs (D_j). The job with the minimum due date should be selected and processed on the machine as a priority. EDD is formulated in Equation (5).

$$\text{Min}(Z_m) = D_j \quad \forall j, \quad (5)$$

5. Proposed MAS-RL Approach

In this section, we introduce the MAS structure and the RL mechanism. In the MAS structure, we explain the agents in the system and the relationships between the agents. The RL mechanism, which provides a learning function to the MAS, is examined.

5.1. Multi-Agent System Structure

The MAS is an AI approach distributing intelligence to different individuals by agents. The MAS is a computerized system that consists of multiple intelligent agents communicating with each other. An agent has a goal, sensors, and actuators. When an agent is put into an environment, the agent should be able to change the environment for its intended goal.

In the MAS, there are multiple agents that can have different goals and different ways of changing the environment. The MAS can be compared to a bee colony. The different types of bees are specialized for different goals, such as searching for resources, making honey, or defending the colony. Each bee type has different actuators to achieve its own goal. The natural swarm intelligence of the bee colony can be imitated with the MAS.

The MAS should be intelligent in order to provide feasible solutions. Intelligence is defined as the "ability to learn". There are three main machine learning methods: supervised, unsupervised, and RL. We used RL to implement the learning ability to the MAS. The aforementioned three types of machine learning are explained in the next section.

The proposed MAS-RL structure designed in this study is shown in Figure 2. Five different types of agents are designed for the MAS-RL, and they all have different goals. All of them have the ability to communicate with each other and take actions toward a goal.

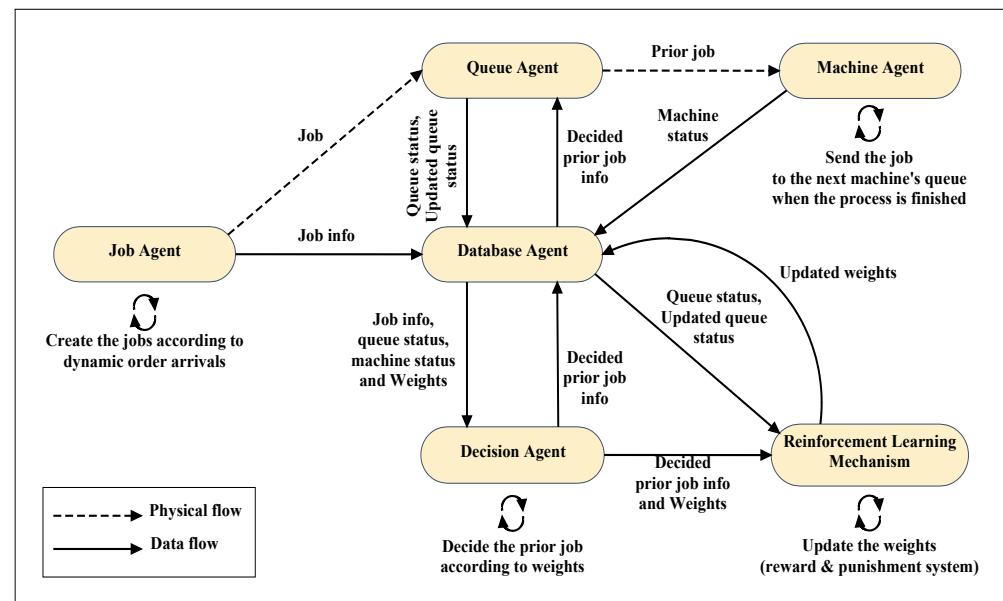


Figure 2. The proposed MAS-RL structure.

In the MAS-RL structure, jobs are created by a Job Agent, and a chain reaction begins when an order arrives. The reaction continues until the maximum number of jobs has been reached. We describe the goals, decisions, and internal mechanisms of the agents.

5.1.1. Job Agents

The main goal of a Job Agent is to create jobs according to incoming orders and report the job's information to the Database Agent. The first aspect of the job information is the job's arrival time to the system. The information about job arrivals is obtained by the Job Agent, generating a random number from the exponential distribution. Another task of the Job Agent is to determine the job type corresponding to the incoming order and to assign processing times for the considered job type. These processing times are obtained by the Job Agent by generating a random number from the normal distribution.

As soon as a job enters the system, its due date is determined. After the assignments, the Job Agent sends jobs to the Queue Agent and reports the job information to the Database Agent.

5.1.2. Queue Agents

A Queue Agent sends the selected job to a Machine Agent when the machine is in idle state. It also provides the current state of the queue to the relevant agents.

Information such as how many jobs are in the queue at the moment, how long the total processing time of these jobs is, and what these jobs' types are, are continuously transferred to the Database Agent. When a job is removed from the queue and sent to the Machine Agent, the information changes are recalculated and reported to the Database Agent.

5.1.3. Machine Agents

A Machine Agent sends the jobs to the queue of another machine when a job is completed on the current machine. The machine's status is monitored as busy or idle by the Machine Agent and when a change occurs that notifies the Database Agent.

5.1.4. Database Agent

A Database Agent stores the incoming information and forwards the information to the other agents. The agent that needs to acquire information requests the information from the Database Agent. The Database Agent acts as an information center for the other agents.

5.1.5. Decision Agent

A Decision Agent decides which job in the queue has the highest priority using the current values in the priority table. After determining the prior job, the Decision Agent transmits the information to the Database Agent and the RL mechanism. The Decision Agent informs the RL mechanism by sending the values from the priority table that it used while determining the prior job. Then, the RL mechanism updates the priority table according to the deviations from the due date of the jobs.

5.2. Reinforcement Learning Mechanism

Machine learning algorithms receive historical input and output data from supervised learning. The supervised learning method allows the algorithm to create outputs as close to the desired result as possible by changing the model between each input/output pair. Supervised learning algorithms include decision trees, neural networks, support vector machines, and linear regression.

The labeled training sets and data are not used in unsupervised learning. Instead, the machine searches the data for less obvious patterns. Machine learning of this type makes decisions by using the data to find patterns. K-means, Hidden Markov models, a Gaussian mixture, and hierarchical clustering models are common unsupervised learning algorithms.

RL is a machine learning type that reflects humans' learning mechanism. The agent learns by interacting with the environment and receives a positive reward or negative reward (punishment). The agent is programmed to seek a long-term reward to reach the goal [69]. The RL mechanism is illustrated in Figure 3. The agent takes an action by looking at the state. The environment changes according to the action taken. According to this change, the agent receives a reward. Then, the loop starts over by looking at the state again.

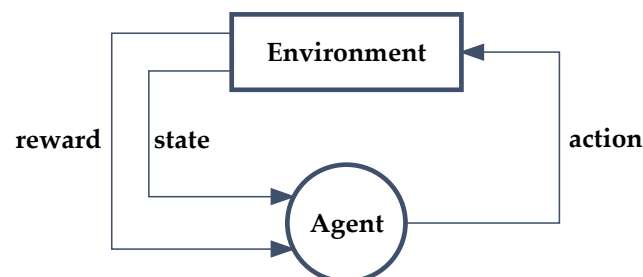


Figure 3. The agent–environment interaction in RL framework.

To use the RL mechanism, a priority table is needed, as in dispatching rules. Therefore, a priority value is defined for each job type on each machine. These values are indicated by W . W values are visualized in Table 5 only for the 3×3 problem, as the size of the problem increases as the job types and the number of machines increase. For the 5×5 and 10×10 problems, the table expands as the job types (i) and machines (m) increase.

Table 5. W values by job type and machine indices (3×3 problem).

$W_{i,m}$		Machine (m)		
		M_1	M_2	M_3
Job type (i)	Type ₁	W_{11}	W_{12}	W_{13}
	Type ₂	W_{21}	W_{22}	W_{23}
	Type ₃	W_{31}	W_{32}	W_{33}

When scheduler agents need to select a job from the corresponding machine's queue, they give priority to the job type with the highest W value. This structure is formulated in Equation (6).

$$\text{Max}(Z_m) = W_{i,m} \quad \forall I, \quad (6)$$

In cases where the queues have more than one job of the same job type, the W values of the jobs are equal. When trying to give priority to one of these jobs, a tie situation occurs. To break the tie, the FIFO rule is used to define the earliest job that came to the queue.

The W values are updated for every job leaving the system. A job leaving the system changes the priority values of the jobs in the same type in the system. For example, a job with job type Type_2 updates W_{21} , W_{22} , and W_{23} when leaving the system. The magnitude of change takes place, as shown in Equation (7) for tardy jobs and as shown in Equation (8) for early jobs.

$$\text{new } W_{i,m} = \text{old } W_{i,m} + \left[\left(T_j / \max(T_j) \right) + (N_i / \max(N_i)) + (M_m / \max(M_m)) \right] / \alpha \quad \forall i, m, \quad (7)$$

$$\text{new } W_{i,m} = \text{old } W_{i,m} - \left(E_j / \max(E_j) \right) / \beta \quad \forall i, m \quad (8)$$

α : Total number of tardy jobs;

T_j : Tardiness of the j th job;

N_i : Total number of jobs of Type_i in the system;

M_m : Total number of jobs waiting in the queue of the m th machine;

β : Total number of early jobs;

E_j : Earliness of the j th job.

By dividing each variable by its maximum value, the effect of these variables on the W value is normalized. This way, the effect of the spike values over the W values is reduced, and the W values become more stable over time.

The update activity occurs in time t . In other words, the $\max(x)$ functions in the Equations (7) and (8) are considered as the greatest measurement value of x up to time t . According to this, the W values increase by a maximum of 3 in a single update, and the update rate slows down over time.

Since early jobs and tardy jobs should affect the W value in opposite directions, and earliness and tardiness are expressed with different notations, two separate equations are presented as W update equations.

The logic of the change in W values is explained as follows.

1. A tardy job affects the W values as much as the delay time (the same applies to early jobs);
2. A tardy job affects the W values as much as the jobs of the same type in the system;
3. A tardy job updates much more than the W value for machines with a long queue length.

The magnitude of the W value shows the priority of the job on the machine. The higher the W value is, the higher the priority of the job type is. The W value increases in the case of tardiness and decreases in the case of earliness. Tardy jobs are more undesirable than early jobs in production systems. For this reason, the change in the W value is greater for tardy jobs than early jobs. While there are three factors affecting the W value for tardy jobs in Equation (7), it is seen that there is only one factor for early jobs in Equation (8).

6. Simulation Model

In order to simulate a real job-shop environment, all input data need to be dynamically and stochastically obtained throughout the simulation period. For this reason, while the simulation is running, input data such as jobs' arrival times, processing times, and due dates are generated according to the probability distributions when needed. The routes and processing times of the jobs used in the 3×3 problem simulation model are given in Table 6. The processing times are randomly generated by the normal distribution. Job type (i) in the table expands to 5 lines for the 5×5 problem and 10 lines for the 10×10 problem. The processing times used for the 5×5 and 10×10 problems are given in Tables 7 and 8, respectively.

Table 6. Routes and processing times (3 × 3 problem).

Job Type (<i>i</i>)	Route (Processing Time) *		
Type ₁	M ₁ [N(25,1)]	M ₂ [N(23,4)]	M ₃ [N(20,9)]
Type ₂	M ₂ [N(25,4)]	M ₃ [N(20,9)]	M ₁ [N(23,1)]
Type ₃	M ₃ [N(25,9)]	M ₁ [N(20,1)]	M ₂ [N(23,4)]

* Each number is in minutes. M: machine; N: normal distribution (mean and variance).

Table 7. Routes and processing times (5 × 5 problem).

Job Type (<i>i</i>)	Route (Processing Time) *		
Type ₁	M ₁ [N(25,1)]	M ₂ [N(23,4)]	M ₃ [N(20,9)]
Type ₂	M ₂ [N(25,4)]	M ₃ [N(20,9)]	M ₁ [N(23,1)]
Type ₃	M ₃ [N(25,9)]	M ₁ [N(20,1)]	M ₂ [N(23,4)]
Type ₄	M ₄ [N(25,1)]	M ₅ [N(20,4)]	M ₁ [N(23,9)]
Type ₅	M ₅ [N(25,4)]	M ₁ [N(20,9)]	M ₂ [N(23,1)]

* Each number is in minutes. M: machine; N: normal distribution (mean and variance).

Table 8. Routes and processing times (10 × 10 problem).

Job Type (<i>i</i>)	Route (Processing Time) *		
Type ₁	M ₁ [N(25,1)]	M ₂ [N(23,4)]	M ₃ [N(20,9)]
Type ₂	M ₂ [N(25,4)]	M ₃ [N(20,9)]	M ₁ [N(23,1)]
Type ₃	M ₃ [N(25,9)]	M ₁ [N(20,1)]	M ₂ [N(23,4)]
Type ₄	M ₄ [N(25,1)]	M ₅ [N(20,4)]	M ₁ [N(23,9)]
Type ₅	M ₅ [N(25,4)]	M ₁ [N(20,9)]	M ₂ [N(23,1)]
Type ₆	M ₆ [N(25,9)]	M ₇ [N(20,1)]	M ₈ [N(23,4)]
Type ₇	M ₇ [N(25,1)]	M ₈ [N(23,4)]	M ₉ [N(20,9)]
Type ₈	M ₈ [N(25,4)]	M ₉ [N(20,9)]	M ₁₀ [N(23,1)]
Type ₉	M ₉ [N(25,9)]	M ₁₀ [N(20,1)]	M ₁ [N(23,4)]
Type ₁₀	M ₁₀ [N(25,1)]	M ₁ [N(23,4)]	M ₂ [N(20,9)]

* Each number is in minutes. M: machine; N: normal distribution (mean and variance).

The jobs' arrival rates are assigned separately for five different scenarios. As the time between arrivals becomes shorter, the jobs' arrivals become more frequent, and the workload of the system increases. The scenarios are presented in Table 9, which represents very low, low, moderate, heavy, and very heavy workloads. The time between arrivals is randomly generated by the exponential distribution.

Table 9. Simulation scenarios.

Scenario	Time Between Arrivals *	Workload
Scenario 1	Exp(80) for each job type	Very low
Scenario 2	Exp(75) for each job type	Low
Scenario 3	Exp(70) for each job type	Moderate
Scenario 4	Exp(65) for each job type	Heavy
Scenario 5	Exp(60) for each job type	Very heavy

* Each number is in minutes. Exp: exponential distribution.

There are different due date assignment methods in the literature for job-shop-scheduling problems. These different methods do not have any advantages over each other. Due to its ease of implementation, one of the "processing time multiplying" methods was used in [70]. In this method, the due date is assigned by the uniform distribution for each job, as shown in Equation (9). When calculating the due date, the arrival time of the job and the estimated total processing time should also be taken into account. After the due date assignment, jobs go to the first machine on their routes.

$$D_j = P_j \times U(10,20) + A_j \quad \forall j, \quad (9)$$

A job that finishes its route on the machines is completed. For completed jobs, the flow time (F_j) and deviation from the due date (Dev_j) are calculated by Equation (1) and Equation (2), respectively. If the Dev_j value is positive, the job is tardy and updates the W values using Equation (7). If the Dev_j value is negative, the job is early and updates the W values using Equation (8).

The simulation model is prepared in the Arena® ver.13.50 package program. Due to its large size, only a small section of the 3×3 problem simulation model is given in Figure 4.

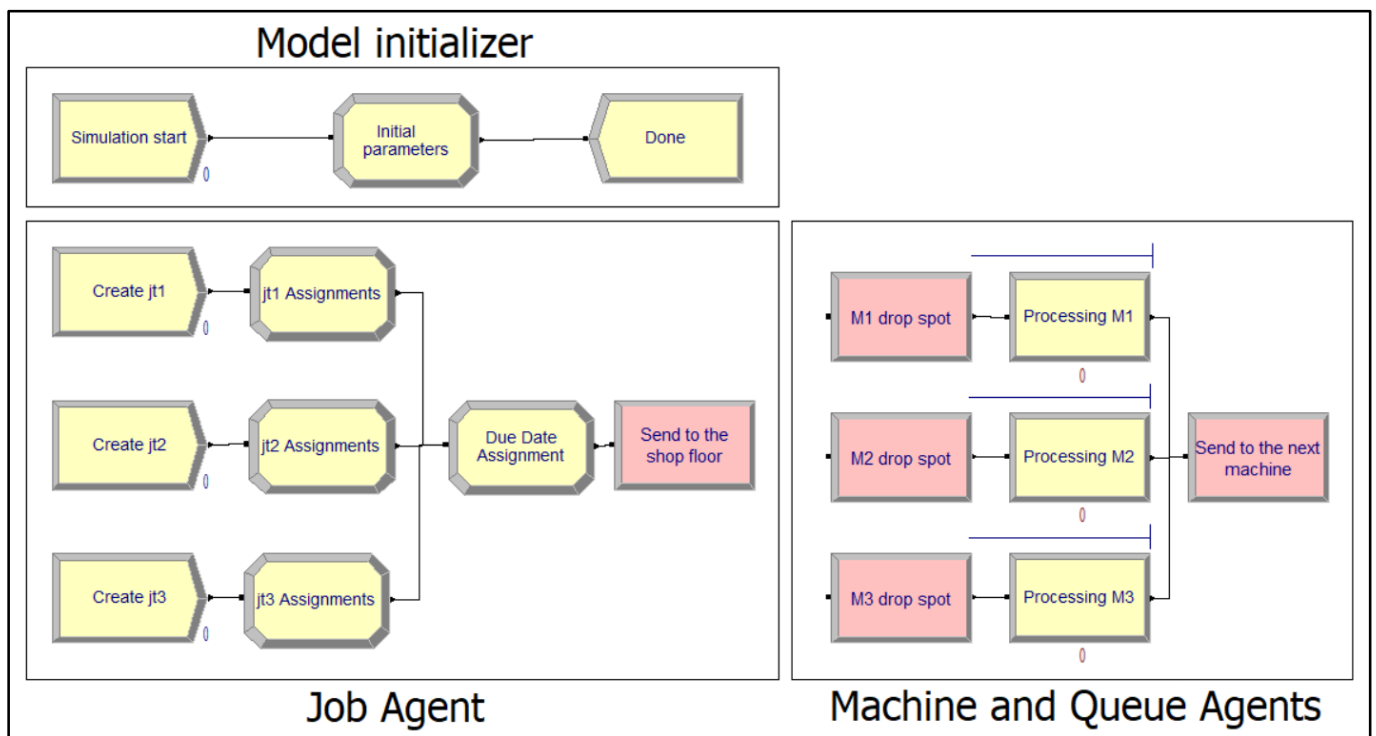


Figure 4. A small section from the 3×3 problem simulation model.

The designed MAS-RL structure is transferred to the event-based simulation model. The momentary status of each job, machine, or queue can be observed by stopping the model at any time. In this way, the job shop can be continuously monitored to determine whether there is a problem with any component.

7. Experimental Results

Since the simulation model works with random numbers, it is necessary to eliminate the effects of extreme values. For this reason, the number of replications is set to 30, and each simulation run ends when 5000 jobs are completed. The average results of 30 replications for the 3×3 , 5×5 , and 10×10 problems are presented in Table 10, Table 11, and Table 12, respectively.

Table 10. Simulation results of the 3×3 problem.

		Proportion of Tardy Jobs (%)	Mean Tardiness (Min.)	Maximum Tardiness (Min.)	Mean Earliness (Min.)	Maximum Earliness (Min.)	Mean flow Time (Min.)	Maximum Flow Time (Min.)	Work in Process	Makespan (Hour)
Method		PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
Scenario 1 Very Low	FIFO	0.0620	25.9	355.3	824.1	1557.1	196.2	1120.7	7.3	66,924.6
	SPT	1.2520	703.7	7210.0	863.0	1488.7	176.7	8192.4	6.6	66,924.6
	EDD	0.0007	0.4	12.8	837.1	1488.7	182.6	1319.7	6.8	66,921.9
	MAS-RL	1.4880	816.7	8946.9	837.9	1534.3	207.1	10,187.6	7.7	66,925.1
Scenario 2 Low	FIFO	2.0720	102.9	1045.5	718.3	1488.7	319.0	1760.6	12.7	62,742.6
	SPT	3.4600	1780.3	60,804.5	840.0	1488.7	273.9	61,818.7	10.9	62,748.0
	EDD	0.8080	21.7	616.1	734.7	1487.6	291.6	2001.2	11.6	62,745.4
	MAS-RL	4.4640	2177.4	62,340.9	804.8	1489.5	350.7	63,219.9	14.0	62,737.5
Scenario 3 Moderate	FIFO	97.1746	13,521.8	36,801.6	31.9	1360.6	14,517.6	37,617.2	621.6	59,198.3
	SPT	8.2614	24,910.4	1777,509.0	813.9	1476.3	2339.9	1778,872.0	483.4	59,078.0
	EDD	96.6794	13,495.2	36,005.7	24.9	1333.9	14,490.6	37,388.6	619.9	59,199.2
	MAS-RL	9.5980	13,124.4	929,348.3	778.0	1522.2	1560.0	930,029.2	619.0	59,197.3
Scenario 4 Heavy	FIFO	99.8687	146,470.2	309,508.1	14.0	1228.1	147,484.6	310,245.0	6795.2	59,525.5
	SPT	9.6920	20,484.6	982,673.0	800.9	1479.1	2267.7	984,130.9	5323.8	58,371.6
	EDD	99.8360	138,305.8	287,823.7	13.9	1228.1	139,319.3	289,232.9	6421.9	59,164.1
	MAS-RL	9.2300	2308.4	74,976.0	753.7	1543.2	549.9	75,725.3	9927.5	61,857.7
Scenario 5 Very Heavy	FIFO	99.9313	318,497.1	654,419.2	20.4	1228.1	319,509.6	655,283.1	15,976.4	61,046.3
	SPT	11.2326	21,244.2	1881,254.0	792.6	1437.9	2728.6	1882,365.0	11,563.3	58,026.9
	EDD	99.8860	263,448.1	538,873.6	14.9	1228.1	264,458.1	540,218.8	13,214.7	59,151.9
	MAS-RL	8.2454	1037.0	33,360.2	748.1	1481.7	419.1	34,138.3	25,073.8	67,228.3

PC: performance criterion; FIFO: first-in–first-out; SPT: shortest processing time; EDD: earliest due date; MAS-RL: multi-agent system with reinforcement learning.

Table 11. Simulation results of the 5×5 problem.

		Proportion of Tardy Jobs (%)	Mean Tardiness (Min.)	Maximum Tardiness (Min.)	Mean Earliness (Min.)	Maximum Earliness (Min.)	Mean Flow Time (Min.)	Maximum Flow Time (Min.)	Work in Process	Makespan (Hour)
Method		PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
Scenario 1 Very Low	FIFO	0.1774	31.8	396.7	807.3	1517.1	212.9	1155.0	7.7	40,159.6
	SPT	2.9273	672.6	6980.3	851.8	1492.3	198.6	7925.5	7.3	40,157.4
	EDD	0.0000	0.0	0.0	812.3	1476.4	207.4	1155.5	7.7	40,161.2
	MAS-RL	2.9066	694.5	11,580.3	833.9	1526.2	225.7	12,299.5	8.3	40,162.1
Scenario 2 Low	FIFO	2.9283	134.0	1463.8	703.2	1518.8	341.8	2200.4	13.4	37,647.4
	SPT	4.8063	1408.8	23,034.2	828.6	1520.7	300.9	23,837.0	11.6	37,648.8
	EDD	1.2274	41.7	1346.2	701.9	1443.2	332.2	2633.9	13.3	37,649.1
	MAS-RL	7.1273	1259.4	34,200.2	802.6	1490.5	366.6	34,911.7	14.3	37,647.2
Scenario 3 Moderate	FIFO	66.9928	9022.7	23,639.6	519.3	1475.3	7188.8	24,333.4	335.9	35,394.7
	SPT	10.2025	11,274.8	631,114.7	802.4	1473.7	1475.4	632,527.4	295.0	35,326.7
	EDD	65.1996	8563.3	24,306.8	460.5	1351.8	6709.3	25,625.7	316.9	35,394.7
	MAS-RL	8.1283	5066.7	95,829.3	769.9	1511.5	1244.8	296,884.5	234.3	35,423.5
Scenario 4 Heavy	FIFO	99.5923	73,912.2	223,541.9	11.5	1235.9	74,921.5	224,328.6	3216.2	35,244.3
	SPT	13.5168	24,535.7	1061,210.0	787.3	1424.7	3677.3	1062,187.0	1880.3	34,383.5
	EDD	99.4782	61,000.6	160,568.2	11.7	1253.6	62,009.0	161,911.7	2998.6	34,803.2
	MAS-RL	11.2910	6647.3	159,869.8	761.1	1510.4	2226.2	321,038.9	3162.7	35,176.7
Scenario 5 Very Heavy	FIFO	99.8029	182,154.6	447,721.8	14.3	1235.9	183,164.0	448,479.4	8475.7	36,380.5
	SPT	14.5394	24,415.0	1141,526.0	778.7	1378.4	3935.3	1142,311.0	4074.0	34,097.2
	EDD	99.7692	140,873.3	320,446.6	14.2	1232.3	141,881.7	321,842.8	7224.5	34,937.7
	MAS-RL	14.3290	12,056.7	279,491.2	761.2	1500.5	3535.1	380,282.3	7659.6	35,638.4

PC: performance criterion; FIFO: first-in–first-out; SPT: shortest processing time; EDD: earliest due date; MAS-RL: multi-agent system with reinforcement learning.

Table 12. Simulation results of the 10 × 10 problem.

		Proportion of Tardy Jobs (%)	Mean Tardiness (Min.)	Maximum Tardiness (Min.)	Mean Earliness (Min.)	Maximum Earliness (Min.)	Mean Flow Time (Min.)	Maximum Flow Time (Min.)	Work in Process	Makespan (Hour)
Method		PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
Scenario 1 Very Low	FIFO	0.0621	28.7	363.7	813.0	1567.4	207.2	1116.6	7.7	20,080.8
	SPT	1.7560	721.4	10,302.0	854.3	1518.9	193.9	11,260.4	7.7	20,079.6
	EDD	0.0567	2.4	235.7	811.6	1567.4	208.5	1522.4	8.0	20,081.5
	MAS-RL	2.4076	748.1	18,821.1	837.2	1569.2	219.3	13,091.2	8.4	20,082.4
Scenario 2 Low	FIFO	1.1907	102.3	965.3	718.6	1529.1	311.1	1670.8	12.3	18,826.3
	SPT	4.3534	1341.6	27,101.8	829.9	1488.7	287.7	27,905.2	11.6	18,827.4
	EDD	0.5360	17.7	398.7	710.3	1455.7	313.4	1735.6	12.8	18,825.0
	MAS-RL	6.3634	1185.1	29,832.9	804.0	1501.2	325.0	36,975.3	13.1	18,824.8
Scenario 3 Moderate	FIFO	50.4454	3762.5	18,931.2	482.6	1503.2	3019.0	19,763.4	219.7	17,668.8
	SPT	10.0694	5601.6	234,970.8	801.8	1467.3	879.2	236,211.9	193.3	17,650.0
	EDD	54.3946	3174.7	13,905.2	436.6	1408.5	2737.2	15,239.6	183.2	17,659.6
	MAS-RL	7.9996	2078.8	31,730.2	767.2	1516.8	703.8	86,913.1	183.0	17,670.2
Scenario 4 Heavy	FIFO	98.8220	35,831.5	115,192.0	16.5	1279.1	36,837.3	116,002.1	1771.3	17,599.8
	SPT	13.8220	17,165.0	878,177.5	784.8	1394.7	2760.8	879,134.1	1135.9	17,239.8
	EDD	98.5106	29,157.2	77,267.3	12.2	1279.1	30,162.4	78,584.2	1473.9	17,363.9
	MAS-RL	11.5424	3677.9	76,001.8	762.2	1466.0	1992.9	157,937.2	1612.4	17,412.1
Scenario 5 Very Heavy	FIFO	99.5094	90,354.6	228,592.7	13.3	1279.1	91,360.0	229,408.6	3977.2	18,181.1
	SPT	15.3220	18,423.5	571,284.7	777.2	1411.5	3194.9	572,492.6	2455.3	17,338.4
	EDD	99.4280	68,201.4	160,346.6	13.0	1279.1	69,208.6	161,772.0	3565.1	17,396.0
	MAS-RL	15.0272	7029.7	142,948.3	762.9	1515.7	2988.3	189,488.4	3614.8	18,002.5

PC: performance criterion; FIFO: first-in–first-out; SPT: shortest processing time; EDD: earliest due date; MAS-RL: multi-agent system with reinforcement learning.

Tables 10–12 should be evaluated by themselves. Moreover, each scenario should be evaluated on its own. The values highlighted in bold show the best result among FIFO, SPT, EDD, and the MAS-RL for that scenario. For example, for the 3×3 problem in Scenario 5, the MAS-RL gave the best result with 8.2454% in PC1. This is also true for the 5×5 and 10×10 problems: the MAS-RL gave the best results for PC1 in Scenario 5.

For each problem size in each scenario, the method that is best in most of the nine performance criteria is highlighted in bold. For example, for the 3×3 problem in Scenario 5, the MAS-RL gave the best results within five of the nine criteria. The same situation was observed in Scenario 4. So the MAS-RL was the best method for five different scenarios: two times for the 3×3 problem, three times for the 5×5 problem, and three times for the 10×10 problem.

When only examining from the aspect of the MAS-RL, from Scenario 1 to Scenario 5, the number of best results given by the MAS-RL was 0-1-2-5-5 for the 3×3 problem, 0-0-4-4-4 for the 5×5 problem, and 0-0-4-4-4 for the 10×10 problem. It can be said that the performance of the MAS-RL increased as it progressed from Scenario 1 to Scenario 5. The MAS-RL gave better results as the workload load increased.

Since PC1, PC2, and PC3 are related to tardiness, these criteria usually symmetrically act with each other. Looking at these criteria, EDD was generally expected to give the best results. However, for all the problem sizes, EDD only gave the best results in Scenario 1 and Scenario 2. The reason for this may be that the workload of the job shop increased a lot since Scenario 3, so it may cause bottlenecks. In systems with bottlenecks, standard dispatching rules may not yield the expected results. In Scenario 4 and Scenario 5, it was observed that the MAS-RL excels in tardiness.

Since PC4 and PC5 are related to earliness, these criteria were examined together. Earliness and tardiness factors are expected to act in opposition to each other. When examined for all problem sizes, it can be said that EDD gave good results for PC4 and PC5. It was observed that the MAS-RL did not achieve good results in terms of earliness.

SPT was expected to give the best results for the PC6 and PC7 criteria in terms of flow time. It is well-known that SPT minimizes the flow time in the single machine scheduling problem. However, this is not the case in the DJSP. For all problem sizes, for the aspect of PC6, SPT (2 out of 5) under low workloads and the MAS-RL (3 out of 5) under heavy workloads gave the best results. For the aspect of PC7, FIFO and EDD shared first place, while the MAS-RL only gave the best results for the 3×3 problem size for Scenario 4 and Scenario 5.

PC8 is a measure that shows the number of jobs in the job shop at any given moment. It is one of the important criteria that indicate the chaos in the job shop. The higher it is, the harder it is to keep track of jobs and scheduling activities. For this reason, PC8 is desired to be low. The rule that is expected to give the best results for PC8 in the literature is SPT and its derivatives. As expected, SPT gave the best results in almost every situation.

One of the frequently used performance criteria for scheduling problems in the literature is PC9. The makespan indicates how long it takes to complete a certain number of jobs. In other words, it shows when the last job exited the system. For all problem sizes, it is seen that SPT gave the best results as the workload increased in Scenarios 3, 4, and 5.

An event-based graph of the W values in the 3×3 problem is given in Figure 5 to examine the curve. As seen in the graph, the W values started from 0 at the beginning of the simulation and made peaks to extreme values. After the peaks, the rate of change in the W value gradually decreased and became stable state. A graph was similarly formed for the 5×5 and 10×10 problems.

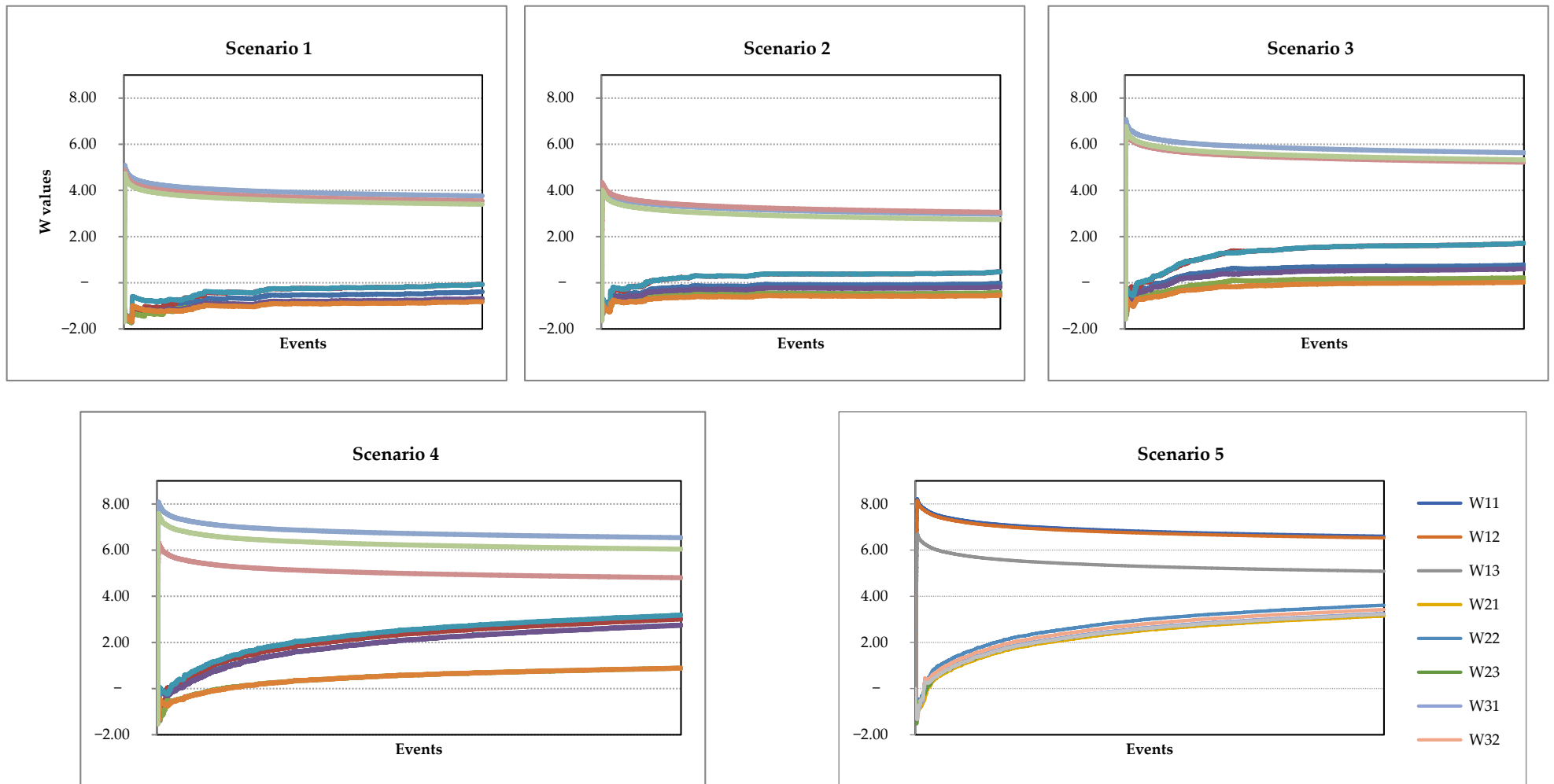


Figure 5. Event-based graph of W values in 3×3 problem.

In the graphs shown in Figure 5, it is seen that a learning curve (LC), which is very common in machine learning studies in the literature, has emerged. The LC is known for initially making hard peaks and becoming stable as time passes [71]. The LC describes a system's performance on a task as a function over some resource to solve that task, as shown in Figure 6.

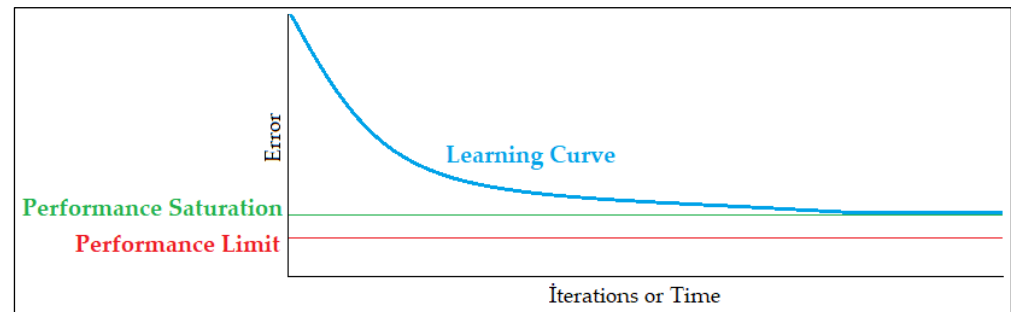


Figure 6. A representation of LC.

In machine learning studies, performance criteria such as the Mean Squared Error (MSE) or the Mean Absolute Percentage Error (MAPE) are often used. In our study, instead of using them, a strategy based on instantly correcting the error occurred was adopted. The MAS-RL constantly monitored the system and updated the W values according to the magnitude of the errors. That is, the error and W values symmetrically proceeded according to each other.

8. Conclusions

In this paper, a MAS-RL approach was proposed to solve the DJSP. The performance of the proposed approach were compared to the FIFO, SPT, and EDD dispatching rules in the literature. Five different scenarios with increasing job arrival rates and nine different performance criteria were used for comparison. Experiments were performed for the 3×3 , 5×5 , and 10×10 problem sizes. The following conclusions were made from the experimental results.

1. As the workload increases, the MAS-RL performs better. From Scenario 1 to Scenario 5, the workload increases along with the performance of the MAS-RL. It is understood that this is caused by two factors. The first factor is that as the workload increases, the number of jobs in the system also increases, so more scheduling activities are needed. The MAS-RL quickly examines the status of all jobs in the system and makes the most appropriate choices. The second factor is that the MAS-RL starts to make more effective decisions after completing its learning stage. Having many jobs in the system at the same time enables the MAS-RL to learn faster and also allows it to apply what it has learned to more jobs.
2. The MAS-RL can successfully overcome tardiness. For all problem sizes, the MAS-RL gave the best results in Scenarios 4 and 5 for all the performance criteria related to tardiness. In the literature, the dispatching rules that work best for tardiness are known as EDD and its derivatives. However, the MAS-RL showed promising results for tardiness, outperforming EDD under heavy workloads.
3. The MAS-RL can reduce the flow time. For the 3×3 problem in Scenarios 4 and 5, the MAS-RL gave the best results for all the performance criteria related to flow time. For the 5×5 and 10×10 problems, the MAS-RL only gave the best results for PC6 (mean flow time).
4. The MAS-RL can give feasible solutions for the aspect of the makespan. The makespan shows how long the duration is to complete a certain number of jobs. For all the problem sizes in Scenario 2, the MAS-RL gave the best results for the makespan. For real businesses, it is not very meaningful to only look at the makespan. Even when

the makespan is optimal, if orders exceed the due date, it would not be long before the business loses its customers. We still included the makespan in our study, as it has been calculated since the first studies of scheduling problems.

5. There is no remarkable relation between the size of the problem and the performance of the MAS-RL. Except for minor differences, the solution methods for all the problem sizes yield similar results.

Another unique contribution of this study to the literature is that each job type could receive different priorities on each machine. In addition, the priorities were reconciled with the RL mechanism so that they could change over time. This technique allowed for more flexible changes to be possible in the production schedule.

In future studies, the MAS-RL can be tested in even larger or smaller systems. Dynamic events such as machine failures and order cancellations can be implemented in future research. A different variety of parameters can be used in the calculation of the W values. This may change the duration of the learning period for the MAS-RL. Researchers can adapt this scheduling method to their own studies for different problem types.

Author Contributions: Conceptualization, A.F.İ.; methodology, A.K.T. and A.F.İ.; software, A.F.İ. and Ç.S.; validation, A.A., Ç.S. and S.E.; formal analysis, A.F.İ.; investigation, A.F.İ.; resources, A.F.İ. and S.E.; data curation, A.F.İ.; writing—original draft preparation, A.F.İ. and Ç.S.; writing—review and editing, A.F.İ. and Ç.S.; visualization, A.F.İ.; supervision, Ç.S., A.K.T. and S.E.; project administration, A.F.İ. and S.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

ACO	Ant colony optimization
AIS	Artificial immune systems
CPPS	Cyber-physical production systems
CR	Critical ratio
DD	Due date
D-FJSP	Dynamic flexible job shop scheduling problem
DJSP	Dynamic job shop scheduling problem
DMAS	Distributed multi-agent system
DSS	Decision support system
EDD	Earliest due date
FBS	Filtered beam search
FIFO	First-in–first-out
FMS	Flexible manufacturing system
GNN	Graph neural network
GRASP	Greedy randomized adaptive search procedure
HFS	Hybrid flow shop
HICA	Hybrid imperialist competitive algorithm
IMS	Intelligent manufacturing systems
JSP	Job shop scheduling problem
MAS	Multi-agent system
MB	Machine breakdowns
NRGA	Non-dominated ranking genetic algorithm
NSGA	Non-dominated sorting genetic algorithm
OA	Order arrivals

OC	Order cancellations
PT	Processing times
RL	Reinforcement learning
SA	Simulated annealing
SEA	Symbiotic evolutionary algorithm
SHFS	Sustainable hybrid flow shop
SPT	Shortest processing time
ST	Setup times
TS	Tabu search
UO	Urgent orders
VNS	Variable neighborhood search
WSI	Weighted sum of indicators

References

- Nelson, R.T.; Holloway, C.A.; Wong, R.M.-L. Centralized Scheduling and Priority Implementation Heuristics for a Dynamic Job Shop Model. *AIIE Trans.* **1977**, *9*, 95–102. [[CrossRef](#)]
- Anciaux, D.; Roy, D.; Vernadat, F. Reactive Shop-Floor Control with a Multi-Agent System. *IFAC Proc. Vol.* **1997**, *30*, 425–430. [[CrossRef](#)]
- Brauer, W.; Weib, G.; Munchen, T.U. Multi-Machine Scheduling—A Multi-Agent Learning Approach. In Proceedings of the International Conference on Multi Agent Systems, (Cat. No. 98EX160). Paris, France, 3–7 July 1998; pp. 42–48.
- Chen, Y.-Y.; Fu, L.-C.; Chen, Y.-C. Multi-agent based dynamic scheduling for a flexible assembly system. In Proceedings of the IEEE International Conference on Robotics and Automation, (Cat. No. 98CH36146). Leuven, Belgium, 20 May 1998; pp. 2122–2127. [[CrossRef](#)]
- Shen, W.; Maturana, F.; Norrie, D. Learning in Agent-Based Manufacturing. In Proceedings of the Artificial Intelligence and Manufacturing Research Planning Workshop, Madison, WI, USA, 26–30 July 1998; pp. 177–183.
- Sousa, P.; Ramos, C. A distributed architecture and negotiation protocol for scheduling in manufacturing systems. *Comput. Ind.* **1999**, *38*, 103–113. [[CrossRef](#)]
- Maturana, F.; Shen, W.; Norrie, D. MetaMorph: An adaptive agent-based architecture for intelligent manufacturing. *Int. J. Prod. Res.* **1999**, *37*, 2159–2173. [[CrossRef](#)]
- Ouelhadj, D.; Hanach, C.; Bouzouia, B. Multi-agent system for dynamic scheduling and control in manufacturing cells. In Proceedings of the 1998 IEEE International Conference on Robotics and Automation, (Cat. No. 98CH36146). Leuven, Belgium, 20 May 1998. [[CrossRef](#)]
- Ouelhadj, D.; Hanachi, C.; Bouzouia, B.; Moualek, A.; Farhi, A. A multi-contract net protocol for dynamic scheduling in flexible manufacturing systems (FMS). In Proceedings of the 1999 IEEE International Conference on Robotics and Automation, (Cat. No. 99CH36288C). Detroit, MI, USA, 10–15 May 1999. [[CrossRef](#)]
- Frey, D.; Nimis, J.; Wörn, H.; Lockemann, P. Benchmarking and robust multi-agent-based production planning and control. *Eng. Appl. Artif. Intell.* **2003**, *16*, 307–320. [[CrossRef](#)]
- Grandgirard, J.; Poinot, D.; Krespi, L.; Nénon, J.P.; Cortesero, A.M. Costs of secondary parasitism in the facultative hyperparasitoid *Pachycrepoides dubius*: Does host size matter? *Entomol. Exp. Appl.* **2002**, *103*, 239–248. [[CrossRef](#)]
- Bongaerts, L.; Monostori, L.; McFarlane, D.; Kádár, B. Hierarchy in distributed shop floor control. *Comput. Ind.* **2000**, *43*, 123–137. [[CrossRef](#)]
- Paternina-Arboleda, C.D.; Das, T.K. A multi-agent reinforcement learning approach to obtaining dynamic control policies for stochastic lot scheduling problem. *Simul. Model. Pr. Theory* **2005**, *13*, 389–406. [[CrossRef](#)]
- Liu, S.; Ong, H.; Ng, K. Metaheuristics for minimizing the makespan of the dynamic shop scheduling problem. *Adv. Eng. Softw.* **2005**, *36*, 199–205. [[CrossRef](#)]
- Wang, Y.-C.; Usher, J.M. Application of reinforcement learning for agent-based production scheduling. *Eng. Appl. Artif. Intell.* **2005**, *18*, 73–82. [[CrossRef](#)]
- Wong, T.; Leung, C.; Mak, K.; Fung, R. Dynamic shopfloor scheduling in multi-agent manufacturing systems. *Expert Syst. Appl.* **2006**, *31*, 486–494. [[CrossRef](#)]
- Wang, S.-J.; Xi, L.-F.; Zhou, B.-H. FBS-enhanced agent-based dynamic scheduling in FMS. *Eng. Appl. Artif. Intell.* **2008**, *21*, 644–657. [[CrossRef](#)]
- Blanc, P.; Demongodin, I.; Castagna, P. A holonic approach for manufacturing execution system design: An industrial application. *Eng. Appl. Artif. Intell.* **2008**, *21*, 315–330. [[CrossRef](#)]
- Xiang, W.; Lee, H. Ant colony intelligence in multi-agent dynamic manufacturing scheduling. *Eng. Appl. Artif. Intell.* **2008**, *21*, 73–85. [[CrossRef](#)]

20. Chaouch, I.; Driss, O.B.; Ghedira, K. A Survey of Optimization Techniques for Distributed Job Shop Scheduling Problems in Multi-factories. In *Cybernetics and Mathematics Applications in Intelligent Systems: Proceedings of the 6th Computer Science On-line Conference 2017 (CSOC2017)*, 26–29 April 2017; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 369–378. [[CrossRef](#)]
21. Guo, Q.-L.; Zhang, M. Multiagent-based scheduling optimization for Intelligent Manufacturing System. *Int. J. Adv. Manuf. Technol.* **2008**, *44*, 595–605. [[CrossRef](#)]
22. Guo, Q.; Zhang, M. A novel approach for multi-agent-based Intelligent Manufacturing System. *Inf. Sci.* **2009**, *179*, 3079–3090. [[CrossRef](#)]
23. Ouelhadj, D.; Petrovic, S. A survey of dynamic scheduling in manufacturing systems. *J. Sched.* **2008**, *12*, 417–431. [[CrossRef](#)]
24. Cossentino, M.; Fortino, G.; Gleizes, M.-P.; Pavón, J. Simulation-based design and evaluation of multi-agent systems. *Simul. Model. Pr. Theory* **2010**, *18*, 1425–1427. [[CrossRef](#)]
25. Moyaux, T.; Liu, Y.; Bouleux, G.; Cheutet, V. An agent-based architecture of the Digital Twin for an Emergency Department. *Sustainability* **2023**, *15*, 3412. [[CrossRef](#)]
26. Asadzadeh, L. A local search genetic algorithm for the job shop scheduling problem with intelligent agents. *Comput. Ind. Eng.* **2015**, *85*, 376–383. [[CrossRef](#)]
27. Aydemir, E.; Koruca, H.I. A New Production Scheduling Module Using Priority-Rule Based Genetic Algorithm. *Int. J. Simul. Model.* **2015**, *14*, 450–462. [[CrossRef](#)]
28. Wang, S.; Wan, J.; Zhang, D.; Li, D.; Zhang, C. Towards smart factory for industry 4.0: A self-organized multi-agent system with big data based feedback and coordination. *Comput. Netw.* **2016**, *101*, 158–168. [[CrossRef](#)]
29. Karnouskos, S.; Leitao, P. Key Contributing Factors to the Acceptance of Agents in Industrial Environments. *IEEE Trans. Ind. Informatics* **2016**, *13*, 696–703. [[CrossRef](#)]
30. Li, K.; Zhou, T.; Liu, B.-H.; Li, H. A multi-agent system for sharing distributed manufacturing resources. *Expert Syst. Appl.* **2018**, *99*, 32–43. [[CrossRef](#)]
31. Zhou, L.; Zhang, L.; Sarker, B.R.; Laili, Y.; Ren, L. An event-triggered dynamic scheduling method for randomly arriving tasks in cloud manufacturing. *Int. J. Comput. Integr. Manuf.* **2017**, *31*, 318–333. [[CrossRef](#)]
32. Gao, K.; Cao, Z.; Zhang, L.; Chen, Z.; Han, Y.; Pan, Q. A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 904–916. [[CrossRef](#)]
33. Wang, J.; Zhang, Y.; Liu, Y.; Wu, N. Multiagent and Bargaining-Game-Based Real-Time Scheduling for Internet of Things-Enabled Flexible Job Shop. *IEEE Internet Things J.* **2018**, *6*, 2518–2531. [[CrossRef](#)]
34. Zhang, C.-L.; Wang, J.-Q.; Zhang, C.-W. Two-agent scheduling on a single parallel-batching machine to minimize the weighted sum of the agents' makespans. *J. Ambient. Intell. Humaniz. Comput.* **2018**, *10*, 999–1007. [[CrossRef](#)]
35. Mohan, J.; Lanka, K.; Rao, A.N. A Review of Dynamic Job Shop Scheduling Techniques. *Procedia Manuf.* **2019**, *30*, 34–39. [[CrossRef](#)]
36. Komma, V.R.; Jain, P.K.; Mehta, N.K. An approach for agent modeling in manufacturing on JADE™ reactive architecture. *Int. J. Adv. Manuf. Technol.* **2010**, *52*, 1079–1090. [[CrossRef](#)]
37. Owliya, M.; Saadat, M.; Anane, R.; Goharian, M. A New Agents-Based Model for Dynamic Job Allocation in Manufacturing Shopfloors. *IEEE Syst. J.* **2012**, *6*, 353–361. [[CrossRef](#)]
38. Leitao, P.; Rodrigues, N.; Turrin, C.; Pagani, A. Multiagent System Integrating Process and Quality Control in a Factory Producing Laundry Washing Machines. *IEEE Trans. Ind. Inform.* **2015**, *11*, 879–886. [[CrossRef](#)]
39. Yu, F.; Wen, P.; Yi, S. A multi-agent scheduling problem for two identical parallel machines to minimize total tardiness time and makespan. *Adv. Mech. Eng.* **2018**, *10*, 1687814018756103. [[CrossRef](#)]
40. Wong, T.; Zhang, S.; Wang, G.; Zhang, L. Integrated process planning and scheduling—Multi-agent system with two-stage ant colony optimisation algorithm. *Int. J. Prod. Res.* **2012**, *50*, 6188–6201. [[CrossRef](#)]
41. Wang, Y.; Liu, H.; Zheng, W.; Xia, Y.; Li, Y.; Chen, P.; Guo, K.; Xie, H. Multi-Objective Workflow Scheduling with Deep-Q-Network-Based Multi-Agent Reinforcement Learning. *IEEE Access* **2019**, *7*, 39974–39982. [[CrossRef](#)]
42. Kim, Y.G.; Lee, S.; Son, J.; Bae, H.; Chung, B.D. Multi-agent system and reinforcement learning approach for distributed intelligence in a flexible smart manufacturing system. *J. Manuf. Syst.* **2020**, *57*, 440–450. [[CrossRef](#)]
43. Lee, W.-C.; Chung, Y.-H.; Wang, J.-Y. A parallel-machine scheduling problem with two competing agents. *Eng. Optim.* **2016**, *49*, 962–975. [[CrossRef](#)]
44. Ahmadi, E.; Zandieh, M.; Farrokh, M.; Emami, S.M. A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms. *Comput. Oper. Res.* **2016**, *73*, 56–66. [[CrossRef](#)]
45. Shiue, Y.-R.; Lee, K.-C.; Su, C.-T. Real-time scheduling for a smart factory using a reinforcement learning approach. *Comput. Ind. Eng.* **2018**, *125*, 604–614. [[CrossRef](#)]
46. Sahin, C.; Demirtas, M.; Erol, R.; Baykasoğlu, A.; Kaplanoğlu, V. A multi-agent based approach to dynamic scheduling with flexible processing capabilities. *J. Intell. Manuf.* **2015**, *28*, 1827–1845. [[CrossRef](#)]
47. Maoudj, A.; Bouzouia, B.; Hentout, A.; Kouider, A.; Toumi, R. Distributed multi-agent scheduling and control system for robotic flexible assembly cells. *J. Intell. Manuf.* **2017**, *30*, 1629–1644. [[CrossRef](#)]
48. Huang, C.-J.; Liao, L.-M. A multi-agent-based negotiation approach for parallel machine scheduling with multi-objectives in an electro-etching process. *Int. J. Prod. Res.* **2012**, *50*, 5719–5733. [[CrossRef](#)]

49. Liu, Y.; Wang, L.; Wang, Y.; Wang, X.V.; Zhang, L. Multi-agent-based scheduling in cloud manufacturing with dynamic task arrivals. *Procedia CIRP* **2018**, *72*, 953–960. [[CrossRef](#)]
50. Jiang, Z.; Jin, Y.; Mingcheng, E.; Li, Q. Distributed Dynamic Scheduling for Cyber-Physical Production Systems Based on a Multi-Agent System. *IEEE Access* **2017**, *6*, 1855–1869. [[CrossRef](#)]
51. Zhang, S.; Wong, T.N. Flexible job-shop scheduling/rescheduling in dynamic environment: A hybrid MAS/ACO approach. *Int. J. Prod. Res.* **2016**, *55*, 3173–3196. [[CrossRef](#)]
52. Barenji, A.V.; Barenji, R.V.; Roudi, D.; Hashemipour, M. A dynamic multi-agent-based scheduling approach for SMEs. *Int. J. Adv. Manuf. Technol.* **2016**, *89*, 3123–3137. [[CrossRef](#)]
53. Shi, L.; Guo, G.; Song, X. Multi-agent based dynamic scheduling optimisation of the sustainable hybrid flow shop in a ubiquitous environment. *Int. J. Prod. Res.* **2019**, *59*, 576–597. [[CrossRef](#)]
54. Luo, S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Appl. Soft Comput.* **2020**, *91*, 106208. [[CrossRef](#)]
55. Baykasoglu, A.; Karaslan, F.S. Solving comprehensive dynamic job shop scheduling problem by using a GRASP-based approach. *Int. J. Prod. Res.* **2017**, *55*, 3308–3325. [[CrossRef](#)]
56. Sel, Ç.; Hamzadayı, A. A simulated annealing approach based simulation-optimisation to the dynamic job-shop scheduling problem. *Pamukkale Univ. J. Eng. Sci.* **2018**, *24*, 665–674. [[CrossRef](#)]
57. Zhang, H.; Roy, U. A semantics-based dispatching rule selection approach for job shop scheduling. *J. Intell. Manuf.* **2018**, *30*, 2759–2779. [[CrossRef](#)]
58. Turker, A.K.; Aktepe, A.; Inal, A.F.; Ersoz, O.O.; Das, G.S.; Birgoren, B. A Decision Support System for Dynamic Job-Shop Scheduling Using Real-Time Data with Simulation. *Mathematics* **2019**, *7*, 278. [[CrossRef](#)]
59. Aydin, M.; Öztemel, E. Dynamic job-shop scheduling using reinforcement learning agents. *Robot. Auton. Syst.* **2000**, *33*, 169–178. [[CrossRef](#)]
60. Kardos, C.; Laflamme, C.; Gallina, V.; Sihn, W. Dynamic scheduling in a job-shop production system with reinforcement learning. *Procedia CIRP* **2021**, *97*, 104–109. [[CrossRef](#)]
61. Erol, R.; Sahin, C.; Baykasoglu, A.; Kaplanoglu, V. A multi-agent based approach to dynamic scheduling of machines and automated guided vehicles in manufacturing systems. *Appl. Soft Comput.* **2012**, *12*, 1720–1732. [[CrossRef](#)]
62. Jana, T.K.; Bairagi, B.; Paul, S.; Sarkar, B.; Saha, J. Dynamic schedule execution in an agent based holonic manufacturing system. *J. Manuf. Syst.* **2013**, *32*, 801–816. [[CrossRef](#)]
63. Leusin, M.E.; Kück, M.; Frazzon, E.M.; Maldonado, M.U.; Freitag, M. Potential of a Multi-Agent System Approach for Production Control in Smart Factories. *IFAC-PapersOnLine* **2018**, *51*, 1459–1464. [[CrossRef](#)]
64. Leusin, M.E.; Frazzon, E.M.; Maldonado, M.U.; Kück, M.; Freitag, M. Solving the Job-Shop Scheduling Problem in the Industry 4.0 Era. *Technologies* **2018**, *6*, 107. [[CrossRef](#)]
65. Sels, V.; Gheysen, N.; Vanhoucke, M. A comparison of priority rules for the job shop scheduling problem under different flow time- and tardiness-related objective functions. *Int. J. Prod. Res.* **2012**, *50*, 4255–4270. [[CrossRef](#)]
66. Holthaus, O.; Rajendran, C. Efficient dispatching rules for scheduling in a job shop. *Int. J. Prod. Econ.* **1997**, *48*, 87–105. [[CrossRef](#)]
67. Jain, A.; Meeran, S. Deterministic job-shop scheduling: Past, present and future. *Eur. J. Oper. Res.* **1999**, *113*, 390–434. [[CrossRef](#)]
68. Yazdani, M.; Aleti, A.; Khalili, S.M.; Jolai, F. Optimizing the sum of maximum earliness and tardiness of the job shop scheduling problem. *Comput. Ind. Eng.* **2017**, *107*, 12–24. [[CrossRef](#)]
69. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; MIT Press: Cambridge, MA, USA, 2018; ISBN 9780262352703.
70. Baykasoğlu, A.; Göçken, M.; Unutmaz, Z.D. New approaches to due date assignment in job shops. *Eur. J. Oper. Res.* **2008**, *187*, 31–45. [[CrossRef](#)]
71. Mohr, F.; van Rijn, J.N. Learning Curves for Decision Making in Supervised Machine Learning—A Survey. *arXiv* **2022**, arXiv:2201.12150.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.