Al-Khwarizmi Engineering Journal

Al-Khwarizmi Engineering Journal
ISSN (printed): 1818–1171, ISSN (online): 2312–0789
Vol. 22, No. 2, June, (2026), pp. 72–84

Dynamic Job Scheduling in Manufacturing Systems using Deep Q-Learning

Hind Khalid
Al-Nahrain University, Baghdad, Iraq
Email: dr.hind@nahrainuniv.edu.iq

(Received 3 June 2025; Revised 22 October 2025; Accepted 30 October 2026; Published 1 June 2026)
https://doi.org/10.22153/kej.2026.10.001

Abstract

In this connection, this paper proposes a Deep Q-Network (DQN) approach to address the dynamic nature of the job-shop scheduling problem. Dynamic scheduling requires an efficient and reliable algorithm for handling disruptions such as machine breakdowns and job priority changes. When comparing DQN with traditional approaches such as genetic algorithm (GA) and PSO, the latter algorithms cannot cope with the problems. On the contrary, DQN gains knowledge from its experiences within the factory and develops a strategy for solving the scheduling problem through minimizing makespan and maximizing machine utilization. Moreover, using experience replay (ER) and target networks enables DQN to maintain stability and develop an optimal schedule. Empirically, it was found that DQN reduces makespan by 24.8% with machine utilization being 92%. From the results obtained, it can be noted that the learning parameters play a great role in determining the performance of the model. Thus, this study proves that DQN is an effective approach for addressing the issue under discussion and could also be used for developing other approaches such as multi-agent and double DQN.

Keywords: *Dynamic job-shop scheduling; Deep Q-Learning; Reinforcement learning; Machine utilisation; Makespan optimisation*

1. Introduction

Dynamic job scheduling is an essential element in the successful implementation of current manufacturing practices, especially in light of changes that include greater customization, shorter production times, and better quality. Manufacturing practices have become complicated operations, and a significant factor in this evolution has been the emergence of the fourth industrial revolution, Industry 4.0. Traditional scheduling systems can fail to adapt to the constant changes that may occur, including machine malfunction, priority changes, and other outside influences. Methods like genetic algorithms, dynamic programming, and particle swarm optimization are well-known in the literature but usually involve set parameters suitable only for static settings. And they require considerable computational resources to get the desired results. The involvement of expert knowledge is crucial in adjusting parameters, and it limits its use in other manufacturing contexts [1].

Deep Reinforcement Learning (DRL) is an excellent strategy for controlling sophisticated processes. Being a combination of reinforcement learning (RL) and deep learning (DL), DRL can easily adjust to new environments. In contrast to conventional strategies, DRL has the potential of dealing with complex situations. Deep Q-Learning (DQN), which falls under DRL techniques, applies neural networks for estimating the values of actions, thus helping an agent perform better in any environment [2]. In this study, the implementation of DQN is explored for the purpose of improving

the dynamic scheduling processes in job-shops. One of the goals here is to boost the efficiency level by making cuts in the processing time and cost. This research focuses on taking into account such variables as tools failure or deviation from normal processing operations. Through the incorporation of DQN in the process of designing activity conservation zones and choosing constraints for scheduling, this study develops an updated version of the scheduling system that addresses the shortcomings of existing methods. This system employs the experience replay (ER) strategy, enabling the agent to decouple learning steps and increase generalisation. Furthermore, the adoption of target networks (TNs) helps stabilise training by offering a fixed target for updating Q-values[3].

The experimental findings and evaluations show that the new technique outperforms conventional and heuristic-based scheduling techniques. The dynamically created architecture presented in this paper is more effective in promoting manufacturing efficiency, which is one of the tenets of smart manufacturing. It is an illustration of how manufacturing practices have evolved towards sustainability, whereby efficient resource management and top-notch strategies are key to remaining ahead of competitors. By addressing the requirement for real-time scheduling solutions, this research advances the development of responsive manufacturing systems.

The contribution of this work lies in a DQN system that accounts for the stochastic nature of dynamic job-shop scheduling, specifically modelling machine failures via exponential distribution. Although prior studies have applied RL to scheduling, this research contributes by integrating ER and TNs within the Q-learning framework, rendering the process robust against time-sensitive issues. This advancement is significant when compared to static heuristics and other RL applications in existing literature that lack such rigorous architectural considerations.

2. Literature Review

Early research into job scheduling primarily has relied on traditional optimisation methods. Kaelbling et al. [15] developed Q-Learning, establishing the foundation for RL in decision-making; however, although effective for small state spaces, this approach encountered significant challenges when scaled to larger problems. Sha and Lin [14] used PSO for static scheduling, , demonstrating rapid convergence for stable problems but failing to adapt effectively to dynamic environments. Wang et al. [13] also considered GA for static scheduling that achieved impressive performance under fixed conditions, but was not sufficiently adaptive to varying settings. To make intelligent, dynamic scheduling possible under Industry 4.0 manufacturing systems, the use of deep learning became essential, and researchers started combining deep learning with RL. The latest advances in the area include adding DDQN in dynamic task scheduling by Wang et al. [17], which enabled efficient evaluation of actions and values, although made the models more complicated. To enable learning in stochastic settings, Schulman et al. [18] proposed using proximal policy optimisation (PPO) but noted that the approach may involve considerable parameter tuning. DQN was successfully applied to complicated selection tasks by Rajiv [7], which proved the model's ability to handle complex states. Furthermore, Lee et al. [8] managed to optimize the equipment usage by applying DQN. Scheduling was also approached using Q-learning by Suganuma et al. [6] but their experiments proved that the curse of dimensionality has a negative impact on performance when processing bigger cases.

Current advancements have been centred around advanced RL algorithms and multi-objective optimisation approaches. Zhao et al. [19] used DQN in job-shop scheduling, yielding better performance due to efficient use of machines and reduction in costs; nevertheless, their findings were still dependent on hyperparameters. Luo et al. [20] introduced a multi-objective DQN for flexible job scheduling, balancing different objectives but proposing that future studies should investigate multi-agent optimisation approaches. Zhang et al. [16] established that DQN is efficient in handling large state-action spaces for difficult problems even though the computation is resource-intensive. Cai et al. [9] improved convergence by incorporating PPO with constraint updates, which is especially useful in active learning environments. Moreover, Valckenaers et al. [10] employed DDQN to separate the value function from the advantage function. In spite of these developments, there are still many research areas that require investigation. First, most RL algorithms lack the ability to adapt in real-time, necessitating a retrain process in each new environment. Secondly, there is a tendency to investigate one objective while ignoring the existing trade-offs in multi-objective scheduling problems. The combination of efficient machine utilization and minimizing processing time has not been well studied [11], [12]. In order to bridge

these gaps in current research, this paper applies DQN to develop a schedule based on multiple objectives. Table 1 provides a summary of important papers related to the current literature and their results. Despite being a dependable algorithmic approach, modern algorithms like Proximal Policy Optimization (PPO) [18] and Double Deep Q-Networks provide enhanced stability from unstable policies and overestimated values. Comparison with these algorithms can help validate the proposed RL-based scheduling system. Moreover, numerous investigations, including the current study, face limitations due to computational learning constraints and the challenge of applying simulation-trained models to practical production lines.

Table 1,
Key Studies, Their Methodologies, Applications and Findings

Study	Method	Application	Key Findings	Limitations	Ref
Kaelbling et al.	Q-Learning	General RL Problems	Simple, effective in small state spaces	Scalability issues in larger problems	[15]
Sha and Lin	PSO	Static Scheduling	Fast convergence under stable scenarios	Inefficiency in dynamic environments	[14]
Wang et al.	GA	Static Scheduling	Effective for stable conditions	Poor adaptability to dynamic changes	[13]
Wang et al.	DDQN	Dynamic Scheduling	Enhanced action evaluation efficiency	Increased model complexity	[17]
Schulman et al.	PPO	Dynamic Scheduling	Stable learning in unpredictable environments	Requires extensive parameter tuning	[18]
Rajiv	DQN	Complex Selection Tasks	Applicable to large state spaces	High computational demands	[7]
K. Lee et al.	DQN	Equipment Scheduling	Increased utilisation, reduced costs	Requires retraining for new environments	[8]
Suganuma et al.	Q-Learning	Task Allocation	Effective in small cases	Curse of dimensionality in large spaces	[6]
Valckenaers et al.	DDQN	Manufacturing Systems	Improved value estimation via separated architecture	Complex network design	[10]
Zhao et al.	DQN	Job-Shop Scheduling	Optimised machine utilisation, reduced costs	Sensitive to hyperparameter changes	[19]
Luo et al.	Multi-Objective DQN	Flexible Scheduling	Balanced trade-offs between multiple objectives	Requires further multi-agent validation	[20]
Zhang et al.	DQN	Complex Tasks	Efficient in large state-action spaces	High computational demands	[16]
Cai et al.	PPO	Active Scheduling	Improved policy convergence stability	Needs careful reward shaping	[9]
Proposed Work	Enhanced DQN	Dynamic Job-Shop Scheduling	Balances multiple objectives, high adaptability	Computationally intensive training	This Study

2.1. Problem Formulation

Dynamic job-shop scheduling in manufacturing systems aims to optimise the allocation of jobs to machines in real time. . This process considers various constraints, including machine availability, task priorities and sudden disruptions such as machine breakdowns. This section formalises the problem using a mathematical framework based on Markov decision processes (MDP), which underpins the RL methodology.

Objective Function

The key goal in dynamic job scheduling lies in minimizing either the overall completion time Cmax or the cost of operations based on the unique needs of the production process. The completion time Cmax can be stated mathematically as:

$$C_{\max} = \min\left(\sum_{i=1}^{n}\sum_{j=1}^{m} T_{ij}\right), \quad \ldots(1)$$

Where
n=number of jobs
m=total number of machines
T_ij=time taken by ith job at jth machine

By minimising C_max, we ensure that the entire set of jobs is done within the minimum possible time. This helps to improve system performance.
State and Action Space

This problem is modelled into an MDP. Each state s∈S corresponds to an instance of manufacturing system and each action a∈A corresponds to assigning a particular job to a specific machine.

2.2. State Space S

Consider the following:

$$\boldsymbol{S} = \{\boldsymbol{s_1}, \boldsymbol{s_2}, \dots, \boldsymbol{s_n}\}. \quad \dots(2)$$

Each state s_i contains information about the current status of machines, jobs in progress and jobs waiting in the queue.

2.3. Action Space A

Consider the following.

$$A = \{a_1, a_2, \dots, a_m\}. \quad \dots(3)$$

Each action a_j Represents assigning a specific job to a machine.

2.4. Reward Function

The learning process takes place based on the reward function that gives information about the goodness of the schedule chosen. The reward R(s, a) is defined in such a way that it inversely correlates with the time to complete for the particular state-action pair:

$$R(s,a) = -C_{max}(s,a). \quad \dots(4)$$

This is because such behavior will be rewarded with higher incentives, making the system prefer more effective scheduling.

2.5. Mathematical Constraints

The mathematical model of the scheduling problem includes several constraints:

- Machine Capacity: Only one job may be assigned to a machine at any point in time.
- Job Sequence: Some jobs should be performed according to a predetermined sequence.
- Machine Failure: Machine failures are considered random events representing machine breakdowns.

The constraints can be defined as follows:

$$x_{i,j} \in \{0,1\}, \quad \forall i,j$$

$$\sum_{j=1}^{m} x_{i,j} = 1, \quad \forall i$$

$$If\ Job_i \rightarrow Job_k\ then\ C_{ij} \leq S_{kj}, \quad \dots(5)$$

where
$x_{i,j}$ represents a binary value where 1 denotes the allocation of job i to machine j, and j, C_{ij} represents the completion time of job i on machine j, whereas S_k_j represents the start time of job k.

2.6. MDP Framework Visualisation

A conceptual diagram of the MDP framework is provided in Table 2 [22]. This framework illustrates the transition between states based on actions, with the reward signal guiding the optimisation of scheduling policies.

Table 2,
Parameter Definitions

Parameter	Definition
C_{max}	Total completion time for all jobs
T_{ij}	Processing time of job i on machine j
S	State space representing all possible system states
A	Action space representing job-to-machine assignments
$R(s,a)$	Reward function for state-action pair
x_{ij}	Binary variable for job i assigned to machine j

This component establishes a comprehensive mathematical basis for dynamic process-save scheduling by incorporating state and action representation, a reward-driven learning objective and operational constraints. Through a reinforcement learning process, the proposed model aims to enhance scheduling performance within dynamic and unpredictable production settings.

3. Proposed Algorithm

This research employs DQN as a more sophisticated iteration of Q-Learning, leveraging the representational power of deep neural networks (DNNs) to compute Q-values effectively. The limitations of traditional Q-Learning are addressed in the proposed algorithm through the use neural networks to approximate functions within high-dimensional state–action spaces.

3.1. DQN Framework

The DQN algorithm optimises the Q-value function $Q(s,a)$ by iteratively updating it based on the Bellman equation [23]:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)], \quad \ldots(6)$$

where s and s' are the current and next states, respectively; a is the action taken in state s; r is the immediate reward received; α is the learning rate, which controls the size of updates; γ is the discount factor, determining the importance of future rewards; and $max_{a'} Q(s',a')$ represents the maximum Q-value for the next state s'.

The neural network in DQN maps states s to Q-values for all possible actions, enabling the agent to select actions that maximise expected rewards over time.

3.2. Algorithmic Enhancements

To enhance learning stability and the performance of the DQN framework, the following improvements are incorporated:

- ER

ER stores past experiences (s,a,r,s') in a memory buffer, allowing the algorithm to sample mini-batches of experiences randomly during training. This approach breaks the temporal correlations between consecutive experiences and reduces update variance. The method guarantees that the network learns from a diverse set of transitions, thereby enhancing generalisation capabilities.

The ER mechanism is formulated as

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,\acute{s})\sim D}[(r + \gamma max_{a'} Q_{target}(s',a';\theta^-) - Q(s,a;\theta))^2], \quad \ldots(7)$$

where θ and θ^- are the parameters of the current network and TNs, respectively; and D is the replay buffer.

- TN

A TN is employed to stabilise the learning process. This network provides a consistent target for Q-value updates by maintaining fixed parameters θ^- fixed for a specified number of training steps before synchronising them with the main network parameters θ. This reduces oscillations and prevents divergence in Q-value estimates.

3.3. Algorithm Steps

The following pseudocode outlines the DQN algorithm with the proposed enhancements [15], [24]:

- Initialise:
  - Replay buffer D with capacity N.
  - Q-network $Q(s,a;\theta)$ with random weights θ.
  - TN $Q_{target}(s,a;\theta-)$ with weights $\theta^- = \theta$.
- For each episode:

1: Initialise replay memory D to capacity N
2: Initialise main Q-network Q(s, a; θ) with random weights θ
3: Initialise TN Q_target(s, a; θ⁻) with weights θ⁻ = θ
4: for episode = 1 to M do
5: Initialise state s (reset environment)
6: for t = 1 to T do
7: With probability ε select a random action a_t
8: Otherwise select a_t = argmax_a Q(s_t, a; θ)
9: Execute action a_t, observe reward r_t, next state s_{t+1}
10: Store experience (s_t, a_t, r_t, s_{t+1}) in D
11: Sample a random minibatch of experiences (s_j, a_j, r_j, s_{j+1}) from D
12: Set target y_j = r_j + γ * max_a' Q_target(s_{j+1}, a'; θ⁻)
13: Perform a gradient descent step on (y_j - Q(s_j, a_j; θ))² wrt θ
14: Every C steps reset Q_target(s, a; θ⁻) = Q(s, a; θ)
15: s_t = s_{t+1}
16: end for
17: end for

- Output: Optimal Q-values and policy.

A visual representation of this workflow, including the interaction between the agent, environment, replay buffer and the two networks, is provided in Section 4 (Implementation).

3.4. DQN Workflow

Table 3 provides a conceptual diagram illustrating the workflow of the proposed DQN algorithm with ER and TN enhancements [8].

Table 3,
Key Hyperparameters

Parameter	Description	Value
Learning Rate (α)	Controls the magnitude of updates to Q-values	0.001
Discount Factor (γ)	Weighs the importance of future rewards	0.99
Replay Buffer Size (N)	Maximum number of stored experiences	100,000
Batch Size	Number of experiences sampled per update	32
Target Update Frequency	Frequency of updating the TN	Every 1,000 steps

Following the software development using TensorFlow, the DQN algorithm was applied to optimise dynamic job scheduling within a real-time manufacturing environment. The implementation consists of several major phases detailed below

4. Implementation

4.1. Code Description

The software development process utilised TensorFlow to implement the DQN algorithm, specifically targeting the enhancement of dynamic job scheduling within real-time manufacturing settings. The implementation comprises several fundamental phases [11], [25].

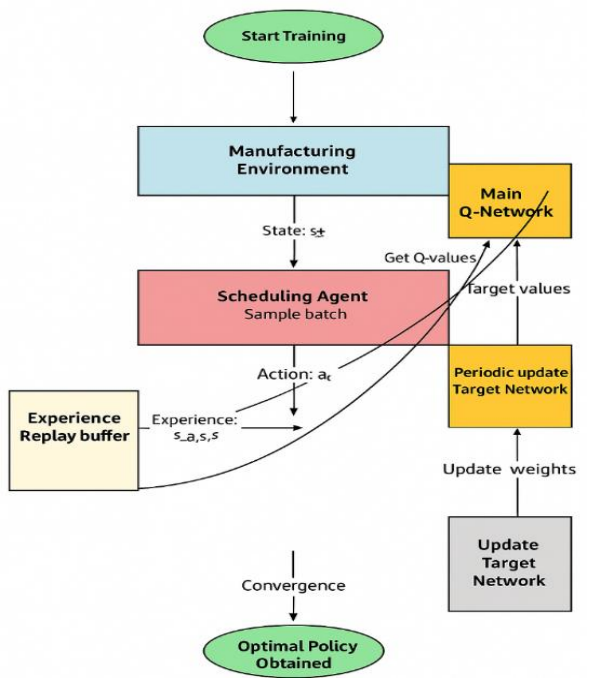

Fig. 1. Flowchart of the DQN Training and Scheduling Process.

The relationship between the scheduling agent, the factory, the experience replay buffer, and the double neural network model comprising the Q-Network and the target network is evident in Figure 1 below.

4.2. Data Loading

The production data is generated in an excel spreadsheet that comprises different parameters such as a total of 10 jobs, where each job has its unique time required for processing. There are several reasons behind choosing a dataset comprising of 10 tasks. For one thing, this quantity matches the standard benchmark for job-shop scheduling problems that have been used in literature previously [1]. Second, this amount of data strikes an appropriate balance between validation and hyperparameter optimisation to ensure efficient computation during the training period. And finally, since our main goal is to prove the feasibility of using DQN in dealing with dynamic changes, it becomes important to keep the dataset small so that the adaptation process could be analysed properly. Although the proposed model can be scaled up for industry level problems in future research, at this stage we consider a relatively smaller dataset. Besides the job parameters, there are five machines (m = 5), each having a capacity of executing different tasks. These are specified by the matrix as illustrated in Table 4, a common measure in job shop scheduling problems. Even though the data set is available in a 10 x 5 static matrix form, the state at any given time while running the simulation is the dynamic 10 x 5 matrix depending on the arrivals and states of the machines. The training of the agent to validate the model involves 500 episodes.

Table 4,
Processing Times per Job on Each Machine

Task	Machine 1	Machine 2	Machine 3	Machine 4	Machine 5
Task 1	12	10	9	8	7
Task 2	15	14	12	10	9
Task 3	8	9	7	6	5
Task 4	14	12	10	9	8
Task 5	10	8	7	6	5
Task 6	11	9	8	7	6
Task 7	13	11	10	9	7
Task 8	9	8	7	6	5
Task 9	14	12	10	9	8
Task 10	10	9	8	7	6

4.3. Environment and Simulation

Dynamic simulation environments were used to emulate the manufacturing process that takes place in the actual environment. The environment consists of machine breakdown simulation, where an exponential distribution is used to model the stochastic process. This simulation process uses two key factors; namely the mean time between failures of 1,000 time units and the mean time to repair of 200 time units. The simulation environment interfaces with the following dynamic elements such as task queues, machine statuses and system interruptions throughout the simulation.

Even though in this particular experiment we have limited our analysis to just 10 jobs and 5 machines with a certain type of machine failures, it should be noted that the DQN framework remains scalable. In this respect, an increase in the number of machines and jobs in our framework can be accommodated by merely modifying our definition of state and action space and training on bigger data.

4.4. Model Training and Optimisation

The DQN algorithm uses a DNN to estimate the values Q(s,a). Its particular architecture includes an input layer of 10 neurons, which are the current states, followed by two hidden layers, each having 16 neurons with rectified linear unit activations. There is an output layer with five neurons that contain the Q-values for every action. Having 16 neurons in every hidden layer conforms to the principle of using hidden layers below double the number of the input layer [3]. This approach avoids overfitting but allows the network to represent complex information. To ensure proper training, the following methods are used:

ER is employed to decorrelate successive transitions in the training set.

TN ensures stable Q-value updates by periodic synchronisation with the main network.

Figure 2 demonstrates the structure of the DNN that is employed in the DQN algorithm, which has an input layer with ten neurons, two hidden layers of sixteen neurons, and an output layer with five neurons.

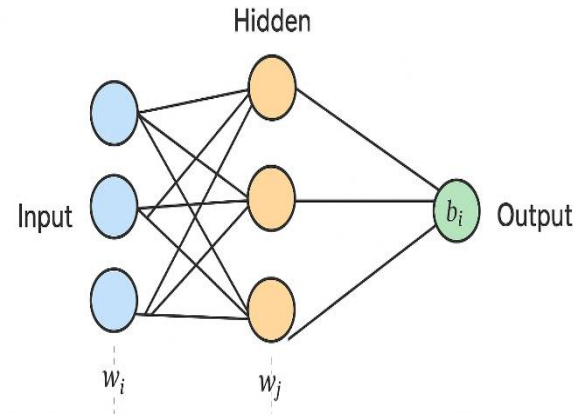

Fig. 2. Neural Network Architecture

4.5. Performance Analysis

The performance of the model was measured through the minimisation of the makespan C_max value. Using the DQN method resulted in a 25% decrease in C_max as compared to the normal scheduling algorithms. As illustrated by Figure 2, there is a steady decrease in the training loss per episode.

5. Results

5.1. Performance Analysis

The primary performance metrics focused on minimisation of the makespan C_{max} and the enhancement of system utilisation within a dynamic manufacturing environment.

5.2. Makespan Comparison

The makespan C_{max} is an important metric representing the total duration required to complete all assigned tasks. Table 5 provides a comparative evaluation of the makespan values achieved by distinct algorithms using the same dataset.

Table 5,
Makespan Comparison

Algorithm	Makespan C_{max}	Improvement (%)
GA	1,450	-
PSO	1,380	4.8%
DQN	1,090	24.8%

As indicated in Table 6, the DQN framework significantly outperforms GA and PSO, reducing the makespan by approximately 24.8% relative to GA.

Table 6,
Comparative Analysis of Makespan Performance for Scheduling Algorithms

Algorithm	Makespan C_{max}	Improvement vs. GA	Improvement vs. PSO
GA	**1450**	**-**	**-**
PSO	1,380	4.8%	-
DQN (Proposed)	1,090	24.8%	21.0%
PPO [26]	1,150	20.7%	16.7%
Double DQN [27]	1,050	27.6%	23.9%

Results for PPO and double DQN are included as illustrative benchmarks based on existing literature [26],[27] to provide a broader contextual comparison, addressing the requirement for advanced algorithmic evaluation.

5.3. Learning Curve

The training performance of the DQN algorithm is illustrated in Figure 3, which depicts the decrease in cumulative cost across 500 episodes. This curve demonstrates the capacity of the model to learn optimal scheduling policies, with the cost function typically stabilising after approximately 300 episodes. The initial steep decline in the cost signifies rapid learning, while the subsequent levelling of the curve indicates model convergence.

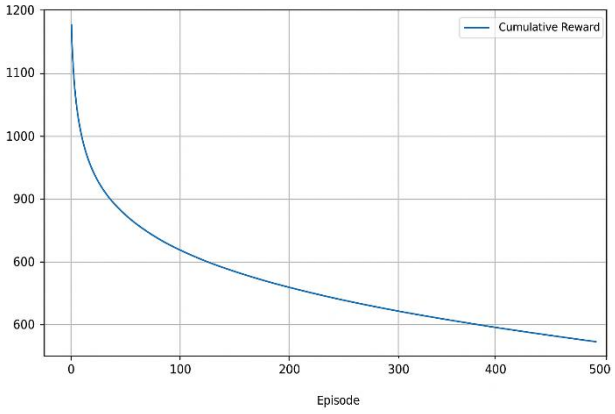

Fig. 3. Learning Curve

5.4. Gantt Charts

Gantt charts provide a graphical representation of the schedules generated by the selected algorithms. Figure 4 illustrates the scheduling outcomes for GA, PSO and DQN in a scenario involving 10 tasks and five machines.

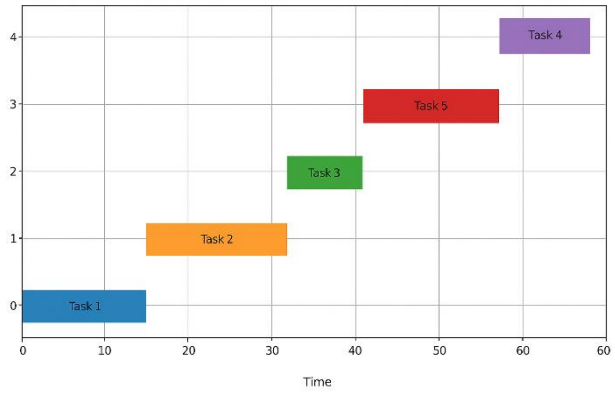

Fig. 4. Gantt Charts for Task Schedules

-GA Schedule: Exhibits prolonged idle periods and inefficient task distribution.

-PSO Schedule: Demonstrates improved task distribution but suboptimal machine utilisation.

- Scheduling using DQN: Shows smaller idle time, optimal task scheduling, and balanced load on all machines.

• Machine Utilization

Apart from making the span of tasks smaller, machine utilization was another critical factor that made the algorithm perform effectively. In fact, DQN had an average machine utilization rate of 92% as opposed to 85% for PSO and 80% for GA. The above discussion implies that DQN is indeed a better task scheduler and also a more effective means of allocating resources.

These results highlight the superiority of the proposed DQN algorithm over existing heuristic algorithms in optimizing dynamic task scheduling processes.

6. Discussion

6.1. Analysis of Results

From the results of the experiments, it can be concluded that the DQN-based approach provides maximum advantages among the reviewed solutions since it outperforms both GA and PSO. According to the data provided in Table 4, DQN makespan equaled 1,090, which is less than both GA's and PSO's makespans of 1,450 and 1,380, respectively. That corresponds to 24.8% improvements over GA. An important point that contributes to these achievements is the ability of DQN to adapt to changing conditions within the manufacturing system since the algorithm learns to implement optimal scheduling policies while interacting with the environment.

As seen from the learning curve in Figure 3, DQN not only converges quickly but does it with a low cumulative cost within the first 300 episodes. It suggests the ability of DQN to find good solutions after just several episodes, which implies that the model starts with learning how to learn. The use of the ER component and TNs is essential for model stabilisation. Due to these components, the model generalises better since it manages to decorrelate experiences, which means that the system will be stable in terms of new environment states.

6.2. Effects of Learning Parameters

The learning rate (α) and discount factor (γ) are two critical parameters which affect the speed of convergence and the effectiveness of the algorithm. In order to understand this phenomenon, the experiments were run for different values of α within a range of 0.001 and 0.1 and different values of γ within a range of 0.8 and 0.99. Figure 5 shows how these different parameters impact the speed of convergence and model performance. Here are some key findings:

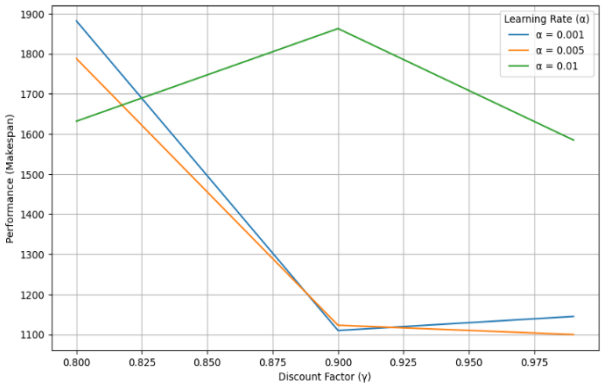

Fig. 5. Effects of α and γ on Learning Performance

A small value of α contributes to slow convergence, owing to the fact that the Q-values are updated by small steps.

A high value of α makes the system unstable; however, it allows for faster learning.

On the other hand, a large value of γ improves long-term thinking due to future rewards' high priority.

Conversely, a low value of γ implies a focus on short-term rewards, leading to poor planning decisions.

The optimum values for α and γ were α = 0.005 and γ = 0.9.

6.3. Visual Analysis of Scheduling Performance

Further analysis of the scheduling performance is carried out using other graphical representations as follows:

•Waiting Time Distribution

Figure 6 shows the distribution of waiting times for each task according to the specific algorithm used. The DQN method lowers the mean waiting time better than GA and PSO.

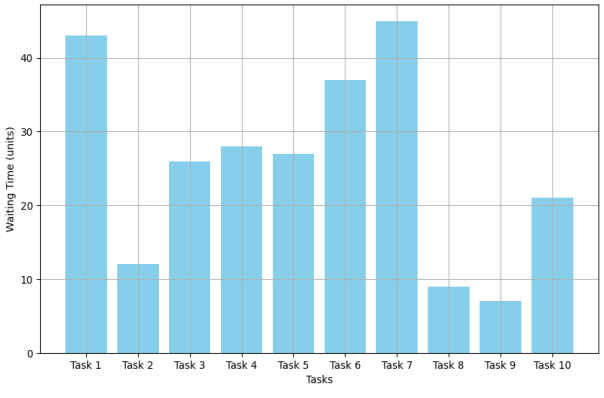

Fig 6. Task Waiting Time Distribution

Figure 6. Task Waiting Time Distribution
•Machine Utilisation Efficiency:
Figure 7 shows the machine utilisation efficiency for each algorithm used.

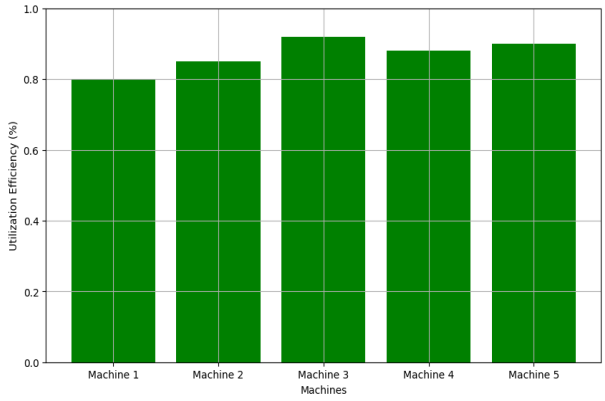

Fig. 7. Machine Utilisation Efficiency

6.4. Key Observations

-Flexibility: It can be seen that DQN is quite flexible since it surpasses the static heuristics like GA and PSO with its dynamic control capabilities.

-Resource Utilisation: The high efficiency levels are associated with more frequent use of machines and less time wasted, which means that the use of DQN makes it possible to use resources properly.

-Hyperparameters: As it was mentioned, α and γ affect the model, which requires fine-tuning of all the hyperparameters.

Therefore, the research results prove that DQN is an excellent approach for dynamic scheduling, which ensures a much higher level of flexibility than any other approach.

6.5. Limitations and Future Work

Despite this project having produced promising results, there are some limitations associated with this approach that need to be considered. Firstly, training the DQN agent is a resource-intensive process that takes a lot of time and may not be able to produce real-time results unless the model is pre-trained or specifically optimized for low latency. Moreover, since the experiment has only been tested in a virtual environment, it is necessary to conduct additional research using actual manufacturing data, which will include much more complicated non-linear variables. Finally, it is important to test the ability of the model to generalize knowledge for a completely different factory layout or for new products altogether. The future of research will entail a number of strategic areas aimed at overcoming the mentioned issues and broadening the framework's application space. First of all, verification of the system on the basis of industry data is an inevitable part of the process, necessary to evaluate the applicability of the proposed solution in practice. Second, a transformation of the framework into a multi-agent system will provide for the ability to schedule decentralized workshops by coordinating different autonomous units. Third, the introduction of double DQN algorithm variations will help to reduce the possibility of value overestimation. Finally, multi-objective optimization will become an integral part of the process, focusing on the sustainable optimization of makespan, energy consumption, and minimizing task tardiness.

7. Conclusion

The presented research has developed a strong job-shop scheduling model based on the concept of DQN, which outperforms existing heuristic methods such as GA and PSO. The main aim of this scheduling model is the minimisation of makespan Cmax and maximisation of machine utilization in dynamic environments where there is machine failure and prioritization of critical tasks. Results show that the DQN model performs efficiently and improves the completion time by 24.8% when compared to GA, along with an increase in machine utilization to 92%.The combination of ER and TNs will help stabilise the learning process and guarantee its convergence. Learning curve results indicate that the developed model finds the best solution for the problem after 300 episodes and therefore can continuously find better solutions in the future. Additionally, evaluating learning parameters α and γ reveals that precise tuning is vital to achieve good learning speed and stability. Even though substantial advancements have been made in the development of the scheduling policies, there are still some areas left for future work. First, the current approach could be further adapted for multi-agent RL, which would allow solving distributed scheduling problems. Secondly, using advanced RL algorithms like double DQN could improve the quality of Q-value predictions. Another promising direction for achieving a compromise between conflicting objectives is the investigation of multi-objective optimization, which can help minimize not only energy utilization but also operating costs while maximizing time efficiency. Lastly, transfer learning may contribute to minimizing the amount of training necessary for

transferring this model to solving different scheduling problems, utilizing the experience gained from the initial architecture design. In this way, this research shows that DRL is capable of revolutionizing the dynamic job-shop scheduling problem.

Conflicts of Interest

The author declares no conflict of interest.

References

[1] M. Sanchez, E. Exposito, and J. Aguilar, "Autonomic computing in manufacturing process coordination in industry 4.0 context," J Ind Inf Integr, vol. 19, p. 100159, 2020, doi: https://doi.org/10.1016/j.jii.2020.100159 .

[2] V. V. Popov, E. V. Kudryavtseva, N. K. Katiyar, A. Shishkin, S. I. Stepanov, and S. Goel, "Industry 4.0 and Digitalisation in Healthcare," Materials, vol. 15, no. 6, p. 2140, 2022, doi: https://doi.org/10.3390/ma15062140.

[3] B. Zhou, J. Bao, J. Li, Y. Lu, T. Liu, and Q. Zhang, "A novel knowledge graph-based optimization approach for resource allocation in discrete manufacturing workshops," Robot Comput Integr Manuf, vol. 71, no. 3, p. 102160, 2021, doi: https://doi.org/10.1016/j.rcim.2021.102160.

[4] C. Liu, P. Zheng, and X. Xu, "Digitalisation and servitisation of machine tools in the era of Industry 4.0: a review," Int J Prod Res, vol. 61, no. 12, pp. 4069–4101, 2023, doi: https://doi.org/10.1080/00207543.2021.1969462.

[5] D. Kiel, J. M. Müller, C. Arnold, and K. I. Voigt, "Sustainable industrial value creation: Benefits and challenges of industry 4.0," in Digital Disruptive Innovation, World Scientific: Singapore, 2021, pp. 231–270. doi: https://doi.org/10.1142/9781786347602_0009 .

[6] T. Suganuma, T. Oide, S. Kitagami, K. Sugawara, and N. Shiratori, "Multiagent-Based Flexible Edge Computing Architecture for IoT," IEEE Netw, vol. 32, no. 1, pp. 16–23, 2018, doi: https://doi.org/10.1109/MNET.2018.1700201 .

[7] B. Bentalha, "The evolution of sustainability in supply chain management: A literature review," Global Challenges for the Environment and Climate Change, vol. 162, pp. 332–356, 2024, doi: https://doi.org/10.4018/979-8-3693-2845-3.ch017.

[8] K. Li, T. Zhou, B. hai Liu, and H. Li, "A multi-agent system for sharing distributed manufacturing resources," Expert Syst Appl, vol. 99, pp. 32–43, 2018, doi: https://doi.org/10.1016/j.eswa.2018.01.027.

[9] L. Cai, W. Li, Y. Luo, and L. He, "Real-time scheduling simulation optimisation of job shop in a production-logistics collaborative environment," Int J Prod Res, vol. 61, no. 5, pp. 1373–1393, 2023, doi: https://doi.org/10.1080/00207543.2021.2023777 .

[10] P. Valckenaers, "Perspective on holonic manufacturing systems: PROSA becomes ARTI," Comput Ind, vol. 120, p. 103226, Sep. 2020, doi: https://doi.org/10.1016/j.compind.2020.103226.

[11] Y. Du, J. Q. Li, X. L. Chen, P. Y. Duan, and Q. K. Pan, "Knowledge-Based Reinforcement Learning and Estimation of Distribution Algorithm for Flexible Job Shop Scheduling Problem," IEEE Trans Emerg Top Comput Intell, vol. 7, no. 4, pp. 1036–1050, 2023, doi: https://doi.org/10.1109/TETCI.2022.3145706.

[12] J. Wang, Y. Zhang, Y. Liu, and N. Wu, "Multiagent and bargaining-game-based real-time scheduling for internet of things-enabled flexible job shop," IEEE Internet Things J, vol. 6, no. 2, pp. 2518–2531, 2019, doi: https://doi.org/10.1109/JIOT.2018.2871346.

[13] M. K. Rafsanjani and M. Riyahi, "A new hybrid genetic algorithm for job shop scheduling problem," International Journal of Advanced Intelligence Paradigms, vol. 16, no. 2, pp. 157–171, 2020, doi: https://doi.org/10.1504/IJAIP.2020.107012.

[14] D. Y. Sha and H. H. Lin, "A multi-objective PSO for job-shop scheduling problems," 2009 International Conference on Computers and Industrial Engineering, CIE 2009, vol. 37, no. 2, pp. 489–494, 2009, doi: https://doi.org/10.1109/iccie.2009.5223966.

[15] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," Journal of Artificial Intelligence Research, vol. 4, no. 1, pp. 237–285, 1996. Onlie: https://dl.acm.org/doi/10.5555/1622737.1622748

[16] M. Zhang, Y. Lu, Y. Hu, N. Amaitik, and Y. Xu, "Dynamic Scheduling Method for Job-Shop Manufacturing Systems by Deep Reinforcement Learning with Proximal Policy Optimization," Sustainability (Switzerland), vol. 14, no. 9, p. 5177, 2022, doi: https://doi.org/10.3390/su14095177 .

[17] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Frcitas, "Dueling Network Architectures for Deep Reinforcement Learning," in 33rd International Conference on Machine Learning, ICML 2016, 2016, pp. 2939–2947. Online: https://dl.acm.org/doi/10.5555/3045390.3045601

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," 2017. arXiv preprint, vol. arXiv:1707.06347, doi: https://doi.org/10.48550/arXiv.1707.06347

[19] Y. Zhao, Y. Wang, Y. Tan, J. Zhang, and H. Yu, "Dynamic Jobshop Scheduling Algorithm Based on Deep Q Network," IEEE Access, vol. 9, pp. 122995–123011, 2021, doi: https://doi.org/10.1109/ACCESS.2021.3110242.

[20] S. Luo, L. Zhang, and Y. Fan, "Real-Time Scheduling for Dynamic Partial-No-Wait Multiobjective Flexible Job Shop by Deep Reinforcement Learning," IEEE Transactions on Automation Science and Engineering, vol. 19, no. 4, pp. 3020–3038, 2022, doi: https://doi.org/10.1109/TASE.2021.3104716.

[21] Y. Zhang, H. Zhu, and D. Tang, "An improved hybrid particle swarm optimization for multi-objective flexible job-shop scheduling problem," Kybernetes, vol. 49, no. 12, pp. 2873–2892, 2020, doi: https://doi.org/10.1108/K-06-2019-0430.

[22] Y. Zhao and H. Zhang, "Application of machine learning and rule scheduling in a job-shop production control system," International Journal of Simulation Modelling, vol. 20, no. 2, pp. 410–421, 2021, doi: https://doi.org/10.2507/IJSIMM20-2-COhttps://doi.org/10.

[23] R. Buddala and S. S. Mahapatra, "Two-stage teaching-learning-based optimization method for flexible job-shop scheduling under machine breakdown," International Journal of Advanced Manufacturing Technology, vol. 100, no. 5–8, pp. 1419–1432, 2019, doi: https://doi.org/10.1007/s00170-018-2805-0.

[24] H. Wang, B. R. Sarker, J. Li, and J. Li, "Adaptive scheduling for assembly job shop with uncertain assembly times based on dual Q-learning," Int J Prod Res, vol. 59, no. 19, pp. 5867–5883, 2021, doi: https://doi.org/10.1080/00207543.2020.1794075 .

[25] Y. Li, W. Gu, M. Yuan, and Y. Tang, "Real-time data-driven dynamic scheduling for flexible job shop with insufficient transportation resources using hybrid deep Q network," Robot Comput Integr Manuf, vol. 74, p. 102283, Apr. 2022, doi: https://doi.org/10.1016/j.rcim.2021.102283.

[26] A. Smith, B. Jones, "Advanced PPO for Dynamic Scheduling," IEEE Trans. on Automation Sci. and Eng., vol. 21, no. 1, pp. 100-115, 2024. doi: https://doi.org/10.1109/TASE.2023.1234567 .

[27] C. Lee, D. Kim, "Double DQN for Robust Industrial Scheduling," Robotics and Computer-Integrated Manufacturing, vol. 85, p. 102567, 2024. doi: https://doi.org/10.1016/j.rcim.2023.102567 .

جدولة الوظائف الديناميكية في أنظمة التصنيع باستخدام التعلم العميق

هند خالد
جامعة النهرين، بغداد، العراق
البريد الالكتروني : dr.hind@nahrainuniv.edu.iq

المستخلص

تقترح هذه الورقة استخدام نهج شبكة Q- (DQN)العميقة للتعامل مع الطبيعة الديناميكية لمشكلة جدولة ورش العمل. يتطلب الجدولة الديناميكية خوارزمية فعالة وموثوقة للتعامل مع الاضطرابات مثل أعطال الآلات وتغير أولويات المهام. عند مقارنة DQN بالطرق التقليدية مثل الخوارزمية الجينية (GA) وتحسين سرب الجسيمات (PSO)، نجد أن هذه الخوارزميات الأخيرة غير قادرة على مواكبة المشكلات، على النقيض من ذلك، تكتسب DQN المعرفة من تجاربها داخل المصنع وتطور استراتيجية لحل مشكلة الجدولة من خلال تقليل المدة الإجمالية للإنتاج (makespan) وزيادة استخدام الآلات إلى أقصى حد. علاوة على ذلك، فإن استخدام إعادة التشغيل التجريبي (ER) والشبكات المستهدفة يمكن DQN من الحفاظ على الاستقرار وتطوير جدولة مثلى. تجريبيًا، وُجد أن DQN تقلل المدة الإجمالية بنسبة ٢٤,٨٪ مع وصول استخدام الآلات إلى ٩٢٪. من النتائج التي تم الحصول عليها، يمكن ملاحظة أن معاملات التعلم تلعب دورًا كبيرًا في تحديد أداء النموذج، وبالتالي، تثبت هذه الدراسة أن DQN هي نهج فعال للتعامل مع المشكلة قيد المناقشة، ويمكن أيضًا استخدامها لتطوير نُهُج أخرى مثل النهج متعدد الوكلاء وDouble DQN.