Title Page

A Novel Carbon Reduction Engineering Method-based Deep Q-learning Algorithm for Energy-efficient Scheduling on a Single Batch-processing Machine in Semiconductor Manufacturing

Min Kong [a,b], Weizhong Wang[a], Muhammet Deveci [c,d,*], Yajing Zhang [a], Xuzhong Wu [a,*], D'Maris Coffman [c]

[a] *School of Economics and Management, Anhui Normal University, Wuhu 241000, PR. China*

[b] *School of Management, Hefei University of Technology, Hefei 230009, PR. China*

[c] *The Bartlett School of Sustainable Construction, University College London, 1-19 Torrington Place, London, WC1E 7HB, UK*

[d] *Department of Industrial Engineering, Turkish Naval Academy, National Defense University, Tuzla, Istanbul 34940, Turkey*

[*] *Corresponding authors:*

muhammetdeveci@gmail.com (Muhammet Deveci); xmuwxz@163.com (Wu Xuzhong)

Carbon Reduction Engineering Method-based Deep Q-learning Algorithm for Energy-efficient Scheduling on a Single Batch-processing Machine in Semiconductor Manufacturing

Abstract

The semiconductor industry is a resource-intensive sector that heavily relies on energy, water, chemicals, and raw materials. Within the semiconductor manufacturing process, the diffusion furnace, ion implantation machine, and plasma etching machine exhibit high energy demands or operate at extremely high temperatures, resulting in significant electricity consumption, which is usually carbon-intensive. To address energy conservation concerns, the industry adopts batch production technology, which allows for the simultaneous processing of multiple products. The energy-efficient parallel batch scheduling problem arises from the need to optimize product grouping and sequencing. In contrast to existing heuristics, meta-heuristics, and exact algorithms, this paper introduces the Deep Q-Network (DQN) algorithm as a novel approach to address the proposed problem. The DQN algorithm is built upon the agent's systematic learning of scheduling rules, thereby enabling it to offer guidance for online decision-making regarding the grouping and sequencing of products. The efficacy of the algorithm is substantiated through extensive computational experiments.

Keywords: Semiconductor manufacturing, Deep Reinforcement Learning, Parallel Batch Scheduling, Less is More, Carbon reduction engineering

1. Introduction

The latest International Energy and Climate Conference has shed light on the carbon neutral and carbon peak timelines for countries worldwide. Achieving these goals requires extensive efforts across various sectors, especially in industries that consume large amounts of energy, such as the semiconductor industry. Nowadays, energy efficiency and emission reduction are essential goals for semiconductor manufacturing enterprises. According to the 2021 Sustainability Report of Taiwan Semiconductor Manufacturing Company, the total consumption of non-renewable energy of TSMC in 2021 is 16410 GWh, exceeding the total consumption of over 2.7 million residents in Taipei City and 7.2% of Taiwan's total energy usage (TSMC 2021). With the semiconductor manufacturing industry's high sensitivity to electricity costs, rising prices can lead to exponential growth in product costs (Derinkuyu et al. 2020; Ho, Hnaien, and Dugardin 2022).

Burn-in testing is essential to semiconductor manufacturing to ensure that the integrated circuits (ICs) can withstand the stresses of regular use over time. During burn-in testing, the ICs are first load in burn-in boards, and then subjected to elevated temperature and electrical stress to accelerate any potential failures during regular operation. This allows manufacturers to identify and address issues before the ICs are shipped to customers. The temperature used for burn-in testing is typically $125\ °C$, considered a 'worst-case' operating condition for the ICs. Burn-in ovens, which can process multiple burn-in boards simultaneously, are widely used in semiconductor burn-in operations. Numerous machines run continuously during semiconductor burn-in operations' batch production and testing phase, consuming stable and high-quality electricity (Huang, Pan, and Guan 2021). Fig. 1 represents the total energy consumption of TSMC over the past five years. In the production process, both renewable and non-renewable energy sources have been gradually increasing in usage.

The utilization rate of renewable energy in actual production is significantly low. Carbon dioxide emissions from energy consumption account for 83% of the company's total greenhouse gas emissions. This situation is mainly attributed to the indirect emissions produced from the procurement of external electricity (TSMC 2021). According to relevant research data, 60% of the global semiconductor industry's carbon emissions in 2021 are caused by manufacturing energy consumption. To effectively mitigate carbon emissions in the semiconductor industry, the main approaches currently being pursued include greenhouse gas substitution, advanced emission reduction methods, process optimization, and the implementation of remote plasma cleaning systems (Pelcat 2023; Zhu et al. 2023; Chien, Peng, and Yu 2016; Liang, Tan, and Li 2023). However, it is important to recognize that the development of advanced production processes and emission reduction systems often entails a lengthy development cycle and significant capital investment. Therefore, enhancing energy management and improving energy efficiency has become one of the key choices for semiconductor enterprises to reduce carbon emissions.

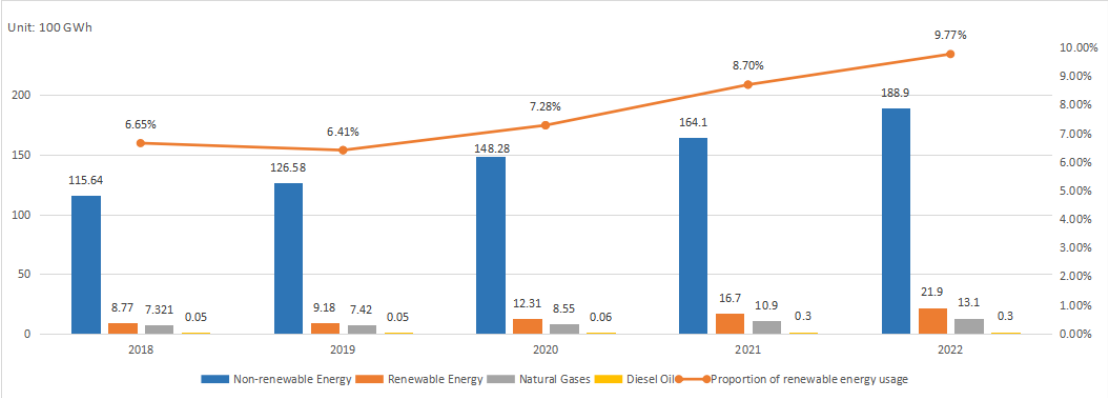


Fig. 1. Total energy consumption of TSMC

The problem of energy efficiency and consumption reduction has received increasing research attention in recent years. The main approaches to achieving energy efficiency and consumption reduction in industry and manufacturing are structural energy efficiency, technical energy efficiency, and management energy efficiency (Gahm et al. 2016; Gao et al. 2020). Structural energy efficiency involves regulating high-energy-consumption industries through macro-level industrial structure optimization, optimizing renewable and non-renewable energy consumption structures, and increasing the proportion of renewable energy use. Technical energy efficiency involves improving energy utilization by developing equipment and upgrading energy-saving processes, such as industrial boiler energy-saving renovation technology. Finally, management energy efficiency involves optimizing the production system's management to enable efficient energy use without upgrading the hardware environment, such as technology, equipment, and processes. Since managed energy efficiency does not require improving and researching the manufacturing hardware environment, it has apparent advantages regarding implementation cost and cycle time. Consequently, research on energy-saving management mechanisms for related manufacturing processes has received significant attention from the management community (Jiang and Powell 2015; Heydar, Mardaneh, and Loxton 2022; Park and Ham 2022).

Overall, the semiconductor industry is known for its resource-intensive nature, with significant energy consumption being a major concern. Therefore, developing an energy-efficient scheduling approach for semiconductor manufacturing is crucial. Additionally, the batch production technology used in the semiconductor industry allows for the simultaneous processing of multiple products,

leading to the need for effective grouping and sequencing strategies. Traditional heuristics, meta-heuristics, and exact algorithms have been widely used to address similar problems. However, their effectiveness may be limited due to the complexity and variability of the batch-processing environment. Hence, the adoption of deep reinforcement learning algorithms, such as the Deep Q-Network (DQN) algorithm, which has shown promising results in solving scheduling problems, is justified. By utilizing the systematic learning capabilities of the DQN algorithm, we aim to optimize the energy efficiency in semiconductor manufacturing, while ensuring efficient grouping and sequencing of jobs.

The paper is structured into six sections, starting with an introduction and then a review of related works on energy-efficient scheduling problem for a single batch-processing machine with arbitrary job sizes (E-SBPM-AJS) problems and the use of deep reinforcement learning for scheduling problems in Section 2. Section 3 briefly introduces the studied E-SBPM-AJS problem, with Section 4 introducing the deep reinforcement learning method used in the paper. Section 5 discusses the comparative experiments, analyzes the experimental results, and finally, Section 6 provides concluding remarks.

2. Literature review

In this section, we provide a comprehensive review of the E-SBPM-AJS model and the application of deep reinforcement learning in scheduling problems.

2.1 Related work of E-SBPM-AJS model

Batch production is a common method for burn-in operations in semiconductor manufacturing (Xu, Chen, and Li 2012; Jia, Li, and Leung 2017; Cigolini et al. 2002). It also often appears in other production scenarios, including heat treatment operations in metalworking (Cheng et al. 2013), 3D printing operations in additive manufacturing (Zhang et al. 2020), parts hardening synthesis operations in aircraft manufacturing (Van etal. 1997), and parts production operations in shoe manufacturing (Fanti 1996). Serial batch (s-batch) and parallel batch (p-batch) production are the two fundamental types of batch manufacturing processes outlined in earlier literature (Fowler and Mnch 2021). This research aims to address the job shop scheduling problem under the parallel batch production model, whereby batches of multiple jobs can be processed simultaneously on a single batch processor. All jobs within a batch must enter the processor simultaneously, after which the batch processor cannot release them until all jobs have been completed, with no ability to interrupt the process.

Previous research has proposed numerous algorithms to tackle the scheduling problem on a single batch-processing machine with arbitrary job sizes (SBPM-AJS). Exact and heuristic algorithms were the first methods to tackle this problem. Uzsoy (1994) introduced two heuristic algorithms for the SBPM-AJS problem, one for minimizing total completion time and another for reducing makespan. Several branch-and-bound (B&B) techniques were also developed (Azizoglu and Webster 2000; Dupont and Dhaenens-Flipo 2002) to solve the SBPM-AJS problem and to improve the B&B algorithm's efficiency. Researchers suggested integrating the column generation technique into the B&B algorithm (Parsa, Karimi and Kashan 2010). Heuristic algorithms have also been applied to solve the SBPM-AJS problem. Jolai Ghazvini and Dupont (1998) proposed the DYNA heuristic algorithm to minimize mean flow time. Chang and Wang (2004) proposed a three-phase heuristic algorithm to reduce the total completion time. Meta-heuristic algorithms have become increasingly popular for solving the SBPM-AJS problem. Some of the commonly used meta-heuristic algorithms include Genetic Algorithm (GA) (Damodaran, Manjeshwar, and Srihari 2006;

Kashan, Karimi, and Jolai 2006a; Kashan, Karimi, and Jolai 2006b; Chou 2007), Ant Colony Optimization (ACO) (Jia and Leung 2015; Parsa, Karimi, and Husseini 2016), Artificial Bee Colony (ABC) (Al-Salamah 2015), and Simulated Annealing (SA) (Melouk, Damodaran, and Chang 2004). The choice of algorithm depends on the problem size, complexity, and computational resources available. The algorithms used in the above papers are summarized in Table 1.

Table 1. Algorithms for the SBPM-AJS problems.

References	Objectives			Algorithms				Notes
	C_{max}	$\sum C_i$	Other	Exact	Heuristic	Meta-Heuristic	Other	
Uzsoy (1994)	☑	☑		☑	☑			B&B
Azizoglu and Webster (2000)			$\sum w_i C_i$	☑				B&B
Dupont and Dhaenens-Flipo (2002)	☑			☑				B&B
Parsa, Karimi and Kashan (2010)	☑			☑				B&P
Ghazvini and Dupont (1998)		☑			☑			DYNA
Chang and Wang (2004)		☑			☑			Heuristic
Damodaran, Manjeshwar, and Srihari (2006)	☑					☑		GA
Kashan, Karimi, and Jolai (2006b)	☑					☑		HGA
Kashan, Karimi, and Jolai (2006a)	☑					☑		HGA
Chou (2007)	☑					☑		GA+ DP
Jia and Leung (2015)	☑					☑		ACO
Parsa, Karimi, and Husseini (2016)		☑				☑		ACO
Al-Salamah (2015)	☑					☑		ABC
Melouk, Damodaran, and Chang (2004)	☑					☑		SA

To render the basic problem model more representative of real-world conditions, researchers have conducted in-depth investigations into the SBPM-AJS problem, integrating realistic production constraints and scenarios such as job families (Kempf, Uzsoy, and Wang 1998; Koh et al. 2005; Rezaeian and Zarook 2018; Alizadeh and Kashan 2019), fuzzy processing times (Kempf, Uzsoy, and Wang 1998), release times/dynamic job arrivals (Zhou et al. 2014; Zhou et al. 2018; Zhou et al. 2021), deteriorating jobs (Kong et al. 2020; Jang et al. 2022), due dates (Zhang, Wu, and Yang 2021; Li et al. 2015), and two-agent scheduling (Tan et al. 2011; Wang et al. 2016). However, the scheduling model with additional constraints and restrictions is more complex than the basic SBPM-AJS model, posing challenges for heuristic and exact algorithms. Consequently, meta-heuristic algorithms are predominantly employed to obtain near-optimal solutions for these problems. We summarize these works in Table 2.

Table 2. Algorithms for the SBPM-AJS problems with different features.

References	Objectives			Features						Algorithms
	C_{max}	$\sum C_i$	Other	Job family	Fuzzy Environment	Release times	Deteriorating jobs	Due date	Two-agent	
Kempf, Uzsoy, and Wang (1998)	☑	☑		☑						Heuristic
Koh et al. (2005)	☑	☑	$\sum w_i C_i$	☑						HGA
Rezaeian and Zarook (2018)	☑		L_{max}	☑		☑		☑		BOGA
Alizadeh and Kashan (2019)	☑			☑						LCA&OIO
Cheng, Li, and Chen (2010)	☑				☑					ACO
Zhou et al. (2014)	☑					☑				Heuristic
Zhou et al. (2018)			L_{max}			☑		☑		PSO
Zhou et al. (2021)	☑					☑		☑		DE
Kong et al. (2020)			$C_{max} + w$				☑			H-DP
Jang et al. (2022)	☑			☑			☑			ACO
Zhang, Wu, and Yang (2021)			E/T					☑		GA
Li et al. (2015)			E/T					☑		GA
Tan et al. (2011)	☑								☑	ACO
Wang et al. (2016)	☑								☑	Heuristic

In recent years, as global environmental and energy problems have grown in urgency, carbon reduction engineering has become a significant research subject. The energy-efficient scheduling method has been paid much attention because of its low cost and little influence on production. As for the batch scheduling problem, researchers have shifted their attention from optimizing time-related objectives to optimizing energy efficiency. Liu and Huang (2014) constructed two multi-objective batch scheduling problems where carbon footprint and peak power were used as energy-saving objectives. A genetic algorithm II (NSGA-II) method based on non dominated sorting was proposed to solve this problem. Zeng et al. (2018) considered a multi-objective flexible flow shop optimization problem with batch machines, and the total power consumption is one of the primary targets for scheduling. A hybrid algorithm combining Tabu search with NSGA-II is presented to solve this problem.

At present, there are few studies of the E-SBPM-AJS problem. Wu, Cheng, and Chu (2021) proposed a series of heuristic algorithms to solve the multi-objective E-SBPM-AJS problem with Time of Use (TOU) electricity tariffs by transforming the original problem into multiple knapsack problem to obtain Pareto solutions about total energy consumption and makespan. Considering the impact of machine power and TOU electricity tariffs on energy consumption costs, Wang et al. (2016) studied a bi-objective E-SBPM-AJS model that minimizes makespan and total electricity consumption costs proposed two types of heuristic algorithms to obtain the Pareto fronts of the problem. Zhang et al. (2017) established an E-SBPM-AJS model with a TOU electricity price strategy and speed scaling machine mechanism, where machine processing speed is positively correlated with machine power consumption (Fang et al. 2016). Zhou et al. (2020) proposed a multi-objective E-SBPM-AJS model with TOU electricity tariffs and job release times.

2.2 Application of deep reinforcement learning in a scheduling problem

Given the excellent performance of reinforcement learning and deep reinforcement learning methods in solving complex dynamic decision-making problems, scholars have considered using them to solve complex production scheduling problems. The Q-learning algorithm is a straightforward method for an agent to learn optimal behavior within a controlled Markov domain (Zhou, Jin, and Gu 2020). Wang and Usher (2005) reshaped the traditional single-machine scheduling process into a Markov decision process. Based on the current state, the Q learning agent must select the appropriate scheduling rule from three potential options to determine the next job to process. Zhao et al. (2020) introduced a collaborative water wave optimization algorithm (CWWO) to tackle a variant of the flow shop scheduling problem. During the CWWO propagation operation, they propose a Q-learning algorithm with variable neighborhood search to ascertain the subsequent wave's position, length, and height. Like the dispatching rule selection proposed by Wang and Usher (2005), Zhang et al. (2012) developed an online R-learning algorithm to solve the parallel machine scheduling problem with minimizing mean weighted tardiness. The R-learning algorithm is an average-reward reinforcement learning algorithm (Schwartz 1993) that selects a job to be processed at each decision time step. Zhang et al. (2013) applied the online algorithm to address the flow shop scheduling problem. Several dispatching rules related to the flow shop scheduling problem, such as SPT, LPT, and FCFS, are designated actions.

Reinforcement learning is generally constrained to small action spaces, sample spaces, and discrete situations. However, more complex optimization problems resembling actual circumstances usually involve large state spaces and continuous action spaces. As a result, a deep reinforcement learning (DRL) algorithm that combines deep learning and reinforcement learning has been

proposed. In the DRL algorithm, the state and action serve as input values for the neural network, which enhances its accuracy through continuous training. The trained neural network model can then guide the reinforcement learning iterative process. However, research on applying deep reinforcement learning algorithms to solve job-shop scheduling problems is scarce. We summarize the current work utilizing deep Q network (DQN), a crucial DRL method, for addressing scheduling problems. For the dynamic scheduling problem in flexible manufacturing systems, Hu et al. (2020) proposed a Petri-net convolution layer based on graph convolutional networks and applied the DQN algorithm with prioritized experience replay, which combines reinforcement learning with deep neural networks. Waschneck et al. (2018) employed the DQN method for a flexible job shop scheduling problem with complex constraints. Palombarini and Martínez (2019) solved the rescheduling problem in a workshop production by constructing a deep Q network. Luo (2020) applied the Double DQN method to address a job shop scheduling problem with new job insertion. The Double DQN algorithm is designed to resolve the overestimation of the Q-value in the DQN algorithm. The above works are summarized in Table 3.

Table 3. Related works on DQN algorithms for scheduling problems.

References	Problems	Algorithms
Wang and Usher (2005)	$J\|r_j, d_j\|\sum U_j$	Q-Learning
Zhao et al. (2020)	$F\|r_j, s_j, d_j\|C_{max}$	Q-Learning
Zhang et al. (2012)	$R\|r_j, d_j\|\frac{1}{n}\sum \omega_j \pi_j$	R-Learning
Zhang et al. (2013)	$F\|r_j, s_j, d_j\|C_{max}$	R-Learning
Hu et al. (2020)	$F\|r_j, s_j\|C_{max}$	DQN
Waschneck et al. (2018)	$J\|d_j, r_j\|T_{max}$	DQN
Palombarini and Martínez (2019)	$J\|r_j, d_j, s_j, prec\|\sum T_j$	DQN
Luo (2020)	$F\|r_j, d_j\|T_{max}$	Double DQN

Inspired by deep reinforcement learning in other combinatorial optimization problems, this study develops a DQN-based algorithm to handle the E-SBPM-AJS problem. The main contributions are summarized below.

(1) We have reconstructed the E-SBPM-AJS model based on the Markov decision process. First, the production environment's state is determined according to the batch processing time and the sum of the batch's job sizes. Then, following classical heuristic rules, ten scheduling actions are designed, and the reward-setting method is explained under each action.

(2) The less is more strategy used to reduce the number of actions to accelerate the DRL algorithm convergence. Using Taguchi's experimental design, 12 orthogonal groups of experiments were designed, and four actions were selected from ten.

3. Problem Formulation

In this section, a MIP model is presented for the E-SBPM-AJS problem. Then, we reconstruct the scheduling problem with the Markov decision process.

3.1 Problem statement

In the burn-in operations of semiconductor manufacturing, the burn-in boards loaded by ICs are tested in the burn-in oven by batches to shorten testing cycle and reduce energy consumption. The burn-in testing time of a batch of boards is determined by the longest burn-in testing time of the

board in the batch. The E-SBPM-AJS problem aims to minimize the total energy consumption of all completed boards while ensuring that the capacity limit of burn-in oven is not exceeded. This requires finding the optimal arrangement of burn-in boards into batches, considering their burn-in testing times and sizes. The notations used for the E-SBPM-AJS problem are as follows.

Notations	**Definitions**
n	Number of burn-in boards
K	Number of batches for burn-in boards
j	Index of burn-in board
k	Index of burn-in board batch
p_j	Burn-in testing time of the jth board, $j = 1, \cdots, n$
s_j	Size of the jth board, $j = 1, \cdots, n$
P_k	Burn-in testing time of the kth batch, $k = 1, \cdots, K$
L	Capacity of the burn-in oven
E	Level of energy consumption per unit of time
Decision variables	**Definitions**
C_k	Completion time of the kth batch, $k = 1, \cdots, K$
TEC	Total Energy Consumption for testing burn-in boards
x_{jk}	1 if burn-in board j assigned to batch k, otherwise 0, $j = 1, \cdots, n$, $k = 1, \cdots, K$

As mentioned above, the E-SBPM-AJS problem involves optimizing the processing schedule for a batching machine to minimize energy consumption while adhering to capacity limits. Hence, the MIP model is given as follows.

$$\text{Minimize } TEC \tag{1}$$

Subject to:

$$\sum_{j=1}^{n} x_{jk} = 1 \qquad k = 1, \cdots, K \tag{2}$$

$$\sum_{k=1}^{K} x_{jk} = 1 \qquad j = 1, \cdots, n \tag{3}$$

$$\sum_{j=1}^{n} s_j x_{jk} \leq L \qquad k = 1, \cdots, K \tag{4}$$

$$P_k \geq p_j x_{jk} \qquad j = 1, \cdots, n, k = 1, \cdots, K \tag{5}$$

$$C_k \geq P_k + C_{k-1} \qquad k = 2, \cdots, K \tag{6}$$

$$TEC \geq E * C_k \qquad k = 1, \cdots, K \tag{7}$$

$$x_{j,k} \in (0, 1) \qquad j = 1, \cdots, n, k = 1, \cdots, K \tag{8}$$

Constraint 2 and 3 guarantee that each burn-in board is tested exactly once. Limiting the total size of burn-in boards tested in a batch, constraint 4 ensures that the burn-in oven's capacity is not exceeded. Constraint 5 defines the burn-in testing time of each batch, while constraint 6 calculates the completion time of each burn-in board batch. The total energy consumption for testing all burn-in boards is calculated according to constraint 7. Lastly, constraint 8 specifies the range of decision variables.

Before developing the DRL algorithm, it is essential to formulate the scheduling process as a Markov Decision Process (MDP). This involves considering state features, actions, and the reward

function. We must determine which burn-in board should be selected at each decision point and which batch should accommodate the selected burn-in board. Once the last burn-in board finishes testing, the entire schedule is generated. In the following sections, we will introduce the state, action, and reward settings.

3.2 State features

Here, the production environment is used to represent the information on testing time and total size of each burn-in board batch. The testing time and size of the burn-in boards affect the number of batches, which is not fixed in the optimal scheduling scheme. The algorithm's efficiency will be negatively affected if the number of batches in the state exceeds the optimal number. Hence, the testing time and total size of batches are recorded in the production environment using $2 \times n$ elements. Although some elements might record empty batches, this ensures the possibility of discovering the optimal scheduling scheme. Let $S_j(t)$ and $P_j(t)$ be the total size and the testing time of the jth batch at decision point t, then the production environment at decision point t could be denoted as $X_t = \{P_1(t), \cdots, P_j(t), \cdots, P_n(t), S_1(t), \cdots, S_j(t), \cdots, S_n(t)\}$. Fig. 2 shows the production environment configurations.

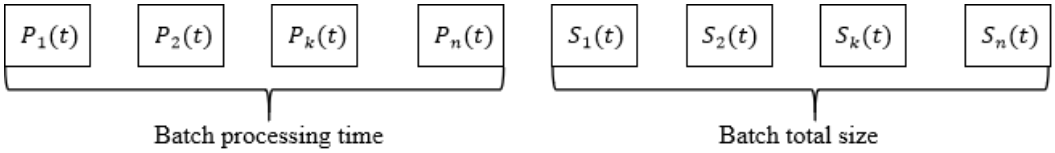


Fig. 2. The production environment configurations

Zhang et al. (2013) provide several guidelines for creating state features. First, they recommend using normalized state characteristics. Second, they emphasize the inclusion of numerical indicators to represent the magnitude of state characteristics. Third, they stress the importance of easily calculable state features. By adhering to these guidelines, we developed the state features for decision point t.

State feature 1 ($X_t(1)$). The maximum value of the ratio of non-empty batch testing time to maximum testing time, which can be written as follows:

$$X_t(1) = \max\{P_1(t), \cdots, P_j(t), \cdots, P_n(t)\} / max(p_j) \quad (9)$$

State feature 2 ($X_t(2)$). The minimum value of the ratio of non-empty batch testing time to maximum testing time, which can be written as follows:

$$X_t(2) = \min\{P_1(t), \cdots, P_j(t), \cdots, P_n(t) | P_j(t) > 0\} / max(p_j) \quad (10)$$

State feature 3 ($X_t(3)$). The mean value of the ratio of non-empty batch testing time to maximum testing time, which can be written as follows:

$$X_t(3) = \text{mean}\{P_1(t), \cdots, P_j(t), \cdots, P_n(t)\} / max(p_j) \quad (11)$$

State feature 4 ($X_t(4)$). The maximum value of the ratio of total burn-in board size to capacity in a non-empty batch, as expressed in the following form:

$$X_t(4) = \max\{S_1(t), \cdots, S_j(t), \cdots, S_n(t)\} / L \quad (12)$$

State feature 5 ($X_t(5)$). The minimum value of the ratio of total burn-in board size to capacity in a non-empty batch, as expressed in the following form:

$$X_t(5) = \min\{S_1(t), \cdots, S_j(t), \cdots, S_n(t) | S_j(t) > 0\} / L \quad (13)$$

State feature 6 ($X_t(6)$). The mean value of the ratio of total burn-in board size to capacity in a non-empty batch, as expressed in the following form:

$$X_t(6) = \text{mean}\{S_1(t), \cdots, S_j(t), \cdots, S_n(t)\} / L \quad (14)$$

3.3 Actions

Considering the capacity constraint of the burn-in oven, we must allocate burn-in boards to distinct batches. Since the objective function aims to minimize the total energy consumption, the testing sequence of these batches is irrelevant. Assuming there are a total of n decision points, we must decide at each decision point which burn-in board should be selected and which batch it should be added to. We propose four ways to choose candidate burn-in boards and four distinct batch organization methodologies. Let UJ_t represent the set of unscheduled jobs at decision time $t \in \{1,2,\cdots,n\}$. The rules for selecting jobs are as follows:

(1) $J_j \leftarrow argmax\{p_j | j \in UJ_t\}$: Select the burn-in board with the largest testing time among the set of unscheduled burn-in boards UJ_t.

(2) $J_j \leftarrow argmin\{s_j | j \in UJ_t\}$: Select the burn-in board with the smallest size among the set of unscheduled burn-in boards UJ_t.

(3) $J_j \leftarrow argmax\{p_j/s_j | j \in UJ_t\}$: Select the burn-in board with the largest ratio between testing time and size among the set of unscheduled burn-in boards UJ_t.

(4) $J_j \leftarrow$ Randomly select from UJ_t: Randomly select a burn-in board from the set of unscheduled burn-in boards UJ_t.

The common methods of grouping burn-in boards include First-Fit (FF) and Best-Fit (BF) (Alahmadi et al. 2014; Li et al. 2021; Hu, Che, and Zhang 2022; Ho et al. 2007). FF assigns a burn-in board to the first batch that can accommodate it, while BF assigns a burn-in board to a batch that can accept the burn-in board while creating the minimum amount of extra space. If no available batches can accommodate the burn-in board, the Empty-Fit (EF) method creates a new, empty batch. On the other hand, the Random-Fit (RF) method assigns the selected burn-in board to a batch that can accommodate it, with the added randomness of selecting the batch. Using these batching and burn-in board selection principles, we construct ten actions summarized in Table 4.

Table 4. Action collection

Actions	Selecting burn-in board	Assigning a burn-in board to batch
1	$J_j \leftarrow argmax\{p_j \mid j \in UJ_t\}$	First-Fit (FF)
2	$J_j \leftarrow argmax\{p_j \mid j \in UJ_t\}$	Best-Fit (BF)
3	$J_j \leftarrow argmin\{s_j \mid j \in UJ_t\}$	First-Fit (FF)
4	$J_j \leftarrow argmin\{s_j \mid j \in UJ_t\}$	Best-Fit (BF)
5	$J_j \leftarrow argmax\{p_j/s_j \mid j \in UJ_t\}$	First-Fit (FF)
6	$J_j \leftarrow argmax\{p_j/s_j \mid j \in UJ_t\}$	Best-Fit (BF)
7	$J_j \leftarrow$ Randomly select from UJ_t	First-Fit (FF)
8	$J_j \leftarrow$ Randomly select from UJ_t	Best-Fit (BF)
9	$J_j \leftarrow argmax\{p_j \mid j \in UJ_t\}$	Empty-Fit (EF)
10	$J_j \leftarrow argmax\{p_j \mid j \in UJ_t\}$	Random-Fit (RF)

3.4 Reward

Initially, the first half of the current production environment is used to log the processing time for each batch. When the processing time of a newly added burn-in board exceeds that of the existing batch, the batch processing time is converted to the processing time of the newly added burn-in board. Consequently, we must choose whether to replace the original burn-in board in the batch with the newly added burn-in board at each decision point. At decision time t, we obtain the set of burn-in boards reserved to represent batch processing time, denoted as J_R, and the set of burn-in boards discarded, denoted as J_d. The total energy consumption is the product of unit energy consumption

and the sum of the processing time for the burn-in boards in J_R, thus we have:

Primarily, the first half of the current production environment is used to log the processing time for each batch. When the processing time of a newly added burn-in board exceeds that of the existing batch, the batch processing time is converted to the processing time of the newly added burn-in board. Consequently, we must choose whether to replace the original burn-in board in the batch with the newly added burn-in board at each decision point. At decision time n, we obtain the set of burn-in boards reserved to represent batch processing time, denoted by J_R, and the set of burn-in boards discarded, denoted by J_d. The total energy consumption is the product of unit energy consumption and the sum of the processing time for the burn-in boards in J_R, thus we have:

$$TEC = E * \sum_{j \in J_R} p_j \tag{15}$$

Additionally, if each burn-in board is individually produced as a batch, the total energy consumption can be expressed as:

$$E * \sum_{j=1}^{n} p_j = E * \sum_{j \in J_R} p_j + E * \sum_{j \in J_d} p_j = TEC + E * \sum_{j \in J_d} p_j \tag{16}$$

It has been determined that the $\sum_{j \in J_d} p_j$ value increases the most when the total energy consumption is the smallest possible. Thus, the reward at each decision point can be set as the total energy consumption of the burn-in boards that were dismissed.

4. DQN-based Algorithm

This section introduces the basic Deep Q-Network (DQN) algorithm and the Dueling Network and then provides a training procedure for the Dueling DQN method.

4.1 DQN algorithm

DQN is a Q-learning algorithm integrating value function approximation with neural network technology. It communicates with the production environment via an agent, monitors the current state, performs the best possible action, and generates a predetermined set of rewards. Combining the state, action, and reward definitions from Section 3, the DQN algorithm is briefly described below.

Assuming a single-batching machine scheduling process is recorded as shown in Fig. 3.

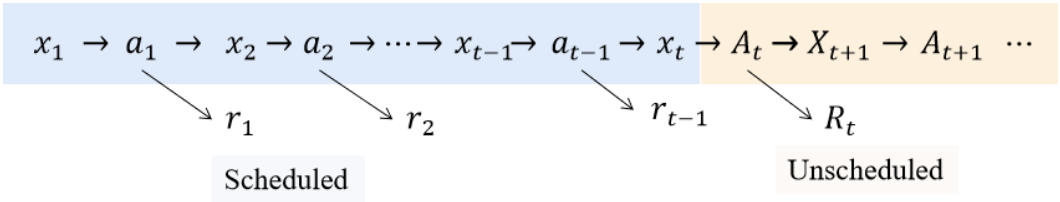


Fig. 3. Whole scheduling process

We have full knowledge of all actions, states, and rewards before time t, but we do not know what will happen at or after t. Therefore, if we assume that the discount rate is γ, we can write down the formula for the discount return (G) as follows:

$$U_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots + \gamma^{n-t} R_n \tag{17}$$

The magnitude of U_t is connected to every possible combination of states and actions that may occur later. Thus, we can express the value of any action as:

$$Q_\pi(x_t, a_t) = \mathbb{E}[U_t | X_t = x_t, A_t = a_t, \pi] \tag{18}$$

Where π is the probability density function of the action, and we call the $Q_\pi(x_t, a_t)$ as the action value function. Then, the value of state s (state value function) is given as:

$$V_\pi(x_t) = \mathbb{E}_{A_t \sim \pi(\cdot|x_t)}[Q_\pi(x_t, A_t)] \tag{19}$$

The optimal action-value function is denoted as $Q_*(x_t, a_t)$, is given as:

$$Q_*(x_t, a_t) = \max_\pi \{Q_\pi(x_t, a_t)\} \tag{20}$$

Given the present state at each decision-making point, $Q_*(x_t, a_t)$ can serve as a guide for determining the appropriate action. The Q-table is used in the Q-learning method to store all the $Q_*(x_t, a_t)$, but it becomes inefficient when the number of possible state and action combinations is huge. To approximate the $Q_*(x_t, a_t)$, neural networks are employed in DQN. Thus, we have:

$$Q_*(x_t, a_t) \approx Q(x_t, a_t; \omega) \tag{21}$$

We then describe how to train DQN with the Temporal Difference (TD) method to determine the ω value of the network $Q(x_t, a_t; \omega)$. In the TD method, we approximate the action value function using the observed reward r_t, hence we have:

$$Q(x_t, a_t; \omega) \approx r_t + \gamma \cdot \max_{a \in A} Q(x_{t+1}, a; \omega) \tag{22}$$

Thus, the loss function $L(\omega)$ is given as:

$$L(\omega) = [Q(x_t, a_t; \omega) - r_t - \gamma \cdot \max_{a \in A} Q(x_{t+1}, a; \omega)]^2 \tag{23}$$

Assuming the gradient of the loss function $L(\omega)$ is $\nabla\omega L(\omega)$, we have:

$$\omega \leftarrow \omega - \nabla\omega L(\omega) \tag{24}$$

4.2 Dueling Network

Dueling Network enhances the structure of the DQN neural network, which also approximates the optimal action-value function. Dueling Network introduces the state value function and the optimal advantage function to define the optimal action-value function, considering the dual values of state and action. The optimal advantage function is defined as:

$$D_*(x_t, a_t) = Q_*(x_t, a_t) - V_*(x_t) \tag{25}$$

Where $V_*(x_t) = \max_\pi \{V_\pi(x_t)\}$. Due to $V_*(x_t) \geq Q_*(x_t, a_t)$, thus $D_*(x_t, a_t) \leq 0$ and $\max_{a_t \in A_t} \{D_*(x_t, a_t)\} = 0$. Thus, the optimal action-value function can be rewritten as:

$$Q_*(x_t, a_t) = V_*(x_t) + D_*(x_t, a_t) - \max_{a_t \in A_t} \{D_*(x_t, a_t)\} \tag{26}$$

We use neural networks $D(x_t, a_t; \omega^D)$ and $V(x_t; \omega^V)$ to approximate $V_*(x_t)$ and $D_*(x_t, a_t)$, respectively. Therefore, we give the dueling network as follows:

$$Q_*(x_t, a_t; \omega) = V(x_t; \omega^V) + D(x_t, a_t; \omega^D) - \max_{a_t \in A_t} \{D_*(x_t, a_t; \omega^D)\} \tag{27}$$

where $\omega \triangleq \{\omega^V; \omega^D\}$. The network structure of $Q_*(x_t, a_t; \omega)$ is shown in Fig. 4. Here, each neuron in the network receives its input from the state variable, which consists of six components. To emphasize the idea of 'less is more', we use the Taguchi approach to narrow the ten potential actions to only 4. The input layer is connected to the three convolutional layers, and the output of the convolution operation is the feature vectors that feed into the state value network and the advantage action network, respectively. Like the advantage action network, the state value network consists of three-layered, fully connected networks. While the state value network has a single output, the advantage action network has results proportional to the number of actions. Finally, the state value network and advantage action network outputs are aggregated as Q-values according to

Eq. (27).

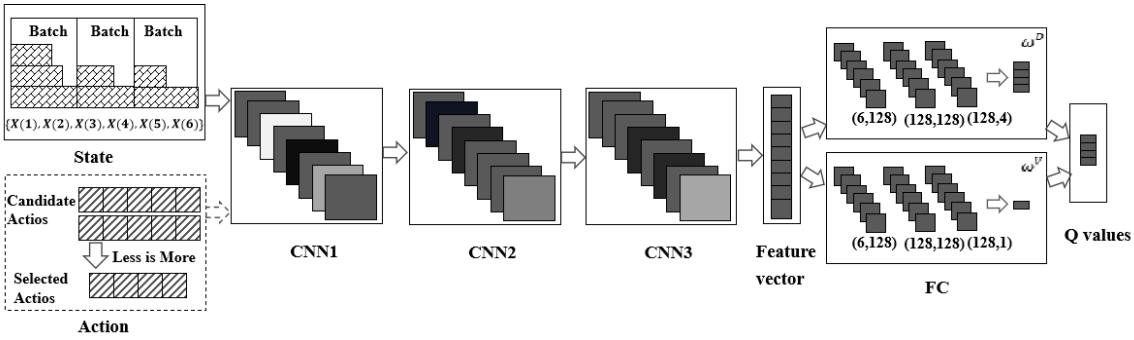


Fig. 4. Network structure of Dueling Network with Less is More Strategy

4.3 Training procedure

The training procedure for dueling DQN is similar to that of basic DQN, with the introduction of the experience replay mechanism and the target network mechanism. The ϵ-greedy method is employed to acquire experience, and the tuples containing x_f, a_f, r_f, x_{f+1} are randomly retrieved from the replay memory D to update the parameters of the dueling network $\omega \triangleq \{\omega^V; \omega^D\}$. The training process can be summarized in Table 5.

Table 5. The training framework of the Dueling DQN

The training process of the Dueling DQN
1. Initialize parameters $max\_episode, \tau$
2. Initialize the replay memory D with capacity N
3. Initialize the action value network Q with random weight ω
4. Initialize the target action value network $\hat{Q}$ with weight $\omega^- = \omega$
5. **while** $episode \leq max\_episode$ **do**
6. Set $t = 1$ and let $x_t = \{0,0,0,0,0,0\}$ #Reset production environment
7. **while** $t \leq n$ **do**
8. Select action $a_t = \begin{cases} Uniform\ sample\ an\ action & if\ \varepsilon > rand(0,1) \\ argmax_a Q(x_t, a, \theta) & Otherwise \end{cases}$
9. Execute action a_t, and observe the next state x_{t+1}, reward r_t
10. **if** $t<n$, **then** $done = 0$, **otherwise** $done = 1$
11. Store transition $(x_t,\ a_t,\ r_t,\ x_{t+1},\ done)$ in D
12. Sample random minibatch of transitions $(x_f,\ a_f,\ r_f,\ x_{f+1},\ done)$ from D
13. Set $y_f = \begin{cases} r_f & if\ done \\ r_f + \gamma \max\limits_{a \in A} Q(x_{f+1}, a; \omega^-) & Otherwise \end{cases}$
14. Calculate loss $L(\omega) = [Q(x_f, a_f; \omega) - y_f]^2$
15. Execute gradient descent process on $L(\omega)$ and update ω
16. If $episode\ \%\ 100=0$, then let $\omega^- = \tau\omega + (1-\tau)\omega^-$

5. Numerical Experiments

This section presents the experimental results comparing the Dueling DQN algorithm to other algorithms, demonstrating its superiority through a series of comparisons. First, the experimental designs are described in Section 5.1, with the Taguchi method used in Section 5.2 to eliminate unnecessary actions. Next, Section 5.3 details the training process for the Dueling DQN algorithm. Finally, Sections 5.4 and 5.5 evaluate the method's efficiency by comparing it to other common

heuristic and meta-heuristic algorithms, showing that it achieves better results.

5.1 Experimental design

We performed experiments on various instances of the E-SBPM-AJS problem, following the setup proposed by Zhou et al. (2021). Table 6 displays the parameters defined for different production arrangements. All algorithms were implemented in Python, and the experiments were run on a personal computer with an Intel Core i7-9700 @ 3.0 GHz CPU and 32.0 GB RAM.

Table 6. Parameter settings of different product configurations.

Parameters	**Value**
The number of burn-in boards (N)	{50, 100, 200, 300}
The size of burn-in boards(s_j)	[1, 20], [10, 30], and [1, 40]
The processing time of burn-in boards(p_j)	[1,50]
The capacity of machine(C)	40
The unit energy consumption (E)	1

Table 6 indicates that we considered different total numbers of burn-in boards (50, 100, 200, and 300) with burn-in board sizes ranging from [1, 20], [10, 30], and [1, 40]. The processing times for the burn-in boards were arbitrarily set between 1 and 50, while the capacity of the batching machine was fixed at 40 for all instances. We generated ten random groups of cases for each type based on the total number of burn-in boards. The parameters used for the Dueling DQN algorithm are listed in Table 7.

Table 7. Parameter settings of the Dueling DQN algorithm

Parameters	**Value**
Number of training episodes	2000
Memory size	20000
Batch size	32
Target update	10000
Epsilon decay	0.0005
Discount factor	0.99
Max epsilon	1.0
Min epsilon	0.1
Alpha	0.2
Beta	0.6
Prior epsilon	0.000001

5.2 Less is More policy

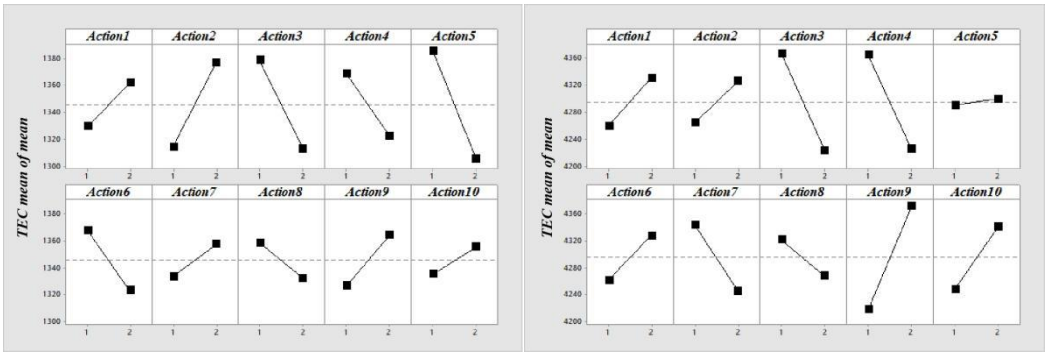
To ensure the efficiency and convergence of the Dueling DQN algorithm, we evaluated ten distinct scheduling rules as actions. However, it is unknown whether all actions can improve algorithm performance or if some actions could increase execution time without improving efficiency. We utilized the Taguchi approach to address this to eliminate potentially underperforming actions (Pei et al. 2022). We conducted orthogonal experiments with ten variables and two levels, with level 1 indicating the selection of an action and level 2 indicating its exclusion. We trained the algorithm for 3000 episodes and recorded the minimum energy consumption achieved during the training process for 12 distinct action combinations as the experiment value. Table 8 presents the experimental values for the 12 action combinations utilized in the orthogonal experiments.

Table 8. Orthogonal Array $L_{12}(2^{10})$ for different instances (100 and 300 burn-in boards).

Trials	Action										The mean of total energy consumption	
	Action1	Action2	Action3	Action4	Action5	Action6	Action7	Action8	Action9	Action10	100	300
1	1	1	1	1	1	1	1	1	1	1	1383.8	4304.3
2	1	1	1	1	1	2	2	2	2	2	1404.2	4425.6
3	1	1	2	2	2	1	1	1	2	2	1259	4239.3
4	1	2	1	2	2	1	2	2	1	1	1327.1	4046.1
5	1	2	2	1	2	2	1	2	1	2	1249.7	4337
6	1	2	2	2	1	2	2	1	2	1	1351.2	4203.7
7	2	1	2	2	1	1	2	2	1	2	1322.2	4033.2
8	2	1	2	1	2	2	2	1	1	1	1259.2	4170.9
9	2	1	1	2	2	2	1	2	2	1	1256.1	4409.5
10	2	2	2	1	1	1	1	2	2	1	1434.4	4354.4
11	2	2	1	2	1	2	1	1	1	2	1417.9	4418
12	2	2	1	1	2	1	2	1	2	2	1481	4595.1

Table 9 shows that we investigate two instances with 100 and 300 burn-in boards, respectively. The optimal total energy consumption obtained through training with Dueling DQN is used as the experimental value for the orthogonal experiments. Based on the empirical analysis of the orthogonal experiments, we get the results illustrated in Fig. 5, and Table 8 presents the best level values for each action. According to the experimental settings and procedures described above, actions 1, 2, 9, and 10 can significantly reduce the total energy consumption in both instances. Therefore, we choose these four heuristic rules as the actions for the Dueling DQN algorithm.

It can be observed that, for the selection of burn-in board, choosing the burn-in board with the largest testing time contributes to the improvement of algorithm efficiency. To enhance the algorithm's global search capability, in addition to the BF and FF rules during the batch formation, the EF and RF rules from actions 9 and 10 need to be considered.



(a) The instance with 100 burn-in boards (b) The instance with 300 burn-in boards

Fig. 5. The mean of total energy consumption for each parameter level of all instances

Table 9. Optimal tuning parameters for total energy consumption (100 and 300 burn-in boards).

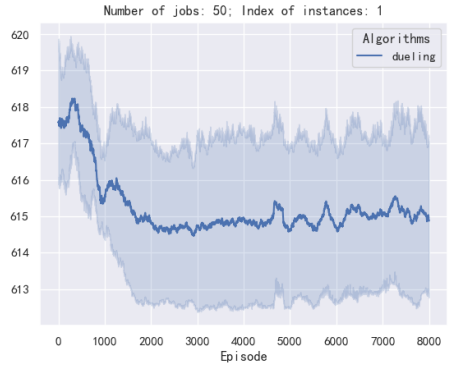
N	Parameters	Action 1	Action 2	Action 3	Action 4	Action 5	Action 6	Action 7	Action 8	Action 9	Action 10
100	Best level	1	1	2	2	2	2	1	2	1	1
	Level value	☑	☑	☒	☒	☒	☒	☑	☒	☑	☑

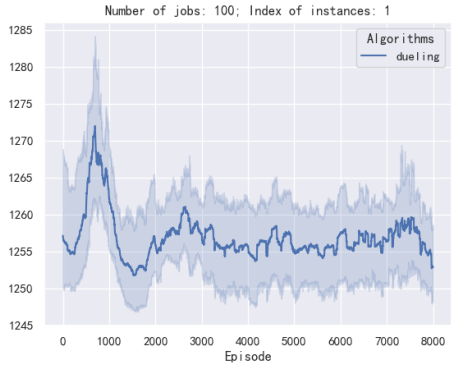
300	**Best level**	1	1	2	2	1	1	2	2	1	1
	Level value	☑	☑	☒	☒	☑	☑	☒	☒	☑	☑

* ☑ indicates that the action is selected, while ☒ implies that it is not.

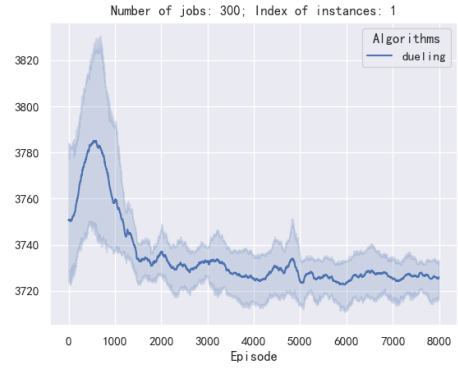
5.3 The training process of Dueling DQN

To evaluate the convergence of the Dueling DQN method, we conducted five separate training sessions on the first type of computation instances, which involve 50, 100, 200, and 300 burn-in boards with varying random seed values. The number of training episodes was set at 8000 for each session. The results of these experiments are displayed in Fig. 6. We can find that the training results of the Dueling DQN algorithm exhibit slight fluctuations around 1000 episodes but converge at 3000 episodes. Thus, in the subsequent experiments, we set the maximum number of training episodes for the Dueling DQN algorithm to 3000.





(a) Convergence curve for 50 burn-in boards (b) Convergence curve for 100 burn-in boards



(c) Convergence curve for 200 burn-in boards (d) Convergence curve for 300 burn-in boards

Fig. 6. Convergence curves for the instances with 50, 100, 200, and 300 burn-in boards

5.4 Comparison with heuristic algorithms

To verify the superiority of Dueling DQN, the total 40 experimental results are recorded in Appendix Table A1. Here, we summarize the results given in Table A1 into Table 9. The BFD-LPT, BFD-RAND, BFD-LPS, and BFD-SJS algorithms are below.

Procedure of BFD-LPT (BFD-RAND, BFD-LPS, and BFD-SJS)	
Step1	Sort all the burn-in boards by LPT (RAND, LPS, SJS) rule
Step2	Batch all the burn-in boards in turn by the BFD rule
Step3	Calculate the total energy consumption for the scheduling solution

In this problem, burn-in boards in the queue can be ordered before being batched according to

different criteria, including Longest Process Time (LPT), Random (RAND) burn-in board sequence, Largest Process Time to Job Size Ratio (LPS), and Smallest Job Size (SJS). The BFD rule requires that jobs be assigned to the batch with the least remaining space to accommodate the job. This rule is applied when grouping batches. Similarly, we also provide the corresponding FFD-LPT, FFD-RAND, FFD-LPS, and FFD-SJS algorithms, where the FFD rule is used to place the job in the batch that can handle it first.

Procedure of FFD-LPT (FFD-RAND, FFD-LPS, and FFD-SJS)	
Step1	Sort all the jobs by LPT (RAND, LPS, SJS) rule
Step2	Batch all the burn-in boards in turn by the FFD rule
Step3	Calculate the total energy consumption for the scheduling solution

The ALL-RAND and LIMA-RAND approaches enable the agent to interact randomly with the production environment by excluding the deep neural network during training. The LIMA-RAND algorithm allows the agent to select an action from only 1, 2, 9, and 10 actions, while the ALL-RAND algorithm grants the agent access to all ten actions. The CPLEX algorithm runs the single batching scheduling problem through the CPLEX solver and selects the optimal solution within 3600 seconds.

Table 10. The comparison results of Dueling DQN and 11 heuristic algorithms.

Algorithms	**Metrics**	**Instances**			
		50	**100**	**200**	**300**
BFD-LPT	**Mean**	587.86	1219.37	2325.56	3494.09
BFD-RAND	**Mean**	670.64	1438.35	2868.49	4383.79
BFD-LPS	**Mean**	672.26	1410.27	2723.19	4127.74
BFD-SJS	**Mean**	805.75	1713.85	3374.79	5095.93
FFD-LPT	**Mean**	691.01	1413.82	2825.41	4173.13
FFD-RAND	**Mean**	664.78	1434.19	2880.96	4396.18
FFD-LPS	**Mean**	672.68	1405.68	2705.42	4102.13
FFD-SJS	**Mean**	801.35	1713.85	3368.11	5081.73
ALL-RAND	**Mean**	614.94	1318.02	2605.27	3966.43
LIMA-RAND	**Mean**	581.72	1229.42	2358.36	3552.88
CPLEX	**Mean**	580.4	1300.87	2923.43	4513.5
Dueling DQN	**Mean**	579.48	1216.46	2317.08	3499.95
BFD-LPT	**Std**	76.362	79.713	113.54	144.836
BFD-RAND	**Std**	65.936	74.494	114.201	118.463
BFD-LPS	**Std**	83.053	83.008	103.804	196.312
BFD-SJS	**Std**	99.282	108.636	130.053	173.664
FFD-LPT	**Std**	92.184	75.107	188.896	150.566
FFD-RAND	**Std**	70.738	74.226	121.921	128.853
FFD-LPS	**Std**	83.296	88.102	105.15	192.207
FFD-SJS	**Std**	92.831	108.636	126.166	176.894
ALL-RAND	**Std**	73.592	85.46	124.004	151.257
LIMA-RAND	**Std**	71.985	80.013	116.473	153.928
CPLEX	**Std**	70.539	82.161	147.164	145.627
Dueling DQN	**Std**	69.832	77.402	105.511	144.042

In Table 10, the mean and standard deviation of the best total energy consumption values obtained

from 10 runs of each instance type are displayed in orange and green, respectively. Darker colors indicate smaller values for both metrics. The results lead to the following conclusions:

(1) The BFD-LPT, LIMA-RAND, and Dueling DQN algorithms perform well, with Dueling DQN being the best among them.

(2) The performance of the CPLEX algorithm deteriorates as the number of burn-in boards increases.

(3) The BFD-SJS and FFD-SJS algorithms perform poorly, indicating that the SJS rule is unsuitable for solving the presented problem. Therefore, the LIMA idea suggests removing the SJS actions from the original ten actions.

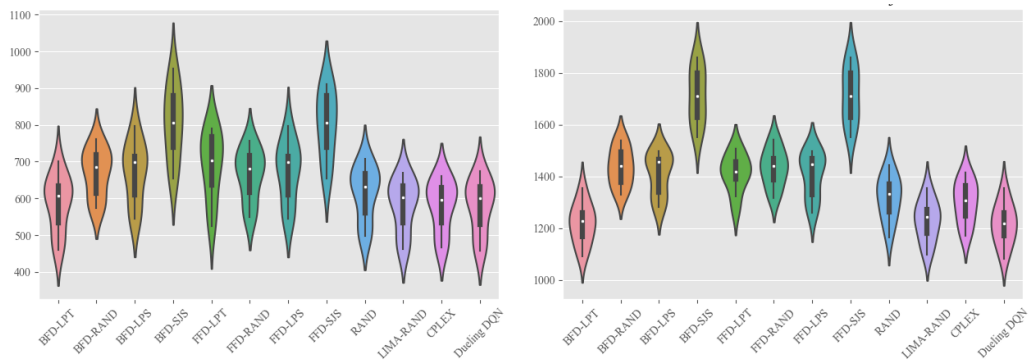
(4) Dueling DQN is superior to LIMA-RAND, demonstrating that the DQN mechanism effectively guides the search in the desired direction.

Fig. 7 visually represents the algorithmic performance using a violin diagram. The figure shows that Dueling DQN consistently achieves the best results, outperforming other algorithms.

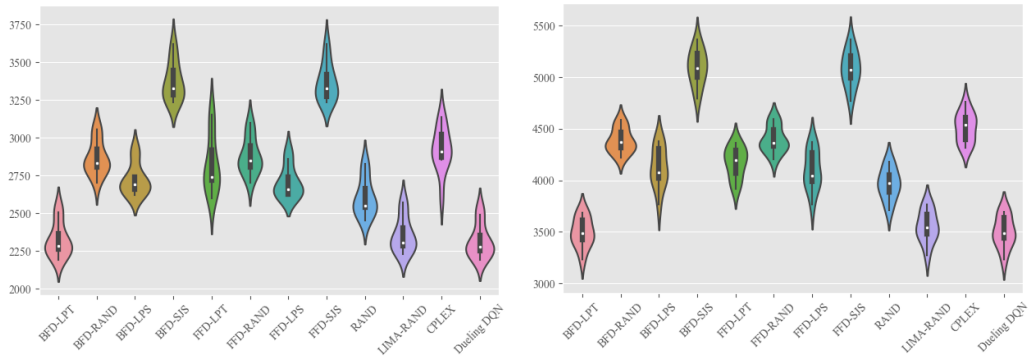
The findings for the instances with 50 burn-in boards are comparable for Dueling DQN, CPLEX, LIMA-RAND, and BFD-LPT, with outcomes hovering around 600. BFD-SJS and FFD-LPS exhibit the poorest performance, with results consistently below par, approaching around 800. The variation across algorithms is relatively small due to the low number of burn-in boards.

For the instances with 100 burn-in boards, the benefits and drawbacks of the algorithms become apparent. The performance of CPLEX degrades as the number of burn-in boards increases, while Dueling DQN, LIMA-RAND, and BFD-LPT maintain relatively similar results, around 1300. BFD-SJS and FFD-LPS demonstrate the lowest performance, with total energy consumption consistently around 1700.

As the number of instances increases to 200 and 300, Dueling DQN, LIMA-RAND, and BFD-LPT produce similar results, with some subtle variations. Again, dueling DQN and BFD-LPT are comparable, with slightly better results than LIMA-RAND. Overall, Dueling DQN outperforms LIMA-RAND and BFD-LPT in most instances, and the difference in outcomes between algorithms increases with the size of cases. Therefore, Dueling DQN is a superior choice for addressing large-scale scheduling problems compared to other algorithms.



(a) Results of instances with 50 burn-in boards (b) Results of instances with 100 burn-in boards



(c) Results of instances with 200 burn-in boards (d) Results of instances with 300 burn-in boards

Fig. 7. Comparison of Dueling DQN algorithm with heuristic algorithms

5.5 Comparison with meta-heuristic algorithms

Meta-heuristic or intelligent optimization algorithms use computational intelligence mechanisms to find the best or satisfactory solution to complex optimization problems (Ahmed et al. 2021). These algorithms are based on principles from various fields, such as biology, physics, chemistry, society, and art, which provide insight into behavior, function, experience, and rules (Wu et al. 2021). Some popular meta-heuristics include Simulated Annealing (Defersha, Obimuyiwa, and Yimer 2022), Genetic Algorithm (Zhang et al. 2020), Differential Evolution Algorithm (Song et al. 2023), Particle Swarm Optimization (Tang et al. 2021), Artificial Fish Swarm Algorithm (Tirkolaee, Goli, and Weber 2020), Immune Algorithm (Li et al. 2020), and more. These algorithms utilize random search techniques within the solution space, but they exhibit variations in terms of search strategy, solution representation, operator manipulation, and global search capability. Given the variations in population settings and iteration mechanisms among the aforementioned algorithms, it would be unsound to terminate them solely based on a predefined maximum number of iterations. To ensure fairness in our evaluations, we have adopted a consistent approach by setting the number of newly generated solutions during the iteration process to be 8000 for all algorithms. Other parameters of all the compared algorithms used in this paper are defined as Table 11.

Table 11. Parameters for SA, GA, PSO, IA, and DE

Algorithms	Parameters	Description	Values
Simulated Annealing(SA)	lc	Number of iteration under every temperature	20
Genetic Algorithm(GA)	prob_mut	Probability of mutation	0.001
	prob_cros	Probability of crossover	0.9
Particle Swarm Optimization(PSO)	w	Weights	0.8
	c1	Individual memory	0.5
	c2	Collective memory	0.5
Immune Algorithm(IA)	T	Threshold for affinity.	0.7
	alpha	Diversity evaluation index	0.95
Artificial Fish Swarm Algorithm(AFSA)	step	Maximum proportion of displacement at each step	0.5
	visual	Maximum perceptual range of the fish	0.3
	delta	Crowding threshold	0.5
Differential	prob_mut	Probability of mutation	0.001

Evolution(DE)	F	Coefficient of mutation	0.005

In this study, we evaluate the performance of the DQN algorithm against these meta-heuristics and present our findings. We use the scikit-opt Python library, available on GitHub, to implement the meta-heuristic algorithms. Two techniques for computing fitness levels are employed, one based on the BFD batching rule and the other based on the FFD batching rule. The framework of BFD-based fitness computation and FFD-based fitness computation is given below.

BFD-based fitness computation/FFD-based fitness computation	
Step1	Given a sequence of N real numbers between 0 and 1, denoted as $x = \{x_1, \cdots, x_N\}$. Here, the current index list of x is $index = \{1,2, \cdots, N\}$.
Step2	Sort x by the ascending order, then update $index$, and finally sort the burn-in boards using $index$.
Step3	Applying the BFD/FFD rule to the burn-in boards that have been sorted.
Step4	Calculate the total energy consumption and utilize it as the fitness value of sequence x.

Most meta-heuristic algorithms begin with an initial solution or population and use a neighborhood generation rule to generate the next-generation solution or population, with the fitness computation method used to evaluate the pros and cons of each individual. Therefore, we constructed 12 meta-heuristics based on the two fitness calculation methods. Fig. 8 depicts the iterative procedure and classification of the meta-heuristic algorithms.

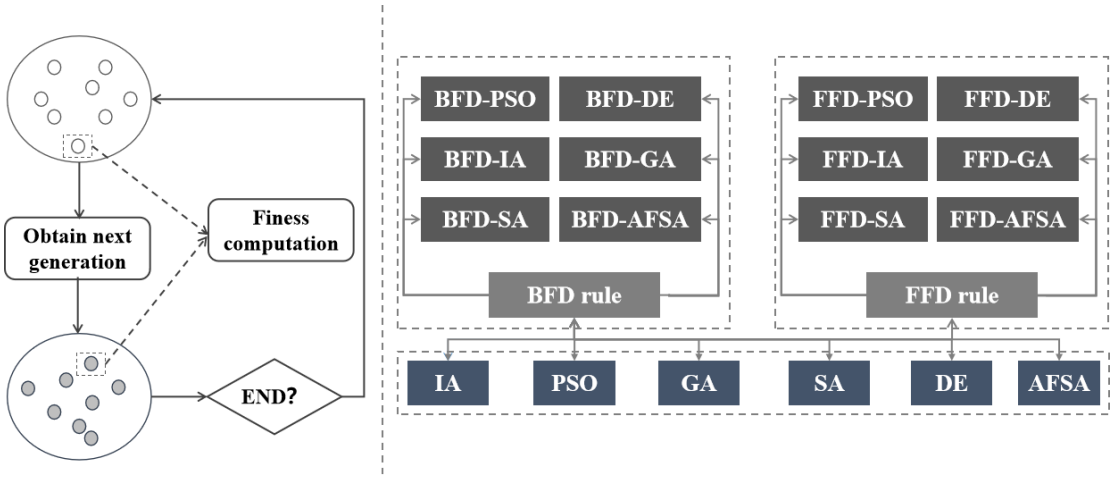


Fig. 8. The detail about the 12 meta-heuristic algorithms

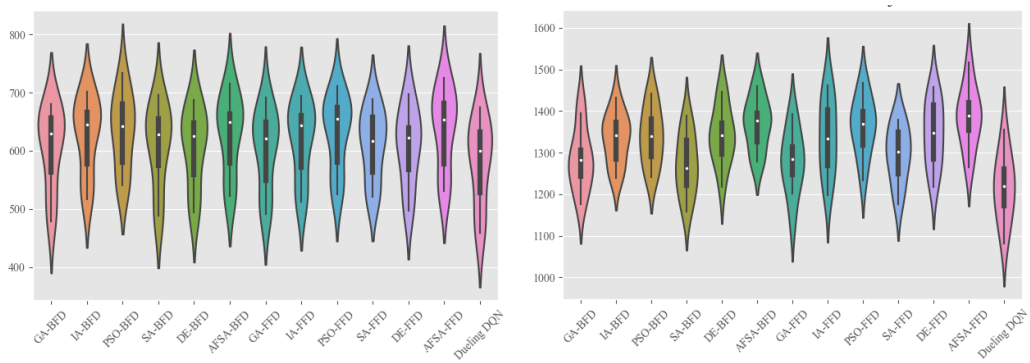
The comprehensive findings of the experiments are presented in Appendix Table A2.

Table 12. The comparison results of Dueling DQN and meta-heuristic algorithms.

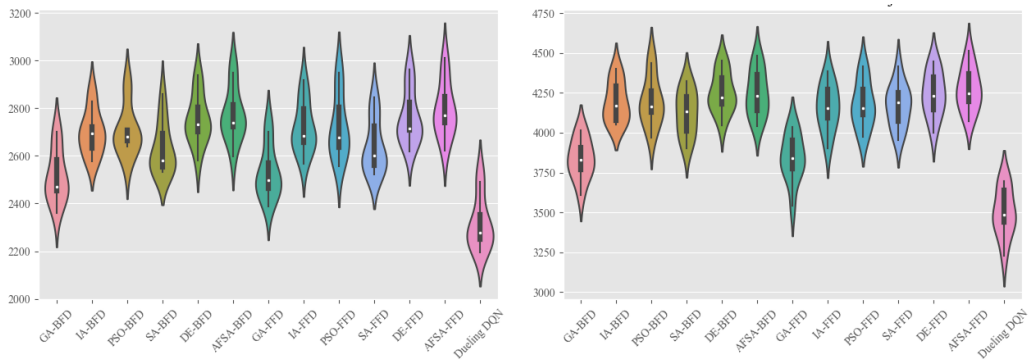
Algorithms	**Metrics**	**Instances**			
		50	**100**	**200**	**300**
GA-BFD	**Mean**	605.56	1285.12	2517.25	3836.04
IA-BFD	**Mean**	626.6	1333.55	2699.82	4193.71
PSO-BFD	**Mean**	633.34	1340.16	2707	4200.77
SA-BFD	**Mean**	608.45	1272.51	2633.16	4124.77
DE-BFD	**Mean**	606.18	1342.47	2754.55	4253.31
AFSA-BFD	**Mean**	628.77	1369.75	2775.87	4248.33
GA-FFD	**Mean**	602.81	1279.05	2531.08	3843.46
IA-FFD	**Mean**	619.49	1339.07	2720.01	4160.88
PSO-FFD	**Mean**	631.81	1363.98	2725.03	4178.76

SA-FFD	**Mean**	610.44	1295.29	2641.56	4170.66
DE-FFD	**Mean**	607.25	1345.71	2763.12	4245.23
AFSA-FFD	**Mean**	637.99	1385.6	2794.95	4273.62
Dueling DQN	**Mean**	579.48	1216.46	2317.08	3499.95
GA-BFD	**Std**	70.64	75.41	112.53	127.76
IA-BFD	**Std**	65.87	61.22	95.60	128.75
PSO-BFD	**Std**	66.29	69.15	107.73	156.69
SA-BFD	**Std**	71.78	73.84	110.20	143.84
DE-BFD	**Std**	67.65	70.44	105.71	128.77
AFSA-BFD	**Std**	68.28	62.81	113.38	146.47
GA-FFD	**Std**	68.99	77.11	111.38	149.45
IA-FFD	**Std**	66.10	90.30	108.46	147.46
PSO-FFD	**Std**	64.07	70.64	136.05	148.40
SA-FFD	**Std**	60.54	68.83	114.54	135.97
DE-FFD	**Std**	66.42	79.68	113.89	142.29
AFSA-FFD	**Std**	70.34	73.98	117.09	136.90
Dueling DQN	**Std**	69.83	77.40	105.51	144.04

Here, we present a summary of the results in Appendix Table A2 into Table 12, which compares the performance of Dueling DQN and meta-heuristic algorithms. The mean and standard deviation metrics for instances with 50, 100, 200, and 300 burn-in boards are presented in Table 12, and a corresponding violin diagram is provided in Fig. 9 for easier comparison. Our findings show that, when comparing mean and standard deviation metrics, Dueling DQN outperforms the meta-heuristic algorithms. This conclusion is also clearly visible in Fig. 9. When dealing with instances of 50 burn-in boards, Dueling DQN exhibits optimal performance, achieving a mean total energy consumption of approximately 600. However, the differences between Dueling DQN and other algorithms are insignificant for instances of this size. As a result, it is difficult to determine which algorithm is the best based on this criterion alone. For example, with 100 burn-in boards, however, the differences between the algorithms become readily apparent. Our results indicate that Dueling DQN performs best when dealing with cases of this size, as the results achieved by Dueling DQN are much lower than those of other algorithms. While the differences in the results shown by different algorithms are subtle, it is clear that Dueling DQN outperforms them significantly. The benefits of Dueling DQN are most pronounced for instances with 200 and 300 burn-in boards. Compared to other algorithms, Dueling DQN consistently produces much lower total energy consumption values. GA-BFD and GA-FFD are the second-best performing algorithms, but they are still outperformed by Dueling DQN, mainly when dealing with more significant instances. Therefore, it can be argued that the superiority of Dueling DQN over competing algorithms becomes increasingly apparent as the size of the cases increases. Overall, our results suggest that Dueling DQN is the best algorithm for solving the batch scheduling problem presented in this study.



(a) Results of instances with 50 burn-in boards (b) Results of instances with 100 burn-in boards



(c) Results of instances with 200 burn-in boards (d) Results of instances with 300 burn-in boards

Fig. 9. Comparison of Dueling DQN algorithm with meta-heuristic algorithms

5.6 Policy implication

The E-SBPM-AJS model and Dueling DQN are essential in management decision-making processes such as planning and allocating manufacturing resources, saving electricity consumption, and shortening product manufacturing cycles. This work can provide the following insights for managers.

1. Improve Production Efficiency: E-SBPM-AJS model and Dueling DQN can help managers optimize the use of resources to avoid waste and delays and improve production efficiency. These models allow managers to develop optimal production schedules to maximize productivity while avoiding avoidable mistakes and costs. We can compare Dueling DQN and heuristic algorithms to illustrate this point. Heuristic algorithms such as BFD-LPT, BFD-RAND, and BFD-LPS are mainly derived from manual production experience and are commonly used in the workshop. Compared to these experiences, Dueling DQN provides a better solution, achieving lower production and manufacturing spans in instances with 50, 100, 200, and 300 burn-in boards.

2. Achieve Cost Control: Using the E-SBPM-AJS model and Dueling DQN can help managers optimize production costs by reducing total energy consumption. According to the comparison results between Dueling DQN and CPLEX, DQN often achieves production scheduling solutions with lower total energy consumption, reducing production costs for enterprises.

3. Better Customer Service: Using the E-SBPM-AJS model and Dueling DQN, managers can create better production plans to reduce manufacturing cycles, shorten product delivery times, and improve customer satisfaction.

6. Conclusions

This study addresses the E-SBPM-AJS problem in the semiconductor manufacturing industry.

Instead of relying on traditional heuristics, meta-heuristics, or exact algorithms, this paper formulates the problem as a Markov decision problem using the deep reinforcement learning framework and trains it using the Dueling DQN algorithm. In addition, this work employs the 'less is more' principle to enhance the algorithm's convergence speed and quality by reducing the action set. The experimental results demonstrate that the proposed Dueling DQN algorithm outperforms existing heuristic and meta-heuristic algorithms. There is a current policy emphasis on ensuring green supply chains in the semiconductor industry in China and elsewhere; this study suggests those strategies should be paired with improved scheduling by the firms themselves.

Furthermore, the proposed algorithm could be extended to handle more complex scheduling scenarios by incorporating additional constraints and objectives, such as energy consumption, resource allocation, and maintenance schedules. Additionally, investigating the generalization capabilities of the algorithm on unseen data or transfer learning to different manufacturing processes and systems could be a promising area of future research. Overall, there is still much potential for advancing the application of deep reinforcement learning in manufacturing scheduling.

Acknowledgments

This research has received financial support from various sources, including the Ministry of Education of Humanities and Social Science Project [grant number 22YJC630050], the China Postdoctoral Science Foundation [grant number 2022M710996], the Educational Commission of Anhui Province [grant number KJ2020A0069], the Natural Science Foundation of Anhui Province [grant numbers 2108085QG291 and 2108085QG287], Anhui Province University Collaborative Innovation Project [grant number GXXT-2021-021], Science and Technology Plan Project of Wuhu [grant number 2021yf49, 2022rkx07], National Natural Science Foundation of China [grant numbers 72101071 and 72071056], the Key Research and Development Project of Anhui Province [grant number 2022a05020023].

Data availability statement

The data that support the findings of this study are available from the authors upon reasonable request.

Reference

Ahmed, A. N., T. Van Lam, N. D. Hung, N. Van Thieu, O. Kisi, and A. El-Shafie. 2021. "A comprehensive comparison of recent developed meta-heuristic algorithms for streamflow time series forecasting problem." Applied Soft Computing 105: 107282.

Alahmadi, A., A. Alnowiser, M. M. Zhu, D. Che, and P. Ghodous. 2014. "Enhanced first-fit decreasing algorithm for energy-aware job scheduling in cloud." International Conference on Computational Science and Computational Intelligence IEEE 2: 69-74..

Alizadeh, N., and A. H. Kashan. 2019. "Enhanced grouping league championship and optics inspired optimization algorithms for scheduling a batch processing machine with job conflicts and non-identical job sizes." Applied soft computing 83: 105657.

Al-Salamah, M. 2015. "Constrained binary artificial bee colony to minimize the makespan for single machine batch processing with non-identical job sizes." Applied Soft Computing 29: 379-385.

Azizoglu, M., and S. Webster. 2000. "Scheduling a batch processing machine with non-identical job sizes." International Journal of Production Research 38 (10): 2173-2184.

Chang, P. C., and H. M. Wang. 2004. “A heuristic for a batch processing machine scheduled to minimise total completion time with non-identical job sizes.” The International Journal of Advanced Manufacturing Technology 24 (7): 615-620.

Cheng, B., K. Li, and B. Chen. 2010. “Scheduling a single batch-processing machine with non-identical job sizes in fuzzy environment using an improved ant colony optimization.” Journal of Manufacturing Systems 29 (1): 29-34.

Cheng, B., W. Qi, S. Yang, and X. Hu. 2013. “An improved ant colony optimization for scheduling identical parallel batching machines with arbitrary job sizes.” Applied Soft Computing Journal 13 (2): 765-772.

Chien, C. F., J. T. Peng, and H. C. Yu. 2016. “Building energy saving performance indices for cleaner semiconductor manufacturing and an empirical study.” Computers and Industrial Engineering 99: 448-457.

Chou, F. D. 2007. “A joint GA+ DP approach for single burn-in oven scheduling problems with makespan criterion.” The International Journal of Advanced Manufacturing Technology 35 (5): 587-595.

Cigolini, R., M. Perona, A. Portioli, and T. Zambelli. 2002. “A new dynamic look-ahead scheduling procedure for batching machines.” Journal of Scheduling 5 (2): 185-204.

Damodaran, P., P. K. Manjeshwar, and K. Srihari. 2006. “Minimizing makespan on a batch-processing machine with non-identical job sizes using genetic algorithms.” International journal of production economics 103 (2): 882-891.

Defersha, F. M., D. Obimuyiwa, and A. D. Yimer. 2022. “Mathematical model and simulated annealing algorithm for setup operator constrained flexible job shop scheduling problem.” Computers and Industrial Engineering 171: 108487.

Derinkuyu, K., F. Tanrisever, N. Kurt, and G. Ceyhan. 2020. “Optimizing day-ahead electricity market prices: increasing the total surplus for energy exchange Istanbul.” Manufacturing and Service Operations Management 22 (4): 700-716.

Dupont, L., and C. Dhaenens-Flipo. 2002. “Minimizing the makespan on a batch machine with non-identical job sizes: an exact procedure.” Computers and operations research 29 (7): 807-819.

Fang, K., N. Uhan, F. Zhao, and J. Sutherland. 2016. “Scheduling on a Single Machine Under Time-of-Use Electricity Tariffs.” Annals of Operations Research 238 (1-2): 199–227.

Fanti, M. P., B. Maione, G. Piscitelli, and B. Turchiano. 1996. “Heuristic scheduling of jobs on a multi-product batch processing machine.” International Journal of Production Research 34 (8): 2163-2186.

Fowler, J. W., and L. Mnch. 2021. “A survey of scheduling with parallel batch (p-batch) processing.” European Journal of Operational Research 298 (1): 1-24.

Gahm, C., F. Denz, M. Dirr, and A. Tuma. 2016. “Energy-efficient scheduling in manufacturing companies: A review and research framework.” European Journal of Operational Research 248 (3): 744-757.

Gao, K., Y. Huang, A. Sadollah, and L. Wang. 2020. “A review of energy-efficient scheduling in intelligent production systems.” Complex and Intelligent Systems 6: 237-249.

Ghazvini, F. J., and L. Dupont. 1998. “Minimizing mean flow times criteria on a single batch processing machine with non-identical jobs sizes, International Journal of Production Economics.” 55 (3): 273-280.

Heydar, M., E. Mardaneh, and R. Loxton. 2022. “Approximate dynamic programming for an

energy-efficient parallel machine scheduling problem." European Journal of Operational Research 302 (1): 363-380.

Ho, M. H., F. Hnaien, and F. Dugardin. 2022. "Exact method to optimize the total electricity cost in two-machine permutation flow shop scheduling problem under Time-of-use tariff." Computers and Operations Research 144: 105788.

Ho, N. B., J. C. Tay, and E. M. K. La. 2007. "An effective architecture for learning and evolving flexible job-shop schedules." European Journal of Operational Research 179 (2): 316-333.

Hu, K., Y. Che, and Z. Zhang. 2022. "Scheduling unrelated additive manufacturing machines with practical constraints." Computers and Operations Research 144: 105847.

Hu, L., Z. Liu, W. Hu, Y. Wang, and F. Wu. 2020. "Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network." Journal of Manufacturing Systems 55: 1-14.

Huang, J., K. Pan, and Y. Guan. 2021. "Multistage stochastic power generation scheduling co-optimizing energy and ancillary services." INFORMS Journal on Computing 33 (1): 352-369.

Jang, J. W., Y. J. Kim, and B. S.Kim. 2022. "A Three-Stage ACO-Based Algorithm for Parallel Batch Loading and Scheduling Problem with Batch Deterioration and Rate-Modifying Activities." Mathematics 10 (4): 657.

Jia, Z., and J. Y. T. Leung. 2015. "A meta-heuristic to minimize makespan for parallel batch machines with arbitrary job sizes." European Journal of Operational Research 240 (3): 649-665.

Jia, Z., X. Li, and J. Y. T Leung. 2017. "Minimizing makespan for arbitrary size jobs with release times on P-batch machines with arbitrary capacities." Future Generation Computer Systems 67: 22-34.

Jiang, D. R., and W. B. Powell. 2015. "Optimal hour-ahead bidding in the real-time electricity market with battery storage using approximate dynamic programming." INFORMS Journal on Computing 27 (3): 525-543.

Kashan, A. H., B. Karimi, and F. Jolai. 2006a. "Effective hybrid genetic algorithm for minimizing makespan on a single-batch-processing machine with non-identical job sizes." International Journal of Production Research 44 (12): 2337-2360.

Kashan, A. H., B. Karimi, and F. Jolai. 2006b. "Minimizing makespan on a single batch processing machine with non-identical job sizes: a hybrid genetic approach." European Conference on Evolutionary Computation in Combinatorial Optimization Springer Berlin. Heidelberg 135-146.

Kempf, K. G., R. Uzsoy, and C. S. Wang. 1998. "Scheduling a single batch processing machine with secondary resource constraints." Journal of Manufacturing Systems 17 (1) : 37-51.

Koh, S. G., P. H. Koo, D. C. Kim, and W. S Hur. 2005. "Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families." International Journal of Production Economics 98 (1): 81-96.

Kong, M., X. B. Liu, J. Pei, Z. P. Zhou, and P. M. Pardalos. 2020. "Parallel-batching scheduling of deteriorating jobs with non-identical sizes and rejection on a single machine." Optimization Letters 14 (4): 857-871.

Li, D., J. Wang, R. Qiang, and R. Chiong. 2021. "A hybrid differential evolution algorithm for parallel machine scheduling of lace dyeing considering colour families, sequence-dependent setup and machine eligibility." International Journal of Production Research 59 (9): 2722-2738.

Li, J., Z. Liu, C. Li, and Z. Zheng. 2020. “Improved artificial immune system algorithm for type-2 fuzzy flexible job shop scheduling problem.” IEEE Transactions on Fuzzy Systems 29 (11): 3234-3248.

Li, Z., H. Chen, R. Xu, and X. Li. 2015. “Earliness–tardiness minimization on scheduling a batch processing machine with non-identical job sizes.” Computers and Industrial Engineering 87: 590-599.

Liang, Y., K. Tan, and Y. Li. 2023. “Implementation Principles of Optimal Control Technology for the Reduction of Greenhouse Gases in Semiconductor Industry.” E3S Web of Conferences 394: 01031.

Liu, C H, and D. H. Huang. 2014. “Reduction of power consumption and carbon footprints by applying multi-objective optimisation via genetic algorithms.” International Journal of Production Research 52 (1-2): 337-352.

Luo, S. 2020. “Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning.” Applied Soft Computing 91: 106208.

Melouk, S., P. Damodaran, and P. Y.Chang. 2004. “Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing.” International journal of production economics 87 (2): 141-147.

Palombarini, J. A., and E. C. Martínez. 2019. “Closed-loop rescheduling using deep reinforcement learning.” IFAC-PapersOnLine 52 (1): 231-236.

Park, M. J., and A. Ham. 2022. “Energy-aware flexible job shop scheduling under time-of-use pricing.” International Journal of Production Economics 248: 108507.

Parsa, N. R., B. Karimi, and A. H. Kashan. 2010. “A branch and price algorithm to minimize makespan on a single batch processing machine with non-identical job sizes.” Computers and Operations Research 37 (10): 1720-1730.

Parsa, N. R., B. Karimi, and S. M. Husseini. 2016. “Minimizing total flow time on a batch processing machine using a hybrid max-min ant system.” Computers and Industrial Engineering 99: 372-381.

Pei, J., H. Wang, M. Kong, N. Mladenovic, and P. M. Pardalos. 2022. “Bi-level scheduling in high-end equipment R&D: when more algorithm strategies may not be better.” International Journal of Production Research 1-32.

Pelcat, M. 2023. “GHG emissions of semiconductor manufacturing in 2021.” Univ Rennes INSA Rennes CNRS IETR–UMR 6164 F-35000 Rennes.

Rezaeian, J., and Y. Zarook. 2018. “An Efficient Bi-objective Genetic Algorithm for the Single Batch-Processing Machine Scheduling Problem with Sequence Dependent Family Setup Time and Non-identical Job Sizes.” Journal of Optimization in Industrial Engineering 1 (2): 65-78.

Schwartz, A.. 1993. “A Reinforcement Learning Method for Maximizing Undiscounted Rewards.” Machine Learning Proceedings 298: 298-305.

Song Y., X. Cai, X. Zhou, B. Zhang, H.Chen, Y. Li, W. Deng, and W. Deng. 2023. “Dynamic hybrid mechanism-based differential evolution algorithm and its application.” Expert Systems with Applications 213: 118834.

Tan, Q., H. P. Chen, B. Du, and X. L. Li. 2011. “Two-agent scheduling on a single batch processing machine with non-identical job sizes.” 2011 2nd International Conference on Artificial Intelligence. Management Science and Electronic Commerce (AIMSEC) IEEE: 7431-7435.

Tang, X., C. Shi, T. Deng, Z. Wu, and L. Yang. 2021. “Parallel random matrix particle swarm

optimization scheduling algorithms with budget constraints on cloud computing systems." Applied Soft Computing 113: 107914.

Tirkolaee, E. B., A. Goli, and G. W. Weber. 2020. "Fuzzy mathematical programming and self-adaptive artificial fish swarm algorithm for just-in-time energy-aware flow shop scheduling problem with outsourcing option." IEEE transactions on fuzzy systems 28 (11): 2772-2783.

TSMC. 2021. "TSMC 2021 Sustainability Report." https://www.tsmc.com/english/aboutTSMC/dc_csr_report.

Uzsoy, R. 1994. "Scheduling a Single Batch Processing Machine with Non-identical Job Sizes." The International Journal of Production Research 32 (7): 1615-1635.

Van, D. R. D. J., A. Van Harten, and P. C. Schuur. 1997. "Dynamic job assignment heuristics for multi-server batch operations : a cost based approach." International Journal of Production Research 35 (11): 3063-3094.

Wang, J. Q., G. Q. Fan, Y. Zhang, C. W., and J. Y. T. Leung. 2016. "Two-agent scheduling on a single parallel-batching machine with equal processing time and non-identical job sizes." European Journal of Operational Research 258 (2): 478-490.

Wang, S., M. Liu, F. Chu, and C. Chu. 2016. "Bi-objective optimization of a single machine batch scheduling problem with energy cost consideration." Journal of Cleaner Production 137: 1205-1215.

Wang, Y. C., and J. M. Usher. 2005. "Application of reinforcement learning for agent-based production scheduling." Engineering Applications of Artificial Intelligence 18 (1): 73-82.

Waschneck, B., A. Reichstaller, L. Belzner, T. Altenmüller, and A. Kyek. "Optimization of global production scheduling with deep reinforcement learning." Procedia Cirp 72: 1264-1269.

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

Wu, P., J. Cheng, and F. Chu. 2021. "Large-scale energy-conscious bi-objective single-machine batch scheduling under time-of-use electricity tariffs via effective iterative heuristics." Annals of Operations Research 296 (1): 471-494.

Wu, Y. 2021. "A survey on population-based meta-heuristic algorithms for motion planning of aircraft." Swarm and Evolutionary Computation 62: 100844.

Xu, R., H. Chen, and X. Li. 2012. "Makespan minimization on single batch-processing machine via ant colony optimization." Computers and operations research 39 (3): 582-593.

Zeng, Z., M. Hong, Y. Man, J. Li, and Z. Yang. 2018. "Multi-object optimization of flexible flow shop scheduling with batch process - Consideration total electricity consumption and material wastage." Journal of Cleaner Production 183 (MAY 10): 925-939.

Zhang, G., Y. Hu, J. Sun, and W. Zhang. 2020. "An improved genetic algorithm for the flexible job shop scheduling problem with multiple time constraints." Swarm and Evolutionary Computation 54: 100664.

Zhang, H., F. Wu, and Z. Yang. 2021. "Hybrid approach for a single-batch-processing machine scheduling problem with a just-in-time objective and consideration of non-identical due dates of jobs." Computers and Operations Research 128: 105194.

Zhang, J., X. Yao, and Y. Li. 2020. "Improved evolutionary algorithm for parallel batch processing machine scheduling in additive manufacturing." International Journal of Production Research 58 (8): 2263-2282.

Zhang, S., A. Che, X. Wu, and C. Chu. 2017. "Improved mixed-integer linear programming model and heuristics for bi-objective single-machine batch scheduling with energy cost consideration."

Engineering Optimization 50 (8): 1380-1394.
Zhang, Z., W. Wang, S. Zhong, and H. U. Kaishun. 2013. “Flow shop scheduling with reinforcement learning.” Asia-Pacific Journal of Operational Research 30 (05): 1350014.
Zhang, Z., Z. Li, L. Na, W. Wang, and K.Hu. 2012. “Minimizing mean weighted tardiness in unrelated parallel machine scheduling with reinforcement learning.” Computers and operations research 39 (7): 1315-1324.
Zhao, F., L. Zhang, J. Cao, and J. Tang. 2020. “A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem.” Computers and Industrial Engineering 153: 107082.
Zhou, H., J. Pang, P. K. Chen, and F. D Chou. 2018. “A modified particle swarm optimization algorithm for a batch-processing machine scheduling problem with arbitrary release times and non-identical job sizes.” Computers and Industrial Engineering 123: 67-81.
Zhou, S., H. Chen, R. Xu, and X. Li. 2014. “Minimising makespan on a single batch processing machine with dynamic job arrivals and non-identical job sizes.” International Journal of Production Research 52 (8): 2258-2274.
Zhou, S., L. Xing, X. Zheng, N. Du, L. Wang, and Q. Zhang. 2021. “A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times.” IEEE transactions on cybernetics 51 (3): 1430-1442.
Zhou, S., M. Jin, and N. Du. 2020. “Energy-efficient scheduling of a single batch processing machine with dynamic job arrival times.” Energy 209: 118420.
Zhu, S., H. Hu, H. Yang, Y. Qu, and Y. Li. 2023. “Mini-Review of Best Practices for Greenhouse Gas Reduction in Singapore’s Semiconductor Industry.” Processes 11 (7): 2120.

Table A1. Comparison results between meta-heuristic algorithms with Dueling DQN

Instances		Algorithms												
		GA-BFD	IA-BFD	PSO-BFD	SA-BFD	DE-BFD	AFSA-BFD	GA-FFD	IA-FFD	PSO-FFD	SA-FFD	DE-FFD	AFSA-FFD	Dueling DQN
50	1	624	648.3	660.9	634	619.4	648.7	620.1	657	662.1	621	613.6	651.9	612.2
	2	647.3	667.5	686	650.9	641.4	664.9	640.1	652.7	677.2	670.8	634.6	683.3	634.7
	3	658.8	661.9	644.1	655.8	649.7	659.6	652.9	662.5	670.1	646.2	641.5	676.6	626.7
	4	611	640.1	623.2	622.5	609.4	642	620.9	620.9	647.5	613	599.9	650.7	581
	5	679.9	696	734.5	695.3	688.1	716	692	682.9	712.1	688.7	696.9	726.2	674.7
	6	478.9	516.6	542.1	489	493.8	522.1	508.1	522.2	547	524.1	497.2	530.3	458.3
	7	505.1	538	540.3	491.8	512	529.6	491.4	512	525.5	521.1	516.4	545.7	475.4
	8	546.8	556	565.4	560.9	541.4	557.3	528.1	556.1	560.7	548.5	556.1	552	510.5
	9	635.2	640.6	641	614.5	630.7	649.7	614.7	633.6	636.9	610.1	630.9	653.9	586.9
	10	668.6	701	695.9	669.8	675.9	697.8	659.8	695	679	660.9	685.4	709.3	634.4
100	1	1306.5	1376.1	1341.7	1299.5	1348.5	1387.7	1269	1300.1	1366.2	1315.7	1355.5	1400.8	1231.5
	2	1243	1367	1309.1	1252.3	1311.1	1371.8	1246.1	1346.1	1371.5	1286.7	1307.4	1367.6	1173.6
	3	1176.6	1238.8	1241.2	1158.6	1218.2	1278.1	1135.7	1198.4	1233.2	1175.1	1216.9	1264.8	1081.6
	4	1413.8	1432.6	1434.4	1354	1444.5	1460.6	1374.4	1462.6	1448.6	1377.9	1423.6	1458.7	1300.3
	5	1395.3	1384.3	1442	1388.3	1446.2	1454.1	1392.7	1453.5	1466.8	1379.4	1457.7	1517.5	1355.4
	6	1274.2	1351.3	1339	1254.4	1336.3	1367.8	1298.6	1322.2	1386.4	1297.6	1340.6	1380.2	1205.7
	7	1290	1320.6	1368.1	1272.5	1374.3	1397.1	1317.2	1400.7	1402.4	1360.6	1428.1	1397.3	1252.7
	8	1245.3	1272.9	1256.2	1212.2	1289.9	1285.8	1248	1258.5	1298.5	1236.8	1275.7	1347.9	1171.9
	9	1204.7	1261	1284.3	1192.9	1290.6	1311.2	1200.8	1244.7	1302.8	1217.8	1263.1	1295.3	1128.6
	10	1301.8	1330.9	1385.6	1340.4	1365.1	1383.3	1308	1403.9	1363.4	1305.3	1388.5	1425.9	1263.3
200	1	2452.7	2718.9	2662.7	2544.4	2732.1	2724.1	2470.3	2645.8	2704.7	2522	2717.6	2735.3	2245.3
	2	2452.8	2667.5	2671.1	2585.2	2717.6	2742.1	2489.8	2663.6	2625.9	2563.1	2714.8	2765.3	2275.4
	3	2483.6	2707.8	2686.8	2639	2697.8	2737.7	2460.6	2662.2	2668.6	2560.3	2711	2778.6	2263.7

	4	2702	2874.9	2912.8	2860	2900.5	2975.7	2701.9	2841.2	2948.4	2760.1	2934.5	3010.2	2489.7
	5	2459.4	2595.5	2663.5	2549.8	2681.6	2719.3	2503.6	2713.2	2622.1	2609.1	2684.7	2690.9	2276.3
	6	2600.1	2724.3	2716.6	2714	2824.5	2831.4	2588.4	2826.4	2836.7	2793.5	2858.8	2872.7	2370.8
	7	2687	2829.4	2875.6	2762.2	2937.7	2948.1	2734.9	2919.3	2940.1	2845.8	2961.8	2947.9	2526.7
	8	2439.3	2682.8	2638.4	2568.8	2733.1	2718.9	2454.8	2655.5	2687.1	2592.1	2708.4	2751.1	2229.2
	9	2359.4	2575.6	2555.6	2533.5	2582.2	2599.5	2387.7	2565.9	2557	2551.4	2619.5	2622.4	2193.4
	10	2536.2	2621.5	2686.9	2574.7	2738.4	2761.9	2518.8	2707	2659.7	2618.2	2720.1	2775.1	2300.3
300	**1**	4012.5	4401.7	4439.1	4299.5	4452.9	4482.7	4027	4388.1	4415.4	4415.2	4448.9	4515.4	3697.2
	2	3725.1	4069.3	4133.3	3968	4183.7	4184.2	3745	4100.7	4110	4107.6	4121.9	4184.5	3356.2
	3	3878.9	4171.1	4207.4	4194.9	4244.4	4269.1	3870.2	4166.2	4146.9	4187.2	4209.9	4257.2	3513.2
	4	3915.7	4360	4278.7	4236.9	4370.3	4397.8	3978.4	4284.5	4300.8	4236.9	4368.5	4386.6	3680.4
	5	3787.8	4053.7	4130	4066	4177.8	4192	3820.6	4142.8	4119.9	4027.2	4222	4239.9	3466.3
	6	3765.2	4109.7	4051.2	3993.4	4163.8	4097	3765.3	3996.5	3976	4059.1	4123.4	4120.9	3434.8
	7	3786.4	4166.7	4183.3	4088.9	4195.8	4126.2	3791.4	4087.2	4166.7	4192.4	4237	4230.4	3441.3
	8	3606.5	4056.9	3969.8	3901.2	4044.8	4042.8	3540.3	3903.1	3986.7	3953.7	4000	4071.1	3227.4
	9	4014.2	4314.5	4465.1	4322	4426.2	4413	4036.3	4302.8	4378.5	4269.9	4419.5	4415.9	3679
	10	3868.1	4233.5	4149.8	4176.9	4273.4	4278.5	3860.1	4236.9	4186.7	4257.4	4301.2	4314.3	3503.7

Table A2. Comparison results between heuristic algorithms with Dueling DQN

Instances		**Algorithms**											
		BFD-LPT	**BFD-RAND**	**BFD-LPS**	**BFD-SJS**	**FFD-LPT**	**FFD-RAND**	**FFD-LPS**	**FFD-SJS**	**RAND**	**LIMA-RAND**	**CPLEX**	**Dueling DQN**
50	**1**	612.2	712.6	657.4	788.7	693.4	677.3	657.4	788.7	627.5	612.2	600	612.2
	2	634.7	730.7	706.9	849.5	774	724.3	706.9	849.5	672.6	635.8	631.2	634.7
	3	640.2	683.9	716.4	821	747.3	693.8	716.4	821	656.7	629.3	625.9	626.7
	4	603.8	652.8	704.1	787.9	686	663.5	704.1	787.9	623.1	584.6	585.2	581

	5	700.9	760.9	797	952.3	786.2	755.9	797.9	908.3	707	670.2	649.7	674.7
	6	459.1	581.8	556.9	654.1	525.3	565.9	556.9	654.1	498.3	462.1	476.5	458.3
	7	488.1	573.5	545.6	677.9	571	548.8	545.6	677.9	508.8	477.2	465.8	475.4
	8	515.6	601	594.5	723.8	621	600.9	594.5	723.8	539.8	515.6	514.9	510.5
	9	589.6	686.8	691.8	890.7	791	686.1	695.1	890.7	638	594.5	593.6	586.9
	10	634.4	722.4	752	911.6	714.9	731.3	752	911.6	677.6	635.7	661.2	634.4
100	**1**	1225.7	1449.8	1460.9	1723.3	1438.6	1455.9	1444.1	1723.3	1333.7	1244.8	1296.3	1231.5
	2	1182.2	1406.2	1362.7	1627.9	1418.9	1416.5	1360.7	1627.9	1276.8	1192.3	1269.6	1173.6
	3	1091	1330.3	1280.8	1551.4	1268.5	1317.5	1258.7	1551.4	1165.1	1098.9	1171	1081.6
	4	1300.5	1537	1461.3	1800.5	1463	1541.9	1490.8	1800.5	1403.5	1313.7	1414.9	1300.3
	5	1355.4	1540.9	1499.1	1858.6	1506.9	1532.4	1497.7	1858.6	1443.5	1356.6	1383.2	1355.4
	6	1227.9	1431.5	1467.4	1701.1	1411.1	1423.6	1467.4	1701.1	1332.6	1240.7	1317	1205.7
	7	1253.4	1507.4	1450.5	1847.5	1418.7	1470.5	1450.5	1847.5	1367	1262.3	1366.1	1252.7
	8	1164.6	1369.7	1299.1	1600.9	1384.2	1378.8	1297.5	1600.9	1259.4	1177.6	1238.2	1171.9
	9	1129.7	1354.6	1330.8	1632.7	1321.9	1341.5	1319.6	1632.7	1226.5	1131.3	1194.2	1128.6
	10	1263.3	1456.1	1490.1	1794.6	1506.4	1463.3	1469.8	1794.6	1372.1	1276	1358.2	1263.3
200	**1**	2254.5	2835.1	2631.9	3277.3	2746.6	2846.7	2619.4	3265.6	2555.9	2278.4	2885.6	2245.3
	2	2290.4	2813.4	2694.3	3301.9	2717	2799.8	2661.3	3301.9	2547.1	2298.3	2855.5	2275.4
	3	2279.2	2845.3	2653.3	3252.3	2735.9	2858.9	2614	3252.3	2540.8	2306	2937.1	2263.7
	4	2505.8	3049.4	2888.3	3623.1	3123.2	3049.2	2862	3623.1	2792.2	2535.9	3082.5	2489.7
	5	2280.3	2826.6	2690.8	3298.8	2717.6	2799.5	2652.7	3298.8	2537.2	2304.6	2882.5	2276.3
	6	2388.3	2952	2745.6	3457.3	2956.5	2975.6	2744.5	3422.7	2678.2	2413.1	3051.2	2370.8
	7	2530.7	3055.2	2923.1	3541.7	3155.6	3097.1	2910.4	3521.2	2828.5	2573.1	3136.1	2526.7
	8	2235.3	2802.4	2666.3	3404.3	2691.1	2829.9	2649	3404.3	2505.3	2267.7	2865.1	2229.2
	9	2190.4	2701.7	2618.8	3236.9	2601	2704.1	2618.8	3236.9	2451.7	2227.3	2612.4	2193.4

	10	2300.7	2803.8	2719.5	3354.3	2809.6	2848.8	2722.1	3354.3	2615.8	2379.2	2926.3	2300.3
300	**1**	3688.9	4583.5	4361.7	5253.7	4324.1	4585.9	4368.2	5253.7	4177	3761.3	4757.2	3697.2
	2	3373.2	4314.7	4004.2	4978.5	3989.4	4320.4	3986.9	4984.9	3796.7	3430.7	4396.1	3356.2
	3	3514.4	4371.9	4081.8	5078.9	4250.9	4359.6	4047.1	5078.9	3984.7	3574.3	4612.5	3513.2
	4	3648.3	4496.9	4380.9	5244.8	4303	4523.8	4327.4	5225	4073.3	3694.3	4638.1	3680.4
	5	3416	4310.8	4028.5	5095.5	4216.6	4346.7	3998.8	5024.7	3924.3	3484.5	4401.1	3466.3
	6	3431.9	4275.1	4016	4903.4	4036.7	4297.2	3967.5	4891.7	3874.3	3477.1	4342.8	3434.8
	7	3453.1	4361.7	4069.4	5072.2	4161.9	4361.7	4041.8	5058.9	3965.7	3505.3	4513.7	3441.3
	8	3228.4	4215.4	3765	4789.4	3914.3	4199.7	3767.2	4772	3706.1	3268.8	4312	3227.4
	9	3664.4	4534.2	4329.1	5363.9	4365.3	4590.2	4283.8	5363.9	4172.4	3753.8	4602.5	3679
	10	3522.3	4373.7	4240.8	5179	4169.1	4376.6	4232.6	5163.6	3989.8	3578.7	4559	3503.7