ORIGINAL RESEARCH **Open access**

BiRNN-Based TD(λ) Algorithm for Energy-Efficient Non-Permutation Flow-Shop Scheduling

Nguyen Van Nam[1], Tran Thi Hoa[1*], Le Minh Duc[2]

Abstract

This study presents an innovative algorithm that integrates temporal difference with bidirectional recurrent neural networks (BiRNNs) to address the multi-objective, energy-efficient non-permutation flow-shop scheduling problem (NFSP). The primary goals of this challenge are to reduce total energy usage and makespan simultaneously. Initially, a mathematical model representing the energy-efficient NFSP is established. The NFSP is then framed as a Markov decision process, featuring a defined action space containing 28 distinct scheduling rules. Taking into account the NFSP's local and global characteristics, a set of 15 specific state features is extracted. Distinct reward functions are subsequently formulated to align with each target objective. Additionally, a BiRNN-based multi-layer perceptron model is employed to extract these NFSP state features. Multiple scheduling policies are produced by applying the TD(λ) algorithm to compute the state value function. To assess the algorithm's performance, a novel test suite for the energy-efficient NFSP is developed, expanding upon standard benchmark problems. Ultimately, comparative evaluations are carried out to validate the efficiency and overall effectiveness of the proposed algorithm.

Keywords Algorithm, BiRNN-Based TD(λ) Algorithm, Energy-efficient, NFSP

*Correspondence:
Tran Thi Hoa
hoa.tran@hust.edu.vn
[1] Department of Intelligent Power Systems, School of Electrical Engineering, Hanoi University of Science and Technology, Hanoi, Vietnam
[2] Department of AI-Based Grid Optimization, Faculty of Engineering, Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam

Introduction

Recently, energy efficiency within flow-shop scheduling has garnered substantial attention, reflecting the growing worldwide commitment to environmental accountability and sustainability [1]. Sustainable, or green, manufacturing seeks to mitigate the ecological footprint of industrial processes without sacrificing productivity or operational efficiency. Investigating energy-efficient scheduling on the shop floor supports the overarching objectives of green manufacturing, fostering a more ecologically responsible industrial landscape [2]. Consequently, tackling the dual challenges of lowering energy use and limiting environmental damage during manufacturing operations has become critically urgent [3].

Generally, the flow-shop scheduling problem is recognized as an NP-hard challenge in most scenarios [4]. While numerous algorithms have been developed to solve it, substantial opportunities remain to improve both the quality and efficiency of the solutions. Zhang and Dietterich [5] initially introduced the use of reinforcement learning (RL) for shop scheduling, sparking an expanding volume of research in this area. Moreover, swift advancements in machine learning have laid the theoretical groundwork for applying deep reinforcement learning (DRL) techniques to flow-shop scheduling tasks [6].

This is an open access article distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (CC BY-NC-SA 4.0), which permits non-commercial use, distribution, reproduction, and adaptation in any medium or format, provided that appropriate credit is given to the original author(s) and source, a link to the license is provided, and any changes made are indicated. Adapted material must be distributed under the same license.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

As intelligent manufacturing evolves, DRL approaches have emerged as vital elements of smart shop scheduling. Currently, the flow-shop scheduling problem is primarily addressed through three techniques: exact algorithms, heuristic algorithms, and metaheuristic algorithms. DRL methods, however, learn from historical data, using feature states as neural network inputs to formulate corresponding action policies and yield diverse scheduling strategies. This methodology effectively simulates the dynamic and complex nature of actual manufacturing environments [7]. Thus, exploring energy-efficient flow-shop scheduling through the lens of DRL holds immense value for intelligent manufacturing.

Building upon prior investigations, this research offers substantial contributions and novel elements, which can be summarized as follows:

(1) Bidirectional recurrent neural networks-temporal difference (BiRNNs-TD(λ)) algorithm for energy-efficient non-permutation flow-shop scheduling problem (NFSP): Significantly, this research represents the initial deployment of BiRNNs merged with the TD(λ) algorithm to tackle the energy-efficient NFSP. Incorporating BiRNNs enables our algorithm to identify contextual information and long-term dependencies, thereby enhancing the model's capacity to generate informed scheduling choices. The TD(λ) component allows the system to glean insights from past experiences and iteratively optimize scheduling policies, yielding efficient and adaptable solutions. By synthesizing these elements into a unique algorithmic structure, the proposed method delivers a robust and holistic approach to solving the multi-objective, energy-efficient NFSP.

(2) Direct application of DRL to NFSP: Unlike earlier studies that integrate DRL with genetic algorithms, this work translates the NFSP directly into a Markov decision process (MDP) and resolves it via a DRL approach. To accomplish this transformation, an action space featuring 28 standard scheduling rules is established as the NFSP input. Moreover, 15 state features are isolated to represent both the local and global attributes of the NFSP. Separate reward functions are also tailored to correspond with the specific goals.

(3) Mathematical model and new test set: A detailed mathematical model is formulated to support the comprehensive study of the energy-efficient NFSP. Utilizing this framework, a novel test suite is generated by expanding upon traditional benchmark problems. Extensive experiments are executed across instances of diverse sizes, illustrating the high performance and efficacy of the introduced algorithm.

Literature Review

During the last ten years, operations research has seen profound progress in machine learning, notably in RL and DL, which have both been extensively applied to solve shop scheduling issues. Nevertheless, the NFSP has received minimal focus in current literature. Therefore, this section seeks to address this gap by investigating the use of DL, RL, and their combined approaches in alternative shop scheduling contexts. By reviewing pertinent research in these areas, we can extract valuable insights and utilize established knowledge to tackle the distinct obstacles associated with the NFSP.

Within the DL domain, artificial neural networks (ANNs) have emerged as highly effective tools for modeling shop scheduling scenarios. Scholars have successfully capitalized on the predictive strengths, parallelism, and resilience of ANNs to overcome these operational hurdles [8]. Specifically, research has frequently utilized ANNs optimized via the standard gradient descent back-propagation algorithm to formulate and resolve shop scheduling tasks. These techniques have yielded encouraging outcomes and are now broadly accepted within the academic community. Applying ANNs in this manner allows investigators to map the intricate dependencies and correlations typical of shop scheduling problems, laying the groundwork for superior and more efficient scheduling strategies.

Liu *et al*. [9] constructed an actor–critic network model utilizing fully connected and convolutional layers. This model not only competes strongly on static benchmark tests but also achieves an effective balance between makespan and execution time under dynamic conditions. Mawson and Hughes [10] suggested using deep neural networks—such as

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

recurrent and feedforward models—to predict workshop environments and energy usage in manufacturing plants, achieving high forecasting accuracy. Intending to reduce makespan in extensive flexible flow shops, Li *et al*. [11] introduced a bilevel framework comprising a double deep Q network and a graph pointer network. This architecture can rapidly generate solutions for massive problems and performs on par with several sophisticated heuristic techniques. ANNs demonstrate distinct benefits in intricate, dynamic job shops possessing extensive historical data. Tian and Zhang [12] presented a model for dynamic job shop scenarios utilizing a long short-term memory network. Additionally, they engineered a data forecasting model for dynamic job-shop scheduling based on DL. Experimental results ultimately proved the proposed model to be highly effective.

Conventional approaches to shop scheduling problems frequently encounter constraints and disadvantages, largely stemming from the high dimensionality and intrinsic complexity of these operational challenges. Acknowledging these hurdles, scholars have shifted their focus toward innovative methodologies, with reinforcement learning (RL) emerging as a prominent solution [13]. RL techniques have attracted substantial interest because of their exceptional decision-making properties, which demonstrate significant potential in managing the intricacies of shop scheduling. Utilizing RL principles, investigators strive to engineer more proficient and effective strategies capable of adapting to the fluctuating conditions of shop scheduling environments. These contemporary techniques provide an alternative viewpoint and can bypass the limitations of classic methods, paving the way for further advancements in shop scheduling research [14].

Regarding dynamic job-shop scheduling scenarios involving machine breakdowns and unpredictable job arrivals, Shahrabi *et al*. [15] introduced an RL framework grounded on the Q-factor. Additionally, they developed a novel technique to compute the reward value based on the quality of parameters chosen during the learning phase. Experimental outcomes indicated that their suggested algorithm outperformed standard variable neighbourhood search techniques and traditional dispatching rules. Luo *et al*. [16] tackled energy-efficient heterogeneous distributed assembly permutation flow-shop scheduling, incorporating product priorities and presented a Q-learning memetic algorithm designed to minimize both overall energy usage and total tardiness. Ren *et al*. [17] formulated an RL model aimed at resolving the flow-shop scheduling problem. They established a value function for this problem by translating the minimization of makespan into the reduction of idle time. Ultimately, their model achieved favourable results when tested against several benchmark instances. For the multi-objective distributed assembly no-wait flow shop scheduling problem, Zhao *et al.* [18] presented a brainstorming optimization concept driven by Q-learning. Within this algorithm, resource distribution across large-scale instances was stabilized via a learning procedure grounded in a clustering mechanism.

When the extraordinary learning proficiency of deep learning (DL) is united with the superior decision-making strengths of RL, the resulting performance in resolving shop scheduling problems is highly promising. This symbiotic integration harnesses the distinct advantages of both methodologies, facilitating more highly productive and streamlined solutions [19]. Xiao *et al*. [20] developed a deep reinforcement learning (DRL) algorithm grounded in the DQN method for the NFSP. This algorithm creatively substituted off-policy Q-learning with an on-policy TD-learning approach. To minimize makespan, the proposed framework was tested against benchmark problems, demonstrating superior efficacy compared to alternative algorithms. To address the intricate and dynamic manufacturing settings found in job shops, Han and Yang [21] introduced a duelling double DQN architecture. This structure merges the flexibility and real-time responsiveness of convolutional neural networks (CNNs) with RL. The algorithm's validity was confirmed using 85 instances from the widely recognized OR-Library. Utilizing DRL, Pan *et al.* [22] engineered a robust optimization algorithm tailored for the permutation flow-shop scheduling problem (PFSP). Their customized neural network bypassed the traditional end-to-end output constraints and utilized RL training to secure enhanced outcomes. Song *et al*. [23] designed an innovative DRL method to deduce high-quality priority scheduling rules for flexible job-shop scheduling problems. By merging machine allocation and operation selection into a unified decision-making process, their experimental findings revealed that the novel algorithm outperformed conventional priority dispatching rules.

The preceding literature review underscores that most DRL-focused research on shop scheduling predominantly targets minimizing makespan as the primary optimization goal. However, within actual manufacturing environments, energy utilization is a critical priority on par with production efficiency [24]. Interestingly, very few investigations have examined

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

the deployment of DRL in energy-efficient shop scheduling, particularly concerning non-permutation flow-shop scenarios. To bridge this gap in the literature, the present research proposes an innovative DRL-oriented algorithm expressly customized for the energy-efficient NFSP.

Problem Description and Formulation

Non-permutation flow-shop

The flow-shop scheduling problem occupies a crucial position within manufacturing management. It can be outlined as follows [25]: a collection of jobs, designated as N = {1, 2, 3, …, n}, requires processing across a specified group of machines, indicated as M = {1, 2, 3, …, m}. Every individual job entails a sequence of m operations that must be executed on the appropriate machinery. Crucially, there are no precedence constraints among the various jobs. If all jobs are required to proceed in an identical sequence on every machine, the configuration is identified as the PFSP. Conversely, if the operational sequence of jobs across the machines is variable and not pre-established, the scenario is classified as the NFSP [26]. Importantly, the NFSP features a considerably broader and more intricate solution space than the PFSP. As outlined by Rossit *et al*. [27], several standard assumptions are typically adopted for the NFSP:

(1) A machine can process merely one job at any given moment.

(2) Only a single machine can operate on a job at a time.

(3) Both machines and jobs are permitted to experience idle time while waiting.

(4) The capacity of the intermediate buffer is ample enough to allow for the reorganization of the job sequence.

The description provided above acts as a broad framework for the NFSP, though it does not necessarily cover every potential variation [28]. By adding or removing particular constraints, diverse classifications of the NFSP can be generated. For this investigation, alongside the four core premises previously detailed, we introduce the subsequent supplementary assumptions:

(1) At the commencement of the scheduling operation, all machines and jobs are in an idle state.

(2) Preemption among the jobs is strictly prohibited.

(3) Every job has a pre-established processing duration on each piece of equipment, indicating the exact time required to complete its operations.

(4) The processing parameters, such as machine specifications and job details, are determined before execution.

(5) The manufacturing process operates continuously and without disruption, guaranteeing a fluid sequence of operations free from pauses or halts.

Collectively, these assumptions establish the precise parameters and constraints of the specific NFSP variation examined in this research, guaranteeing that the mathematical problem formulation accurately reflects the interdependencies and trade-offs among the disparate objectives.

Multi-objective optimisation model

Multi-objective optimisation problem

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

Typically, a multi-objective optimization framework featuring k distinct targets can be formulated as **Eq. 1**, following the definitions provided by Yenisey and Yagmahan [29]:

$$F(x) = min\{f_1(x), f_2(x), \ldots, f_k(x)\}$$
$$x = (x_1, x_2, \ldots, x_n)^T \in X, F(x) \in Y \quad (1)$$

Wherein f1(x), f2(x), …, fk(x) denote the k objective functions under optimization, x represents the decision vector, X signifies the decision space, y corresponds to the objective vector, and Y constitutes the target space.

Addressing the NFSP requires identifying the ideal execution order of all jobs across every machine to minimize the designated objective criteria. For scheduling challenges involving multiple goals, the primary aim is generally to determine the Pareto front. This front comprises a collection of solutions where advancing one objective inevitably degrades performance in at least one other objective.

A given strategy x1 is said to strongly Pareto dominate an alternative strategy x2 if the subsequent conditions (2) and (3) hold:

$$\forall i \in \{1, 2, \ldots, k\}, f_i(x_1) \leq f_i(x_2) \quad (2)$$

$$\exists j \in \{1, 2, \ldots, k\}, f_j(x_1) < f_j(x_2) \quad (3)$$

A strategy x1 qualifies as Pareto efficient if no competing strategy exists that can simultaneously surpass its performance across all objectives. The aggregation of these Pareto-efficient strategies constitutes the Pareto set. Conversely, the Pareto front is established by projecting this Pareto set onto the target space, illustrating the ideal compromises among conflicting objectives and delivering a holistic overview of top-tier solutions within the multi-objective optimization context.

Notations

Indices	
i	Index of machines. ($1 \leq i \leq m$)
j	Index of jobs. ($1 \leq j \leq n$)
k	Index of operations. ($1 \leq k \leq m$)
Sets	
J	Set of jobs, J = {; ; …}. It contains a batch of flow-shop jobs
M	Set of machines, M = {; …; }
O_i	Set of operations, = {; ; …; }
Parameters	
m	Total number of machines in the process
n	Total number of jobs to be scheduled
o_{jk}	The kth operation of job j_j

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

MT_{ijk}	The processing time of operation o_{jk} on machine m_i
MT_j	The total processing time of job j_j
W_{ijk}	The waiting time of job j_j before processing on machine m_i
W_j	The total waiting time of job j_j on all machines
C_j	The makespan of job j_j
ET_{jk}	The end time of operation o_{jk}
ES_i	The energy consumption for machine m_i startup
EC_i	The energy consumption for machine m_i shutdown
EW_{ijk}	The energy consumption per unit time during operation o_{jk} processing on machine m_i
EN_i	The energy consumption per unit time when machine m_i is idle
Decision variables	
α_{ijk}	Take value 1 if operation o_{jk} is processed on machine m_i; otherwise, 0
L_{ij}	Take value 1 if the jth job is processed on machine m_i; otherwise, 0
S_{ijk}	Take value 1 if the jth job occupies the kth position on machine m_i; otherwise, 0

Model formulation

Utilizing the previously established notations, the mathematical framework for the energy-conscious NFSP is constructed as follows:

$$\begin{aligned} minC_{max} &= \max_{j=1}^{n} C_j = \max_{j=1}^{n}(M_j + W_j) \\ &= \max_{j=1}^{n} \sum_{i=1}^{m}(\alpha_{ijk} \times MT_{ijk} + \alpha_{ijk} \times W_{ijk}) \end{aligned} \tag{4}$$

$$\begin{aligned} minE = &\sum_{i=1}^{m}(ES_i + EC_i) + \sum_{j=1}^{n}\sum_{i=1}^{m}(\alpha_{ijk} \times MT_{ijk} \times EW_{ijk}) \\ &+ \sum_{i=1}^{m}\left\{EN_i \times \left[\max_{j=1,k=1}^{j=n,k=m}(\alpha_{ijk} \times ET_{jk})\right.\right. \\ &\left.\left. - \sum_{j=1}^{n}\sum_{k=1}^{m}(\alpha_{ijk} \times MT_{ijk})\right]\right\} \end{aligned} \tag{5}$$

s.t.

$$C_{max} \geq Ci, \forall i; \tag{6}$$

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

$$\sum_{m\in M}\alpha_{ijk}=1,\forall i,j,k; \quad (7)$$

$$C_1\geq\sum_{m\in M}\sum_{o\in O}(\alpha_{ijk}\times MT_{ijk}),\forall i,k \quad (8)$$

$$C_j\geq C_{j-1}+\sum_{m\in M}\sum_{o\in O}(\alpha_{ijk}\times MT_{ijk}),\forall i,k,\forall j\neq 1 \quad (9)$$

$$\sum_{i=1}^{m}L_{ij}=1,\forall j\leq n \quad (10)$$

$$\sum_{j=1}^{n}L_{ij}=1,\forall i\leq m \quad (11)$$

$$ET_{jk}\leq ET_{j,k+1}-\alpha_{ijk}\times MT_{ijk},\forall i,j,k \quad (12)$$

$$\sum_{i=1}^{m}S_{ijk}=1,\forall j\leq n \quad (13)$$

$$\sum_{j=1}^{n}S_{ijk}=1,\forall i\leq m \quad (14)$$

The initial objective function presented in **Eq. 4** is designed to minimize the total makespan, which represents the overall completion duration for the entire set of jobs. The secondary objective function detailed in **Eq. 5** targets the reduction of the aggregate energy consumption across all machinery. This total power footprint comprises three distinct elements: the energy consumed during job processing, the standby energy expended. At the same time, machines are idle, and the electricity required for machine startup and shutdown operations.

Constraint (6) ensures that the total makespan aligns with the finalization time of the terminal operation on the prior machine. Constraint (7) dictates that an individual operation can only be executed on a single machine at a time. Constraints (8-9) guarantee that the calculated makespan respects the foundational flow-shop configuration. Constraints (10-11) enforce the restriction that a job cannot occupy multiple machines concurrently, and similarly, a single machine can only process one job at any given moment. Constraint (12) highlights that a subsequent operation within a flow-shop job cannot commence until its immediate predecessor has been fully executed. Finally, Constraints (13-14) govern the processing sequence of the flow-shop jobs.

In practical manufacturing, flow-shop settings are inherently dynamic and complex. Unplanned events frequently arise—such as equipment breakdowns, tool switchouts, and operational delays—which introduce further complications into the production workflow. Quantifying the precise total energy consumption within a flow shop is an intricate task because it involves numerous power-consuming factors, including production machinery, facility lighting, ventilation fans, and material transport mechanisms. Although calculating the complete energy footprint of an entire shop floor is exceptionally difficult, the electricity consumed directly by the processing machines represents the vast majority of the total energy usage in a flow-shop context.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

Given these practical circumstances, this investigation centers on an NFSP involving m machines and n jobs. The production workflow is configured to optimize two criteria simultaneously: minimizing the total makespan and reducing energy use. For the energy consumption objective, the analysis is confined exclusively to machine-related electricity consumption, as it constitutes the primary share of power demands within the flow-shop environment.

Multi-Objective Optimisation Algorithm Based on DRL

Conversion of scheduling problems

A fundamental challenge when applying DRL to production scheduling involves transforming the operational problem into a valid mathematical structure, such as an MDP or a semi-MDP. For the NFSP, this conversion process rests on three primary architectural elements:

(1) Construction of action space: The initial phase involves defining the action space for the machinery based on the decision variables that serve as inputs for the target NFSP. This is accomplished by leveraging traditional heuristic rules widely utilized in flow-shop scheduling, which act as guidelines to establish the available actions for each machine.

(2) Extraction of state features: The subsequent phase requires isolating relevant state characteristics from the manufacturing environment, keeping the unique properties of the flow shop in mind. These state features capture the real-time status of the system, offering indispensable data for automated decision-making. This extraction process is tailored to reflect the practical conditions of the active shop floor.

(3) Definition of reward functions: To steer the policy learning process, appropriate reward structures must be formulated based on the objective functions of the underlying scheduling problem. These reward functions dictate the relative benefit or penalty of executing a particular action within a given state. The mathematical design of these rewards is explicitly tied to the optimization targets, providing the essential reinforcement signals that enable the DRL algorithm to learn and enhance its decision-making proficiency.

Construction of action space

At its core, resolving a flow-shop scheduling problem requires making systematic choices regarding the sequencing of jobs across every machine and identifying the ideal starting times for their processing. In parallel, within an MDP framework, it is vital to determine the explicit actions chosen by each machine during the state transition $st+1 \leftarrow st$. Consequently, formulating an appropriate action space stands as a foundational requirement.

Traditionally, the design of this action space depends on heuristic dispatching rules. While such rules can produce optimal schedules for small-scale problem configurations, they frequently display a short-sighted approach when scaled to larger scenarios, culminating in low-quality, suboptimal solutions. Conversely, DRL capitalizes on an ongoing estimation of the state value function, denoted as $V(S)$, which maintains a positive correlation with long-term cumulative rewards. By selecting actions guided by this expected value, the DRL agent can overcome the localized, short-sighted flaws of heuristic scheduling rules, thereby securing superior performance.

In this study, we design an action space comprising 28 frequently utilized scheduling rules to serve as inputs for the considered NFSP, encompassing elementary constructive heuristics alongside their hybrid combinations [30]. The exact configuration of this action space is mapped out in **Table 1.** Following the principles of the Johnson–Bellman rule, the waiting queue Qi ($1 \le i \le m - 1$) is split into two distinct sub-collections, SJi,1 and SJi,2. Furthermore, Condition 1 is formally expressed as $\exists 2 \le i \le m, Jj \in Qi: pi-1,k + pi,k < pi,j$.

Table 1. Action space.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

No.	Rules	Description
1	FCFS	First come, first served. Prioritise the first job in queue Q_i
2	JB1	Select the job in the sorted job list $SJ_{i,1}$ according to SPT. $SJi;1 = \{J_j \mid p_{i;j} \le p_{i+1,j}\ ,\ J_j\ \ Q_i\ \}$
3	JB2	Select the job in the sorted job list $SJ_{i,2}$ according to LPT. $SJ_{i,2} = \{J_j \mid p_{i,j} > p_{i+1,j},\ J_j \in Q_i\}$
4	JB3	When condition 1 is true, select the job with the minimum $p_{A,j}$ from the sorted job list $SJ_{i,3}$. $p_{A,j} = p_{i,j} + p_{i+1,j}$, $SJ_{i,3} = \{J_j \mid p_{i,j} \ge \max\{p_i{+}1_{,j}\},\ J_j \in\ Q_i\ \}$
5	JB4	When condition 1 is true, select the job with the maximum $p_{B,j}$ from the sorted job list $SJ_{i,4}$. $p_{B,j} = p_{i+1,j} + p_{i+2,j}$, $SJ_{i,4} = \{J_j \mid p_{i+2,j} \ge \max\{p_{i+1,j}\},\ J_j \in Q_i\}$
6	SPT	Shortest Processing Time rule. The job with the minimum processing time is prioritized.
7	LPT	Longest Processing Time rule. The job with the maximum processing time is prioritized.
8	LWKR	Least Work Remaining. Prioritize the job with the smallest remaining processing load.
9	MWKR	Most Work Remaining. Prioritize the job with the largest remaining processing load.
10	SPT/TWK	Select the job that minimizes the ratio of its operation time to the total processing time.
11	LPT/TWK	Select the job that maximizes the ratio of its operation time to the total processing time.
12	SPT/TWKR	Choose the job with the minimum ratio of its operation time to its remaining processing time.
13	LPT/TWKR	Choose the job with the maximum ratio of its operation time to its remaining processing time.
14	SPT × TWK	Prioritize the job with the smallest product of operation time and total processing time.
15	LPT × TWK	Prioritize the job with the largest product of operation time and total processing time.
16	SPT × TWKR	Prioritize the job with the smallest product of operation time and total remaining processing time.
17	LPT × TWKR	Prioritize the job with the largest product of operation time and total remaining processing time.
18	SRM	Shortest Remaining Machining rule. Select the job with the minimum remaining machining time, excluding the current operation.
19	LRM	Longest Remaining Machining rule. Select the job with the maximum remaining machining time, excluding the current operation.
20	SSO	Shortest Subsequent Operation. Choose the job with the smallest duration of the next operation.
21	LSO	Longest Subsequent Operation. Choose the job with the largest duration of the next operation.
22	SPT + SSO	Select the job that minimizes the sum of current operation time and the next operation time.
23	LPT + LSO	Select the job that maximizes the sum of current operation time and the next operation time.
24	SPT/SSO	Choose the job that minimizes the ratio of current operation time to subsequent operation time.
25	LPT/LSO	Choose the job that maximizes the ratio of current operation time to subsequent operation time.
26	SSJF	Shortest Sum Job First. Select the job with the smallest total processing time from the first operation to the second-to-last operation.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

27	LSJF	Longest Sum Job First. Select the job with the largest total processing time from the first operation to the second-to-last operation.
28	-	The machine is currently idle.

Extraction of state features

In the context of scheduling optimizations, isolating state characteristics within the DRL framework is closely linked to defining the action space. Based on the practical conditions of the flow shop, a set of 15 quantitative state features is derived for every machine [31]. Generally, these state characteristics fulfill the following criteria:

(1) The state characteristics must encapsulate the core variables of the flow shop and accurately mirror ongoing environmental variations, spanning both global metrics and local metrics.

(2) Every state feature must be contained within a unified, standard feature collection.

(3) The state metrics should possess sufficient generality to be applicable across diverse scheduling scenarios.

(4) State indicators must be quantifiable, with distinct numerical values signifying unique environmental conditions.

(5) The computation of these state features should be straightforward and computationally efficient.

(6) Normalized state indicators ought to generalize across varying scheduling problems using a relatively standardized scale.

Meeting these conditions ensures that the isolated state features precisely capture the critical data required by the DRL model, thereby supporting swift and reliable decision-making throughout the scheduling cycle. Let fi,s represent the sth state feature of the ith machine within the flow shop; the individual state metrics designated for each piece of equipment are outlined in **Table 2.**

Table 2. State features for each machine.

Machine m_i	State features
$1 \le i \le m-2$	10 real-valued features $f_{i,k}$ $(1 \le k \le 10)$
	5 binary features $f_{i,k}$ $(11 \le k \le 15)$
$i = m-1$	8 real-valued features $f_{m\text{-}1,k}$ $(1 \le k \le 10)$
	3 binary features $f_{m\text{-}1,k}$ $(k = 11, 12, 15)$
$i = m$ (the last machine)	8 real-valued features $f_{m,k}$ $(1 \le k \le 10)$
	1 binary feature $f_{m,k}$ $(k = 15)$

Detailed formulations for the 15 state characteristics are articulated below in **Eqs. 15-29**.

fi,1 represents the proportion of total jobs currently residing in queue Qi, reflecting how jobs are distributed across the various machines.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

$$f_{i,1} = \frac{|Q_i|}{n}, 1 \le i \le m \quad (15)$$

fi,2 corresponds to the ratio of the mean processing duration of all jobs in queue Qi on machine mi relative to pi, signifying the active workload allocated to each machine.

$$f_{i,2} = \frac{\sum_{j \in Q_i} p_{i,j}}{p_i |Q_i|}, 1 \le i \le m \text{ and } Q_i \ne \emptyset, p_i = \frac{\sum_{i=1}^{n} p_{i,j}}{n} \quad (16)$$

Where pi denotes the overall mean processing duration across all jobs.

fi,3 is defined as the ratio of the average processing time of all jobs within the initial i queues Qk on machine mi to pi, which characterizes the residual workload for each machine.

$$f_{i,3} = \frac{\sum_{k=1}^{i} \sum_{j \in Q_k} p_{i,j}}{p_i \sum_{k=1}^{i} |Q_k|}, 1 \le i \le m \text{ and } Q_i \ne \emptyset \quad (17)$$

fi,4 represents the ratio of the longest processing duration among all jobs in queue Qi on machine mi to pi, highlighting the maximum time requirement within the current queue.

$$f_{i,4} = \max_{j \in Q_i} \{p_{i,j}\} / p_i, 1 \le i \le m \text{ and } Q_i \ne \emptyset \quad (18)$$

fi,5 expresses the ratio of the shortest processing duration for all jobs in queue Qi on machine mi relative to pi, indicating the minimum time requirement in the active queue.

$$f_{i,5} = \min_{j \in Q_i} \{p_{i,j}\} / p_i, 1 \le i \le m \text{ and } Q_i \ne \emptyset \quad (19)$$

fi,6 signifies the ratio of the residual processing duration of the work in progress (WIP) on machine mi to pi, serving as an indicator of whether machine mi is occupied or disengaged. This metric resolves to 0 if machine mi is unengaged.

$$f_{i,6} = \frac{p_{i,WIP} - pt_{i,WIP}}{p_i}, 1 \le i \le m \quad (20)$$

Where $pt_{i,WIP}$ denotes the duration for which the WIP has already been processed on machine mi.

fi,7 corresponds to the ratio between the longest residual processing duration of all jobs in queue Qi and the absolute maximum processing duration across all jobs, indicating the upper limit of processing time for queued jobs.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

$$f_{i,7} = \frac{\max\limits_{j \in Q_i}\left\{\sum\limits_{k=i}^{m} p_{k,j}\right\}}{\max\limits_{1 \le j \le n}\left\{\sum\limits_{k=i}^{m} p_{k,j}\right\}}, 1 \le i \le m \text{ and } Q_i \neq \varnothing \quad (21)$$

fi,8 evaluates the ratio of the shortest residual processing duration of all jobs in queue Qi to the maximum processing duration of the entire job set, illustrating the lower bound of processing time for waiting jobs.

$$f_{i,8} = \frac{\min\limits_{j \in Q_i}\left\{\sum\limits_{k=i}^{m} p_{k,j}\right\}}{\max\limits_{1 \le j \le n}\left\{\sum\limits_{k=i}^{m} p_{k,j}\right\}}, 1 \le i \le m \text{ and } Q_i \neq \varnothing \quad (22)$$

fi,9 represents the ratio of the peak Ri,j value within queue Qi compared to the peak value found across all jobs, providing an index of the relative maximum processing duration of queued jobs.

$$f_{i,9} = \frac{\max\limits_{j \in Q_i} R_{i,j}}{\max\limits_{1 \le j \le n} R_{i,j}}, 1 \le i \le m \text{ and } Q_i \neq \varnothing, R_{i,j} = \frac{p_{i,j}}{p_{i+1,j}} \quad (23)$$

Where Ri,j denotes the ratio of processing time on machine mi to that on machine mi+1.

fi,10 calculates the ratio of the lowest Ri,j value within queue Qi to the absolute highest Ri,j value among all jobs, which indicates the relative minimum processing duration of waiting jobs.

$$f_{i,10} = \frac{\min\limits_{j \in Q_i} R_{i,j}}{\max\limits_{1 \le j \le n} R_{i,j}}, 1 \le i \le m \text{ and } Q_i \neq \varnothing \quad (24)$$

fi,11 specifies whether machine mi is eligible to execute action JB1, returning 1 if the machine can implement action JB1.

$$f_{i,11} = \begin{cases} 1, & SJ_{i,1} \neq \varnothing \\ 0, & \text{otherwise} \end{cases}, 1 \le i \le m-1 \quad (25)$$

fi,12 indicates whether machine mi is permitted to execute action JB2, returning a value of 1 if machine mi can deploy action JB2.

$$f_{i,12} = \begin{cases} 1, & SJ_{i,2} \neq \varnothing \\ 0, & \text{otherwise} \end{cases}, 1 \le i \le m-1 \quad (26)$$

fi,13 checks if machine mi can employ action JB3, taking a value of 1 if machine mi is capable of adopting action JB3.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

$$f_{i,13} = \begin{cases} 1, & SJ_{i,3} \neq \varnothing \text{ and Condition 2 is true} \\ 0, & \text{otherwise} \end{cases}, 1 \leq i \leq m-2 \quad (27)$$

Where Condition 2 is true when.

fi,14 dictates whether machine mi can choose action JB4, equaling 1 if machine mi is authorized to execute action JB4.

$$f_{i,14} = \begin{cases} 1, & SJ_{i,4} \neq \varnothing \text{ and Condition 2 is true} \\ 0, & \text{otherwise} \end{cases}, 1 \leq i \leq m-2 \quad (28)$$

fi,15 assesses whether machine mi can engage in action 28, holding a value of 1 if machine mi can select action 28, which signifies that the machine is currently unengaged.

$$f_{i,15} = \begin{cases} 1, & f_{i,1} = 0 \text{ or Condition 1 is true} \\ 0, & \text{otherwise} \end{cases}, 1 \leq i \leq m-2 \quad (29)$$

Definition of reward functions

Within the scheduling setup, operations progress through a continuous sequence of environmental states, selected actions, and associated feedback rewards. At the onset of scheduling, the environment occupies an initial state designated as s0. In this opening phase, the entire set of jobs resides within the primary waiting queue Q1 of machine m1, and every piece of equipment is unengaged. The scheduling sequence initiates when machine m1 executes action ai, whereas the other machines implement action a28 to denote their inactive state. Upon the finalization of each operation, the system transitions to an updated state st, triggering each machine to select its next action. This cyclical routine persists until the subsequent operation concludes, causing a transition into the next state st+1. Over the course of this cycle, the learning agent is allocated a reward rt+1.

In a DRL framework, the agent's primary objective is to maximize its long-term total reward through strategic action selection. Achieving this requires formulating suitable reward functions, which are generally derived from the core objective functions of the scheduling problem. Within this scheduling framework, the reward granted during each state transition, represented as rt+1, is computed based on the processing duration, mirroring the chronological interval separating consecutive state transitions. These reward mechanisms are structured to deliver critical feedback to the agent, steering its policy optimization.

As scheduling advances to completion and all operations are executed, the waiting queues clear out, steering the system into its terminal state. At this juncture, a finalized scheduling configuration is realized, marking the conclusion of the entire process.

Under the DRL paradigm, the agent optimizes its policy by evaluating the instantaneous rewards gained at each distinct state transition. By breaking down the overarching objective functions into localized, immediate rewards, the agent learns to choose actions that maximize the total long-term reward, ultimately yielding highly efficient and practical scheduling arrangements.

Minimizing the makespan typically requires maximizing equipment utilization rates to eliminate inactive intervals and boost production throughput. Conversely, elevated machine utilization can expand the overall energy footprint because the machinery operates for extended periods. Both the total makespan and power usage in an NFSP are tightly linked to the machine utilization index δi(t), as shown in **Eq. 30**, which specifies whether machine mi is active or inactive.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

$$\delta_i(t) = \begin{cases} 0, & \text{machine } m_i \text{ is busy at time } t \\ -1, & \text{machine } m_i \text{ is idle at time } t \end{cases} \quad (30)$$

Aggregate energy usage incorporates machine activation/deactivation cycles, active job processing, and equipment standby modes. Nevertheless, only the power drawn during actual processing and idle intervals depends directly on the machine utilization rate δi(t). In contrast, the energy expended during machine startup and shutdown remains entirely independent of δi(t), being dictated solely by the physical properties of the machine itself.

To facilitate makespan minimization, the reward function r1 is specified as outlined in **Eq. 31**. To minimize energy use, drawing from **Eq. 5**, the reward function r2 is established as shown in **Eq. 32**. As demonstrated in Zhang *et al*. [32], minimizing the overall makespan is mathematically equivalent to maximizing the cumulative reward r1, with the formal proof detailed in **Eq. 33.** Similarly, minimizing energy usage is equivalent to maximizing the cumulative reward r2, as proven in **Eq. 34.**

$$r_1 = \frac{1}{m}\sum_{i=1}^{m}\int_{t}^{t+1}\delta_i(t) \quad (31)$$

Where m signifies the machine count, t and t + 1 denote the timestamps associated with state st and state st+1, respectively, and r1 corresponds to the reward acquired by the agent.

$$r_2 = \frac{1}{m}\sum_{i=1}^{m}\left(EN_i\int_{t}^{t+1}\delta_i(t) + \sum_{j=1}^{n}\left(EW_{ijk}\int_{t}^{t+1}\delta_i(t)\right) + ES_i + EC_i\right) \quad (32)$$

Where m indicates the total number of machines, t and t + 1 represent the times for state st and state st+1, and r2 denotes the reward allocated to the agent.

$$\begin{aligned} R &= \sum_{t=1}^{T} r_k = \frac{1}{m}\sum_{t=1}^{T}\sum_{i=1}^{m}\int_{\tau=t_{k-1}}^{t_k}\delta_i(\tau) \\ &= \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{T}\int_{\tau=t_{k-1}}^{t_k}\delta_i(\tau) \\ &= \frac{1}{m}\sum_{i=1}^{m}\int_{\tau=0}^{C_{max}}\delta_i(\tau) = \frac{1}{m}\sum_{i=1}^{m}T_i \\ &= \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n}MT_{ijk} - C_{max} \end{aligned} \quad (33)$$

Where Ti represents the aggregate non-productive time on machine mi during the processing phase. Once the volume of jobs and machines is established, the expression $\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n}MT_{ijk}$ becomes a constant value; therefore, minimizing the overall makespan is equivalent to maximizing the accumulated reward r1.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

$$
\begin{aligned}
R = \sum_{t=1}^{T} r_k &= \frac{1}{m}\sum_{t=1}^{T}\sum_{i=1}^{m}\left(EN_i\int_t^{t+1}\delta_i(t)\right. \\
&\left.+\sum_{j=1}^{n}\left(EW_{ijk}\int_t^{t+1}\delta_i(t)\right)+ES_i+EC_i\right) \\
&=\frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{T}\int_{\tau=t_{k-1}}^{t_k}\left(EN_i\int_t^{t+1}\delta_i(t)\right) \\
&+\frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{T}\sum_{j=1}^{n}\left(EW_{ijk}\int_t^{t+1}\delta_i(t)\right) \\
&+\frac{1}{m}\sum_{i=1}^{m}(ES_i+EC_i) \\
&=\frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{T}\int_{\tau=t_{k-1}}^{t_k}\left(EN_i\int_t^{t+1}\delta_i(t)\right) \\
&+\sum_{j=1}^{n}\sum_{i=1}^{m}(\alpha_{ijk}\times MT_{ijk}\times EW_{ijk})+\sum_{i=1}^{m}ES_i+\sum_{i=1}^{m}EC_i \\
&=-\frac{1}{m}\sum_{i=1}^{m}(T_i\times EN_i)+E_W+E_M \\
&=-E_N+E_W+E_M
\end{aligned}
\tag{34}
$$

Where EM, EW, and EN denote the energy requirements for machine startup/shutdown, active job execution, and idle machine standby, respectively. Furthermore, when the machine and job counts remain static, EM and EW act as constants, meaning that minimizing power consumption directly maximizes the accumulated reward r2.

The reward functions ri(i = 1, 2) possess the following three distinctive traits:

(1) ri dynamically relays the immediate consequences of selected actions back to the environment.

(2) ri mirrors the primary optimization targets, meaning that maximizing the cumulative reward directly minimizes the target objectives.

(3) ri remains entirely independent of the NFSP dimensions, allowing it to be seamlessly applied across problems of varying scales.

DRL algorithm

Bidirectional recurrent neural networks

Deep learning (DL) represents a branch of machine learning that employs multi-layered networks known as perceptrons to glean insights from datasets [33]—the core emphasis of DL centers on extracting meaningful representations from

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

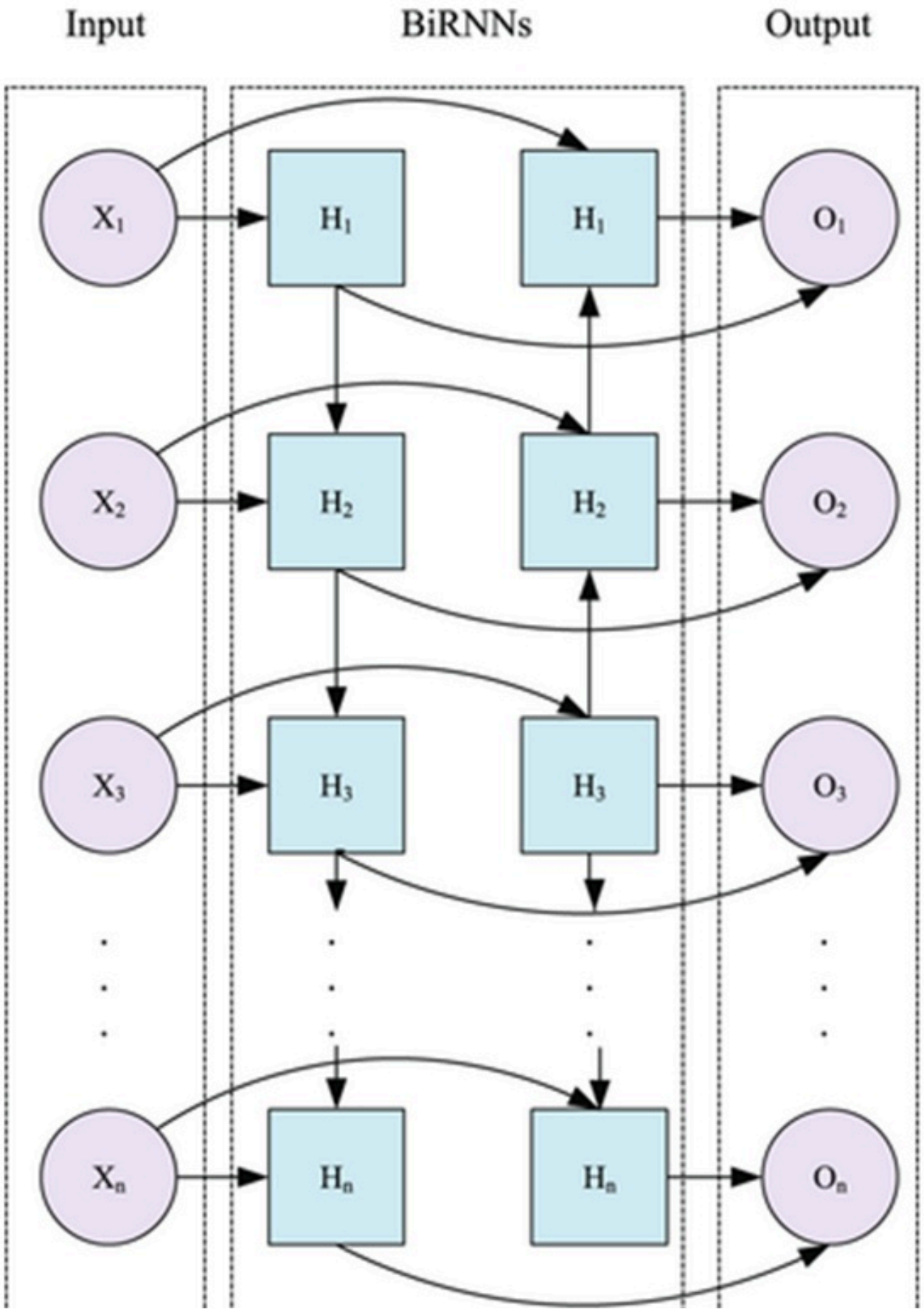
underlying data inputs. A variety of DL structures have been introduced—such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and deep belief networks—which are widely utilized across multiple domains. In the present research, bidirectional recurrent neural networks (BiRNNs) are used to train the developed model and learn from the established action space.

As a specialized category of artificial neural networks (ANNs), RNNs excel at managing sequential datasets. Beyond the structural components found in standard feedforward configurations, RNNs integrate recurrent connections or feedback loops [34]. This structural design empowers the network to store and exploit feature data from historical inputs, ensuring that subsequent outputs are influenced by both the immediate input and prior historical states.

Nevertheless, conventional RNN architectures face challenges related to vanishing gradients during back-propagation, which restricts their capacity to capture long-range dependencies. When dealing with sequences lacking strict chronological order, RNN frameworks often achieve better outcomes by analyzing data sequences in both progressive and backward directions.

Serving as a variant of standard RNNs, BiRNNs integrate an extra recurrent layer to extract context from both edges of a sequence when computing output approximations. This dual-direction analysis strengthens the representation capacity of RNN models for non-chronological sequences. Integrating BiRNNs into the developed methodology facilitates the comprehensive collection of both preceding and subsequent contextual details throughout the scheduling decision framework. By analyzing input configurations in both directions, BiRNNs enhance the system's ability to decode intricate correlations and chronological patterns typical of flow-shop scheduling environments. This provides a foundation for sophisticated decision-making and superior flexibility in volatile manufacturing settings, ultimately boosting scheduling efficiency. **Figure 1** illustrates a BiRNN structure featuring a single hidden layer.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press



© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

learning. A major advancement was documented by Hinton and Salakhutdinov [35], who verified that architectures containing multiple hidden layers possess significantly higher representational capacity than shallower networks. Additionally, they unveiled a “layerwise pre-training” technique designed to mitigate training difficulties by establishing highly effective initialization parameters for these neural structures. **Figure 2** details an MLP configuration equipped with two dense internal layers. Within this diagram, data transmission follows the designated arrows, and the pathways are linked to specific operational weights. To maintain optimal parameter optimization efficiency, the quantity of internal layers is generally capped at three in real-world scenarios.

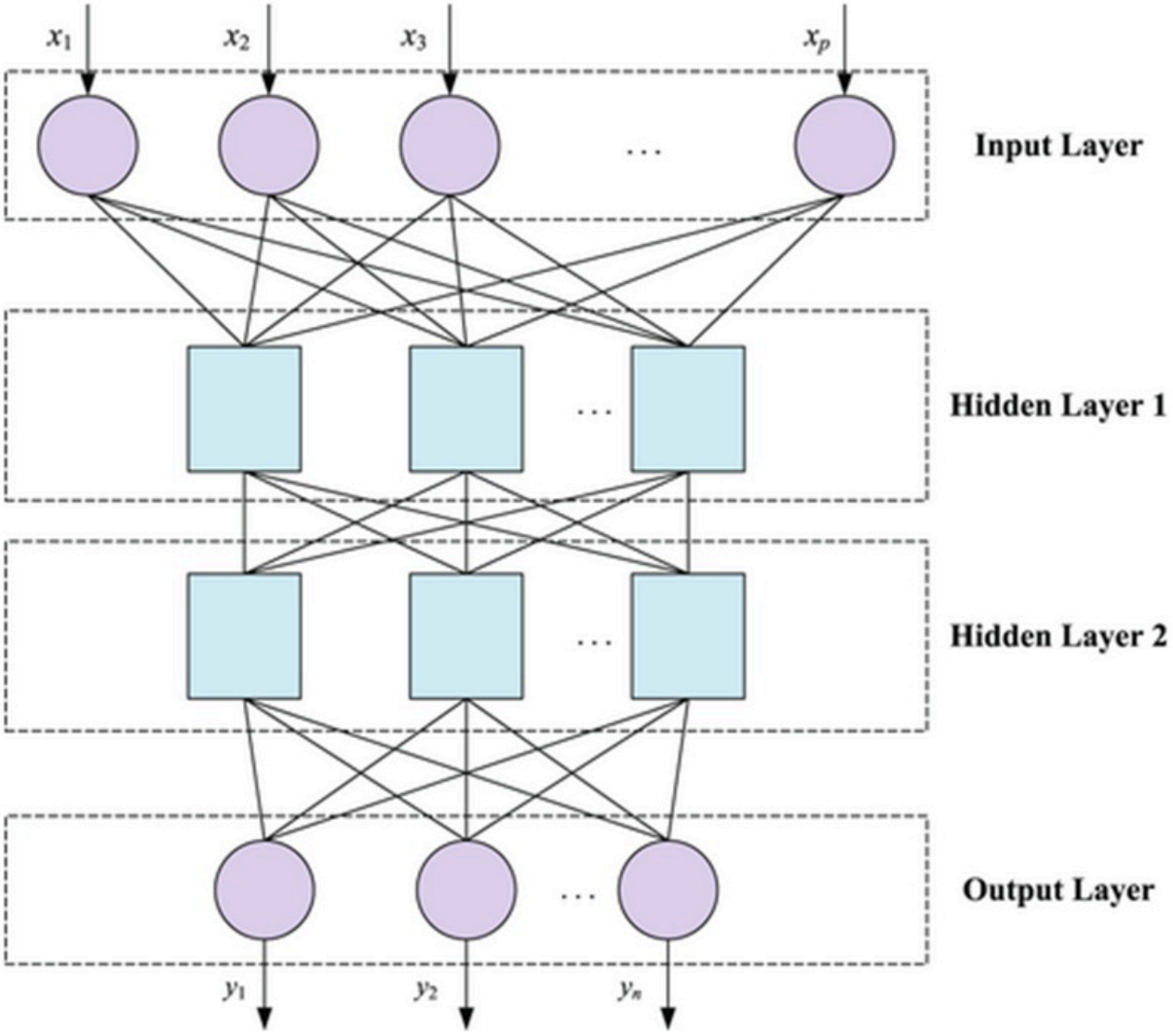


Figure 2. Multi-layer perceptron neural network model.

Optimizing multi-layer feedforward systems frequently relies on the Error Back-propagation technique, which operates across two primary phases: the forward pass and the backward pass. The backward calculation starts at the final output layer of the MLP structure. The error metrics during this feedback phase can be formulated as **Eq. 35** [36]. Through the computation of partial derivatives relative to the cost function, individual weight parameters are adjusted as defined in **Eq. 36.**

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

$$E = E(y_1, y_2, \ldots, y_n) = \sum_{i}^{n} (y_i - t_i)^2 \quad (35)$$

Where yi represents the output value, and ti represents the average of all output values.

$$\omega = \omega - \alpha \times \frac{\partial E}{\partial \omega} \quad (36)$$

Where α indicates the learning rate.

Within the developed BiRNNs-TD(λ) framework, the optimized network configuration incorporates a total of six layers: a single input layer, four internal layers, and a single output layer, as depicted in **Figure 3.** The receptive input layer contains (m − 2) × 15 + 11 + 9 nodes to capture the comprehensive state attributes of all processing machinery. The final layer consists of a single node that yields the computed state value output. The internal MLP sub-model integrates four distinct hidden layers, combining two LSTM-configured BiRNN structures alongside two fully connected Dense layers. The respective hidden layers are configured with 128, 64, 10, and 1 output nodes, utilizing a linear activation function throughout these intermediate stages.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

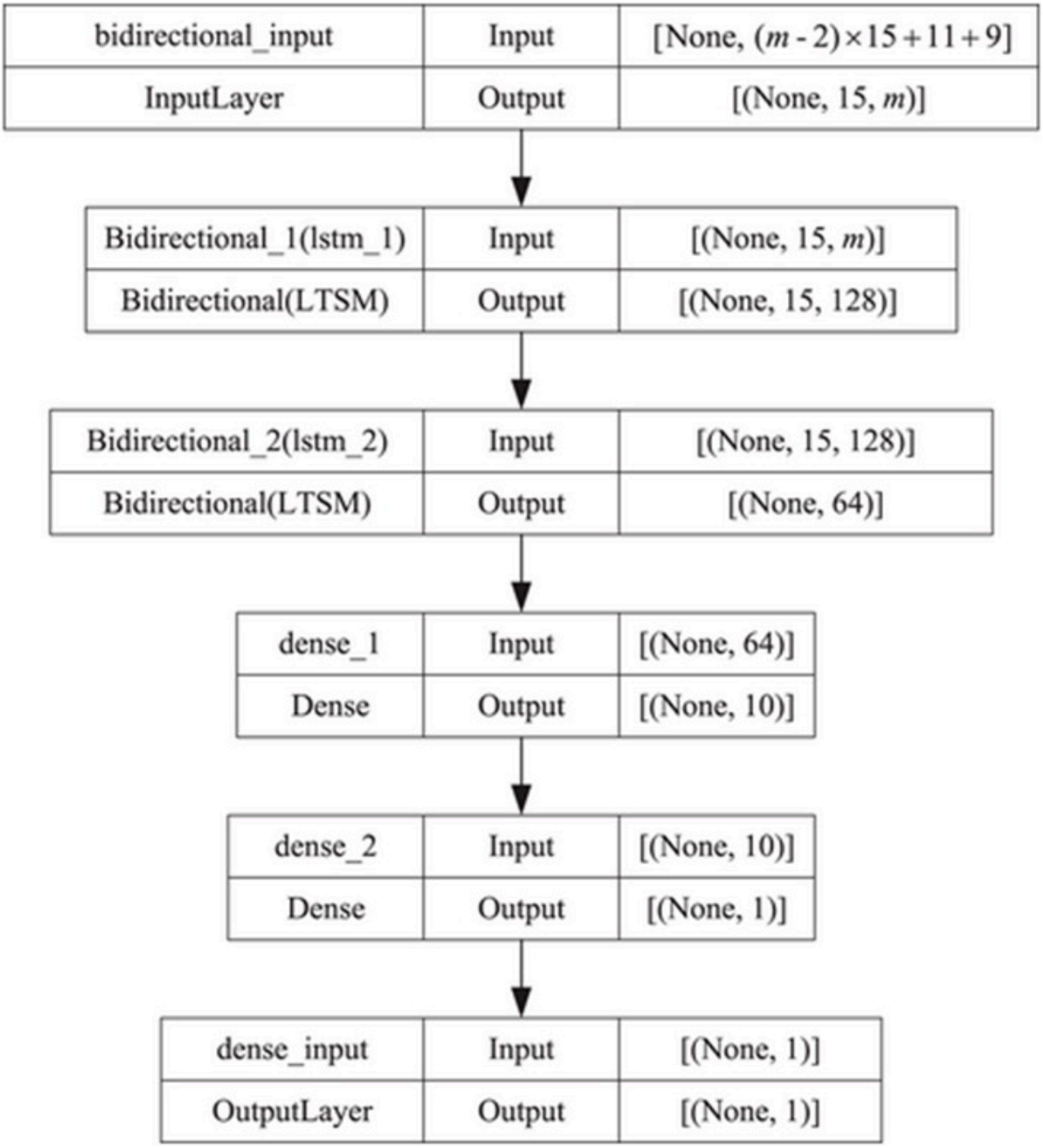


Figure 3. Components of the trained network model.

Throughout the model optimization phase, the internal network weights are tuned via the Adam optimization algorithm, utilizing a mean squared error loss criterion. The detailed workflow governing this network optimization is diagrammed in **Figure 4.**

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

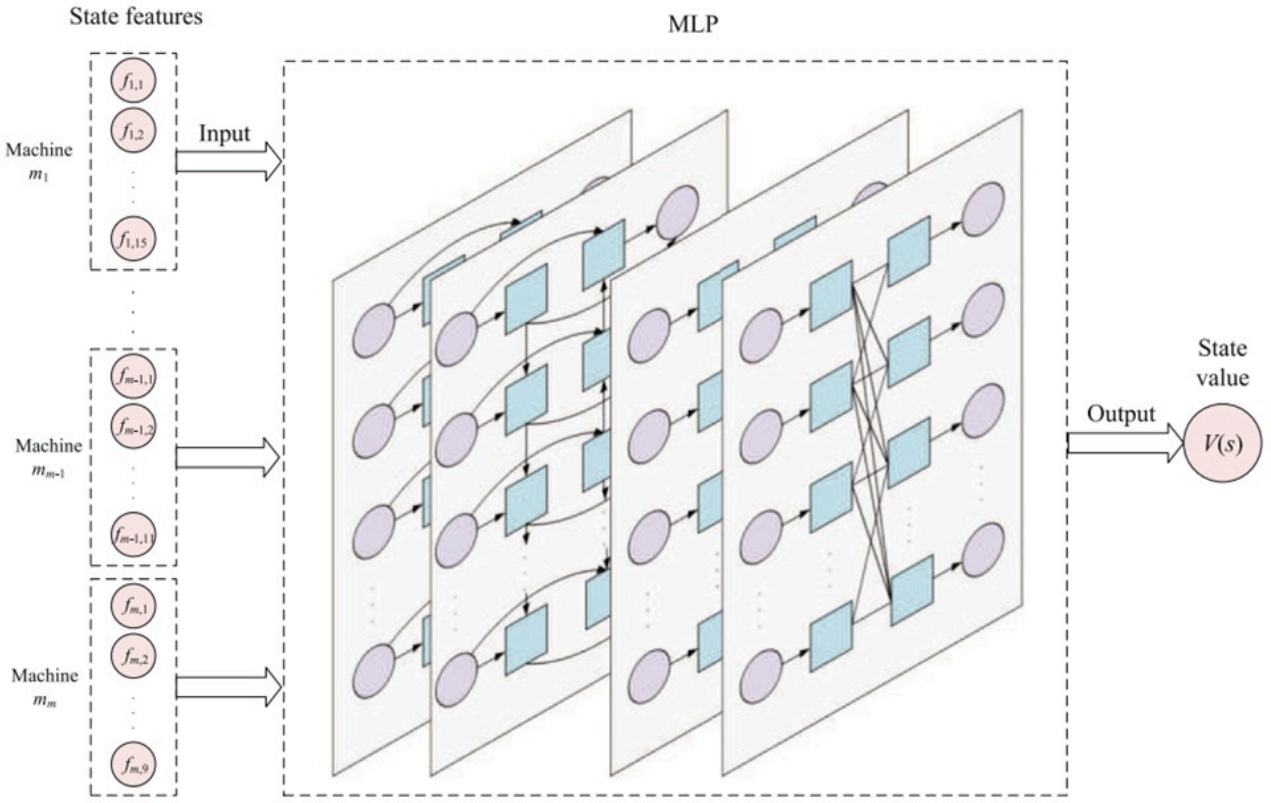


Figure 4. The specific process of network training.

TD(λ) algorithm

The temporal difference framework was originally introduced by Sutton in 1988 [37], with its convergence properties validated through empirical testing. Subsequently, the TD methodology has served as a cornerstone for numerous reinforcement learning approaches. In a manner comparable to Monte Carlo techniques, the TD framework incorporates two key computational steps: TD prediction for evaluating state values, and TD control geared toward deriving the most effective operational policy. Relative to Monte Carlo approaches, the TD scheme demands lower computational overhead, and its primary formulation is given by **Eq. 37.**

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \quad (37)$$

Where α denotes the learning rate or step size and $R_{t+1} + \gamma V(S_{t+1})$ indicates the TD target.

Following the logic of **Eq. 36,** during a transition from an active state to a subsequent state, the value associated with the initial state is already established. This computed value exhibits minimal variance relative to the succeeding state value, although minor deviations can occur. The resulting TD error is typically defined via **Eq. 38.**

$$\delta_t \doteq R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \quad (38)$$

A greedy execution strategy dictates that the operational agent chooses an option from the available action space A(st) utilizing the active state st, thereby moving toward the specific state that yields the maximum value V(s) within the adjacent accessible state space S. This greedy selection routine is depicted in **Eq. 39.** Based on the principles of the

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

Bellman equation, this exploit-driven strategy becomes the optimal policy π* once the underlying value function converges through iterative updates.

$$a_t^\star = \arg\max_{a \in A(s_t)} V(s_t) = \arg\max_{a \in A(s_t)} [r_{t+1} + \gamma V(s_{t+1})] \quad (39)$$

Within reinforcement learning, balancing exploration against exploitation constitutes a core dilemma in decision-making. Exploration means the agent avoids restricting its choices to the immediate best option at a given decision step, instead trying alternative actions that might provide greater long-term returns. Conversely, exploitation entails leveraging acquired knowledge, meaning the agent defaults to state transitions previously found to maximize immediate returns. While exploration typically leads to a more gradual convergence rate, it retains the capacity to locate global optima. In contrast, exploitation accelerates convergence but exposes the model to the risk of stagnation in local optima. To effectively balance these two tendencies, an ϵ-greedy selection policy can be applied to regulate choices during agent-environment cycles. Here, the agent targets the state offering the highest value V(s) within the accessible state space S with a likelihood of ϵ, while choosing a state from S purely at random with a likelihood of 1 − ϵ. The probability p(st, a) of executing action a at state st is formulated in **Eq. 40.**

$$p(s_t, a) = \begin{cases} 1 - \epsilon + \dfrac{\epsilon}{|A(s_t)|} & , a_t = a_t^\star \\ \dfrac{\epsilon}{|A(s_t)|} & , a_t \neq a_t^\star \end{cases} \quad (40)$$

Where A(st) indicates the action space in state st and |A(st)| indicates the number of takeable actions.

In the previously described TD(0) implementation, the Markov chain evaluates a single step, with subsequent state metrics derived by estimating the active value function from the preceding state's values. Correspondingly, the Markov chain within the TD(λ) method extends its horizon across n future steps. The structural weight allocated to each discrete step is defined by (1 − λ)λn−1, which diminishes as n grows larger, guaranteeing that the cumulative sum of all weights equals 1. The value function V(λ)(st) undergoes updates driven by the calculated return λ. The iterative formulation of the TD(λ) scheme is outlined in **Eq. 41.**

$$V^{(\lambda)}(s_t) = r_{t+1} + \gamma \left[(1 - \lambda) max_{A(s_t)} V(s_t) + \lambda V^{(\lambda)}(s_{t+1}) \right] \quad (41)$$

Generally, the multi-step TD(λ) approach achieves rapid convergence compared to the single-step TD(0) framework. Combining reinforcement learning with temporal-difference mechanisms allows our model to gather insights from practical experience and continuously refine its scheduling strategies. Consequently, the TD(λ) methodology is chosen to handle value function updates throughout this investigation. The operational steps of this TD(λ) framework are detailed in **Algorithm 1.**

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

- Input: Learning rate $\alpha \in (0, 1)$, state value function $V(s)$
- Output: The optimal policy π^*
- 1: Initialise state S_0, *episode*
- 2: if *episode* $\leq$ MAX_EPISODE then
- 3: Updating the state values $V(S)$ according to Eq. 40
- 4: Record the state values $V(S)$
- 5: Generate a *new episode*
- 6: Take action according to policy π to obtain R, S'
- 7: while state S' is not terminated do
- 8: Update state $S = S'$
- 9: end while
- 10: else
- 11: Terminate the current *episode*
- 12: Output the optimal policy π^*
- 13: end if

Algorithm 1. TD(λ) algorithm

Multi-objective reinforcement learning

In contrast to conventional single-criterion RL approaches, multi-objective reinforcement learning (MORL) embeds several simultaneous targets directly into the feedback loop between the agent and its environment, assigning an independent reward signal to each distinct objective function. This architecture effectively models situations where an individual actor must address an array of competing targets concurrently. Through this mechanism, MORL can comprehensively navigate the search domain and construct a diverse collection of non-dominated options.

MORL is explicitly tailored to handle sequential choice problems that involve multiple competing performance measures. For any given target under a static policy π, a specific value function exists that satisfies the criteria of the Bellman equation. **Eq. 42** illustrates the vector form of the state-action value function, which similarly complies with the Bellman formulation. Accordingly, the ideal vector value function is formulated in **Eq. 43**, facilitating the direct deduction of the optimal policy as shown in **Eq. 44.**

$$MQ^{\pi}(s,a) = \left[Q_1^{\pi}(s,a), Q_2^{\pi}(s,a), \cdots, Q_N^{\pi}(s,a)\right]^{\mathrm{T}} \quad (42)$$

$$MQ^{*}(s,a) = \max_{\pi} MQ^{\pi}(s,a) \quad (43)$$

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

$$\pi^*(s,a) = \operatorname{argmax}_a MQ^*(s,a) \quad (44)$$

Where MQπ(s, a) represents the vector state-action value function.

Frameworks for MORL integrate a variety of methodologies, such as linear scalarization models, interaction-driven schemes, ϵ-constraint techniques, Pareto-centric models, and reward-value-based systems, each presenting unique computational properties. For this research, the reward-value-based methodology is selected. The Pareto front is identified by performing a fast non-dominated sorting procedure on the total reward accumulated during each training episode. Notably, MORL techniques constitute an emerging field of study; thus, universally accepted validation metrics for these systems remain unavailable. Many current algorithms detailed in the academic literature are customized for narrow, distinct operational tasks, often lacking the versatility required for thorough algorithmic validation. As a result, benchmarking these different techniques is problematic due to discrepancies in task definition and algorithmic execution. Additionally, the lack of an established baseline Pareto frontier for these scenarios complicates the determination of definitive performance benchmarks for standalone algorithms.

- Input: Given learning rate $\alpha \in (0, 1)$, constants *Batch_Size*, *Max_Episode*, *S*, initialise experience replay memory *D*
- Output: The optimal policy $\pi*$
- 1: Initialise *episode* = 0, behaviour network model *V* and its parameter θ, target network model $\tilde{V}$ and its parameter $\tilde{\theta}$
- 2: if *episode* ≤ MAX_EPISODE then
- 3: Reset the environment *E*, initialise state s_0, pre-process $\phi_0 = \phi(s_0)$
- 4: while $|D| < Batch_Size$ do
- 5: Take action $a_t = (a_1, a_2, , a_n)^T$ by ϵ- greedy policy
- 6: Update state s_{t+1}, pre-process $\phi_{t+1} = \phi(s_{t+1})$
- 7: Attach one-step interaction sequence $(\phi_t, r_{t+1}, \phi_{t+1})$ to *D*
- 8: end while
- 9: Take small batch samples $(\phi_t, r_{t+1}, \phi_{t+1})$ randomly from *D*
- 10: Calculate target state value $V(s_t)$
- 11: Update recursively parameter per $\tilde{\theta}$ *S*-step interaction
- 12: else
- 13: Terminate the current *episode*
- 14: Output the optimal policy $\pi*$
- 15: end if

Algorithm 2. BiRNNs-TD(λ) algorithm

BiRNNs-TD(λ) algorithm

In this work, the NFSP is converted into an MDP structure, where the isolated state characteristics are fed into an MLP architecture centered around BiRNN blocks. By integrating the multi-step temporal difference routine (TD(λ)) standard in RL, action valuations are derived from the underlying state values. This approach allows the identification of high-performing strategies to execute throughout the production cycle, yielding a variety of unique scheduling configurations. The operational schematic of the BiRNNs-TD(λ) procedure is illustrated in **Algorithm 2.**

Within the BiRNNs-TD(λ) scheme, the core interaction sequences between the processing agent and the manufacturing environment during state transitions st+1 ← st are structured as follows:

(1) Supply the active state st and employ the target network function approximator to calculate expected state values.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

(2) Determine the immediate best action at using the state value metrics in accordance with **Eq. 45.**

(3) Finalize the environmental transition to capture an updated state st+1 and an associated reward rt+1 following the execution of action at.

$$a_t = \arg max_{a \in A(s)} \left[r^a_{s_t, s_{t+1}} + \gamma V(\Phi(s_{t+1})|\theta) \right] \quad (45)$$

Throughout these state transitions, the operational agent treats the produced experience tuple e_t = (s_t, a_t, r_t,s_{t+1}) as input data to iteratively update the function approximator. The precise workflow unfolds as follows:

(1) Archive the experience vector et within the experience replay buffer D immediately after the state transition finishes.

(2) Once the replay buffer D contains a sufficient volume of historical data, randomly extract a mini-batch of samples {(Φ_t ,r_{t+1} ,Φ_{t+1})} from D to systematically adjust the parameter weights θ of the behavioral network.

(3) Synchronize the parameter values by updating $\theta^- \leftarrow \theta$ after every S-step cycle of interaction, thereby maintaining structural alignment between the behavioral and target networks.

The detailed interaction loop between the decision agent and the environment is diagrammed in **Figure 5.**

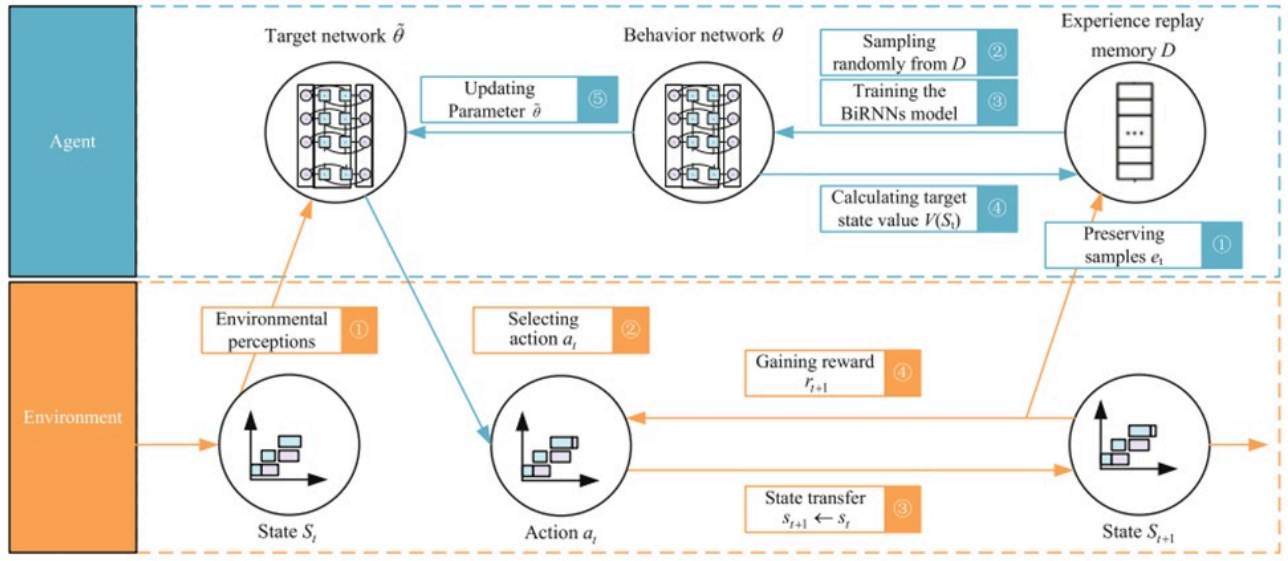


Figure 5. Interaction process between the agent and the environment.

Experimental Study

Benchmark problems and experimental platform

Considering the fundamental significance and broad applicability of the traditional NFSP, a dedicated test collection for the NFSP was assembled utilizing established benchmark problems drawn from the standard PFSP. In this investigation, the formulated mathematical model incorporates supplementary variables associated with the energy usage required for the activation and deactivation of machine mi (ESi or ECi), the energy consumed per unit of time while processing operation ojk on machine mi (EWijk), and the power drawn per unit of time while machine mi remains idle (ENi). To

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

validate the efficacy and computational efficiency of the developed algorithm, the evaluation suite was compiled by isolating test scenarios of varying magnitudes from three widely recognized benchmark collections.

The small-scale scenarios are extracted from the benchmark set introduced by Carlier [38], referred to as Car1–Car8. The large-scale scenarios are derived from datasets presented by Reeves [39] and Taillard [40], designated as Rec01–Rec42 and Ta001–Ta120, respectively.

Regarding the energy requirements for equipment activation and deactivation, it is generally anticipated that the energy necessary to start a machine exceeds that needed to shut it down, while maintaining a consistent correlation between the two. That is, an elevated ESi value should map to an elevated ECi value. To accomplish this, ESi values are generated stochastically within the interval [500, 999], and this range is proportionally mapped to the interval [100, 599] to determine the corresponding ECi. For instance, if ESi equals 500, the associated ECi is assigned a value of 100. This identical proportional mapping strategy is utilized to compute EWijk and ENi. In particular, EWijk is generated randomly within the bounds of [6, 10], and this domain is mapped to [1, 5] to establish the corresponding ENi value.

An illustrative example for the Car1 instance is detailed in **Table 3**, demonstrating how energy-centric parameters are integrated into the baseline benchmark problem.

Table 3. Processing data of instance Car1.

	m_5	m_4	m_3	m_2	m_1
j_1	412	245	142	12	375
j_2	398	278	758	452	632
j_3	765	534	124	876	12
j_4	499	120	523	542	460
j_5	999	124	789	101	528
j_6	123	375	632	245	796
j_7	452	896	543	230	532
j_8	785	543	214	124	14
j_9	463	210	753	527	257
j_{10}	259	258	214	896	896
j_{11}	988	765	501	302	532
ES	840	907	789	693	948
EC	440	507	389	293	548
EW	6	7	9	8	10
EN	1	2	4	3	5

Extensive experiments were executed employing the BiRNNs-TD(λ) methodology to resolve the NFSP, thereby validating both its effectiveness and operational efficiency. The experiments were implemented in Python 3.10 using the PyCharm integrated development environment. Execution took place on a personal computer equipped with an Intel Core i7

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

processor operating at 3.40 GHz and a discrete graphics processing unit (Nvidia GeForce RTX 3080). The reinforcement learning framework—comprising the foundational Environment module, Machine structure, and Job structure—was developed utilizing the OpenAI Gym RL library.

OpenAI Gym supplies a uniform architecture for the primary constituent functions of the Environment class, encompassing init, reset, step, render, and close. The developed algorithm was driven by an active agent that engages in iterative interactions with the simulated environment. The deep neural network acting as the function approximator for the agent was constructed via Keras, whereas the internal BiRNN architecture was built upon the TensorFlow backend.

This testing configuration ensures a thorough assessment of the proposed algorithm's capability to resolve the NFSP, leveraging the robust computational capabilities of Python, OpenAI Gym, Keras, and TensorFlow.

Parameters setting

The choice of algorithmic parameters substantially influences both the caliber of the final solution and the velocity of model training. Although a universally accepted baseline for parameter tuning in DRL remains absent, several foundational guidelines apply. Within the framework of the ϵ-greedy policy, balancing the trade-off between exploration and exploitation is critical. At the outset, ϵ ought to be assigned an elevated value to promote broad exploration across the policy space, then progressively reduced to exploit the identified optimal strategy. Customarily, ϵ is subjected to exponential decay at a predefined rate.

The discount factor γ dictates the influence of future state values on the overall accumulated reward and is generally configured to a value approaching 1. This guarantees that anticipated future rewards factor heavily into the ongoing decision-making sequence. The learning rate, which governs the degree to which novel insights overwrite previously acquired knowledge, typically resides within the 0–1 range.

Following comprehensive empirical testing, precise parameter values were established, as detailed in **Table 4.** These specific configurations were chosen because they have proven capacity to deliver optimal performance within the context of the newly proposed algorithm.

Table 4. The values of parameters in the BiRNNs-TD(λ) algorithm.

Parameters	Value	Description
BATCH SIZE	64	Number of samples utilized to compute gradients during neural network training.
D	5000	Maximum capacity of the experience replay buffer, from which training samples are randomly drawn.
MAX_EPISODE	200	Upper limit on the number of interactions between the learning agent and the environment.
LEARNING_RATE	1.0×10^{-3}	Step size that determines how much the agent's parameters are updated during learning.
γ	0.998	Discount factor used in the iterative formulation of the optimal value function, controlling the influence of future rewards on cumulative return in the Markov process.
ϵ	1.0	Parameter in the ϵ-greedy policy that balances exploration and exploitation when searching the state space.
DECAY_RATE	0.996	Exponential decay factor applied to ϵ over the training process.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

λ	0.8	Trace-decay parameter in the TD(λ) method that regulates the contribution of past rewards.

Experimental results and analysis

The introduced BiRNNs-TD(λ) algorithm is tested against the pre-assembled benchmark problems outlined previously. Within this framework, BiRNNs process the incoming machine state attributes to output a corresponding state value. This computed state value subsequently drives the decision-making protocol embedded in the TD(λ) algorithm.

From the established evaluation suite, 12 distinct, representative problems of varying sizes were chosen. Every problem scenario is subjected to 200 training episodes, with the resultant solutions from each episode aggregating to form the solution space for that specific problem. This collective solution space is stratified into distinct tiers utilizing a fast non-dominated sorting technique, as visualized in **Figure 6**, where distinct colors denote separate non-dominated fronts. Importantly, the values on the vertical axis have been modified by deducting the baseline fixed energy expenditure corresponding to the static operational time in each episode. The true Pareto front for each evaluated problem corresponds to the lowest tier within the overall solution space. The exact Pareto front for each specific problem scenario is detailed in **Table 5.**

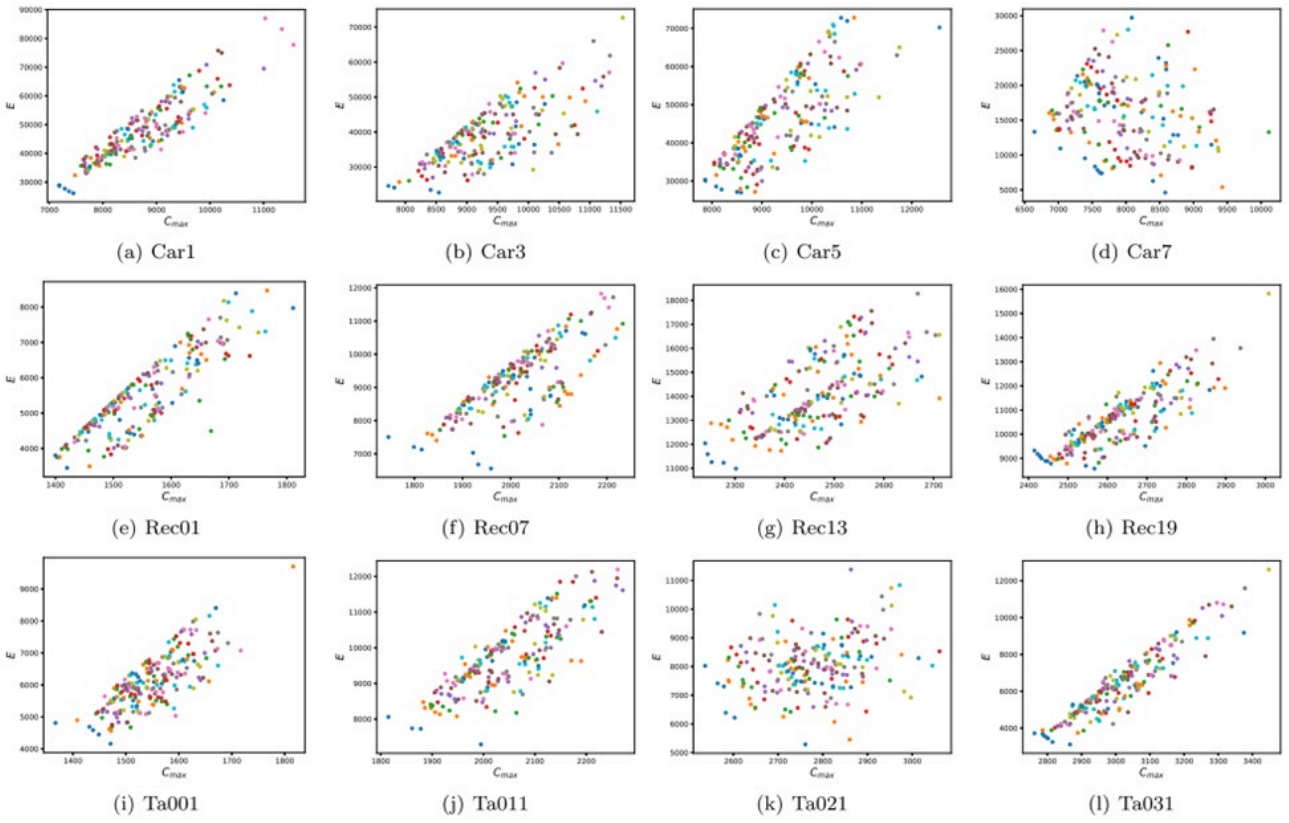


Figure 6. Scatter chart of the solution space.

Table 5. Pareto fronts of the instances.

Instances	Objectives	Values								

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

Car1(11 × 5)	C_{max}	7184	7187	7291	7369	7449				
	E	28,927	28,643	27,688	26,785	26,150				
Car3(12 × 5)	C_{max}	7725	7820	8410	8547					
	E	24,583	24,130	23,440	22,687					
Car5(10 × 6)	C_{max}	7865	7873	8086	8194	8505	8584			
	E	30,318	30,049	28,578	27,734	27,143	26,955			
Car7(7 × 7)	C_{max}	6643	7029	7393	7541	7579	7615	7641	8401	8582
	E	13,325	10,946	9531	8353	7837	7454	7395	6277	6112
Rec01(20 × 5)	C_{max}	1400	1403	1406	1420					
	E	3803	3750	3747	3449					
Rec07(20 × 10)	C_{max}	1746	1799	1815	1922	1933	1959			
	E	7506	7205	7127	7032	6682	6558			
Rec13(20 × 15)	C_{max}	2240	2245	2253	2277	2302				
	E	12,042	11,589	11,259	11,229	10,991				
Rec19(30 × 10)	C_{max}	2415	2423	2429	2433	2443	2448	2497	2545	2567
	E	9322	9181	9093	8997	8897	8885	8791	8697	8581
Ta001(20 × 5)	C_{max}	1367	1431	1438	1449	1471				
	E	4810	4689	4587	4450	4158				
Ta011(20 × 10)	C_{max}	1815	1861	1878	1995					
	E	8061	7743	7731	7292					
Ta021(20 × 20)	C_{max}	2537	2565	2579	2583	2603	2761			
	E	8023	7712	7306	6392	6218	5287			
Ta031(50 × 5)	C_{max}	2762	2786	2790	2797	2802	2815	2866		
	E	3722	3682	3586	3511	3442	3244	3118		

Using the specific solution (6643, 13,325) located on the Pareto front of the Car7 scenario as an illustration, the corresponding Gantt chart generated by the BiRNNs-TD(λ) method is displayed in **Figure 7.** The dotted red line indicates the exact moment of each sequential state transition. The processing sequence for the jobs on the machine follows the pattern 4-5-6-3-2-7-1. The operational agent progresses from the starting state S0 to the terminal state S43 through a sequence of 43 distinct decisions required to finalize all operational tasks. These choices encompass pinpointing the exact start times for operations, assigning operations to specific machinery, dictating if a machine should enter an idle phase, and related choices. Crucially, the chronological gap between consecutive state transitions is measured in milliseconds, illustrating that the optimized agent processes decisions with near-instantaneous speed.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

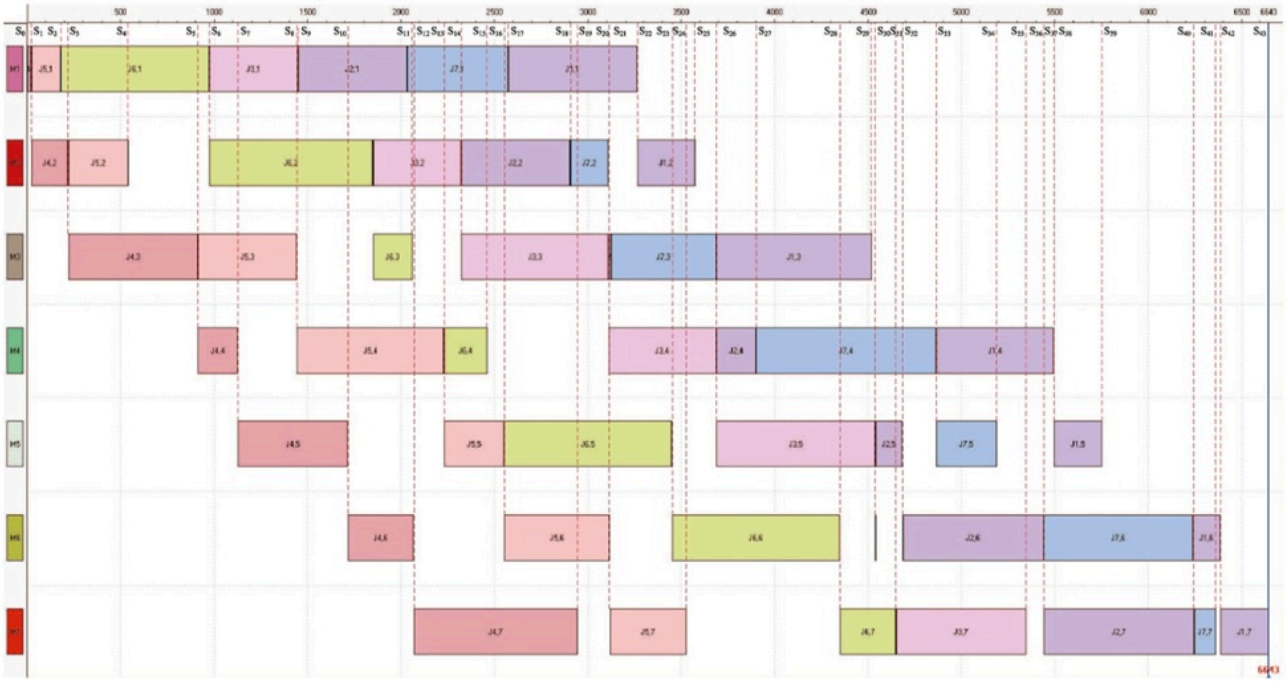


Figure 7. The Gantt chart of instance Car7.

Because DRL algorithms function through an ongoing interaction cycle involving a single agent and the environment—unlike typical population-based meta-heuristic algorithms—the total count of solutions present in the Pareto set generated by the BiRNNs-TD(λ) method is comparatively modest. During the testing phase, it became evident that adjusting various parameters directly alters the empirical outcomes, with the learning rate exhibiting a particularly strong influence. Modifying the learning rate yields distinctly different Pareto fronts. Using instance Ta001 as a case study, when the learning rate is adjusted to 1 × 10−3, 8 × 10−4, and 5 × 10−4, the generated Pareto fronts are illustrated in **Figure 8.**

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

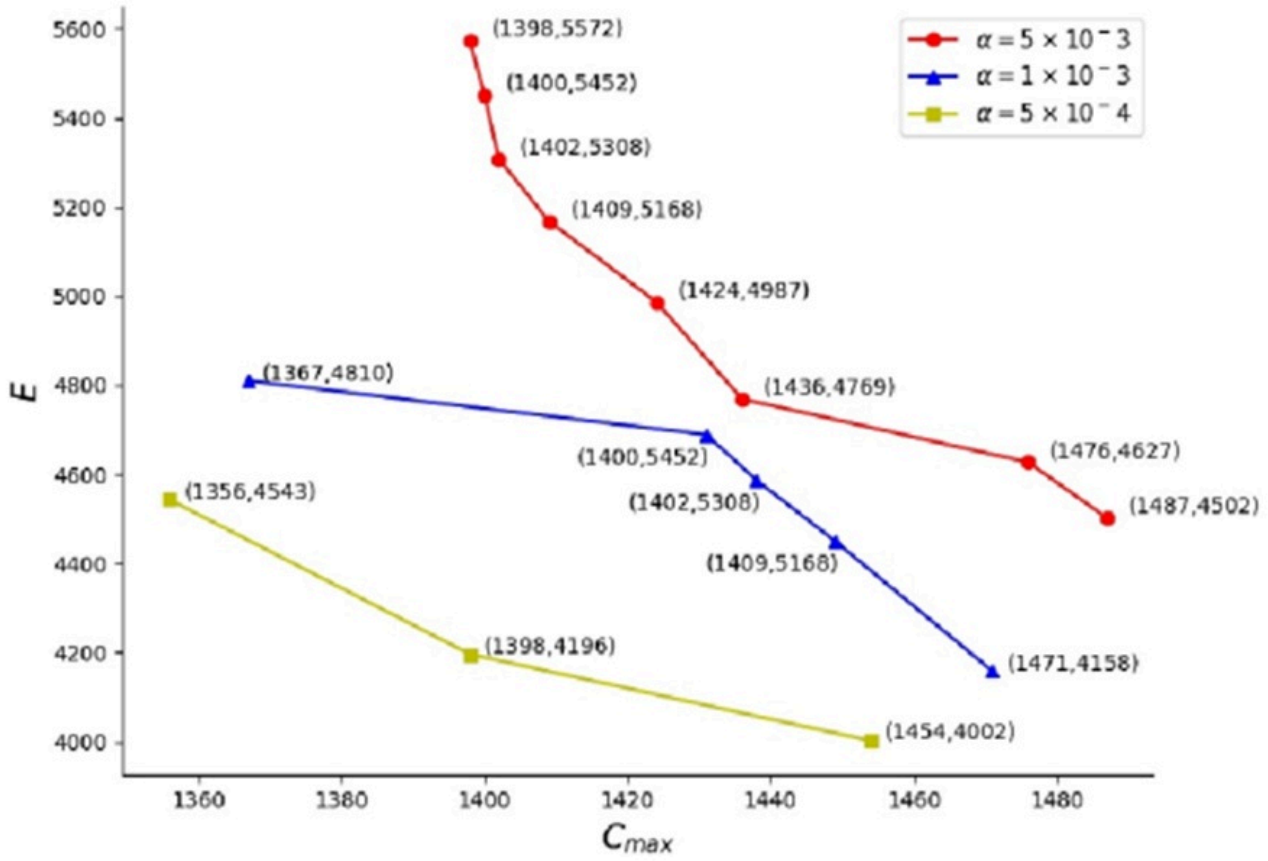


Figure 8. Pareto fronts of Ta001 with different learning rates.

As clearly demonstrated in **Figure 8**, the learning rate heavily dictates the solution caliber for the multi-objective NFSP when processed by the BiRNNs-TD(λ) algorithm. Typically, a diminished learning rate yields enhanced solution quality, although it might produce fewer viable solutions along the actual Pareto front. Nevertheless, it is critical to recognize that an overly restrictive learning rate can cause sluggish convergence during neural network parameter tuning. Standard DRL frameworks lack mechanisms to auto-adjust the learning rate as training progresses. The ideal learning rate fluctuates depending on the specific NFSP instance. Even within a single problem scenario, dynamically scaling the learning rate relative to training progression yields better results than adhering to a static rate. Consequently, during practical experimentation, it is recommended to start with an elevated learning rate early in the iteration cycle to ensure parameter convergence. This rate should then be systematically decreased as iterations progress until the network parameters are largely optimized, thereby increasing the likelihood of securing an optimal solution.

To evaluate how frequently each action from the defined action suite is utilized in the multi-objective NFSP, a statistical review of the actions selected by the agent during the instance training phases was conducted, as mapped out in **Figure 9**. A heightened frequency of action utilization signals the agent's strong preference for deploying that specific maneuver during training to secure maximized rewards. Based on **Figure 9**, it is evident that operational choices including JB1, JB2, JB3, SPT, LPT, SPT/TWK, SPT*TWK, SRM, LRM, SSO, LSO, SPT + SSO, LPT + LSO, SPT/SSO, and LPT/LSO were executed in excess of 50 times. This high execution frequency underscores their vital impact and utility in constructing the final Pareto front. These specific maneuvers are fundamental to the overall optimization strategy and play a pivotal role in securing high-tier solutions.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

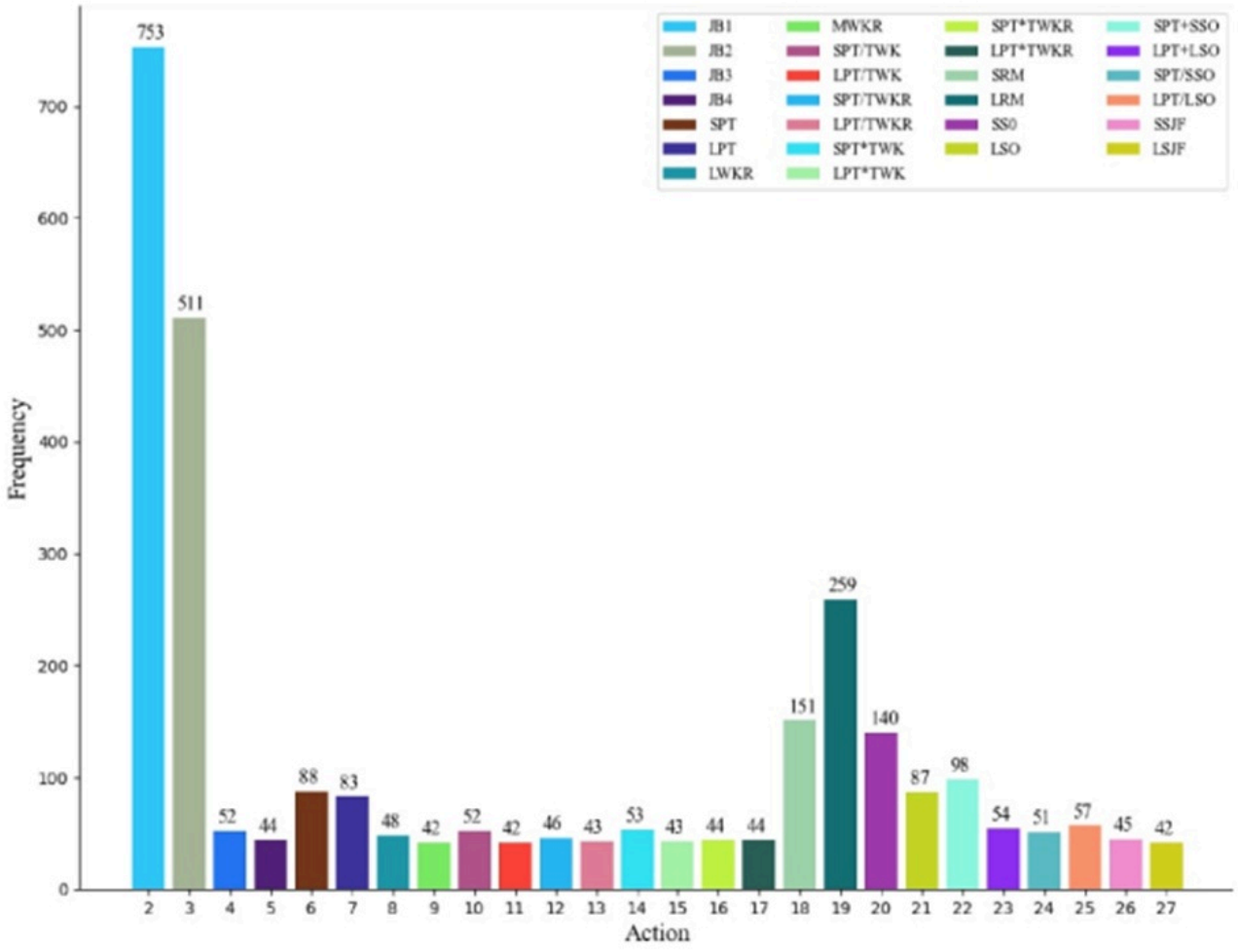


Figure 9. The frequency of each action.

To thoroughly gauge the operational capabilities of the introduced BiRNNs-TD(λ) method, it was benchmarked against the advanced LSTM-TD(0) architecture introduced in Wang *et al*. [41]. The LSTM-TD(0) architecture was specifically developed to tackle the NFSP, focusing on minimizing total makespan. It leverages an LSTM-based neural network for isolating state attributes and applies a single-step TD(0) protocol to guide decision-making. The average relative percentage deviation (ARPD) metric is routinely utilized to measure the comparative performance of multi-objective DRL systems, as defined in **Eq. 46**. A lower ARPD metric signifies superior algorithmic efficiency and a closer approximation to the true Pareto front.

$$\mathrm{ARPD} = \frac{1}{n}\sum_{i=1}^{n}\frac{C_{i,max} - C^{*}_{max}}{C^{*}_{max}} \times 100\% \qquad (46)$$

Where n is the number of solutions in the Pareto front, $C_{i,\,\max}$ represents the ith solution in the Pareto front, and $C^{*}_{\max}$ indicates the optimal solution of the two algorithms.

To validate the enhanced capabilities of the developed algorithm, both the BiRNNs-TD(λ) and the LSTM-TD(0) frameworks were deployed against the identical batch of formulated test instances, operated within an identical hardware/software environment, and initialized with identical tuning parameters. The resulting Pareto fronts generated by both systems are graphed in **Figure 10**, whereas the boxplot illustrating the comparative performance metrics of the two algorithms over various test cases is shown in **Figure 11**, computed via **Eq. 45.**

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

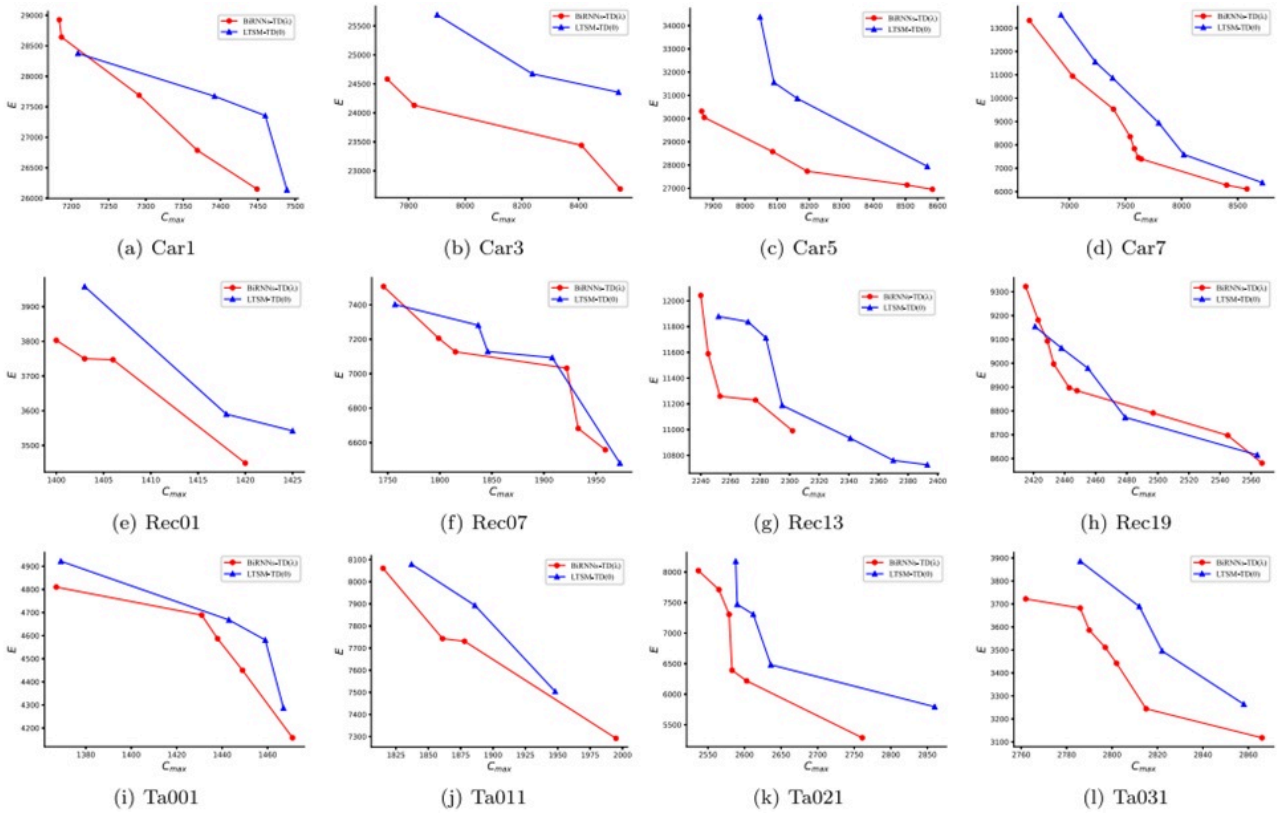
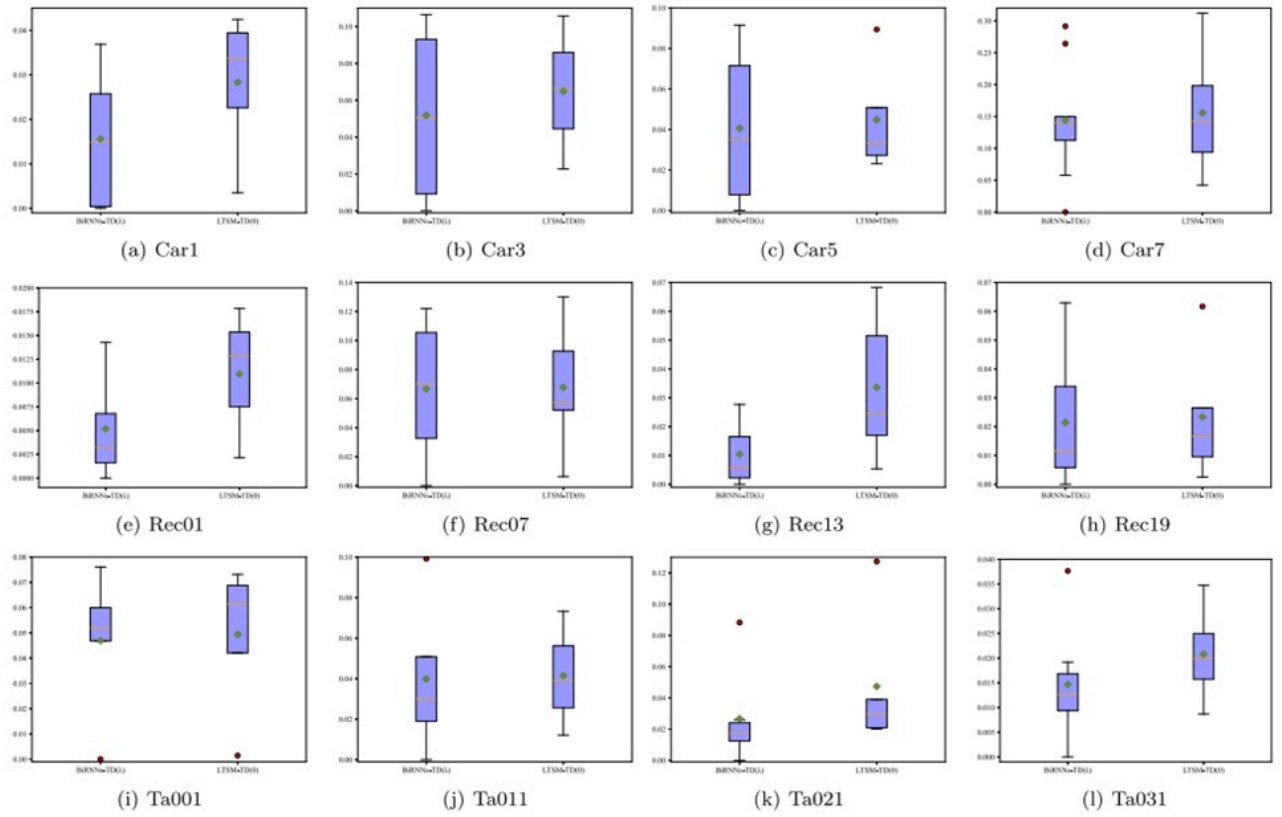


Figure 10. The Pareto fronts obtained by both algorithms.



© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

Figure 11. The boxplot of different instances.

The configuration of the Pareto front serves as a vital index for judging the efficacy of multi-objective optimization strategies. In standard bi-objective optimization contexts, this front is conventionally visualized as a curve. The proximity of this resulting curve to the origin of the 2D Cartesian plane directly reflects the high caliber of the generated Pareto front. As illustrated in **Figure 10**, the curve depicted in red denotes the Pareto front generated by the BiRNNs-TD(λ) approach, whereas the blue curve outlines the front generated by the LSTM-TD(0) method. The visual data clearly confirms that BiRNNs-TD(λ) generates a greater volume of Pareto-optimal solutions and exceeds the performance of LSTM-TD(0) across nearly all tested scenarios. Additionally, data from **Figure 11** confirms that the BiRNNs-TD(λ) model yields reduced ARPD metrics compared to the LSTM-TD(0) model in the majority of cases, supplying extra validation regarding the overall superiority of the newly proposed BiRNNs-TD(λ) framework.

Conclusion

In the present study, the DRL-driven BiRNNs-TD(λ) architecture was introduced to solve the energy-efficient variation of the NFSP, with the dual objectives of minimizing total makespan alongside overall energy usage. Initially, a specialized mathematical model was constructed to represent the energy-efficient NFSP accurately. Following this, the NFSP was reformulated into a standard MDP format, and a comprehensive action space comprising 28 standard scheduling protocols was assembled to serve as the input mechanism for the analyzed NFSP. By analyzing both the macro and micro characteristics of the NFSP environment, a collection of 15 precise state features was derived. Furthermore, specific reward frameworks were formulated to synchronize with the distinct optimization targets. To successfully isolate and process the state features, an MLP structure anchored by BiRNN blocks was utilized. The multi-step TD(λ) approach was integrated to compute the underlying state value function and formulate varying operational policies. Moreover, a customized evaluation dataset tailored for the energy-efficient NFSP was established, drawing from well-known classical benchmark problems. The empirical findings validated both the functional effectiveness and the computational efficiency of the developed algorithm.

Despite these successes, it is crucial to recognize specific constraints inherent to the proposed framework, which outline clear pathways for subsequent research and enhancement. Primarily, the caliber of the generated solutions relies heavily on precise parameter configurations, especially concerning the learning rate. Future research initiatives could center on engineering a neural network architecture capable of autonomously modulating the learning rate relative to ongoing training progression, thereby boosting both system adaptability and overall robustness. Secondly, although the current algorithm successfully translates the core scheduling problem into an MDP framework, it currently demonstrates limited capacity to manage conditional problem constraints. Subsequent investigations might explore adapting this algorithmic structure to tackle more intricate dynamic job-shop scheduling scenarios, thereby integrating supplementary operational constraints and facilitating real-time responsive decision-making. Finally, the currently deployed state features and action space within the model might harbor redundant elements. Future enhancement efforts could focus on streamlining the operational action space by eliminating maneuvers that are rarely selected during training, alongside optimizing the isolated state features, all without diminishing the accurate representation and description of the underlying problem.

To summarize, this research presented the innovative DRL-driven BiRNNs-TD(λ) approach to resolve the energy-efficient NFSP, successfully restricting both the total makespan and aggregate energy consumption. The empirical data generated from our testing protocols verified the algorithm's heightened effectiveness and efficiency relative to the benchmark LSTM-TD(0) approach. Nevertheless, future academic efforts must seek to resolve the noted limitations, including the automation of parameter tuning, the expansion of the algorithm's scope to encompass highly complex scheduling dilemmas, and the meticulous refinement of both state characteristics and available action spaces. Overcoming these specific hurdles will drive continued progress in energy-efficient flow-shop scheduling, thereby supporting the broader implementation of intelligent manufacturing methodologies and ecologically sustainable production practices.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

Acknowledgements

None

Conflict of interest

None

Financial support

This research was partially supported by the National Key R&D Programme for International Cooperation (grant no. 2022YFE0114200), the Project of International Cooperation and Exchanges NSFC (grant no. 51861165202), the National Natural Science Foundation of China (grant nos. 51575211, 51705263, 51805330), and the 111 Project of China (grant no. B16019).

Ethics statement

None

Received: 27 Dec 2022 Revised: 04 Mar 2023 Accepted: 21 Mar 2023
Published online: 15 June 2024

Rights and permissions

Open Access The author(s) retain copyright. This article is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. It may be shared and adapted for non-commercial purposes with appropriate attribution, an indication of changes, and distribution of adaptations under the same license. Third-party material may be subject to separate terms identified in its credit line. View the license at https://creativecommons.org/licenses/by-nc-sa/4.0/.

References

1. Zhang C, Song W, Cao Z, Zhang J, Tan PS, Chi X. A review on learning to solve combinatorial optimisation problems in manufacturing. IET Collab Intell Manuf. 2023;5(1):e12072. https://doi.org/10.1049/cim2.12072.

2. Kong M, Pei J, Liu X, Pardalos PM, Mladenović N, Yang H. Energy-efficient scheduling model and method for assembly blocking permutation flow-shop in industrial robotics field. Artif Intell Rev. 2024;57(3):60. https://doi.org/10.1007/s10462-023-10649-3.

3. Zhong L, Luo Q, Duan H, Zhong Y, Li J, Wen X. Improved discrete cuckoo-search algorithm for mixed no-idle permutation flow shop scheduling with consideration of energy consumption. IET Collab Intell Manuf. 2021;3(4):345-55. https://doi.org/10.1049/cim2.12025.

4. Xie J, Gao L, Pan QK, Tasgetiren MF, Li X, Zhang C. Review on flexible job shop scheduling. IET Collab Intell Manuf. 2019;1(3):67-77. https://doi.org/10.1049/iet-cim.2018.0009.

5. Zhang W, Dietterich TG. A reinforcement learning approach to job-shop scheduling. In: Mellish CS, editor. IJCAI: Proceedings of the 14th International Joint Conference on Artificial Intelligence. Vol 95. San Francisco: Morgan Kaufmann Publishers Inc.; 1995. p. 1114-20.

6. He Z, Zhang Y, Wang H, Zhao F, Xu X, Liu Y. Improved Q-learning algorithm for solving permutation flow shop scheduling problems. IET Collab Intell Manuf. 2022;4(1):35-44. https://doi.org/10.1049/cim2.12042.

7. Sun X, Yang Y, Wang L, Zhou M, Li X, Liu S. A deep reinforcement learning based approach for dynamic distributed blocking flowshop scheduling with job insertions. IET Collab Intell Manuf. 2022;4(3):166-80. https://doi.org/10.1049/cim2.12060.

8. Destouet C, Tlahig H, Bettayeb B, Mazari B, Benbelkacem S, Benfriha K. Flexible job shop scheduling problem under Industry 5.0: a survey on human reintegration, environmental consideration and resilience improvement. J Manuf Syst. 2023;67:155-73. https://doi.org/10.1016/j.jmsy.2023.01.004.

9. Liu CL, Chang CC, Tseng CJ. Actor-critic deep reinforcement learning for solving job shop scheduling problems. IEEE Access. 2020;8:71752-62. https://doi.org/10.1109/access.2020.2987820.

10. Mawson VJ, Hughes BR. Deep learning techniques for energy forecasting and condition monitoring in the manufacturing sector. Energy Build. 2020;217:109966. https://doi.org/10.1016/j.enbuild.2020.109966.

11. Li L, Lin Q, Zhang X, Gong D, Li J, Wang Z. Bilevel learning for large-scale flexible flow shop scheduling. Comput Ind Eng. 2022;168:108140. https://doi.org/10.1016/j.cie.2022.108140.

12. Tian W, Zhang H. A dynamic job-shop scheduling model based on deep learning. Adv Prod Eng Manag. 2021;16(1):23-36. https://doi.org/10.14743/apem2021.1.382.

13. Wang H, Liu M, Shen W. Industrial-generative pre-trained transformer for intelligent manufacturing systems. IET Collab Intell Manuf. 2023;5(2):e12078. https://doi.org/10.1049/cim2.12078.

14. Wang H, Liu Y, Zhou M, Yang J, Zhao F, Zhang C. Multi-objective reinforcement learning framework for dynamic flexible job shop scheduling problem with uncertain events. Appl Soft Comput. 2022;131:109717. https://doi.org/10.1016/j.asoc.2022.109717.

15. Shahrabi J, Adibi MA, Mahootchi M. A reinforcement learning approach to parameter estimation in dynamic job shop scheduling. Comput Ind Eng. 2017;110:75-82. https://doi.org/10.1016/j.cie.2017.05.026.

16. Luo C, Zhao F, Wang L, Yang B, Wang H, Zhang Y. A Q-learning memetic algorithm for energy-efficient heterogeneous distributed assembly permutation flowshop scheduling considering priorities. Swarm Evol Comput. 2024;85:101497. https://doi.org/10.1016/j.swevo.2024.101497.

17. Ren J, Ye C, Yang F. Solving flow-shop scheduling problem with a reinforcement learning algorithm that generalizes the value function with neural network. Alex Eng J. 2021;60(3):2787-800. https://doi.org/10.1016/j.aej.2021.01.030.

18. Zhao F, Li G, Wang L, Yang B, Liu Y, Zhang Y. A reinforcement learning-driven brain storm optimisation algorithm for multi-objective energy-efficient distributed assembly no-wait flow shop scheduling problem. Int J Prod Res. 2023;61(9):2854-72. https://doi.org/10.1080/00207543.2022.2070786.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

19. Li S, Wang H, Liu Y, Yang J, Zhao F, Zhang C. Machine learning-based scheduling: a bibliometric perspective. IET Collab Intell Manuf. 2021;3(2):131-46. https://doi.org/10.1049/cim2.12004.

20. Xiao P, Zhang Z, Liu Q, Wang L, Li X, Chen Y. Non-permutation flow shop scheduling problem based on deep reinforcement learning. Comput Integr Manuf Syst. 2021;27:193-206.

21. Han BA, Yang JJ. Research on adaptive job shop scheduling problems based on dueling double DQN. IEEE Access. 2020;8:186474-95. https://doi.org/10.1109/access.2020.3029868.

22. Pan Z, Wang L, Wang W, Liu Y, Zhao F, Zhang C. Deep reinforcement learning based optimization algorithm for permutation flow-shop scheduling. IEEE Trans Emerg Top Comput Intell. 2021;7(4):983-94. https://doi.org/10.1109/tetci.2021.3098354.

23. Song W, Chen X, Li Q, Cao Z, Zhang J, Wang H. Flexible job-shop scheduling via graph neural network and deep reinforcement learning. IEEE Trans Ind Inf. 2022;19(2):1600-10. https://doi.org/10.1109/tii.2022.3189725.

24. Meng L, Zhang C, Ren Y, Li X, Gao L, Wang W. Constraint programing for solving four complex flexible shop scheduling problems. IET Collab Intell Manuf. 2021;3(2):147-60. https://doi.org/10.1049/cim2.12005.

25. Šeda M. Mathematical models of flow shop and job shop scheduling problems. Int J Pure Math Sci. 2007;1(7):307-12.

26. Singh H, Oberoi JS, Singh D. Multi-objective permutation and non-permutation flow shop scheduling problems with no-wait: a systematic literature review. Oper Res. 2021;55(1):27-50. https://doi.org/10.1051/ro/2020055.

27. Rossit DA, Tohmé F, Frutos M. The non-permutation flow-shop scheduling problem: a literature review. Omega. 2018;77:143-53. https://doi.org/10.1016/j.omega.2017.05.010.

28. Goli A, Ala A, Hajiaghaei-Keshteli M. Efficient multi-objective meta-heuristic algorithms for energy-aware non-permutation flow-shop scheduling problem. Expert Syst Appl. 2023;213:119077. https://doi.org/10.1016/j.eswa.2022.119077.

29. Yenisey MM, Yagmahan B. Multi-objective permutation flow shop scheduling problem: literature review, classification and current trends. Omega. 2014;45:119-35. https://doi.org/10.1016/j.omega.2013.07.004.

30. Panwalkar SS, Iskander W. A survey of scheduling rules. Oper Res. 1977;25(1):45-61. https://doi.org/10.1287/opre.25.1.45.

31. Zhang Z, Wang L, Wang S, Zhou M, Li X, Zhao F. Flow shop scheduling with reinforcement learning. Asia Pac J Oper Res. 2013;30(5):1350014. https://doi.org/10.1142/s0217595913500140.

32. Zhang Z, Wang L, Wang S, Zhou M, Li X, Zhao F. Minimizing mean weighted tardiness in unrelated parallel machine scheduling with reinforcement learning. Comput Oper Res. 2012;39(7):1315-24. https://doi.org/10.1016/j.cor.2011.07.019.

33. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436-44. https://doi.org/10.1038/nature14539.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press

34. Du X, Cai Y, Wang S, Zhang L, Li H, Zhao Q. Overview of deep learning. In: 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC). IEEE; 2016. p. 159-64.

35. Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science. 2006;313(5786):504-7. https://doi.org/10.1126/science.1127647.

36. Kiranyaz S, Ince T, Gabbouj M. Real-time patient-specific ECG classification by 1-D convolutional neural networks. IEEE Trans Biomed Eng. 2015;63(3):664-75. https://doi.org/10.1109/tbme.2015.2468589.

37. Sutton RS. Learning to predict by the methods of temporal differences. Mach Learn. 1988;3(1):9-44. https://doi.org/10.1007/bf00115009.

38. Carlier J. Ordonnancements a contraintes disjonctives. Oper Res. 1978;12(4):333-50.

39. Reeves CR. A genetic algorithm for flowshop sequencing. Comput Oper Res. 1995;22(1):5-13. https://doi.org/10.1016/0305-0548(93)e0014-k.

40. Taillard E. Benchmarks for basic scheduling problems. Eur J Oper Res. 1993;64(2):278-85. https://doi.org/10.1016/0377-2217(93)90182-m.

41. Wang Z, Zhang J, Song W, Cao Z, Li X, Liu Y, et al. Solving non-permutation flow-shop scheduling problem via a novel deep reinforcement learning approach. Comput Oper Res. 2023;151:106095. https://doi.org/10.1016/j.cor.2022.106095.

© 2024 The Author(s). | Published by Institute of Smart Energy Systems Press