Li, Funing et al.

Article — Published Version

A transformer-based deep reinforcement learning approach for dynamic parallel machine scheduling problem with family setups

Journal of Intelligent Manufacturing

Provided in Cooperation with:
Springer Nature

Suggested Citation: Li, Funing et al. (2024) : A transformer-based deep reinforcement learning approach for dynamic parallel machine scheduling problem with family setups, Journal of Intelligent Manufacturing, ISSN 1572-8145, Springer US, New York, NY, Vol. 36, Iss. 7, pp. 4735-4768,
https://doi.org/10.1007/s10845-024-02470-8

This Version is available at:
https://hdl.handle.net/10419/330625

Journal of Intelligent Manufacturing (2025) 36:4735–4768
https://doi.org/10.1007/s10845-024-02470-8

A transformer-based deep reinforcement learning approach for dynamic parallel machine scheduling problem with family setups

Funing Li[1,2] · Sebastian Lang[2,3] · Yuan Tian[4] · Bingyuan Hong[5] · Benjamin Rolf[2] · Ruben Noortwyck[1] · Robert Schulz[1] · Tobias Reggelin[2]

Received: 5 March 2024 / Accepted: 16 July 2024 / Published online: 8 August 2024

Abstract

The parallel machine scheduling problem (PMSP) involves the optimized assignment of a set of jobs to a collection of parallel machines, which is a proper formulation for the modern manufacturing environment. Deep reinforcement learning (DRL) has been widely employed to solve PMSP. However, the majority of existing DRL-based frameworks still suffer from generalizability and scalability. More specifically, the state and action design still heavily rely on human efforts. To bridge these gaps, we propose a practical reinforcement learning-based framework to tackle a PMSP with new job arrivals and family setup constraints. We design a variable-length state matrix containing full job and machine information. This enables the DRL agent to autonomously extract features from raw data and make decisions with a global perspective. To efficiently process this novel state matrix, we elaborately modify a Transformer model to represent the DRL agent. By integrating the modified Transformer model to represent the DRL agent, a novel state representation can be effectively leveraged. This innovative DRL framework offers a high-quality and robust solution that significantly reduces the reliance on manual effort traditionally required in scheduling tasks. In the numerical experiment, the stability of the proposed agent during training is first demonstrated. Then we compare this trained agent on 192 instances with several existing approaches, namely a DRL-based approach, a metaheuristic algorithm, and a dispatching rule. The extensive experimental results demonstrate the scalability of our approach and its effectiveness across a variety of scheduling scenarios. Conclusively, our approach can thus solve the scheduling problems with high efficiency and flexibility, paving the way for application of DRL in solving complex and dynamic scheduling problems.

Keywords Deep reinforcement learning · Dynamic parallel machine scheduling · New job arrivals · Family setups · Multi-head attention

✉ Funing Li
funing.li@ift.uni-stuttgart.de

✉ Sebastian Lang
sebastian.lang@ovgu.de

Yuan Tian
yutian@ethz.ch

Bingyuan Hong
hongby@zjou.edu.cn

Benjamin Rolf
benjamin.rolf@ovgu.de

Ruben Noortwyck
ruben.noortwyck@ift.uni-stuttgart.de

Robert Schulz
robert.schulz@ift.uni-stuttgart.de

Tobias Reggelin
tobias.reggelin@ovgu.de

[1] Institute of Mechanical Handling and Logistics, University of Stuttgart, Holzgartenstraße 15B, 70174 Stuttgart, Germany

[2] Institute of Logistics and Material Handling Systems, Otto von Guericke University Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany

[3] Fraunhofer Institute for Factory Operation and Automation IFF, Sandtorstraße 22, 39106 Magdeburg, Germany

[4] Institute for Building Material, ETH Zürich, Stefano-Franscini-Platz 5, 8093 Zürich, Switzerland

[5] National-Local Joint Engineering Laboratory of Harbor Oil and Gas Storage and Transportation Technology, Zhejiang Provincial Key Laboratory of Petrochemical Pollution Control, School of Petrochemical Engineering and Environment, Zhejiang Ocean University, Zhoushan 316022, China

Introduction

Modern international supply chains have been increasingly getting more sophisticated, which poses a great challenge for the scheduling of manufacturing processes. For example, the unforeseen arrival of new jobs can disturb planned static schedules and dramatically decrease the production efficiency (Ouelhadj & Petrovic, 2009). Additionally, the diversified demands of the global market require grouping jobs into distinct families based on their processing characteristics, with each shift from one family to another incurring additional setup time. An appropriate solution is hence supposed to adaptively optimize the processing sequence of jobs across various families under the dynamic arrival of jobs.

The scheduling problem in the manufacturing process has been commonly formulated as a parallel machine scheduling problem (PMSP), which offers a practical mathematical framework for optimization and development across diverse domains, for example, in the fields of semiconductor manufacturing (Zhang & Chen, 2022), thin-film-transistor liquid crystal display manufacturing (Shin & Leon, 2004), and offshore oil and gas industry (Abu-Marrul et al., 2021).

In this work, we focus on a particular instance of the PMSP characterized by constraints related to new job arrivals and family setups. This approach effectively abstracts the scheduling process pertinent to the modern manufacturing environment described previously. The objective function is to minimize the total tardiness. A detailed mathematical formulation is defined in Sect. "Problem formulation". It is worth noticing that this problem is non-trivial, since a previous study without the consideration of these two constraints has been proven to be NP-hard (Biskup et al., 2008).

Under this formulation, the majority of conventional approaches for tackling these problems can be broadly classified into two categories: rule-based methods and metaheuristic algorithms (Đurasević & Jakobović, 2023). However, rule-based methods often fail to deliver high-quality results, since they rely on predefined rules that do not account for the unique characteristics of each specific scheduling problem. Meanwhile, metaheuristic algorithms require numerous iterations for yielding an appropriate solution for one instance, which is computationally expensive. Furthermore, both conventional approaches struggle to respond to dynamic environments.

To obtain a proper solution in a time-efficient manner, several research studies have employed reinforcement learning (RL), including deep reinforcement learning (DRL) based approach to solve the PMSP (Kayhan & Yildiz, 2023). RL is a promising branch in the field of machine learning and has achieved remarkable development in many challenging decision-making tasks, such as controlling the tokamak plasma for nuclear fusion (Degrave et al., 2022) and discovering faster matrix multiplication algorithms (Fawzi et al., 2022). In the RL-based approach, an agent is employed to learn an optimal policy through interaction with the environment. This mechanism aligns well with the process of most scheduling problems, making RL a promising alternative solution for the PMSP.

However, even though RL-based approaches show superiority over rule-based methods and metaheuristic algorithms in solving PMSPs, they still have limitations. First, most of them do not tackle the scheduling problem in an end-to-end manner, which means they cannot select job directly based on the raw information of the manufacturing environment. On the contrary, they rely on hand-crafted state features and predefined dispatching rules (Guo et al., 2020; Luo, 2020; Luo et al., 2022), which require extensive domain knowledge and lead to tedious work. Moreover, with the integration of hand-crafted state features and pre-defined rules, human bias might be introduced, and the potential of the data-driven method, i.e., RL, could not be fully leveraged. Second, the majority of the existing RL-based approaches need to be re-designed and re-trained when being applied to larger instances (Lang et al., 2020; Liu et al., 2020), which is highly time-consuming and hence restricts their practical applicability in diverse and evolving manufacturing environments.

To the best of our knowledge, the only work that breaks-through these two limitations is proposed by Li et al. (2024). They represent the PMSP instance with variable-length matrix and then employ a recurrent neural network (RNN) based DRL agent to process the matrix, enabling an end-to-end scheduling process with high scalability. However, this approach equally suffers from significant drawbacks. First, the RNN models process the independent jobs in a sequential manner, making the index of jobs can affect the performance, which is a highly undesirable attribute in the PMSP scenarios. Additionally, the sequential processing of state matrix could also impact the performance when managing a large number of independent jobs, as capturing relationships between distantly positioned jobs in the matrix becomes increasingly complex. More crucially, the state matrix solely contains the information of the current idle machine, neglecting the overall state of all machines at a given decision time point. This restricted perspective limits the agent's ability to make globally optimal decisions, as it is only considering the immediate circumstances of a single machine. Furthermore, purely expanding the matrix to encompass information regarding all machines brings new challenges, particularly the need to consider the relationships between all jobs and all machines, in addition to solely comparing jobs with each other. Handling these dual distinct dependencies simultaneously could also be problematic for RNN models.

Inspired by the current advantages of Transformer model (Vaswani et al., 2017) over RNN models in the field of natural language processing (NLP), we propose a Transformer-based DRL framework for solving PMSP. To do so, we tailor the

Transformer model to fit it for the scheduling problem, since it has originally been designed specifically for NLP. This enables us to overcome the aforementioned limitations and the drawbacks of the RNN-based DRL approach. The tailored Transformer model utilizes the multi-head attention mechanism to process the available jobs in a parallel manner, avoiding the impact of job index on performance. Meanwhile, this mechanism is capable of effectively process diverse relationships in one state matrix, allowing the state matrix to be extended to include all the global information.

With the motivations above, we train this Transformer-based DRL agent with the proximal policy optimization (PPO) algorithm (Schulman et al., 2017), to minimize the total tardiness of PMSPs with family setups and new job arrivals constraints. An overview of the proposed approach is demonstrated in Fig. 1 and the contribution of this work is three-fold:

1. We present a more comprehensive state representation for the PMSP that considers not only the information of all jobs and the current idle machine but also the global state of all machines at the decision time point. This expanded representation enables the agent to make more globally optimal decisions. Meanwhile, this novel state representation retains scalability, allowing the agent trained on a small-size instance to be directly applied to large-size instances, which would be time- and resource-efficient.
2. We represent the DRL agent with an elaborately modified Transformer model featuring a two-head attention mechanism. The proposed agent can efficiently handle the multiple intertwined dependencies introduced by the novel state representation, which contains diverse information about machines and jobs and thus increases its complexity. Additionally, the agent can directly output the index of the selected job as the action, providing a direct scheduling process without relying on any pre-defined rules. This end-to-end framework not only leverages the potential of the data-driven method, but also reduces tedious manual work involved in defining and selecting rules. Furthermore, the Transformer-based agent processes jobs in parallel to ensure that the performance is not affected by the sequence of jobs in the matrix, as expected in industrial applications.
3. We validate the superiority of the proposed agent by comparing it with agents employing one-head and four-head attention mechanisms. We also execute extensive numerical experiments with various parameter configurations to demonstrate the generalization capability of the proposed agent. A learning-based method, a metaheuristic method, and a classic dispatching rule are taken into comparison to illustrate the superiority of the proposed approach. Moreover, an ablation experiment is conducted to illustrate the importance of the global information, where comparative DRL agents are trained without the state of all machines.

The remainder of this article is organized as follows. Section "Literature review" introduces the related works on solving dynamic scheduling problems. Section "Background" presents the background of RL and Transformer model. Section "Problem formulation" formulates the mathematical model of the problem addressed in this paper. Section "Proposed approach" establishes the details of the proposed approach. Section "Numerical experiments" provides the results of numerical experiments. Section "Discussion" carries out a discussion on the application of the proposed approach. The conclusions are finally drawn in Sect. "Conclusion".

Literature review

Scheduling problems, including PMSP, involve assigning jobs to machines over time to optimize given objectives, such as minimizing total tardiness or makespan. This kind of problem is central to various application domains, for instance semiconductor manufacturing (Ghaedy-Heidary et al., 2024) and vehicle planning (Zhang et al., 2021). Challenges in these problems arise from diverse constraints such as new job arrivals, sequence-dependent setup times (SDST), and machine availability, and many of these problems are inherently NP-hard (Blazewicz et al., 1991), indicating that they do not have known polynomial-time solutions and can be computationally intensive to solve. Therefore, these problems have attracted considerable attention from researchers to develop innovative and efficient methods.

Non-RL-based methods

To obtain the optimal solution of a scheduling problem, many researchers formulate the problem to a mathematical model and solve it with optimization techniques. For instance, Heydari and Aazami (2018) address a job shop scheduling problem (JSP) under SDST constraint considering two objective function, namely makespan minimization and maximum tardiness minimization. They first mathematically describe the problem as a mixed integer nonlinear programming model and then convert it into mixed integer linear programming (MILP) model. A set of Pareto optimal solutions is obtained by utilizing ε-constraint method. Hu et al. (2024) consider an unrelated PMSP with new job arrivals constraint to minimize the total tardiness. They formulate the problem into MILP model and then solve it by employing a commercial solver. Optimal solutions are yield on several small-scale-instances with 20 jobs. However, due to the NP-hard nature of the scheduling problems, mathematical modeling-based

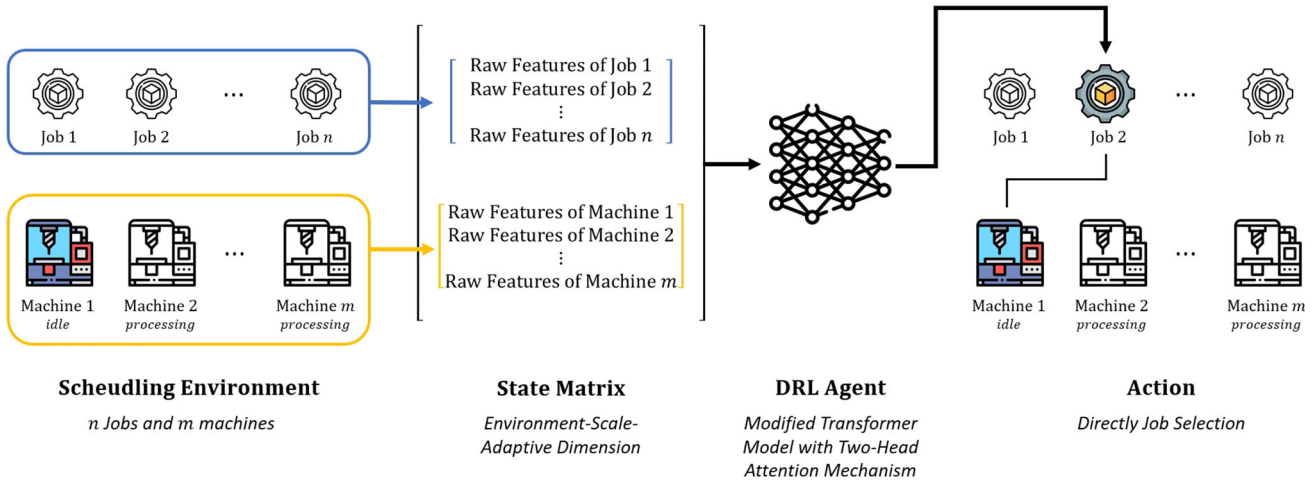

Fig. 1 Overview of the proposed approach

solutions become extremely computationally intensive when tackling large-scale-instances. Therefore, Hu et al. (2024) also develop a metaheuristic method based on neighborhood search for addressing large-scale-instances up to 200 jobs, on which the metaheuristic method outperforms the mathematical modeling-based method in terms of both solution quality and computational time.

Under this motivation, the metaheuristic methods are widely utilized for solving scheduling problems (Ezugwu, 2024; Pellerin et al., 2020; Para et al., 2022). Rolf et al. (2020) propose a genetic algorithm (GA) to to assign a given set of scheduling rules to several specific time point during the scheduling process. The combination of rules provided by this GA method can outperform any of the composite dispatching rule in solving the addressed hybrid flow shop scheduling problem. Although the metaheuristics is less computationally intensive relative to the mathematical modeling-based approach, they rely on numerous iterations for yielding an appropriate solution for one instance, restricting their applications in scenarios requiring immediate decision-making.

To obtain a real-time solution, many rule-based approaches are developed. Xiong et al. (2017) propose four dispatching rules for a dynamic JSP with the consideration of extended technical precedence constraints and the due date tightness. The proposed rules have promising performance when the due date is relatively loose. Didden et al. (2023) take the entire manufacturing system into consideration by utilizing multiple agents, where the release of jobs, their assignment on the machines, and their sequencing within the queue are all realized by the respective agents through different rules. Meanwhile, they also propose several algorithms to learn weights for proper linear combinations of the rules. In the computational experiments, their proposed MAS outperforms other multi-agent systems and common dispatching rules in several performance measures, including minimizing mean weighted tardiness. However, due to the pre-defined nature of rule-based approaches, they get comparatively poor results on instances with different environment settings, for example different due date tightness. This inability to respond to the changes imitates their application in variable manufacturing environments. In addition, designing new rule or selecting new combination of rules for new environments requires substantial expert knowledge and leads to tedious work.

Therefore, approaches that can provide appropriate solutions in real time under dynamic environments are highly desired.

RL-based methods

In contrast to non-RL-based approaches, RL-based approaches learn a policy through interaction with the environment, enabling them to provide proper results in real time and to respond to environment changes immediately.

Over the last half decade, the application of RL techniques for solving scheduling problems has been subject of many publications. The extensive research in this field is also reflected by a multitude of systematic literature reviews covering RL for scheduling general (Kayhan & Yildiz, 2023; Shyalika et al., 2020) or with focus on specific application domain, for instance production and smart manufacturing (Esteso et al., 2023; Li et al., 2023; Panzer & Bender, 2022; Waubert de Puiseau et al., 2022; Panzer et al., 2021; Wang et al., 2021), maintenance and repair (Ogunfowora & Najjaran, 2023), supply chain management (Rolf et al., 2023), cloud computing (Hou et al., 2024; Zhou et al., 2021), wireless networks (Hurtado Sánchez et al., 2022; Frikha et al., 2021; Erick & Folly, 2020; Yau et al., 2013) or power systems and

smart grids (Yu et al., 2021; Arwa & Folly, 2020; Zhang et al., 2019, 2018).

In view of the large number of existing literature analyses, a detailed review of RL applications for scheduling problems seems redundant and would go beyond the scope of this paper. In accordance with the topic of this paper, we want to focus our literature review on the following two aspects: (1) solving the PMSP with RL, and (2) deploying RL-trained Transformer models for scheduling problems in general. With this two perspectives, our approach not only demonstrates significant improvements in solving PMSP with RL but also offers valuable insights into the utilization of Transformer models for addressing a wide range of scheduling challenges.

RL methods for computing the PMSP

We first review the applications of RL technique on the PMSP based on a recent survey article of Kayhan and Yildiz (2023). Their analyses include and even go beyond those of other review articles that examine RL for the PMPS in detail, such as the work of Wang et al. (2021).

In summary, Kayhan and Yildiz (2023) investigate 13 publications describing various RL techniques for computing different variants of the PMSP. The majority of publications consider some PMSP with unrelated machines (Csáji & Monostori, 2005; Zhang et al., 2007; Iwamura et al., 2009; Palombarini & Martínez, 2009; Zhang et al., 2011; Palombarini & Martínez, 2012a, b; Zhang et al., 2012; Ábrahám et al., 2019; Zhou et al., 2020), two publications a PMSP with identical machines (Yuan et al., 2013, 2016), and only one publication a PMSP with uniform machines (Guo et al., 2020). More precisely, the problems analyzed are subject to different constraints. These include machine breakdowns (Csáji & Monostori, 2005; Palombarini & Martínez, 2012a, b; Yuan et al., 2016), precedence constraints (Csáji & Monostori, 2005; Zhang et al., 2011; Ábrahám et al., 2019) and SDST (Zhang et al., 2007, 2011; Palombarini & Martínez, 2012b). Furthermore, different objective functions are taken into, such as makespan (Csáji & Monostori, 2005; Ábrahám et al., 2019; Zhou et al., 2020), total tardiness (Palombarini & Martínez, 2009, 2012a, b), weighted total tardiness (Zhang et al., 2007, 2012), total costs (Zhang et al., 2011) or some mult-criteria objective function (Iwamura et al., 2009; Yuan et al., 2013, 2016; Guo et al., 2020). Looking at the applied RL technique, most publications rely on some Q-Learning strategy (Csáji & Monostori, 2005; Zhang et al., 2007; Iwamura et al., 2009; Zhang et al., 2012; Ábrahám et al., 2019; Zhou et al., 2020; Yuan et al., 2013, 2016; Guo et al., 2020). Only a few publications analyze other RL algorithms, such as SARSA (Zhang et al., 2011) or relational RL strategies (Palombarini & Martínez, 2009, 2012a, b).

Another interesting aspect is how exactly the trained RL agent solves the corresponding PMSP, i. e. how the actions of an agent construct and manipulate a schedule. The vast majority of publications (Zhang et al., 2007, 2011, 2012; Zhou et al., 2020; Yuan et al., 2013, 2016; Guo et al., 2020) describes an agent that constructs a schedule indirectly by selecting a heuristic, for instance a dispatching rule, from a set of heuristics, which in turn determines the next scheduling decision (e. g. selecting a job to be processed on an idle machine). The remaining publications tend to describe individual solutions. In one reported approach, the agent assigns tasks of jobs to a machines, i. e. each action is associated with a machine (Csáji & Monostori, 2005), while another publication suggests to train an agent that selects in each time step a parameter from a set of parameters to compute a utility function, which in turn prioritizes the selection of job operations and machines (Iwamura et al., 2009). Three publications, which, however, stem from the same main author, describe an approach, in which an RL-trained agent repairs unfeasible schedules by selecting from a set of predefined repair operations (Palombarini & Martínez, 2009, 2012a, b). Only one publication presents an agent that selects the next task of a job to be processed on an idle machine (Ábrahám et al., 2019), making it comparable to the approach described in this article. However, since the authors use tabular Q-learning, the state and action space is tailored exactly to one problem instance, which makes it difficult to apply the learned strategy to problem instances of different dimensions.

RL-based approaches also yield promising achievements in solving real-world PMSPs. Rodríguez et al. (2022) consider the scheduling problem in the context of predictive maintenance for a set of parallel machines. To minimize the machine breakdown time and prevent failure, they proposed a DRL-based approach for technicians scheduling. A deep neural network, in particular a Multi-Layer Perceptron (MLP), is utilized to represent of the agent, which can select a proper technician for the current maintenance task based on the availability and skills of each job. Zhang et al. (2021) model the electric vehicles (EVs) charging scheduling problem as a PMSP and employ the DRL-based approach to minimize the total charging time. The agent can assign a suitable charging station for each EV based on the availability of all stations. This approach can significantly reduce the total charging time on all EVs in comparison of two heuristic-based baselines. Paeng et al. (2021) investigate the PMSP with family setups constraint in a semiconductor manufacturing environment. To be more specific, they train an MLP with RL algorithm to minimize the total tardiness. The trained network computes five matrices consisting of respectively five sets of statistical information, based on which it selects a suitable family of jobs. The particular selection of jobs inside the selected family is executed by a dispatching rule. Their method exhibits significant advantages over rule-based and metaheuristic methods on eight given datasets.

However, although the aforementioned works demonstrate advantages over non-learning-based methods with respect to efficiency and quality, they still encounter limitations. First, most of them cannot perform the scheduling process directly, in contrast, they rely on a set of predefined dispatching rules as actions for outputs, and several statistical information derived from the scheduling environment as states for inputs. The combination of dispatching rules cannot cover all possible solutions and is not guaranteed to cover the optimal solution, while the statistical information does not provide a complete description of the whole scheduling environment. Therefore, the potential of RL as a data-driven method is not be fully leveraged under this non-end-to-end manner. Moreover, this kind of RL methods offers restricted flexibility, since the dispatching rules set may require to be re-selected when applied to different scheduling environment, which requires extensive domain knowledge and leads to tedious human effort.

Second, despite the fact that some RL-based methods can schedule directly, they usually cannot be flexible enough to handle problems with arbitrary scales. These methods require re-designed and re-trained when being applied to larger instances with more machines or jobs, since they are implemented through pre-defined tables or neural networks with fixed dimensional, i.e., MLP. The re-training process is highly time-consuming and hence restrict their practical applicability in diverse and evolving production environments. The only work that can directly solve PMSP instances of any scales is proposed by Li et al. (2024) based on the framework of Lang et al. (2019). However, as elaborated in Sect. "Introduction", this work does not take the global information of the scheduling environment into consideration, and the manner it processes the state matrix does not align with the nature of PMSPs.

To this end, we develop a novel DRL-based framework with highly scalability for solving the proposed PMSP, where the scheduling process is executed end-to-end and the complete information of the environment is considered. A Transformer model is employed for representation of the DRL agent. The upper part of Table 1 summarizes all the aforementioned literature on applying RL to PMSP and provides a comparison with our method.

To further emphasize the novelty of our work and clarify the research gap, in the next subsection, we will briefly discuss whether and in which context RL-trained Transformer models have already been investigated for scheduling problems in general.

Transformer-based RL methods for computing general scheduling problems

Compared to the general literature on RL for scheduling problems, the application of Transformers for scheduling problems is still in its infancy. Against this background, we could only investigate seven publications describing the application of RL-trained Transformer models for computing scheduling problems. Due to the scarcity of literature and the lack of detailed review articles, we will discuss the papers on the use of RL-trained Transformers for scheduling problems in more detail.

Chen et al. (2022) present an RL-trained Transformer, which is trained on the selection of dispatching rules in a dynamic job shop scheduling problem. The input of the Transformer is a disjunctive graph describing the scheduling problem. A particularity of the implementation is that the Transformer is capable to extract and generate higher features of the disjunctive graph representation to facilitate scheduling decisions. The objective is to minimize the makespan. Results show that the makespan is in average 11.67% smaller compared to single dispatching rules, metaheuristics and non-Transformer RL strategies.

Wang et al. (2022) consider a cloud manufacturing scheduling problem, in which an agent allocates manufacturing tasks to enterprises, while optimizing a multi-criteria objective function including the minimization of makespan and total cost as well as the maximization of the reliability. A state comprises the task to be allocated at the current point in time, described by several attributes, as well as attributes of all enterprises to which the task can be allocated. An outstanding particularity of the authors approach is that it utilizes the multi-criteria objective function as reward function, which in turn implies that the engineering of a separate time-step based reward function is not necessary.

Xu and Zhao (2022) investigate a cloud job scheduling problem, in which jobs composed of different tasks must be allocated to computing resources. In particular, each task of each job must be allocated in three stages. First, each task must be allocated to a data center. Second, each task must be allocated to a server node within the selected data center. Third, each task must be allocated to an application container of the selected server node. The authors present a multi-agent RL approach, in which allocation decisions of each stage are conducted by a separate agent. Thus, the agent's actions are associated with data centers, server nodes, and application containers, respectively. The objective is the minimization of the total energy costs. Each agent is not represented by a whole Transformer model, but incorporates a self-attention layer within its model architecture. The purpose of the self-attention layer is to learn key features from the raw input states. A state comprises the CPU, RAM and hard drive capacity of each application container, server node, and data center, respectively.

Zhao et al. (2022) present an RL-trained Transformer and evaluate its performance on a job shop scheduling problem, with the objective of minimizing the makespan. Here, the actions of the agent point to the available jobs that can be

processed in the next time step, while a state comprises multiple attributes of all available jobs. The authors approach performs in average better than several dispatching rules and a GA. In three problem instance, the RL-trained Transformer is even able to compute the optimal solution.

Chen et al. (2023) train a Transformer with RL to compute solutions for a job shop scheduling problem. A special aspect of their implementation is that, analogous to the work of Chen et al. (2022), the complete problem is described as disjunctive graph and considered as a single, forward propagable state. Thus, the intention of using a Transformer is to autonomously derive significant features from the problem data to determine appropriate scheduling decisions. The output of the agent is a global sequence of tasks from which the local processing sequence of tasks on each machine can be derived. The Transformer model outperforms multiple dispatching rules and a Tabu search algorithm. For large-scale problems, the authors approach performs also better than Google OR-Tools.

Chen et al. (2023) consider a cloud edge task scheduling problem comparable to the problem in the paper of Wang et al. (2022). The authors pursue to optimize a multi-criteria objective function including the minimization of the production lead time, production costs, and the variance of utilization as well as the maximization of the work rate of scheduling resources. To do so, the agent observes a one-dimensional vector as a state, which contains, for instance, information about the locations of scheduling resources, the number of unfinished scheduling tasks, and the number of scheduling tasks currently in process. For each observed state, the agent selects a dispatching rule from a set of 9 dispatching rules. The agent comprises self-attention layers to extract additional features from a concatenated vector consisting of the current state as well as the selected action and the received reward, both from the last time step. The authors state that their approach solve the cloud edge task scheduling problem more effectively in comparison to single dispatching rules and other RL algorithms.

Song et al. (2023) investigate an RL-trained Transformer for minimizing the makespan in a dynamic job shop scheduling problem as in the work of Chen et al. (2022). In particular, the agent selects in each time step a dispatching rule from a set of 14 dispatching rules to determine scheduling decisions. As in the aforementioned paper (Chen et al., 2022, 2023), a Transformer model is used to autonomously generate states from the disjunctive graph representation of the scheduling problem. The reward function compares the utilization of machines in the current and in the previous time step, where an increase of the utilization is rewarded and a decrease is penalized. The authors approach outperforms the use of single dispatching rules, a GA and other reinforcement learning methods by in average 15.93%.

The literature on solving scheduling problems in general utilizing Transformer-based RL methods is summarized in the lower part of Table 1. It is evident that these works still do not simultaneously break through the two limitations stated above.

The first limitation is from the perspective of direct scheduling, i.e., end-to-end scheduling. In particular, many approaches equally execute the scheduling process in a indirect fashion by employing a set of dispatching rules as the action space. As we discussed in the previous subsection, this action representation does not leverage the data-driven nature of the learning-based approach. The selection of dispatching rules requires extensive domain knowledge, while the selected combination is not guaranteed to cover the optimal solution. Additionally, several works apply Transformer-based RL for solving JSP. Instead of utilizing the raw features of the scheduling environment, these problems are typically formulated as disjunctive graphs, which describe the complex sequences of operations of jobs across multiple machines. However, such formulation is less applicable to PMSP, since the challenges of PMSP lie in efficient job allocation rather than operation sequencing. Moreover, the process of setting up this graph-based model requires significant domain expertise, increasing complexity in application.

The second limitation is from the perspective of scalability, namely whether the method can be applied retraining-free to environments with more machines and jobs. Several works also rely on the state representation, whose dimension depends on the number of jobs or machines, and therefore require time-consuming re-training when being applied to significant larger instances. Under the modern dynamic manufacturing environment, this representation restricts the application of the RL-based approach. It is worth noting that the work of Zhao et al. (2022) has realized a certain scalability by proposing a state matrix representation with variable number of rows. In particular, each row corresponds to the features of an available job and the approach can therefore process instance with any jobs. However, to embed the machine-related information into the state representation, the dimension of the job features depends on the number of machines. When the practical scheduling environment has a larger machine number than the predefined dimension, the tedious re-training process is inevitable.

With the previous motivation, we first develop a state representation that can adapt to instances with arbitrary numbers of jobs and machines. The state contains the raw features of all available jobs and machines, providing a global perspective for decision-making. To process the state matrix appropriately, we elaborately modify the Transformer model to represent the RL agent. The agent automatically maps the raw information of the scheduling environment to the priority of each job, providing an end-to-end scheduling manner.

Table 1 Literature summary

RL for PMSP						
Work	Agent	State matrix	State dimension	Action	Re-training free	End-to-end
Ábrahám et al. (2019)	Tabular	Job features	Fixed, Scale-Dependent	Job index	No	Yes
Csáji and Monostori (2005)	Tabular	Job & machine features	Fixed, Scale-Dependent	Job index	No	Yes
Iwamura et al. (2009)	Tabular	Job & machine features	Fixed, Scale-Dependent	Job index	No	Yes
Palombarini and Martínez (2009)	Tabular	Statistical & relational data	16, Scale-Independent	4 swap actions	Yes	No
Palombarini and Martínez (2012a)	Tabular	Statistical & relational data	21, Scale-Independent	4 swap actions	Yes	No
Palombarini and Martínez (2012b)	Tabular	Statistical & relational data	21, Scale-Independent	4 swap actions	Yes	No
Zhang et al. (2011)	Tabular	Statistical data	Fixed, Scale-Dependent	5 dispatching rules	No	No
Zhang et al. (2012)	Tabular	Statistical data	Fixed, Scale-Dependent	5 dispatching rules	No	No
Zhang et al. (2007)	Tabular	Job & machine features	Fixed, Scale-Dependent	5 dispatching rules	No	No
Zhou et al. (2020)	MLP	Machine features	Fixed, Scale-Dependent	Job index	No	Yes
Yuan et al. (2013)	Tabular	Statistical data	4, Scale-Independent	3 dispatching rules	Yes	No
Yuan et al. (2016)	Tabular	Statistical data	3, Scale-Independent	3 dispatching rules	Yes	No
Guo et al. (2020)	Tabular	Statistical data	5, scale-independent	3 dispatching rules	Yes	No
Rodríguez et al. (2022)	MLP	Job & machine features	Fixed, scale-dependent	Job index	No	Yes
Zhang et al. (2021)	MLP	Machine features	Fixed, scale-dependent	Job index	No	Yes
Paeng et al. (2021)	MLP	Statistical data	5, Scale-Independent	Family index	Yes	No
Ours	Transformer	Job & machine features	Variable, scale-adaptive	Job index	Yes	Yes

RL integrated transformer model for scheduling problems in general					
Work	Problem	State representation	Action	Re-training free	End-to-end
Chen et al. (2022)	Job shop scheduling problem	Disjunctive graph	8 dispatching rules	Yes	No
Wang et al. (2022)	Cloud manufacturing scheduling problem	Scale-Dependent Vector	Job index	No	Yes
Xu and Zhao (2022)	Cloud computing job scheduling	Scale-Dependent Vector	Job index	No	Yes
Zhao et al. (2022)	Job shop scheduling problem	Scale-Dependent Matrix	Job index	No	Yes
Chen et al. (2023)	Job shop scheduling problem	Disjunctive graph	Job index	Yes	No
Chen et al. (2023)	Cloud manufacturing scheduling problem	Scale-Dependent Vector	9 dispatching rules	Yes	No
Song et al. (2023)	Graph structure	Disjunctive graph	20 dispatching rules	Yes	No
Ours	Parallel machine scheduling problem	Scale-adaptive matrix	Job index	Yes	Yes

Background

In this section, we briefly review the basic concepts related to RL and Transformer, which form the foundation of our proposed framework.

Basic concepts in RL

RL represents a promising branch of machine learning, concentrating on learning an appropriate policy through interaction with the environment to attain predefined objectives. These environments for training the RL agent are commonly formulated as Markov Decision Process (MDP), offering a structured mathematical framework to describe and analyze decision-making scenarios.

An MDP can be framed as a five-tuple representation (S, A, P, R, γ), where S is a set of all states representing the different conditions that the agent can encounter in the environment, A is a set of actions that the agent can execute, P is the transition probability function describing the likelihood of state transition under a given action, R is the reward function that assigns the reward to the agent for each state transition due to an action, γ is the discount factor that balances immediate versus long-term rewards.

The RL agent interacts with the MDP-formulated environment under a particular policy π. To evaluate the policy, the state value function V is a crucial tool (Sutton & Barto, 2018).

It is defined for each state and represents the expected cumulative reward the agent can achieve starting from that state while following policy π. The value function for state s following policy π can be mathematically formulated as:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \middle| s_t = s\right], \tag{1}$$

where $\pi(s_t)$ is the action taken by the agent in state s_t and can be denoted as a_t. The goal of RL is to find a optimal policy that maximizes the discounted cumulative reward $J(\pi)$ and is given by the following expression:

$$J(\pi) = \max_{\pi} \sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \tag{2}$$

Basic concept of Transformer

The Transformer model is proposed by Vaswani et al. (2017) and has achieved state-of-the-art performance in various fields, such as natural language processing (Ouyang et al., 2022), computer vision (Dosovitskiy et al., 2020), life science(Rives et al., 2021) and so on. Unlike traditional models that rely on recurrent layers, the Transformer utilizes a mechanism called 'attention' to process sequential data with variable-length, which enables it with more parallelization (Lin et al., 2022).

Figure 2 (left) depicts the attention mechanism, illustrating the transformation of input sequence X into output sequence Y. Prior to entering the attention mechanism, each element of the input sequence X is transformed into three distinct vectors—query Q, key K, and value V—by being multiplied with respective weight matrices as shown as follows:

$$\begin{aligned} Q &= XW^Q \\ K &= XW^K \\ V &= XW^V \end{aligned} \tag{3}$$

Subsequently, the attention mechanism processes these vectors to compute the output Y, which can be mathematical expressed by the following equation:

$$Y = Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_K}}\right)V, \tag{4}$$

where d_K is the dimension of the key vector K. The term $\frac{1}{\sqrt{d_K}}$ servers as a scaling factor, effectively normalizing the dot product to prevent excessively large values, thereby maintaining numerical stability in the attention scores.

For processing sequence with multiple dependencies is the multi-head attention mechanism particularly beneficial. As demonstrated in the Fig. 2 (right), the Q, K and V vectors are segmented into multiple heads, where each head independently performs the aforementioned attention function. The outputs from these heads are then concatenated to form the final result. The mathematical expression of multi-head mechanism is given as follows:

$$MultiHead(Q, K, V) = Concat(head_1, \ldots head_i, \ldots head_h) \tag{5}$$

Here, h represents the total number of heads, and the ith head, denoted as $head_i$, is calculated using $Attention(Q, K, V)$. This segmentation into multiple heads allows the model to capture a more diverse range of information from the input sequence, enhancing its ability to understand and represent complex relationships within the data.

Problem formulation

In this study, we address a complex variant of the PMSP that incorporates the dynamic arrivals of new job along with family setups constraints. The problem is first described in Sect. "Problem description" and a mathematical model is given in Sect. "Mathematical model".

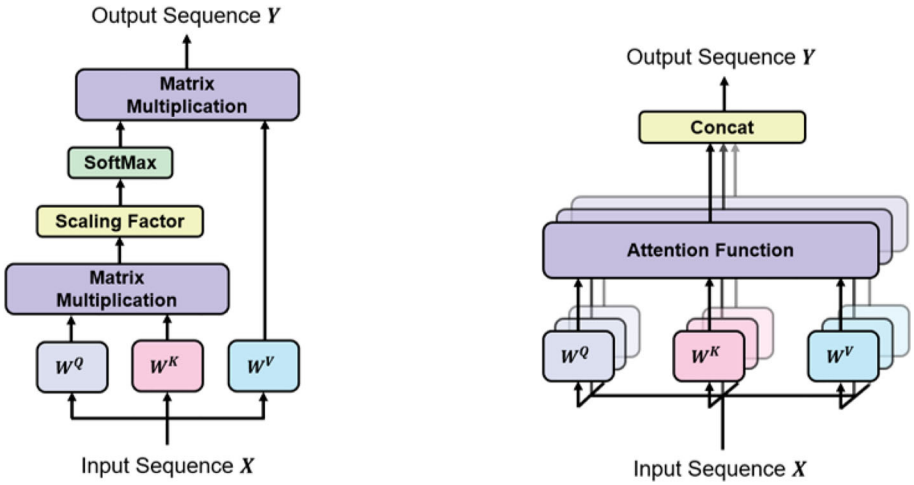

Fig. 2 Concept of the attention mechanism (left) and the multi-head attention mechanism (right)

Problem description

PMSP involves assigning n independent jobs to m parallel operating machines. The set of jobs and machines can be represented as $J = \{J_1, \ldots J_j, \ldots, J_n\}$ and $M = \{M_1, \ldots M_i, \ldots, M_m\}$, respectively. Each job J_j has an individual processing time p_j and a due date d_j, while each machine M_i possesses a specified processing speed v_i. The effective processing time for job J_j on machine M_i is thus calculated as p_j/v_i. If a job is completely processed after its own due date, it incurs tardiness. The objective of this work is optimizing the assignment of jobs to machines to minimize the sum of tardiness incurred on all jobs, which is referred as total tardiness and denoted as TT. For a given set of jobs, the TT can be calculated as follows:

$$TT = \sum_{j=1}^{n} \max(0, C_j - d_j), \tag{6}$$

where C_j is the completion time of job J_j. Figure 3 demonstrates a small instance of PMSP that contains only two machines and five jobs. The length of each block indicates the processing time of the corresponding job, and the due date of each job is marked on the horizontal axis. In the shown schedule, tardiness occurs on Job 5.

Furthermore, to obtain a more realistic representation of modern manufacturing environments, we take the following two constraints into the consideration:

1. The new job arrivals constraints. The new job arrivals constraints. In modern dynamic manufacturing environments, new jobs can arrive unpredictably, necessitating real-time adjustments to the schedule. Under this constraint, each job J_j is associated with a release date r_j, indicating that the job can only be processed after its release date.
2. The family setups. In real manufacturing environments, jobs are usually categorized into several families according to their characteristics, where sequential processing two jobs from different families requires an additional setup time in between (Liaee & Emmons, 1997). The number of families is denoted as N_f and the family of job J_j is expressed as f_{J_j}. The family of the previous job processed by machine M_i is termed as its setup state and is denoted as f_{M_i}. Figure 4 shows the previous instance under the constraint of family setups, where the five jobs are divided into two families. Two additional setup times are required before processing of Job 4 and Job 5, respectively. Moreover, these two setup times lead to a new tardiness on Job 4 and an increased tardiness on Job 5.

This problem formulation, combining the dynamic nature of job arrivals with family setups constraints, presents a realistic and challenging scenario typical in modern manufacturing and service systems, reflecting the intricacies of scheduling in a globalized, dynamic environment. To maintain focus on the core aspects of this challenge without losing the essence of the problem, we introduce the following simplifications:

1. Each machine can immediately start to process a job after the setup is finished.
2. Each machine can process only one job at a time, and each job can be processed on only one machine.
3. There is no moving time for the jobs.

Mathematical model

Then we mathematically describe the PMSP tackled in this paper based on the mixed integer formulation developed by Avalos-Rosales et al. (2015).

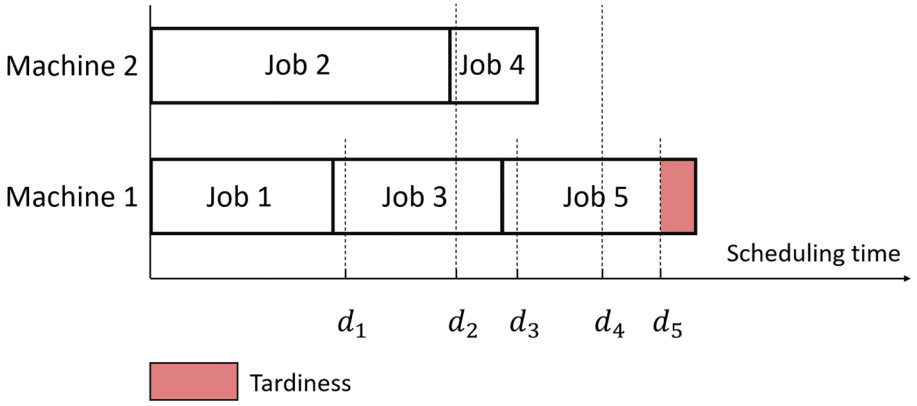

Fig. 3 An instance of PMSP with two machines and five jobs

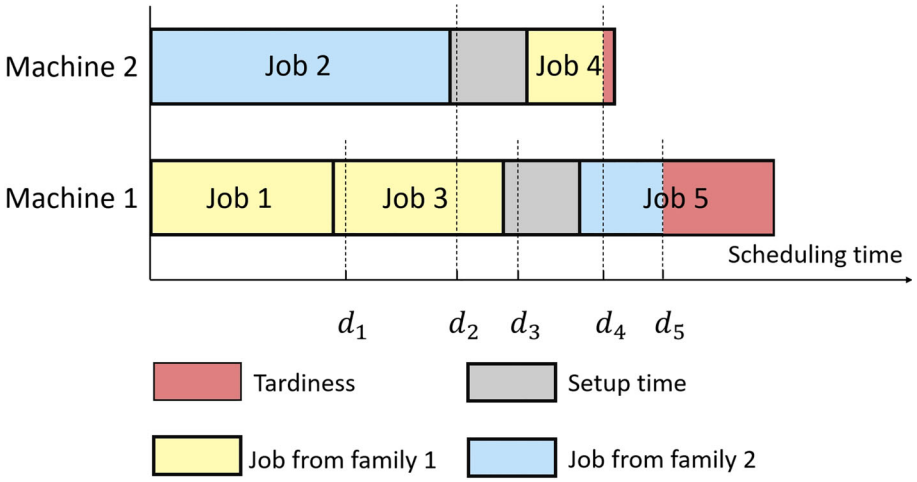

Fig. 4 An instance of PMSP with two machines and five jobs under the constraint of family setups

Although the proposed DRL approach solves the problem in a model-free manner, which does not inherently rely on a precise mathematical model (Schulman et al., 2017), we still present the mathematical model for the following two benefits. First, it provides an exact definition of the scope of our problem and clarifies the parameters, constraints, and objective. Second, this precise formulation aids in standardizing the problem across various studies, establishing a solid foundation for follow-up researchers that aim to explore alternative solutions or enhancements.

The notations required for modeling are listed below.

1. *Parameters:*

- n: total number of jobs
- m: total number of machines
- j, g: index of jobs, $j, g = 1, 2, \ldots, n$
- i, k: index of machines, $i = 1, 2, \ldots, m$
- J: the set of jobs
- M: the set of parallel machines
- p_j: the processing time of job J_j
- r_j the release date of job J_j
- d_j: the due date of job J_j
- v_i: the processing speed of machine M_i
- S_{jg}: setup time for job J_g when it immediately follows job J_j (equal to 0 if J_g and J_j come from same family, equal to 10 otherwise)
- V: a sufficient large constant

2. *Decision variables:*

- C_j: the completion time of Job J_j
- X_{ijg}: 1 if J_j is a predecessor of J_g on machine M_i, 0 otherwise
- Y_{ij}: 1 if job J_j is assigned to machine M_i, 0 otherwise

Moreover, to support the problem formulation, a dummy job is introduced at the start and end on each machine and J' denotes the set of jobs includes J and the dummy jobs. The processing times and setup times related to the dummy jobs are considered 0. Our model can be therefore stated as:

Objective function:

$$Minimize \sum_{j=1}^{n} \max\left(0,\ C_j - d_j\right) \tag{7}$$

Subject to:

$$\sum_{i \in M} Y_{ij} = 1, \quad \forall j \in J \tag{8}$$

$$Y_{ig} = \sum_{j \in J', j \neq g} X_{ijg}, \quad \forall g \in J, \quad \forall i \in M \tag{9}$$

$$Y_{ij} = \sum_{g \in J', g \neq j} X_{ijg}, \quad \forall j \in J, \quad \forall i \in M \tag{10}$$

$$\sum_{j \in J} X_{i0j} \leq 1, \quad \forall i \in M \tag{11}$$

$$C_j + V(1 - Y_{ij}) \geq \frac{p_j}{v_i} + r_j, \quad \forall i \in M, \quad \forall j \in J \tag{12}$$

$$C_g - C_j + V(1 - X_{ijg}) \geq S_{jg} + \frac{p_g}{v_i} \quad \forall j \in J', \quad \forall g \in J, \quad j \neq g, \quad \forall i \in M \tag{13}$$

$$C_0 = 0 \tag{14}$$

$$X_{ijg} \in \{0, 1\}, \quad \forall j \in J', \quad \forall g \in J', \quad j \neq g, \quad \forall i \in M \tag{15}$$

$$Y_{ij} \in \{0, 1\}, \quad \forall j \in J, \quad \forall i \in M \tag{16}$$

$$C_j > 0, \quad \forall j \in J \tag{17}$$

Objective (7) minimizes the total tardiness of the solution. Constraint (8) imposes that each job is assigned to one and only one machine. Constraint (9) and constraint (10) ensure that each job on the machine it is assigned to has one and only one predecessor and one successor, respectively. Constraint (11) establishes that at most, one job is scheduled as the first job on each machine. Constraint (12) makes sure that a job can only be processed after its arrival time. Constraint (13) forbids overlapping among the jobs with respect to family setups and machine speeds. Constraint (14) sets the completion time of the dummy job at the start on each machine to 0. Constraint (15)–(17) define the domain of the variables.

Proposed approach

In this section, the details of the proposed Transformer-based DRL approach are successively provided, including the state and action representations, the structure of the neural network that represents the DRL agent, the reward function, and the architecture of the PPO algorithm for training.

State representation

The state representation is designed based on the scheduling framework of the problem illustrated in Fig. 5. According to the framework, the state is discrete and the decision point is defined as every time a machine becomes idle. State transition occurs when a job is selected for this current idle machine.

Table 2 Job features and machine features in the state matrix

State features in the state matrix	
Job features	Processing time p_j
	Due date d_j
	Family f_{J_j}
Machine features	Running time τ
	Family of the previous job f_{M_i}
	Speed of the machine v_i

To enable the proposed approach to conduct the scheduling process in an end-to-end fashion, the raw features of all available jobs and all machines should be contained in the state matrix. The job features and machine features are shown in Table 2, where the machine feature running time τ is computed by adding the current time t to the remaining processing time of the job being processed on the corresponding machine. It is worth noting that $\tau_i = t$ if M_i is idle at time point t.

Meanwhile, to allow the approach can process instances of arbitrary scale, we design the state matrix to be variable-length. Li et al. (2024) proposed a variable-length state representation, which can be summarized in Fig. 6. The number of rows is equal to the number of available jobs, where each row contains the value of the features of the corresponding job and the value of the features of the machine that is idle at that decision time point.

However, this state representation only includes the idle machine at each decision time point and omits other working machines, which can result in a myopic manner. To this end, we propose a novel variable-length state representation, which is demonstrated in Fig. 7. As shown in the figure, a new matrix containing the information of all machines is concatenated to the matrix in Fig. 6. We apply zero padding on the matrix with all machine information to make the number of columns of the two matrices equal. Since there are n_t available jobs and m machines at time t, and each job and machine can be described by 3 features, the shape of the state matrix at time t is $(n_t + m) \times (3 + 3)$. Additionally, the number of rows decreases with the selection of jobs and increases with the arrival of new jobs.

This expansion of the state matrix introduces an additional dependency, capturing not only the interactions among jobs but also the relationships between jobs and machines. This more comprehensive view of the scheduling environment allows for a richer and more global understanding of the decision-making landscape.

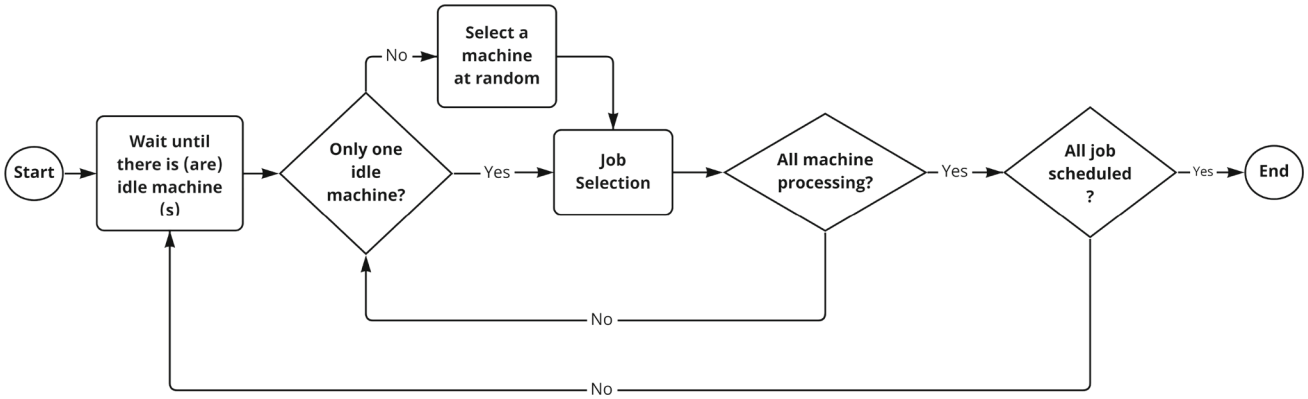

Fig. 5 Overall scheduling framework

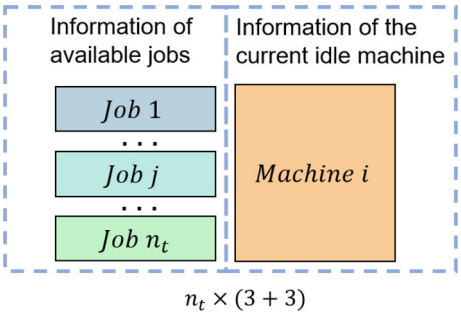

Fig. 6 Variable-length state matrix with only jobs information

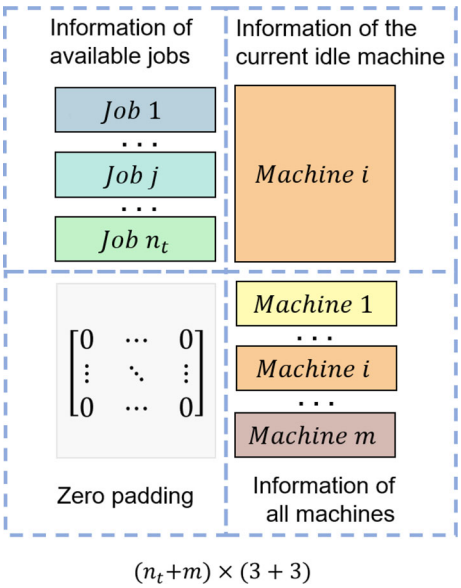

Fig. 7 Variable-length state matrix with global information of all jobs and all machines

Action representation

To realize the end-to-end scheduling manner, the agent should calculate a priority for each job at each decision time point, based on which the job could be selected directly without any pre-defined rules. Therefore, we define the action representation as the priorities of all available jobs in the current environment. Assuming that the number of available jobs at time point t is n_t, so the action of the DRL agent is an n_t-dimensional vector. Just like the state space, the action space also decreases with the selection of jobs and increases with the arrival of new jobs.

Then we apply the Softmax function to the output of the DRL agent, to convert the priorities into a probability distribution. In the training process, we conduct the job selection based on this probability distribution, where the job with higher priority corresponds to a higher probability of being selected. While during testing, we make the agent greedily select the job with the highest priority.

Training algorithm and neural network structure

In this work, we adopt the PPO algorithm (Schulman et al., 2017) as the learning algorithm, which is realized by actor-critic architecture. The actor network takes the current state as input, then executes the previously mentioned task of calculating the priority for each available job. The critic network also takes the current state as input but estimates the value of the states based on the current policy, which is a scalar and utilized for updating the actor network. Algorithm 1 illustrates the interaction mechanism between the actor network and the critic network.

We leverage the Transformer model Vaswani et al. (2017) to construct both the actor network and the critic network, which can process variable-length matrix in a parallel manner. To be more specific, the Transformer model utilized in this work employs solely the encoder component and discards the positional encoding. This modification ensures that the sequence of jobs in the state matrix does not impact performance, which is key in applying the Transformer model to the PMSP.

The architecture of the actor network is shown in Fig. 8. In the actor network, the Transformer model is composed

Algorithm 1 PPO Algorithm, Actor-Critic Style

Initialize policy parameters θ (Actor), old policy parameters $\theta_{old} \leftarrow \theta$, and value function parameters ϕ (Critic)
for each episode **do**
  Run the Actor with policy $\pi_{\theta_{old}}$ in environment for T timesteps
  Compute advantage estimates $A_1, \ldots, A_T$ utilizing Critic
  for each epoch **do**
    Optimize the surrogate objective L wrt θ (Actor)
  end for
  Update the Actor's policy: $\theta_{old} \leftarrow \theta$
  Update the Critic to fit value function
end for

of N_{Block} k-dimensional Transformer blocks. These Transformer blocks encode the state matrix into an encoded matrix with constant number of rows and k columns. Then, to obtain the priority of each job, we utilize the Multi-Layer Perceptron (MLP) to map the encoded matrix from k-dimensional to 1-dimensional. Since only the first n_t rows of this matrix correspond to the job information, we apply a mask operation on the last m rows to filter out the redundant machine information. It is noticeable that this mask operation will not lead to myopic behavior, since the machine information has been already encoded into the whole matrix by the Transformer model and taken into account of the jobs' priorities. Finally, we use the Softmax function to convert the priority vector into a probability distribution, where the job with a higher priority has a higher probability of being selected. The final output of the actor network is the index of the selected job that sampled from this distribution.

Figure 9 present the architecture of the critic network. The Transformer model implemented in the critic network has the same architecture as in the actor network. Since the final output of critic network is a scalar that indicate the value of the corresponding state, we straightforwardly sum the encoded matrix by columns to get a k-dimensional vector. Then the vector is mapped to the state value by an MLP.

It is crucial to note that the proposed new state matrix contains two distinct dependencies as described previously, providing global information while also presenting challenges for processing. To process this state matrix appropriately, we leverage the multi-head attention mechanism in the Transformer model.

Two perspectives of the relationship need to be considered for selecting the most suitable job for the current idle machine. The first perspective is how each job is influenced by other jobs, involving the assessment of factors like processing times, due dates and the relative urgency between jobs. The second perspective is the interactions between jobs and machines. This perspective involves considering the impact of machine allocation on overall scheduling efficiency. For instance, determining whether a job suitable for a currently idle machine might be better allocated to a machine that will be idle in the future.

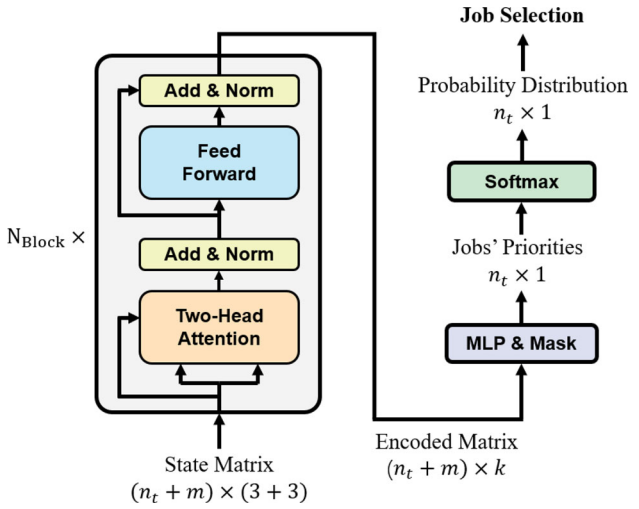

Fig. 8 Architecture of the actor network

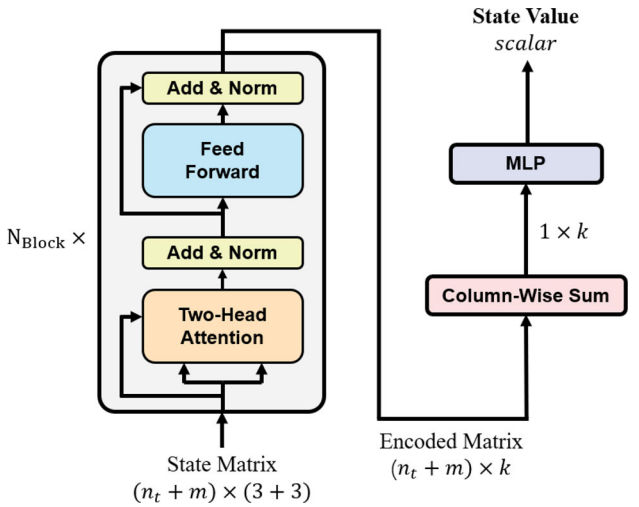

Fig. 9 Architecture of the critic network

The multi-head attention mechanism employs an architecture, where each attention head first processes the matrix from its own perspective separately, and then the results of each head are synthesized through a concatenation operation. This architecture enables the model to obtain multiple perspectives on the matrix and thus handle multiple dependencies within one matrix. In our work, the number of attention heads is intuitively chosen to be two, corresponding to the previous description of the state matrix. This two-head attention mechanism is proven to be efficient in the subsequent numerical experiments.

Reward function design

Given that the objective is to minimize total tardiness, equating to the cumulative sum of all delays incurred in the scheduling process, it becomes intuitive in our MDP frame-

work to define the reward as the negative of tardiness. However, this direct approach, where the reward function is solely dependent on the negative of total tardiness, can present challenges in the learning process of the agent. Since due dates are sampled from an identical distribution for a particular instance, tardiness tend to incur in the later stages of the scheduling process. Therefore the agent will struggle to evaluate a long sequence of decisions from the rewards that are given only at the end of those decisions.

Algorithm 2 Reward calculation of the second training stage

Initialization: $s_t \leftarrow s_0$, $n_t \leftarrow n$, $reward \leftarrow 0$, $Tardiness \leftarrow 0$, $TT \leftarrow 0$
while $n_t \neq 0$ **do**
  $a_t \leftarrow \pi_\theta(s_t)$
  $i \leftarrow a_t$
  if $f_i = f_M$ **then**
    for $j \leftarrow 0$ to n_t **do**
      if $f_j \neq f_M$ **then**
        $reward \leftarrow reward + R_{shape}$
        break
      end if
    end for
  else
    for $j \leftarrow 0$ to n_t **do**
      if $f_j = f_M$ **then**
        $reward \leftarrow reward - R_{shape}$
        break
      end if
    end for
  end if
  if $t + p_i/v_M > d_i$ **then**
    $Tardiness \leftarrow (t + p_i/v_M - d_i)$
  else
    $Tardiness \leftarrow 0$
  end if
  $reward \leftarrow reward + Tardiness$
  $TT \leftarrow TT + Tardiness$
  $s_t \leftarrow s_{t+1}, n_t \leftarrow n_t - 1$
end while
if $Total\ Tardiness = 0$ **then**
  $reward \leftarrow reward + R_{opt}$
end if

To address this challenge and enhance the learning efficiency of the agent, we adopt the technique of reward shaping (Ng et al., 1999). Reward shaping involves incorporating intermediate rewards to the original reward function to provide more frequent and informative feedback to the agent, ensuring a effective and stable learning process. The Algorithm 2 illustrates the procedure to calculate the reward.

The rationale behind our reward shaping strategy is rooted in the observation that frequent changes in setup state consequently lead to increased processing time. Hence, a schedule of high quality intuitively features fewer setup changes, as each additional setup can heighten the probability of incurring tardiness for the subsequent jobs. In particular, the agent receives a positive shaping reward R_{shape} when it selects a job that identical to the setup state among jobs from different families, thereby avoiding additional setup time; Conversely, the agent is penalized with a negative shaping reward $-R_{shape}$ if it chooses a job from a different family, despite the availability of same-family jobs. Moreover, when the total tardiness remains 0, imply an optimal schedule, an additional positive reward R_{opt} is granted. In this work, we set R_{shape} and R_{opt} to constants with values of 1 and 200, respectively.

Numerical experiments

In this section, we first provide the details of the training process of the agent. The training process is then compared with two agents with a one-head attention mechanism and a four-head attention mechanism, respectively, to show the superiority of the two-head attention mechanism. To demonstrate the generalization capacity of the proposed agent, it is further applied on larger instances without re-training. Meanwhile, the aforementioned comparative agents with different number of attention heads, together with an RNN-based DRL method (Li et al., 2024), a metaheuristics method (Rolf et al., 2020), and a dispatching rule, are taken into comparison.

The training process of the agent

We develop our DRL agent in PyTorch. The proposed agent is trained in an RL environment that simulates a PMSP. We construct the environment based on OpenAI Gym (Brockman et al., 2016) and a Python-based Discrete-Event Simulation (DES) library called Salabim (van der Ham, 2018). The training process and the comparison process that follows are conducted on a PC with Intel Core i911900KF@3.50GHz CPU, 16GB RAM, and a single Nvidia RTX 3080 GPU. The entire training process takes 9028.87 sec on this PC, with the neural network updating process executed on the GPU and lasting for 2473.29 sec.

A production environment of a manufacturing company in Baden-Wuerttemberg, Germany, is selected as the instance for training the DRL agent. To maintain confidentiality, specific values such as processing times and due dates have been reasonably modified. Details of this environment are outlined in Table 3.

This production environment contains 12 machines with 2 different speeds and 80 initial jobs from 8 families. New jobs enter the production environment in batches of 10 at a constant time interval of 10 time units after the 20th time unit. A total of 5 such batches are added to simulate dynamic and fluctuating production demands that commonly occur in the company mentioned above.

We utilize the parameters due date tightness r and due date range R to generate the modified due date for all jobs accord-

Table 3 Parameter settings of production environment for training

Parameter	Value
Total number of machines m	12
Number of initial jobs n_0	80
Number of batches N_B	5
Number of new jobs per batch n_{new}	10
Total number of families of jobs N_F	8
Speed of machines v_i	{1, 1.25}
Processing time of a job p_j	Unif [5, 15]
Average tardiness factor r	0.1
Relative range of due dates R	0.5
Setup time S	10

ing to the following uniform distribution, which is proposed by Potts and Van Wassenhove (1985):

$$U\left(MP\left(1-r-\frac{R}{2}\right), MP\left(1-r+\frac{R}{2}\right)\right), \tag{18}$$

where the indicator MP could be considered as the modified cumulative processing time of all jobs and is calculated as:

$$MP = \sum_{j=1}^{n} p_j/m + (N_s \cdot S)/m, \tag{19}$$

where N_s is the total number of setups. And n is the total number of jobs, which is calculated as $n = n_0 + N_B * n_{new}$. However, the precise value of N_s cannot be known until the end of the scheduling process. Since its maximum possible value and minimum possible value are equal to the total number of jobs n (each job is not from the same family as the previous one) and the number of families N_F (not changing families until all jobs from one family have been processed), respectively. Therefore, the value of N_s is determined as $N_s = \frac{n+N_F}{2}$, i.e., the average of the maximum possible value and the minimum possible value.

Since the due date of a new job cannot be earlier than its arrival time, we modify the uniform distribution in (18) to generate the due dates of the new jobs as follows:

$$U\left(\max\left((r_j + p_{max} + S), MP\left(1-r-\frac{R}{2}\right)\right), MP\left(1-r+\frac{R}{2}\right)\right), \tag{20}$$

where the p_{max} is the longest possible processing time and S is the setup time. This distribution ensures that every new job must not incur tardiness if it is processed immediately after it arrives in the production environment.

Table 4 Hyperparameter Settings for Training

Hyperparameter	Value
Number of episodes	12,000
Learning rate before optimal solution found η_1	1e−4
Learning rate after optimal solution found η_2	1e−5
Discount factor γ	0.99
Clip range ϵ	0.2
Batch size	64
Optimizer	Adam

Table 5 Architecture for the transformer model and the MLP utilized in the actor and critic networks

Architecture of the transformer model	
Number of Transformer blocks N_{Block}	2
Transformer model dimension k	128
Number of heads	2
Discount factor γ	0.99
Clip range ϵ	0.2
Batch size	64
Optimizer	Adam
Architecture of the MLP	
Number of layers	2
Number of neurons in the output layer	1

In the training instance, the r and R are set to be 0.1 and 0.5, respectively. A relatively small due date tightness and a relatively large due date range guarantee that a effective policy yields an optimal solution with 0 total tardiness, while a poor policy yields large total tardiness.

Table 4 gives the hyperparameter setting for the training process. It is noticeable that we utilize a smaller learning rate after an optimal solution with 0 total tardiness is found in order to achieve a stable training process (Bengio, 2012).

Table 5 gives the architecture for the Transformer model and the MLP utilized in the actor and critic networks. Since the dimension in the Transformer model is 128 and the number of heads is 2, each attention head contains 64 dimensions.

To verify the effectiveness of this two-head attention mechanism, we train two comparative Transformer-based agents: one with a one-head mechanism and another with a four-head mechanism, keeping all other hyperparameters, including the model dimension, consistent. As a result, each attention head in the one-head and four-head mechanisms contains 128 and 32 dimensions, respectively. The proposed agent, the comparative agents with one-head and four-head mechanisms are denoted in this work as Two-Head Agent, One-Head Agent, and Four-Head Agent, respectively.

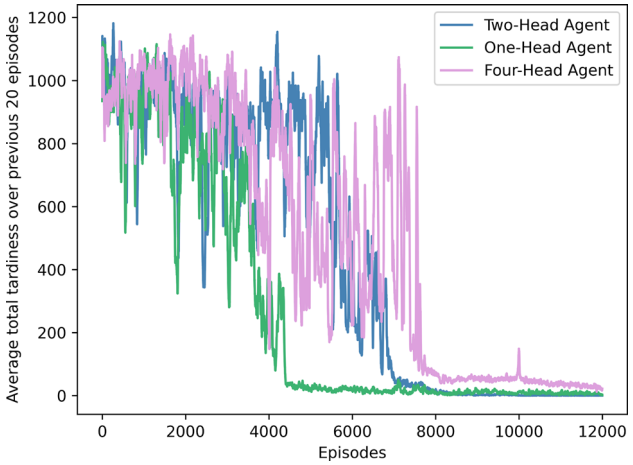

Fig. 10 Average total tardiness over previous 20 episodes obtained by the Two-Head Agent (blue curve), One-Head Agent (green curve), and Four-Head Agent (pink curve) in the whole training process (Color figure Online)

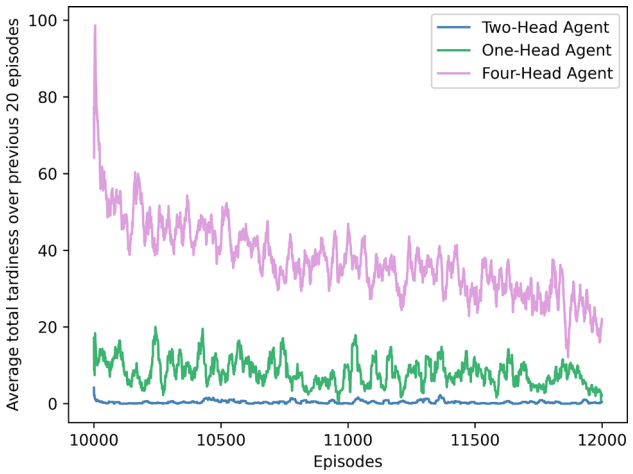

Fig. 11 Average total tardiness over previous 20 episodes obtained by the Two-Head Agent (blue curve), One-Head Agent (green curve), and Four-Head Agent (pink curve) in the last 2000 episodes (Color figure Online)

The whole training processes of the proposed agent and the comparative agent are illustrated in Fig. 10, in which the abscissa is the number of episodes, and the ordinate is the average number of setups that the agent obtained in the previous 20 episodes. Meanwhile, Fig. 11 demonstrates the last 2000 episodes of these training processes.

From the results, it is evident that the agent with one attention head exhibits a swift decline in tardiness in the initial stages of the training process, which might be due to the simpler structure of the one-head mechanism and the more dimensions that each attention head possesses. However, this simple structure is insufficient for capturing the two different relationships, i.e., the relationship among jobs and the interrelationship between jobs and machines, in the state matrix. Therefore, this agent still fluctuates in the late stage of training. Contrastingly, the proposed agent utilizing the two-head attention mechanism displays a more stable process. Throughout the training process, a consistent reduction in total tardiness is observed, ultimately nearing zero in the late stage of training.

Moreover, increasing the number of attention heads does not induce further improvements within the same training time frame. As observed in Fig. 11, the total tardiness given by the agent with the four-head mechanism is still in a decreasing trend in the last 2,000 episodes, implying that a more prolonged training period might be necessary for convergence.

In conclusion, the comparison validates the superiority of the RL agent with the two-head attention mechanism in addressing this proposed problem.

Table 6 Parameter settings of different production environments for testing

Parameter	Value
Total number of machines m	{16, 20}
Number of initial jobs n_0	{200, 300, 400}
Number of batches N_B	{6, 8}
Number of new jobs per batch n_{new}	{10, 15}
Total number of families of jobs N_F	{6, 8}
Speed of machines v_i	{1, 1.25}
Processing time of a job p_j	Unif [5, 15]
Average tardiness factor r	{0.2, 0.4}
Relative range of due dates R	{0.2, 0.6}
Setup time S	10

Generalization capability of the trained agent

To evaluate the generalization of the proposed approach, the agent trained in the previous section is then employed on much larger instances without re-training. These large testing instances, drawn from the more extensive production environments of the companies mentioned above, are similarly modified to protect confidentiality. The parameters defining these larger instances are detailed in Table 6.

The results are compared with those of the comparative agents with respectively one-head and four-head mechanisms. Moreover, to demonstrate the superiority of the proposed approach, we take an RNN-based DRL method (Li et al., 2024), a metaheuristic method (Rolf et al., 2020), and a classic dispatching rule called earliest due date (EDD) into the comparison. The RNN-based DRL method is also trained on the training instance generated with the parameter in Table 3, while the hyperparameter and the state representation are consistent with the original work. This method

Table 7 Parameter settings for GA approach

Parameters for GA-rules	
Number of generations	200
Number of individuals	200
Composite rules	EDD, SPT, FIFO
Crossover probability	0.8
Mutation probability	0.8
Initial number of switches in each gene	5
Probability of add a new switch	0.2
Probability of drop a existing switch	0.1

is denoted as RNN Agent in this work. The metaheuristic method utilizes GA to assign four dispatching rules during the scheduling process and therefore can handle dynamic scheduling problems, where the accurate information of all jobs cannot be known in advance. This metaheuristic method is denoted as GA-Rules in this work, and its parameters are listed in Table 7. The dispatching rule EDD always assigns the highest priority to the job with the earliest due date.

The total tardiness obtained by all the aforementioned approaches on all large instances is given in Tables 8, 9, 10, and 11, with the best result on each instance highlighted in bold. Each Table corresponds to one set due date configuration.

First, it can be observed from Tables 8, 9, 10, and 11 that the proposed DRL approach consistently demonstrates high performance across scenarios with various scales and due date settings. Specifically, within each of the four sets of 48 instances, our approach outperforms the comparative methods on 42, 47, 41, and 47 instances, respectively, indicating robust scalability of the proposed DRL framework to the varying conditions.

Moreover, in the instance sets from Tables 9 and 11, our approach shows a significant advantage by outperforming on 47 out of 48 instances from each set. It is worth noting that these two sets are generated with a larger value of the parameter R, providing the instances a wider range of due dates. This suggests that our DRL approach excels at adapting to wide variations in due dates compared to other comparative methods.

Furthermore, our approach yields optimal solutions on several instances from the sets of Tables 8 and 9, where the due date tightness is relatively low. In particular, the proposed approach incurs no tardiness on 30 and 31 instances from the sets in Tables 8 and 9 respectively, validating the efficiency of our method.

In addition, Table 12 indicates the computational time required by each approach to solve instances of each scale. The result given in Table 12 is the average computational time over the four different due date configurations, as the computational time is primarily influenced by the instance scale rather than the due date configuration. Specifically, Table 12 reveals the computational efficiency of our approach. The agent is able to solve the largest instance with 520 jobs (400 initial jobs and 120 new jobs) within 3 sec, which requires considerably more time with metaheuristics. Although the rule-based method EDD uses less computational time due to its simpler procedures, the quality of results produced by the trained DRL agent significantly surpasses that of the dispatching rules, thus confirming its superiority.

Given the widespread application of dispatching rules in real-world scheduling (Pinedo, 2022), we set the rule-based method EDD as our benchmark and evaluate the performance of our proposed approach together with all the other comparative approaches based on their improvement over this benchmark. We calculate the improvement using the following equation:

$$Improvement = \frac{TT_{EDD} - TT_{Approach}}{TT_{EDD}} \times 100\%, \quad (21)$$

where TT_{EDD} represents the total tardiness obtained by EDD and $TT_{Approach}$ is the total tardiness obtained by the approach to be evaluated. The improvements achieved by the Transformer-based agents, i.e., the Two-Head Agent, the One-Head Agent, and the Four-Head Agent, on the instances with the aforementioned four due date configurations, are shown in Fig. 12. Meanwhile, Fig. 13 depict the improvements obtained by the Two-Head Agent, the RNN Agent, and the GA-Rules on the instances with the due date configurations of Tables 8, 9, 10, and 11.

From the figures, it can be observed that the proposed Two-Head Agent demonstrates robust and consistent performance improvement over the benchmark across all instances and due date configurations. In comparison, other methods show more variable improvements and might not consistently outperform the benchmark EDD. This indicates the superior ability of the proposed agent to learn effectively from the state matrix and adapt to different scheduling scenarios.

Moreover, another noticeable observation from the figures is the significantly larger improvement demonstrated by the Two-Head Agent in due date configurations with a larger due date range R, which brings about a considerable disparity between optimal and sub-optimal scheduling decisions and further underscores the robustness of our proposed approach.

Ablation study

In this work, we propose a comprehensive state matrix containing information about all jobs and all machines, enabling the DRL agent to make globally informed decision. To validate the benefit of the proposed state representation with global information, we train the proposed DRL agent with a

Table 8 When $r = 0.2$ and $R = 0.2$

m	N_B	n_{new}	n_0	N_F	I	Two-head agent	One-head agent	Four-head agent	RNN agent	GA-rules	EDD
16	6	10	200	6	1	**0.00**	18.73	**0.00**	1101.42	1324.41	2029.21
				8	2	2.30	57.47	**0.43**	1328.65	2226.04	2171.13
			300	6	3	**0.00**	13.49	129.09	1499.37	2719.95	3796.35
				8	4	**0.00**	127.78	72.97	2039.18	4062.74	4354.60
			400	6	5	**20.94**	260.67	204.47	1983.16	4770.81	6157.96
				8	6	22.41	289.46	**3.16**	3095.12	6773.77	7371.07
		15	200	6	7	**0.00**	13.27	**0.00**	1118.78	1799.89	2561.55
				8	8	**1.51**	74.90	33.14	1322.67	3063.69	3009.78
			300	6	9	**0.00**	53.14	204.10	1489.61	3552.21	4404.63
				8	10	**0.00**	306.24	97.16	2026.55	4640.21	5471.20
			400	6	11	**28.16**	564.09	399.86	2068.59	5873.25	7445.76
				8	12	73.23	379.16	**13.40**	3184.51	8197.79	8086.59
	8	10	200	6	13	**0.00**	56.35	**0.00**	1143.89	1641.65	2163.35
				8	14	2.38	83.76	**0.00**	1325.81	2440.29	2686.15
			300	6	15	**0.00**	24.68	130.77	1390.60	3255.30	3891.09
				8	16	**0.00**	36.61	192.29	2024.59	4337.60	4809.39
			400	6	17	**19.57**	317.88	53.93	2166.32	5245.11	7170.25
				8	18	27.87	282.24	**21.45**	3218.80	7297.32	7417.79
		15	200	6	19	**0.00**	3.80	**0.00**	1139.56	2696.68	3532.82
				8	20	**1.88**	72.24	162.52	1330.47	3625.10	3297.92
			300	6	21	**0.00**	57.74	276.09	1393.49	4276.31	5064.79
				8	22	**0.00**	477.77	96.32	2069.00	6052.17	6440.25
			400	6	23	**27.40**	573.64	98.20	2162.17	6729.27	7493.93
				8	24	70.70	901.84	**47.78**	3043.91	8584.00	10,219.71
20	6	10	200	6	25	**0.00**	**0.00**	**0.00**	1176.76	1092.59	1665.97
				8	26	**0.00**	3.74	0.43	1386.44	1862.36	1986.89
			300	6	27	**0.00**	7.38	129.09	1513.92	2208.53	3048.94
				8	28	**0.00**	57.72	72.97	2027.74	3410.99	3135.38
			400	6	29	**3.77**	38.14	204.47	1958.93	3790.35	4769.56
				8	30	**0.56**	34.56	3.16	2731.73	4512.54	6496.98
		15	200	6	31	**0.00**	69.27	0.00	1207.78	1466.89	1921.64
				8	32	**0.00**	67.13	33.14	1421.62	2275.74	2433.09
			300	6	33	**0.00**	30.84	204.10	1512.82	2855.28	3604.80
				8	34	**0.00**	48.14	97.16	1964.48	4250.13	4308.25
			400	6	35	**18.81**	286.02	399.86	1922.56	4918.91	5827.36
				8	36	14.03	189.80	**13.40**	2891.56	6002.63	7113.95
	8	10	200	6	37	**0.00**	**0.00**	**0.00**	1201.92	1106.80	1876.01
				8	38	**0.00**	1.86	**0.00**	1396.82	2007.42	2114.79
			300	6	39	**0.00**	6.81	130.77	1526.42	2654.24	2726.09
				8	40	**0.00**	63.58	192.29	1949.75	3727.60	4086.08
			400	6	41	**0.74**	38.45	53.93	2027.83	4113.09	5813.88
				8	42	**0.00**	73.86	21.45	2877.42	6011.26	5841.23
		15	200	6	43	**0.00**	94.53	**0.00**	1151.10	1619.07	2542.37
				8	44	**0.00**	104.20	162.52	1432.73	3129.09	3252.72
			300	6	45	**0.00**	30.49	276.09	1456.85	3286.45	3889.68
				8	46	**0.00**	72.67	96.32	1986.06	5006.99	5220.21
			400	6	47	**18.57**	395.99	98.20	1926.24	5491.26	6608.55
				8	48	**13.26**	349.34	47.78	2662.76	6520.33	7587.52

Table 9 When $r = 0.2$ and $R = 0.6$

m	N_B	n_{new}	n_0	N_F	I	Two-head agent	One-head agent	Four-head agent	RNN agent	GA-rules	EDD
16	6	10	200	6	1	**0.00**	10.73	54.26	244.26	31.31	117.02
				8	2	**0.00**	3.29	134.64	373.68	304.82	281.52
			300	6	3	**0.00**	9.51	248.40	132.99	11.54	56.97
				8	4	**0.00**	60.80	163.15	535.78	295.84	507.94
			400	6	5	**8.34**	70.26	558.80	577.24	119.57	555.51
				8	6	**9.47**	628.22	50.91	407.45	575.82	369.41
		15	200	6	7	**0.00**	52.20	127.05	217.94	20.11	99.74
				8	8	**0.00**	6.87	397.51	485.10	343.52	407.89
			300	6	9	**0.00**	94.32	775.08	269.33	4.97	206.50
				8	10	**25.55**	107.60	210.32	667.39	379.57	629.16
			400	6	11	**37.42**	282.09	1162.32	755.78	252.19	765.12
				8	12	**49.91**	182.65	275.92	687.66	815.70	653.00
	8	10	200	6	13	**0.00**	4.29	46.33	29.26	5.25	16.98
				8	14	**0.00**	2.45	122.05	468.24	270.44	384.73
			300	6	15	**0.00**	18.49	263.22	275.16	33.30	215.52
				8	16	**0.00**	126.15	338.13	767.28	414.18	753.71
			400	6	17	**8.27**	147.87	826.07	382.45	148.62	355.94
				8	18	**21.78**	638.92	189.99	789.93	643.52	783.66
		15	200	6	19	**0.00**	105.92	142.67	463.92	43.43	372.03
				8	20	**0.00**	4.51	561.36	618.89	301.86	557.37
			300	6	21	**0.00**	133.83	921.63	310.04	20.78	259.92
				8	22	**39.82**	297.54	202.57	908.37	567.43	1007.38
			400	6	23	**41.71**	235.52	1373.96	342.47	168.31	286.97
				8	24	**128.49**	372.87	465.35	750.55	608.86	995.82
20	6	10	200	6	25	**0.00**	19.47	4.25	273.43	7.28	126.36
				8	26	**0.00**	0.91	105.85	248.92	347.23	129.80
			300	6	27	**0.00**	0.00	125.91	190.36	0.00	100.46
				8	28	**0.00**	40.94	28.72	304.80	435.46	213.26
			400	6	29	**0.00**	37.09	817.13	224.40	19.61	151.83
				8	30	**38.90**	428.08	46.54	709.20	341.19	698.03
		15	200	6	31	**0.00**	0.00	336.31	219.27	0.00	78.34
				8	32	**0.00**	2.74	85.58	408.80	311.31	311.84
			300	6	33	**0.00**	12.94	254.16	343.53	12.43	269.11
				8	34	**0.00**	114.73	252.11	956.12	251.96	941.85
			400	6	35	**9.70**	206.76	1154.76	599.09	58.08	570.13
				8	36	86.61	254.81	**47.29**	861.64	367.64	1280.97
	8	10	200	6	37	**0.00**	17.88	85.35	311.41	1.07	181.87
				8	38	**0.00**	0.00	112.26	337.96	333.32	226.16
			300	6	39	**0.00**	0.00	322.59	204.49	18.67	118.29
				8	40	**0.00**	43.41	85.85	517.59	248.46	454.01
			400	6	41	**0.00**	56.94	634.42	561.57	97.48	522.92
				8	42	**41.71**	68.70	62.36	882.34	532.38	873.03
		15	200	6	43	**0.00**	1.05	744.89	378.76	28.24	253.16
				8	44	**0.00**	0.00	104.05	470.34	424.20	390.76
			300	6	45	**0.00**	20.36	307.60	169.62	16.69	90.75
				8	46	**0.80**	196.14	237.00	833.90	326.59	802.27
			400	6	47	**9.64**	192.46	1220.46	474.06	148.81	428.75
				8	48	**78.90**	414.02	161.10	956.36	474.48	1063.66

Table 10 When $r = 0.4$ and $R = 0.2$

m	N_B	n_{new}	n_0	N_F	I	Two-head agent	One-head agent	Four-head agent	RNN agent	GA-rules	EDD
16	6	10	200	6	1	46.17	139.48	**0.33**	2961.39	5331.70	7150.95
				8	2	**4.14**	288.47	104.10	3411.17	6549.21	6837.36
			300	6	3	**4.06**	167.15	209.05	5079.02	11,320.51	12,089.78
				8	4	**16.43**	241.18	149.61	6248.75	13,087.50	13,737.04
			400	6	5	**34.59**	331.50	817.73	8109.42	18,664.33	21,145.58
				8	6	**215.67**	310.60	353.49	9776.97	21,642.12	21,060.12
		15	200	6	7	**0.39**	458.35	87.67	2987.56	6847.38	9228.52
				8	8	**49.46**	479.53	582.02	3405.19	9116.79	9063.44
			300	6	9	**20.14**	248.21	370.12	5058.45	12,244.66	14,475.80
				8	10	**147.28**	462.64	529.67	6217.98	15,581.46	16,299.12
			400	6	11	**92.35**	720.80	693.66	7995.26	21,442.40	22,880.27
				8	12	401.02	574.41	**178.33**	9650.28	24,874.00	24,867.09
	8	10	200	6	13	15.56	203.12	**4.37**	2941.88	6442.37	8074.04
				8	14	**25.51**	427.41	290.28	3482.14	7757.20	8647.51
			300	6	15	**0.02**	179.70	262.28	5050.31	12,323.21	13,862.60
				8	16	**81.94**	242.52	478.53	6162.91	14,075.31	15,866.60
			400	6	17	**32.37**	419.83	650.97	7935.52	20,280.06	23,982.01
				8	18	**258.01**	418.47	265.54	9861.54	23,878.40	23,528.58
		15	200	6	19	**36.27**	558.30	177.04	2954.91	8854.70	10,438.30
				8	20	**99.71**	592.56	1058.90	3418.15	11,086.77	11,546.01
			300	6	21	**8.43**	376.82	963.30	5240.59	15,038.18	17,608.21
				8	22	**81.90**	378.10	641.43	6200.10	19,002.68	18,330.77
			400	6	23	**91.41**	867.45	622.50	7929.40	24,853.40	25,722.85
				8	24	450.08	1561.61	**123.05**	10,183.27	27,927.32	29,149.36
20	6	10	200	6	25	**153.96**	311.71	360.81	2723.43	4157.60	5328.49
				8	26	**36.46**	179.14	89.99	3164.49	5330.87	5284.85
			300	6	27	109.65	73.78	**27.54**	4594.64	7984.88	10,406.00
				8	28	**0.72**	645.87	77.23	5639.30	9775.35	10,838.51
			400	6	29	**2.93**	119.89	219.63	7181.64	14,208.93	17,017.52
				8	30	**158.51**	419.84	315.24	8605.16	17,170.99	16,488.16
		15	200	6	31	**8.41**	475.67	200.37	2690.76	5595.34	6180.03
				8	32	**74.78**	443.00	240.14	3288.52	6695.03	7530.58
			300	6	33	**1.06**	82.36	212.02	4609.55	10,293.38	11,915.45
				8	34	**34.40**	284.04	386.20	5545.82	12,450.70	12,658.38
			400	6	35	**18.02**	437.68	541.64	7043.04	17,684.10	17,678.17
				8	36	308.77	**234.78**	381.77	8691.21	18,895.02	21,214.08
	8	10	200	6	37	**24.03**	471.99	589.52	2721.41	5225.67	5719.39
				8	38	**71.04**	136.57	113.61	3272.10	6406.58	6420.88
			300	6	39	**3.33**	40.49	73.39	4740.69	9820.49	11,373.85
				8	40	**146.19**	451.16	463.74	5507.84	11,225.21	12,605.89
			400	6	41	**2.91**	44.10	278.24	7045.53	16,365.37	18,669.36
				8	42	**144.47**	304.03	285.58	8650.58	18,566.24	19,049.10
		15	200	6	43	**23.76**	607.24	185.58	2751.93	6869.98	8015.15
				8	44	**72.27**	453.22	317.00	3266.98	8043.95	9352.59
			300	6	45	**5.61**	136.88	304.14	4665.87	12,374.29	14,163.48
				8	46	**8.97**	328.61	405.05	5665.78	14,869.31	14,557.19
			400	6	47	**17.92**	549.64	310.61	7234.04	20,204.48	20,527.35
				8	48	275.85	406.23	**262.50**	8723.96	22,069.70	23,364.61

Table 11 when $r = 0.4$ and $R = 0.6$

m	N_B	n_{new}	n_0	N_F	I	Two-Head Agent	One-Head Agent	Four-Head Agent	RNN Agent	GA-Rules	EDD
16	6	10	200	6	1	**25.39**	406.82	1516.66	3622.76	2878.31	3570.44
				8	2	**330.60**	616.29	1043.82	3758.47	4918.24	3765.44
			300	6	3	**25.99**	386.46	2298.45	6643.99	5284.87	6214.25
				8	4	**258.59**	1220.44	1258.56	6720.88	8862.74	7472.29
			400	6	5	**400.68**	1456.37	4262.90	10,515.96	10,323.75	10,587.64
				8	6	**13.26**	2150.59	1295.55	11,278.01	13,761.94	11,061.19
		15	200	6	7	**125.63**	188.11	1568.24	3705.54	3450.77	5206.54
				8	8	**254.12**	821.49	986.47	3788.59	5473.19	5233.70
			300	6	9	211.64	**131.34**	2955.57	6676.99	6186.94	7407.39
				8	10	**148.35**	1111.66	1037.81	6920.96	10,059.67	8918.27
			400	6	11	**417.81**	1480.07	4429.10	10,219.74	12,309.97	12,805.12
				8	12	**123.74**	2598.32	1530.32	11,081.32	14,749.94	13,496.90
	8	10	200	6	13	**111.18**	302.53	1741.85	3692.49	3310.34	4489.72
				8	14	**399.20**	789.83	888.87	3792.08	5199.12	4183.05
			300	6	15	**102.85**	256.77	2683.14	6728.03	5863.94	7396.95
				8	16	**285.61**	1595.35	1237.83	6784.19	9368.73	8405.25
			400	6	17	**295.29**	1982.21	4685.29	10,835.14	10,513.06	12,245.54
				8	18	**103.55**	2172.47	2140.91	11,373.84	13,762.24	13,064.89
		15	200	6	19	**62.74**	241.35	1877.19	3686.58	3989.61	5792.31
				8	20	**213.93**	763.89	1215.39	3799.28	6821.45	5773.19
			300	6	21	**280.64**	429.56	3194.64	6757.27	7896.30	9025.34
				8	22	**372.71**	1404.02	1038.18	6880.90	11,424.31	9818.58
			400	6	23	**133.95**	1384.19	4672.94	10,504.50	13,154.92	15,313.94
				8	24	**460.72**	2515.59	1640.32	11,279.38	17,112.18	15,050.43
20	6	10	200	6	25	**38.24**	467.11	1093.72	3315.83	1957.50	3033.37
				8	26	**180.29**	279.53	663.80	3581.55	4064.48	2694.82
			300	6	27	**167.11**	1118.51	1976.91	6019.69	3849.59	4767.17
				8	28	**268.67**	757.88	1253.45	6301.62	6989.34	5513.89
			400	6	29	**158.36**	1357.93	3453.67	8914.99	7144.53	8719.31
				8	30	**76.33**	1626.01	1291.49	9752.14	10,415.62	9059.62
		15	200	6	31	**27.12**	352.47	1523.36	3247.77	2479.06	3392.96
				8	32	**211.47**	670.68	1034.16	3519.95	4880.66	4269.91
			300	6	33	**186.81**	653.57	2240.94	5933.06	5073.22	6881.76
				8	34	**44.29**	1161.44	1263.48	6209.93	8138.57	7497.43
			400	6	35	**223.60**	1082.73	3638.00	9119.18	9022.35	10,424.01
				8	36	**293.52**	1828.92	1143.71	9793.55	12,437.21	10,985.52
	8	10	200	6	37	**60.58**	455.88	1102.16	3364.37	2772.30	3547.03
				8	38	**287.34**	299.70	868.22	3511.00	4623.51	3359.79
			300	6	39	**235.09**	784.35	2191.95	5757.40	4840.20	5223.34
				8	40	**11.34**	974.05	1433.23	6168.23	7476.14	5592.32
			400	6	41	**460.61**	1312.39	3657.63	9016.44	8312.10	9809.14
				8	42	**54.63**	1715.53	1274.34	9827.66	12,088.68	10266.85
		15	200	6	43	**47.98**	595.65	1969.52	3327.31	3508.53	4733.53
				8	44	**323.44**	589.71	1153.86	3522.91	5512.15	4372.65
			300	6	45	**54.91**	1127.61	3005.91	5870.84	5985.67	7469.84
				8	46	**161.64**	1240.65	1354.32	6226.04	8701.32	7801.52
			400	6	47	**77.15**	1402.04	3517.44	9008.06	10,334.84	12,457.98
				8	48	**30.89**	2073.25	1343.22	9748.86	14,727.77	13,680.51

Table 12 Comparison of the computational time (sec) of all approaches for solving instances with different scales

m	N_B	n_{new}	n_0	N_F	I	Two-head agent	One-head agent	Four-head agent	RNN agent	GA-rules	EDD
16	6	10	200	6	1	1.6819	1.6931	1.6330	0.9990	606.2796	0.0750
				8	2	1.0368	1.0504	1.0518	0.9947	559.3764	0.0137
			300	6	3	1.4653	1.4683	1.4819	1.9985	594.3887	0.0145
				8	4	1.4774	1.4837	1.4653	2.0112	607.9018	0.0142
			400	6	5	2.0436	1.9685	2.0260	3.3294	609.2435	0.0147
				8	6	2.0694	2.0817	2.0645	3.3007	621.5323	0.0144
		15	200	6	7	1.3354	1.4241	1.3350	0.9943	569.1803	0.0137
				8	8	1.3349	1.2980	1.3521	0.9977	560.5388	0.0135
			300	6	9	1.7797	1.8561	1.7893	1.9895	587.9272	0.0138
				8	10	1.7698	1.8066	1.7769	1.9805	593.1280	0.0143
			400	6	11	2.2249	2.2340	2.1536	3.2850	625.9401	0.0146
				8	12	2.2178	2.1667	2.1637	3.3098	635.4886	0.0157
	8	10	200	6	13	1.3045	1.2756	1.2965	1.0019	589.0219	0.0133
				8	14	1.3003	1.3128	1.3430	0.9923	564.4404	0.0126
			300	6	15	1.7779	1.7405	1.7892	1.9884	615.1295	0.0138
				8	16	1.7310	1.7263	1.7094	1.9793	602.1144	0.0135
			400	6	17	2.1542	2.1590	2.0887	3.3049	637.5972	0.0140
				8	18	2.2145	2.2976	2.1463	3.3170	636.7360	0.0155
		15	200	6	19	1.4918	1.5161	1.4183	1.0008	587.5337	0.0132
				8	20	1.5142	1.4844	1.4793	1.0045	574.6021	0.0133
			300	6	21	2.0137	1.9205	1.9711	1.9767	579.1564	0.0141
				8	22	1.9781	1.9311	1.9156	1.9912	585.1665	0.0135
			400	6	23	2.4235	2.3102	2.2791	3.2871	617.3433	0.0142
				8	24	2.4126	2.4218	2.3812	3.2977	635.8131	0.0146
20	6	10	200	6	25	1.6834	1.1855	1.6450	1.0032	600.5205	0.0128
				8	26	1.0808	1.0989	1.0749	0.9894	618.4399	0.0129
			300	6	27	1.4588	1.6453	1.5168	1.9713	612.4707	0.0134
				8	28	1.5187	1.7158	1.5205	1.9678	620.8952	0.0134
			400	6	29	2.1584	2.1932	2.1137	3.2938	651.2294	0.0147
				8	30	2.2122	2.3371	2.1617	3.3019	633.4716	0.0141
		15	200	6	31	1.3614	1.5021	1.3732	0.9910	602.1207	0.0135
				8	32	1.4003	1.3576	1.3921	0.9915	613.9460	0.0130
			300	6	33	1.8452	1.8506	1.8767	1.9703	625.5788	0.0142
				8	34	1.9233	1.7259	1.8171	1.9734	609.1460	0.0144
			400	6	35	2.3889	2.3709	2.3023	3.2963	606.4953	0.0143
				8	36	2.3200	2.3361	2.2747	3.3097	642.0526	0.0146
	8	10	200	6	37	1.3210	1.3310	1.3343	0.9800	578.0813	0.0129
				8	38	1.3685	1.3852	1.3327	1.0187	605.3223	0.0130
			300	6	39	1.8498	1.9101	1.8030	1.9895	593.3739	0.0139
				8	40	1.8435	1.8333	1.7879	1.9741	598.5103	0.0137
			400	6	41	2.3223	2.3397	2.2658	3.3776	620.5863	0.0145
				8	42	2.3755	2.3075	2.2995	3.3028	653.6900	0.0147
		15	200	6	43	1.5892	1.3829	1.5474	0.9817	592.5830	0.0138
				8	44	1.5306	1.4861	1.5388	1.0000	526.0414	0.0144
			300	6	45	2.0302	2.0250	2.0135	1.9848	566.1585	0.0147
				8	46	2.0477	2.0784	2.1254	1.9582	576.3336	0.0142
			400	6	47	2.3595	2.4073	2.5140	3.3076	617.0207	0.0150
				8	48	2.5440	2.4524	2.3882	3.2933	639.6812	0.0148

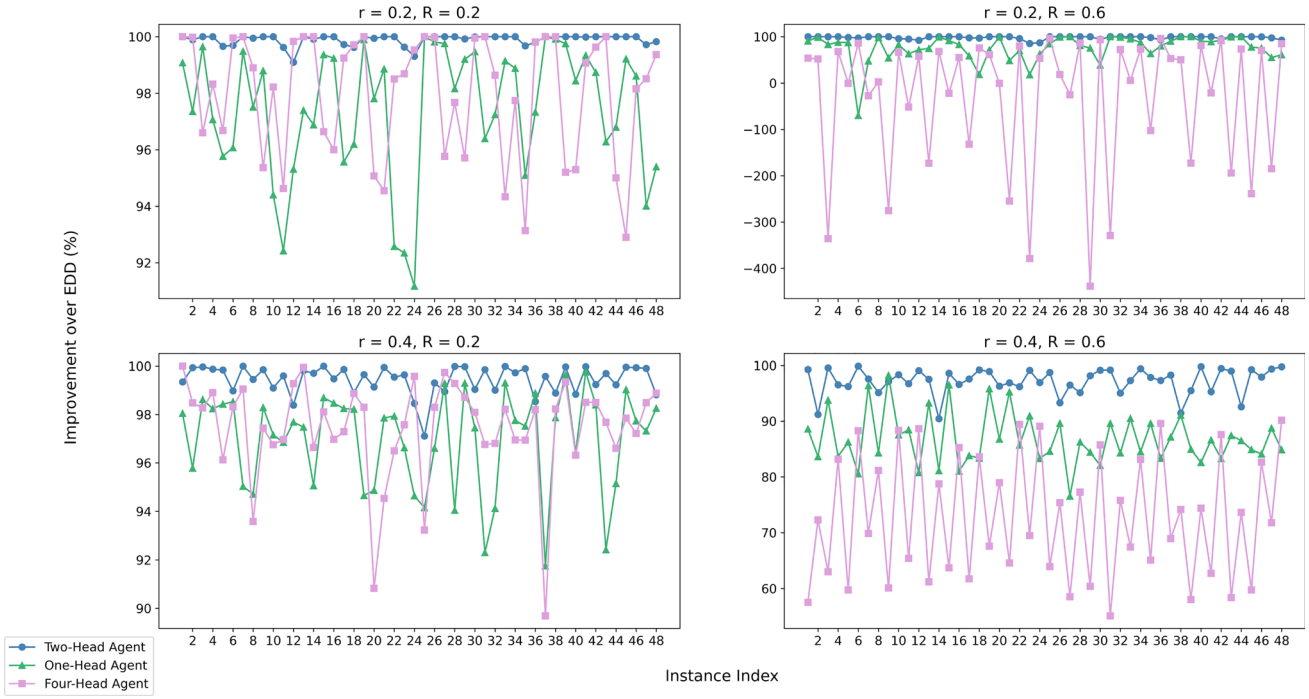

Fig. 12 Performance improvement on each instance of transformer-based agents over the EDD method across different due date configurations

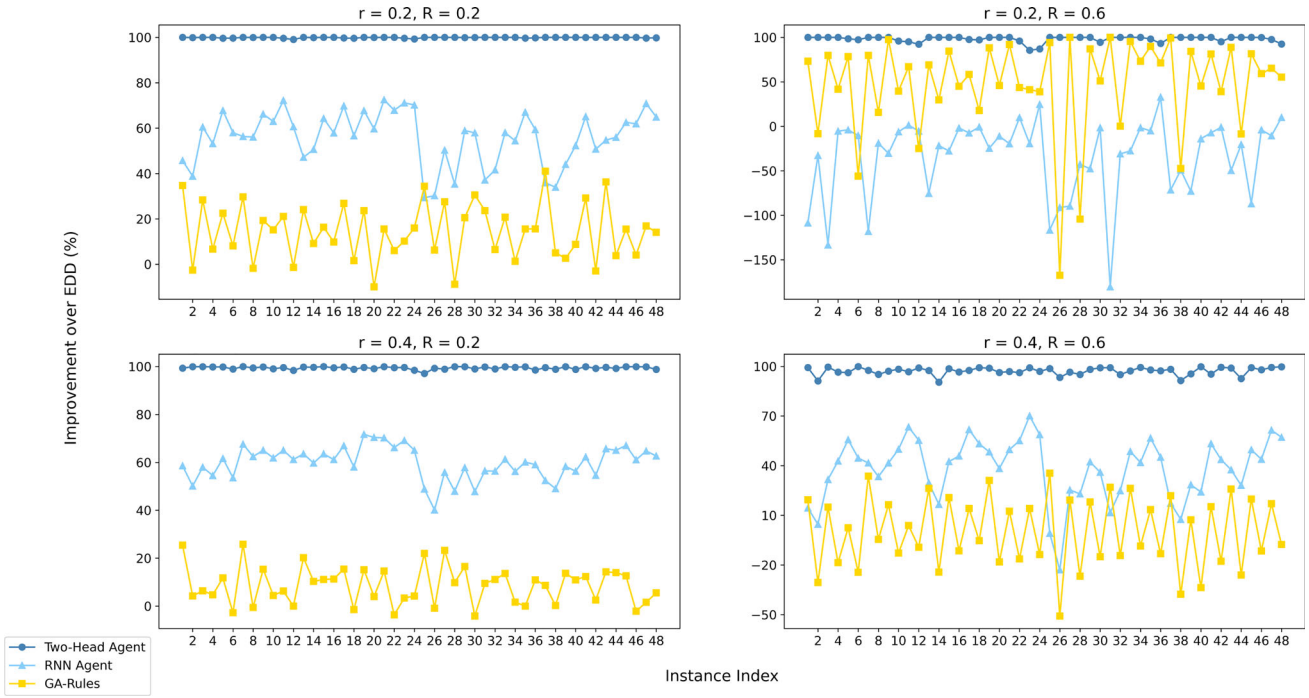

Fig. 13 Performance improvement on each instance of proposed Two-head agent, RNN agent, and GA over the EDD method across different due date configurations

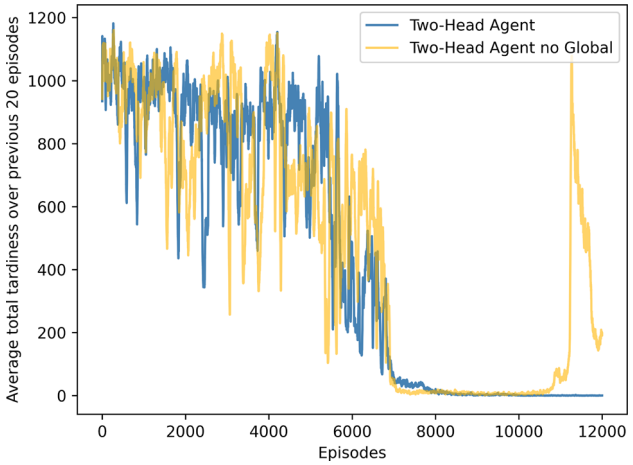

Fig. 14 Training process of the proposed Two-Head Agent with (blue curve) and without (yellow curve) global information (Color figure Online)

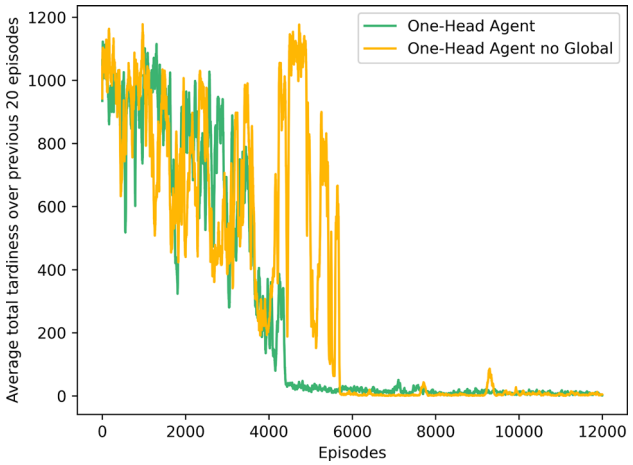

Fig. 15 Training process of the comparative One-Head Agent with (green curve) and without (yellow curve) global information (Color figure Online)

comparative state matrix as previously shown in Fig. 6, which does not provide any information of processing machines and contains solely the relationships among jobs. The configuration of the training instance and all other hyperparameters remain the same.

To further demonstrate the advantage of including global information, we also train the comparative One-Head Agent on the comparative state matrix, where the one-head attention architecture aligns with the state matrix containing only one category of relationship. This training process will eliminate the explanation, that the worse performance on the comparative state matrix obtained by the proposed agent is solely due to the mismatch between the two-head architecture and the comparative state matrix.

Figures 14 and 15 present the training process with and without the global information of the Two-Head Agent and the One-Head Agent, respectively,

where the training curves without global information are additionally labeled with "no Global" and shown in yellow.

It is evident from these figures that without the inclusion of global information, the training processes of both agents exhibit significantly more fluctuations. Particularly, the Two-Head Agent tends to diverge abruptly after initially converging, leading to substantial deviation from the optimal performance. Additionally, compared to the training process of the One-Head Agent on the state matrix without global information, it yields a smoother training process on the proposed comprehensive state matrix. Despite the fact that the proposed state matrix contains two distinct dependencies and does not align with the one-head attention mechanism.

Moreover, we further investigate the generalization capability of the agents trained without global information. They are employed to solve the four sets of instances from Tables 8, 9, 10, and 11. Their results are demonstrated in Tables 13, 14, 15, and 16 in Appendix, where the best result on each instance is also highlighted in bold. Our proposed approach maintains its superiority by outperforming on 42, 42, 34, 43 instances in each Table, respectively. Their improvements over EDD are illustrated in Fig. 16. It is worth noting that the Two-Head Agent trained on the comprehensive state matrix still realizes the most stable performance, verifying the robustness of the proposed approach.

In summary, the fluctuating training processes and the worse generalization capabilities could be attributed to the lack of comprehensive relationships that global information provides. Without the state of all machines, the agent might struggle to accurately assess the impact of individual job selection from a broader perspective and thus can only make myopic decisions. For instance, consider two similar scenarios where the available jobs and idle machines are identical. However, the jobs currently being processed on all the other machines are completely different. A proper action in the first scenario might lead to large tardiness in the second scenario, making it difficult to maintain stability during training and finally converge on a robust policy. Therefore, considering global information allows the agent to obtain a deeper understanding of the environment, offering a significant advantage in decision-making and overall performance.

Discussion

Through numerical experiments in various scenarios, our approach demonstrates significant scalability and stable effectiveness. A further discussion on its application will be carried out in this section.

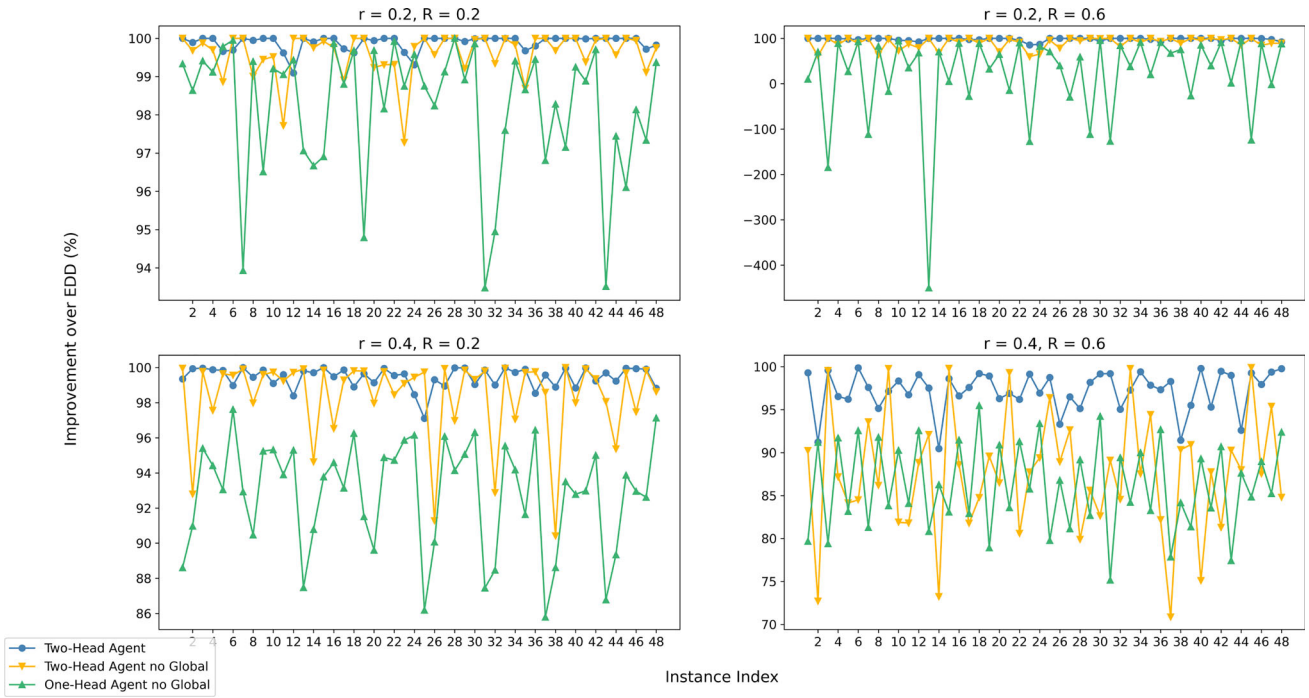

Fig. 16 Performance improvement on each instance of proposed Two-Head Agent, Two-Head Agent no Global, and One-Head Agent no Global over the EDD method across different due date configurations

Potential applications

The approach would find promising application the modern manufacturing. Specifically, several key components of manufacturing systems such as product scheduling on production lines (Paeng et al., 2021), Automated Guided Vehicle scheduling in warehouses (Miao et al., 2023), and scheduling of maintenance technicians (Rodríguez et al., 2022) have been investigated by being considered as PMSPs. Our approach can provide a high-quality solution in real-time for instances of any scale, providing the manufacturer with efficient scheduling process and consequent economic benefit.

Moreover, since the PMSP is an essential optimization problem, the utility of our method extends beyond the bounds of manufacturing into multiple domains where scheduling is a critical component. For instance, in the healthcare domain, the assignment of operations to surgeons and machines can be modeled as a PMSP (Kanoun et al., 2023; Burdett & Kozan, 2018). The efficiency and scalability of our approach offers the potential to dynamically give proper scheduling based on surgeon availability as well as the time tightness of the operations, ensuring that the healthcare system remains responsive and efficient under varying circumstances.

Similarly, in the cloud computing field, the scheduling of virtual machines for cloud data centers can also be described as PMSP (Tian et al., 2013). The scheduling problem in this domain is characterized by fluctuating demand and a strong requirement for real-time response, where the scalability and the immediate response capability of our approach could be particularly beneficial.

Practical implementation

By formulating the real-world information into the form of our proposed state representation as shown in Fig. 7, our approach could be seamlessly integrated into the existing decision-making system and then provide effective solution in real-time, without requiring any pre-defined rules.

While for the scenarios where the requirement on real-time response is less stringent, the explorative ability of the RL-based approach can be leveraged (Sutton & Barto, 2018). Instead of directly selecting the job with the highest probability, it can convert the priority of the job into a probability and then select the job according to the probability. The overall performance can be improved by repeating the process several times and selecting the schedule with the best result.

Furthermore, an industrial software with user-friendly interface could be developed based on the algorithm we proposed in this article and would be useful for the further transfer of our method. Since our code for the algorithm is open source, if a software is to be developed, the focus should solely be on the packaging process and user interface design.

Conclusion

In this work, we develop a novel DRL approach to address the PMSP, characterized by new job arrivals and family setups constraints.

The main contributions of our approach are its high scalability and the end-to-end scheduling manner. To be more specific, our approach can directly select the most appropriate job based on the raw information from the scheduling environment of arbitrary scales, reducing laborious manual work and thus increasing the applicability in various scenarios.

To achieve this, we first propose a variable-length state matrix that captures comprehensive information about both jobs and machines. The variable-length nature of the state representation allows it to adapt to scheduling environment of any scale. Meanwhile, the detailed representation is crucial for the DRL agent to perform an informed and global decision-making process. To process the multiple dependencies introduced by the rich information in the state matrix, we then utilize an elaborately modified Transformer model with two-head attention mechanism to represent the DRL agent. The Transformer model processes inputs in a way that enables it to dynamically handle matrix of any size. After training, the Transformer-based DRL agent can compute the raw features of jobs and machines and handle the dependencies between them. The globally optimal job is selected by directly outputting its index and without reliance on any pre-defined rules.

The experimental results validate the scalability and effectiveness of our approach. We first train the proposed DRL agent on a instance with 80 jobs and 12 machines. The training process is analyzed through the comparison with two comparative DRL agents with one-head and four-head attention mechanism respectively. The analysis demonstrates the superiority of the proposed two-head agent in terms of efficiency and stability.

We then apply the well-trained agent on four sets of large-scale instances, where each set has a different due date setting and contains 48 instances with scales ranging from 200 jobs and 16 machines to 400 jobs and 20 machines. Across numerous diverse and large-sized instances, our method exhibits remarkable scalability and effectiveness. In addition, we compare our approach with an RNN-based DRL methods, a metaheuristic algorithm, a standard dispatching rule, and the two previous DRL agents with different number of attention heads. The proposed approach outperforms all the comparative methods on a majority of instances. In particular, it yields the best solution on 42, 47, 41, and 47 instances in the four sets respectively.

Furthermore, an ablation study demonstrates the benefit of the novel state representation with global information. The agents trained on the state without the global information yield more fluctuating training processes, and obtain worse generalization capabilities.

Looking ahead, there are several promising avenues for future research building on the foundation of this work. Given that the PMSP is a specific instance of the broader job shop scheduling problem, an immediate extension of our approach could be its application to this wider context. This adaptation would test the versatility and robustness of our model in even more complex scheduling scenarios. Additionally, incorporating more dynamic constraints into the model, such as machine breakdowns and maintenance schedules, could significantly enhance its applicability in real-world manufacturing environments.

Appendix A: Generalization capabilities of the agents trained without global information

The DRL agents trained in the Sect. “Ablation study” are further employed to solve the instances in Tables 8, 9, 10, and 11. The results are shown in Tables 13, 14, 15 and 16.

The proposed agent, the agent with two-head-attention trained without global information, and the agent with one-head-attention trained without global information are denoted as Two-Head Agent, Two-Head Agent no Global, and One-Head Agent no Global, respectively.

Table 13 When $r = 0.2$ and $R = 0.2$

m	N_B	n_{new}	n_0	N_F	I	Two-head agent	Two-head agent no global	One-head agent no global
16	6	10	200	6	1	**0.00**	**0.00**	13.51
				8	2	**2.30**	6.98	29.54
			300	6	3	**0.00**	4.56	22.32
				8	4	**0.00**	12.99	38.26
			400	6	5	20.94	70.04	**13.08**
				8	6	22.41	**0.00**	3.43
		15	200	6	7	**0.00**	**0.00**	155.46
				8	8	**1.51**	29.87	18.11
			300	6	9	**0.00**	24.20	153.59
				8	10	**0.00**	26.23	43.45
			400	6	11	**28.16**	169.91	70.33
				8	12	73.23	**0.00**	45.61
	8	10	200	6	13	**0.00**	**0.00**	63.55
				8	14	**2.38**	6.62	89.46
			300	6	15	**0.00**	2.98	120.20
				8	16	**0.00**	11.33	6.20
			400	6	17	**19.57**	78.45	85.84
				8	18	27.87	**0.00**	23.60
		15	200	6	19	**0.00**	**0.00**	183.96
				8	20	**1.88**	25.10	10.37
			300	6	21	**0.00**	34.89	93.13
				8	22	**0.00**	44.01	4.71
			400	6	23	**27.40**	204.21	93.49
				8	24	70.70	**21.68**	42.94
20	6	10	200	6	25	**0.00**	**0.00**	20.76
				8	26	**0.00**	8.45	34.99
			300	6	27	**0.00**	**0.00**	26.61
				8	28	**0.00**	**0.00**	**0.00**
			400	6	29	**3.77**	37.65	51.61
				8	30	**0.56**	1.94	9.06
		15	200	6	31	**0.00**	**0.00**	125.34
				8	32	**0.00**	16.10	122.91
			300	6	33	**0.00**	0.28	86.64
				8	34	**0.00**	7.21	25.34
			400	6	35	**18.81**	76.04	78.14
				8	36	14.03	**0.00**	38.85
	8	10	200	6	37	**0.00**	**0.00**	59.87
				8	38	**0.00**	6.80	36.27
			300	6	39	**0.00**	**0.00**	77.62
				8	40	**0.00**	**0.00**	30.62
			400	6	41	**0.74**	36.00	64.78
				8	42	**0.00**	2.36	17.04
		15	200	6	43	**0.00**	**0.00**	164.87
				8	44	**0.00**	14.04	83.08
			300	6	45	**0.00**	**0.00**	151.49
				8	46	**0.00**	2.83	97.18
			400	6	47	**18.57**	58.70	175.92
				8	48	**13.26**	18.74	47.37

Table 14 When $r = 0.2$ and $R = 0.6$

m	N_B	n_{new}	n_0	N_F	I	Two-head agent	Two-head agent no global	One-head agent no global
16	6	10	200	6	1	**0.00**	**0.00**	105.15
				8	2	**0.00**	109.51	83.58
			300	6	3	**0.00**	**0.00**	162.07
				8	4	**0.00**	69.13	54.65
			400	6	5	8.34	**0.00**	405.91
				8	6	**9.47**	39.23	26.05
		15	200	6	7	**0.00**	**0.00**	211.08
				8	8	**0.00**	148.59	69.71
			300	6	9	**0.00**	3.79	241.19
				8	10	**25.55**	171.22	33.13
			400	6	11	**37.42**	95.73	496.15
				8	12	**49.91**	133.90	208.98
	8	10	200	6	13	**0.00**	**0.00**	93.41
				8	14	**0.00**	122.86	113.79
			300	6	15	**0.00**	**0.00**	204.91
				8	16	**0.00**	60.48	79.23
			400	6	17	8.27	**0.00**	455.07
				8	18	**21.78**	82.53	86.81
		15	200	6	19	**0.00**	**0.00**	251.36
				8	20	**0.00**	166.22	196.73
			300	6	21	**0.00**	5.46	297.74
				8	22	**39.82**	88.20	97.16
			400	6	23	**41.71**	115.82	652.11
				8	24	**128.49**	342.49	172.25
20	6	10	200	6	25	**0.00**	4.87	36.88
				8	26	**0.00**	28.05	78.16
			300	6	27	**0.00**	**0.00**	130.18
				8	28	**0.00**	10.87	87.60
			400	6	29	**0.00**	0.02	321.44
				8	30	38.90	**3.91**	29.16
		15	200	6	31	**0.00**	**0.00**	177.65
				8	32	**0.00**	57.36	48.66
			300	6	33	**0.00**	**0.00**	167.08
				8	34	**0.00**	82.37	80.32
			400	6	35	9.70	**0.01**	456.06
				8	36	86.61	**85.33**	113.13
	8	10	200	6	37	**0.00**	**0.00**	59.42
				8	38	**0.00**	24.81	55.31
			300	6	39	**0.00**	**0.00**	150.02
				8	40	**0.00**	18.12	66.18
			400	6	41	**0.00**	0.01	315.95
				8	42	41.71	**25.74**	81.33
		15	200	6	43	**0.00**	**0.00**	249.67
				8	44	**0.00**	57.75	19.21
			300	6	45	**0.00**	**0.00**	203.19
				8	46	**0.80**	123.23	90.73
			400	6	47	**9.64**	45.78	436.64
				8	48	**78.90**	116.28	126.76

Table 15 When $r = 0.4$ and $R = 0.2$

m	N_B	n_{new}	n_0	N_F	I	Two-head agent	Two-head agent no global	One-head agent no global
16	6	10	200	6	1	46.17	**2.98**	814.77
				8	2	**4.14**	492.60	616.69
			300	6	3	**4.06**	27.22	555.26
				8	4	**16.43**	335.97	766.10
			400	6	5	**34.59**	74.07	1469.48
				8	6	215.67	**92.38**	501.49
		15	200	6	7	**0.39**	8.27	652.32
				8	8	**49.46**	183.23	864.12
			300	6	9	**20.14**	57.32	688.40
				8	10	147.28	**41.70**	762.83
			400	6	11	**92.35**	175.55	1396.35
				8	12	401.02	**68.32**	1168.69
	8	10	200	6	13	15.56	**6.35**	1010.80
				8	14	**25.51**	464.88	796.43
			300	6	15	**0.02**	19.86	862.22
				8	16	**81.94**	551.28	857.56
			400	6	17	**32.37**	170.80	1644.47
				8	18	258.01	**42.98**	882.05
		15	200	6	19	36.27	**20.21**	885.40
				8	20	**99.71**	234.86	1200.21
			300	6	21	**8.43**	40.62	901.59
				8	22	**81.90**	283.02	966.72
			400	6	23	**91.41**	233.38	1064.37
				8	24	450.08	**160.70**	1120.64
20	6	10	200	6	25	153.96	**13.57**	735.83
				8	26	**36.46**	461.13	524.78
			300	6	27	109.65	**5.98**	407.16
				8	28	**0.72**	327.66	635.28
			400	6	29	**2.93**	24.33	840.40
				8	30	158.51	**111.22**	608.18
		15	200	6	31	**8.41**	11.95	775.53
				8	32	**74.78**	536.63	868.66
			300	6	33	**1.06**	4.48	531.06
				8	34	**34.40**	372.76	736.02
			400	6	35	**18.02**	48.12	1478.41
				8	36	308.77	**49.92**	754.26
	8	10	200	6	37	**24.03**	80.91	812.29
				8	38	**71.04**	615.40	730.92
			300	6	39	3.33	**0.00**	739.23
				8	40	**146.19**	253.91	908.79
			400	6	41	**2.91**	13.90	1310.02
				8	42	144.47	**122.18**	950.20
		15	200	6	43	**23.76**	155.38	1059.85
				8	44	**72.27**	433.73	996.42
			300	6	45	**5.61**	43.21	866.81
				8	46	**8.97**	367.94	1024.56
			400	6	47	**17.92**	32.51	1514.99
				8	48	**275.85**	321.48	669.41

Table 16 When $r = 0.4$ and $R = 0.6$

m	N_B	n_{new}	n_0	N_F	I	Two-head agent	Two-head agent no global	One-head agent no global
16	6	10	200	6	1	**25.39**	349.14	726.01
				8	2	**330.60**	1028.04	331.23
			300	6	3	**25.99**	27.21	1280.53
				8	4	**258.59**	957.68	619.18
			400	6	5	**400.68**	1680.07	1783.63
				8	6	**13.26**	1714.70	822.16
		15	200	6	7	**125.63**	333.97	974.12
				8	8	**254.12**	723.44	429.81
			300	6	9	211.64	**15.95**	1198.47
				8	10	**148.35**	1614.91	868.05
			400	6	11	**417.81**	2330.87	2038.46
				8	12	**123.74**	1506.21	1003.54
	8	10	200	6	13	**111.18**	354.10	860.45
				8	14	**399.20**	1120.41	574.66
			300	6	15	102.85	**14.51**	1251.16
				8	16	**285.61**	958.44	716.49
			400	6	17	**295.29**	2232.42	2093.20
				8	18	**103.55**	1991.63	591.73
		15	200	6	19	**62.74**	604.11	1220.69
				8	20	**213.93**	780.89	525.37
			300	6	21	280.64	**62.50**	1481.66
				8	22	**372.71**	1907.05	855.19
			400	6	23	**133.95**	1877.25	2180.25
				8	24	**460.72**	1596.13	995.22
20	6	10	200	6	25	**38.24**	109.65	613.70
				8	26	**180.29**	298.26	356.14
			300	6	27	**167.11**	350.96	899.31
				8	28	**268.67**	1108.59	596.85
			400	6	29	**158.36**	1256.16	1508.52
				8	30	**76.33**	1574.04	522.80
		15	200	6	31	**27.12**	369.38	842.73
				8	32	**211.47**	659.69	452.11
			300	6	33	186.81	**14.48**	1086.51
				8	34	**44.29**	934.55	750.50
			400	6	35	**223.60**	582.14	1745.07
				8	36	**293.52**	1958.56	803.38
	8	10	200	6	37	**60.58**	1033.77	785.52
				8	38	**287.34**	324.70	532.06
			300	6	39	**235.09**	473.85	973.27
				8	40	**11.34**	1393.32	598.69
			400	6	41	**460.61**	1200.77	1612.62
				8	42	**54.63**	1921.90	955.12
		15	200	6	43	**47.98**	460.27	1068.15
				8	44	**323.44**	524.69	542.03
			300	6	45	54.91	**7.60**	1133.10
				8	46	**161.64**	973.27	860.06
			400	6	47	**77.15**	576.38	1839.91
				8	48	**30.89**	2080.92	1041.40

Acknowledgements The work by Bingyuan Hong was supported by the Basic Public Welfare Research Program of Zhejiang Province (No. LQ23E040004), and Zhoushan Science and Technology Project (No. 2021C21011). The authors are grateful to all study participants.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability The code for realizing the proposed approach and training the agent could be found at https://github.com/ASnowbow/Transformer4PMSP. The data of the instances for testing is also available and can be found in the "Validation" folder. All instances are divided into four folders according to their due date settings, corresponding to Tables 8, 9, 10, and 11. Each instance is named according to the scale, e.g., an instance with 20 machines, 400 initial jobs, 8 batches of new jobs, and 15 new jobs in each batch, and all jobs divided into 8 families is named "20 m_400j_8b_15new_8f". In this instance all the raw data is saved as CSV files.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

Ábrahám, G., Auer, P., Dósa, G., Dulai, T., & Werner-Stark, Á. (2019). A reinforcement learning motivated algorithm for process optimization. *Periodica Polytechnica Civil Engineering, 64*(4), 961–970.

Abu-Marrul, V., Martinelli, R., Hamacher, S., & Gribkovskaia, I. (2021). Matheuristics for a parallel machine scheduling problem with non-anticipatory family setup times: Application in the offshore oil and gas industry. *Computers & Operations Research, 128*, 105162.

Arwa, E. O., & Folly, K. A. (2020). Reinforcement learning techniques for optimal power control in grid-connected microgrids: A comprehensive review. *IEEE Access, 8*, 208992–209007.

Avalos-Rosales, O., Angel-Bello, F., & Alvarez, A. (2015). Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times. *The International Journal of Advanced Manufacturing Technology, 76*(9), 1705–1718.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade: Second edition* (pp. 437–478). Springer.

Biskup, D., Herrmann, J., & Gupta, J. N. D. (2008). Scheduling identical parallel machines to minimize total tardiness. *International Journal of Production Economics, 115*(1), 134–142.

Blazewicz, J., Dror, M., & Weglarz, J. (1991). Mathematical programming formulations for machine scheduling: A survey. *European Journal of Operational Research, 51*(3), 283–300.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. arXiv preprint arXiv:1606.01540).

Burdett, R. L., & Kozan, E. (2018). An integrated approach for scheduling health care activities in a hospital. *European Journal of Operational Research, 264*(2), 756–773.

Chen, S., Huang, Z., & Guo, H. (2022). An end-to-end deep learning method for dynamic job shop scheduling problem. *Machines, 10*(7), 573.

Chen, R., Li, W., & Yang, H. (2023). A deep reinforcement learning framework based on an attention mechanism and disjunctive graph embedding for the job-shop scheduling problem. *IEEE Transactions on Industrial Informatics, 19*(2), 1322–1331.

Chen, Z., Zhang, L., Wang, X., & Wang, K. (2023). Cloud-edge collaboration task scheduling in cloud manufacturing: An attention-based deep reinforcement learning approach. *Computers & Industrial Engineering, 177*, 109053.

Csáji, B. C., & Monostori, L. (2005). Stochastic approximate scheduling by neurodynamic learning. *IFAC Proceedings Volumes, 38*(1), 355–360.

Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., & Riedmiller, M. (2022). Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature, 602*(7897), 414–419.

Didden, J. B., Dang, Q. V., & Adan, I. J. (2023). Decentralized learning multi-agent system for online machine shop scheduling problem. *Journal of Manufacturing Systems, 67*, 338–360.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., & Houlsby, N. (2020). An image is worth 16 × 16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Đurasević, M., & Jakobović, D. (2023). Heuristic and metaheuristic methods for the parallel unrelated machines scheduling problem: A survey. *Artificial Intelligence Review, 56*(4), 3181–3289.

Erick, A. O., & Folly, K. A. (2020). Reinforcement learning approaches to power management in grid-tied microgrids: A review. In *2020 Clemson University Power Systems Conference (PSC)* (pp. 1–6). IEEE.

Esteso, A., Peidro, D., Mula, J., & Díaz-Madroñero, M. (2023). Reinforcement learning applied to production planning and control. *International Journal of Production Research, 61*(16), 5772–5789.

Ezugwu, A. E. S. (2024). Metaheuristic optimization for sustainable unrelated parallel machine scheduling: A concise overview with a proof-of-concept study. *IEEE Access, 12*, 3386–3416.

Fawzi, A., Balog, M., Huang, A., Hubert, T., Romera-Paredes, B., Barekatain, M., & Kohli, P. (2022). Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature, 610*(7930), 47–53.

Frikha, M. S., Gammar, S. M., Lahmadi, A., & Andrey, L. (2021). Reinforcement and deep reinforcement learning for wireless internet of things: A survey. *Computer Communications, 178*, 98–113.

Ghaedy-Heidary, E., Nejati, E., Ghasemi, A., & Torabi, S. A. (2024). A simulation optimization framework to solve stochastic flexible job-shop scheduling problems-case: Semiconductor manufacturing. *Computers & Operations Research, 163*, 106508.

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

Heydari, M., & Aazami, A. (2018). Minimizing the maximum tardiness and makespan criteria in a job shop scheduling problem with

sequence dependent setup times. *Journal of Industrial and Systems Engineering, 11*(2), 134–150.

Hou, H., Agos Jawaddi, S. N., & Ismail, A. (2024). Energy efficient task scheduling based on deep reinforcement learning in cloud environment: A specialized review. *Future Generation Computer Systems, 151*, 214–231.

Hu, K., Che, Y., Ng, T. S., & Deng, J. (2024). Unrelated parallel batch processing machine scheduling with time requirements and two-dimensional packing constraints. *Computers & Operations Research, 162*, 106474.

Hurtado Sánchez, J. A., Casilimas, K., & Caicedo Rendon, O. M. (2022). Deep reinforcement learning for resource management on network slicing: A survey. *Sensors, 22*(8), 3031.

Iwamura, K., Mayumi, N., Tanimizu, Y., & Sugimura, N. (2009). A study on real-time scheduling for holonic manufacturing systems—Determination of utility values based on multi-agent reinforcement learning. In *Holonic and multi-agent systems for manufacturing: 4th international conference on industrial applications of holonic and multi-agent systems* (pp. 135–144). Springer.

Kanoun, S., Jerbi, B., Kamoun, H., & Kallel, L. (2023). Multi-objective mathematical models to resolve parallel machine scheduling problems with multiple resources. *Yugoslav Journal of Operations Research, 33*(4), 577–600.

Kayhan, B. M., & Yildiz, G. (2023). Reinforcement learning applications to machine scheduling problems: A comprehensive literature review. *Journal of Intelligent Manufacturing, 34*(3), 905–929.

Lang, S., Behrendt, F., Lanzerath, N., Reggelin, T., & Müller, M. (2020). Integration of deep reinforcement learning and discrete-event simulation for real-time scheduling of a flexible job shop production. In *2020 winter simulation conference (WSC)* (pp. 3057–3068).

Lang, S., Schenk, M., & Reggelin, T. (2019). Towards learning-and knowledge-based methods of artificial intelligence for short-term operative planning tasks in production and logistics: research idea and framework. *IFAC-PapersOnLine, 52*(13), 2716–2721.

Liaee, M. M., & Emmons, H. (1997). Scheduling families of jobs with setup times. *International Journal of Production Economics, 51*(3), 165–176.

Li, F., Lang, S., Hong, B., & Reggelin, T. (2024). A two-stage RNN-based deep reinforcement learning approach for solving the parallel machine scheduling problem with due dates and family setups. *Journal of Intelligent Manufacturing, 35*(3), 1107–1140.

Lin, T., Wang, Y., Liu, X., & Qiu, X. (2022). A survey of transformers. *AI Open, 3*, 111–132.

Liu, C. L., Chang, C. C., & Tseng, C. J. (2020). Actor-critic deep reinforcement learning for solving job shop scheduling problems. *IEEE Access, 8*, 71752–71762.

Li, C., Zheng, P., Yin, Y., Wang, B., & Wang, L. (2023). Deep reinforcement learning in smart manufacturing: A review and prospects. *CIRP Journal of Manufacturing Science and Technology, 40*, 75–101.

Luo, S. (2020). Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Applied Soft Computing, 91*, 106208.

Luo, S., Zhang, L., & Fan, Y. (2022). Real-time scheduling for dynamic partial-no-wait multiobjective flexible job shop by deep reinforcement learning. *IEEE Transactions on Automation Science and Engineering, 19*(4), 3020–3038.

Miao, M., Sang, H., Wang, Y., Zhang, B., & Tian, M. (2023). Joint scheduling of parallel machines and agvs with sequence-dependent setup times in a matrix workshop. *Computers & Industrial Engineering, 185*, 109621.

Ng, A. Y., Harada, D., & Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th international conference on machine learning (ICML)* (pp. 278–287). Morgan Kaufmann Publishers Inc.

Ogunfowora, O., & Najjaran, H. (2023). Reinforcement and deep reinforcement learning-based solutions for machine maintenance planning, scheduling policies, and optimization. *Journal of Manufacturing Systems, 70*, 244–263.

Ouelhadj, D., & Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling, 12*(4), 417–431.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., & Lowe, R. (2022). Training language models to follow instructions with human feedback. In *Advances in neural information processing systems* (Vol. 35, pp. 27730–27744). Curran Associates, Inc.

Paeng, B., Park, I. B., & Park, J. (2021). Deep reinforcement learning for minimizing tardiness in parallel machine scheduling with sequence dependent family setups. *IEEE Access, 9*, 101390–101401.

Palombarini, J., & Martínez, E. (2009). Learning to repair plans and schedules using a relational (deictic) representation. *Computer Aided Chemical Engineering, 27*, 1377–1382.

Palombarini, J., & Martínez, E. (2012). Smartgantt—An intelligent system for real time rescheduling based on relational reinforcement learning. *Expert Systems with Applications, 39*(11), 10251–10268.

Palombarini, J., & Martínez, E. (2012). Smartgantt—An interactive system for generating and updating rescheduling knowledge using relational abstractions. *Computers & Chemical Engineering, 47*, 202–216.

Panzer, M., Bender, B., & Gronau, N. (2021). Deep reinforcement learning in production planning and control: A systematic literature review. In *Proceedings of the 2nd conference on production systems and logistics (CPSL 2021)* (pp. 535–545). publish-Ing.

Panzer, M., & Bender, B. (2022). Deep reinforcement learning in production systems: A systematic literature review. *International Journal of Production Research, 60*(13), 4316–4341.

Para, J., Del Ser, J., & Nebro, A. J. (2022). Energy-aware multi-objective job shop scheduling optimization with metaheuristics in manufacturing industries: A critical survey, results, and perspectives. *Applied Sciences, 12*(3), 1491.

Pellerin, R., Perrier, N., & Berthaut, F. (2020). A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research, 280*(2), 395–416.

Pinedo, M. L. (2022). *Scheduling: Theory, algorithms, and systems* (6th ed.). Springer.

Potts, C. N., & Van Wassenhove, L. N. (1985). A branch and bound algorithm for the total weighted tardiness problem. *Operations Research, 33*(2), 363–377.

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., & Fergus, R. (2021). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences, 118*(15), e2016239118.

Rodríguez, M. L. R., Kubler, S., de Giorgio, A., Cordy, M., Robert, J., & Le Traon, Y. (2022). Multi-agent deep reinforcement learning based predictive maintenance on parallel machines. *Robotics and Computer-Integrated Manufacturing, 78*, 102406.

Rolf, B., Jackson, I., Müller, M., Lang, S., Reggelin, T., & Ivanov, D. (2023). A review on reinforcement learning algorithms and applications in supply chain management. *International Journal of Production Research, 61*(20), 7151–7179.

Rolf, B., Reggelin, T., Nahhas, A., Lang, S., & Müller, M. (2020). Assigning dispatching rules using a genetic algorithm to solve a hybrid flow shop scheduling problem. *Procedia Manufacturing, 42*, 442–449.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Shin, H., & Leon, V. J. (2004). Scheduling with product family set-up times: An application in TFT LCD manufacturing. *International Journal of Production Research, 42*(20), 4235–4248.

Shyalika, C., Silva, T., & Karunananda, A. (2020). Reinforcement learning in dynamic task scheduling: A review. *SN Computer Science, 1*(6), 306.

Song, L., Li, Y., & Xu, J. (2023). Dynamic job-shop scheduling based on transformer and deep reinforcement learning. *Processes, 11*(12), 3434.

Sutton, R. S. , & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.

Tian, W., Xiong, Q., & Cao, J. (2013). An online parallel scheduling method with application to energy-efficiency in cloud computing. *The Journal of Supercomputing, 66*(3), 1773–1790.

van der Ham, R. (2018). Salabim: Discrete event simulation and animation in Python. *Journal of Open Source Software, 3*(27), 767.

Vaswani, A. , Shazeer, N. , Parmar, N. , Uszkoreit, J. , Jones, L. , Gomez, A. N., & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc.

Wang, L., Pan, Z., & Wang, J. (2021). A review of reinforcement learning based intelligent optimization for manufacturing scheduling. *Complex System Modeling and Simulation, 1*(4), 257–270.

Wang, X., Zhang, L., Liu, Y., Zhao, C., & Wang, K. (2022). Solving task scheduling problems in cloud manufacturing via attention mechanism and deep reinforcement learning. *Journal of Manufacturing Systems, 65*, 452–468.

Waubert de Puiseau, C., Meyes, R., & Meisen, T. (2022). On reliability of reinforcement learning based production scheduling systems: A comparative survey. *Journal of Intelligent Manufacturing, 33*(4), 911–927.

Xiong, H., Fan, H., Jiang, G., & Li, G. (2017). A simulation-based study of dispatching rules in a dynamic job shop scheduling problem with batch release and extended technical precedence constraints. *European Journal of Operational Research, 257*(1), 13–24.

Xu, Y. , & Zhao, J. (2022). Actor-critic with transformer for cloud computing resource three stage job scheduling. In *Proceedings of the 2022 7th international conference on international conference on cloud computing and big data analytics (ICCCBDA)* (pp. 33–37). IEEE.

Yau, K., Kwong, K. H., & Shen, C. (2013). Reinforcement learning models for scheduling in wireless networks. *Frontiers of Computer Science, 7*(5), 754–766.

Yuan, B. , Wang, L. , & Jiang, Z. (2013). Dynamic parallel machine scheduling using the learning agent. In *2013 IEEE international conference on industrial engineering and engineering management (IEEM 2013)* (pp. 1565–1569). IEEE.

Yuan, B., Jiang, Z., & Wang, L. (2016). Dynamic parallel machine scheduling with random breakdowns using the learning agent. *International Journal of Services Operations and Informatics, 8*(2), 94.

Yu, L., Qin, S., Zhang, M., Shen, C., Jiang, T., & Guan, X. (2021). A review of deep reinforcement learning for smart building energy management. *IEEE Internet of Things Journal, 8*(15), 12046–12063.

Zhang, X., & Chen, L. (2022). A general variable neighborhood search algorithm for a parallel-machine scheduling problem considering machine health conditions and preventive maintenance. *Computers & Operations Research, 143*, 105738.

Zhang, D., Han, X., & Deng, C. (2018). Review on the research and practice of deep learning and reinforcement learning in smart grids. *CSEE Journal of Power and Energy Systems, 4*(3), 362–370.

Zhang, C., Liu, Y., Wu, F., Tang, B., & Fan, W. (2021). Effective charging planning based on deep reinforcement learning for electric vehicles. *IEEE Transactions on Intelligent Transportation Systems, 22*(1), 542–554.

Zhang, Z., Zhang, D., & Qiu, R. C. (2019). Deep reinforcement learning for power system: An overview. *CSEE Journal of Power and Energy Systems, 6*(1), 213–225.

Zhang, Z., Zheng, L., Hou, F., & Li, N. (2011). Semiconductor final test scheduling with zSarsa(λ, k) algorithm. *European Journal of Operational Research, 215*(2), 446–458.

Zhang, Z., Zheng, L., Li, N., Wang, W., Zhong, S., & Hu, K. (2012). Minimizing mean weighted tardiness in unrelated parallel machine scheduling with reinforcement learning. *Computers & Operations Research, 39*(7), 1315–1324.

Zhang, Z., Zheng, L., & Weng, M. X. (2007). Dynamic parallel machine scheduling with mean weighted tardiness objective by Q-Learning. *The International Journal of Advanced Manufacturing Technology, 34*, 968–980.

Zhao, L. , Shen, W. , Zhang, C. , & Peng, K. (2022). An end-to-end deep reinforcement learning approach for job shop scheduling. In *Proceedings of the 2022 IEEE 25th international conference on computer supported cooperative work in design (CSCWD)* (pp. 841–846). IEEE.

Zhou, G., Tian, W., & Buyya, R. (2021). Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions. arXiv preprint arXiv:2105.04086.

Zhou, L., Zhang, L., & Horn, B. K. (2020). Deep reinforcement learning-based dynamic scheduling in smart manufacturing. *Procedia CIRP, 93*, 383–388.