Contents lists available at ScienceDirect

Engineering

journal homepage: www.elsevier.com/locate/eng

Research
Intelligent Manufacturing—Article

考虑阻塞约束的分布式异构混合流水车间多目标强化学习调度方法

孙雪颜[a]，沈卫明[a,*]，范家昕[a,*]，Birgit Vogel-Heuser[b]，Fandi Bi[b]，张春江[a]

[a] State Key Laboratory of Intelligent Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan, China
[b] Institute of Automation and Information Systems, Technical University of Munich, Garching, Germany

ARTICLE INFO

Article history:
Received 16 November 2023
Revised 12 April 2024
Accepted 28 November 2024
Available online 20 December 2024

关键词
多目标马尔可夫决策过程
多智能体深度强化学习
近端策略优化
分布式混合流水车间调度
阻塞约束

摘要

本文针对旨在同时最小化总拖期和总能耗的、考虑阻塞约束的分布式异构混合流水车间调度问题（distributed heterogeneous hybrid blocking flow-shop scheduling problem, DHHBFSP），提出了一种改进的近端策略优化（improved proximal policy optimization, IPPO）算法进行实时决策。首先，建立了DHHBFSP问题的多目标马尔可夫决策过程模型，其中奖励函数用向量表示，而不是与目标相关的标量值。将每个工厂视为一个工厂智能体（factory agent, FA）负责选择未安排工件，并通过所提出的IPPO进行训练，以提高决策质量。多个FA异步决策，分配随机到达的工件。为了更好地利用训练数据，IPPO提出了两阶段训练策略，分别从单策略和双策略数据中学习。本文提出的IPPO算法在随机生成的实例上进行了测试，并与未改进的PPO、调度规则、多目标元启发式和多智能体强化学习方法进行了比较。实验结果表明，所提出的策略对PPO的性能有显著的提升，并且所提出的IPPO在收敛性和解决方案质量方面都优于其他的调度方法。

© 2024 THE AUTHORS. Published by Elsevier LTD on behalf of Chinese Academy of Engineering and Higher Education Press Limited Company. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. 引言

智能制造已成为制造业发展的主流趋势，生产调度是制造业发展的关键技术[1]。混合流水车间调度问题广泛存在于化工、食品加工、炼钢等工业领域[2]。在部分混合流水车间中，由于特殊的加工步骤或工艺约束，两个连续阶段之间没有缓冲区。当下游机器不可用时，当前机器上已经完成的工件只能暂时留在该机器上，因此会阻塞下一个工件加工，直到下游机器可用。阻塞约束增加了生产线周期时间和机器的无效占用率。因此，研究混合阻塞流水车间调度问题对于提高生产效率、降低生产成本具有重要意义。当机器数量大于5时，阻塞流水车间调度问题（blocking flow-shop scheduling problem, BFSP）已被证明是非确定性多项式困难（non-deterministic polynomial-hard, NP-hard）问题[3]。由于BFSP是分布式异构混合BFSP（distributed heterogeneous hybrid BFSP, HHBFSP）的子问题，因此DHHBFSP也是一个NP-hard优化问题。

分布式制造可以实现原材料的高效分配和生产力的最优组合，从而实现高质量、低成本产品的快速生产[4]。混合流水车间调度问题结合了分布式制造和阻塞约束，可以被定义为分布式混合BFSP（distributed hybrid BFSP, DHBFSP）[5]。一般在DHBFSP中，生成任务由几个相同

* Corresponding authors.
E-mail addresses: wshen@ieee.org (W. Shen), fanjx@hust.edu.cn (J. Fan).

2095-8099/© 2024 THE AUTHORS. Published by Elsevier LTD on behalf of Chinese Academy of Engineering and Higher Education Press Limited Company. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
英文原文：Engineering 2025, 46(3): 278–291
引用本文：Xueyan Sun,Weiming Shen, Jiaxin Fan, Birgit Vogel-Heuser, Fandi Bi, Chunjiang Zhang. Deep Reinforcement Learning-based Multi-Objective Scheduling for Distributed Heterogeneous Hybrid Flow Shops with Blocking Constraints. *Engineering*, https://doi.org/10.1016/j.eng.2024.11.033

的阻塞车间协作完成。然而，在实际的分布式制造问题中，由于生产条件不同，车间中的机器数量和机器加工时间因地而异，从而导致能耗不同[6]。此外，在大规模个性化生产中，具有异构需求的工件会在加工过程中动态出现，需要即时做出调度决策。研究动态DHHBFSP，可以有效减少资源空闲时间和等待时间，从而提高设备和人员的利用效率。

能源消耗增加是全球关注的重大问题，想要提高能源效率、实现低碳发展目标，应着重解决生产运营中的瓶颈问题[7]。钢铁生产过程通常是一种有阻塞约束的流水制造过程，会消耗大量能源。钢铁行业采用高温炉生产钢铁，已成为第二大能源消耗行业[8]。2013年，全球最终工业能源消耗总量为113 131 PJ，其中，钢铁行业占18% [9]。Hernandez等[10]总结了三条炼钢路线和九个独立工厂的能源强度、火用强度和资源效率结果。由于能源损失巨大，全球钢铁生产的实际资源效率仅为32.9%。同样，药品制造涉及各种化学反应和合成，这些都与高能耗有关[11]。

分布式BFSP（distributed BFSP, DBFSP）主要采用精确方法[12]、启发式规则[13]和元启发式算法[5]进行求解。精确方法可以获得最优解，但不适用于大规模问题。单一启发式规则虽然高效，但对于大规模问题效果不佳。元启发式方法针对大规模的复杂问题也可以获得更优的调度方案，但其对动态生产状态的响应能力较差，并且与启发式规则相比耗时较长。强化学习（reinforcement learning, RL）与智能体相结合，可以有效地利用生产过程中的实时数据，做出有效的生产决策，并快速响应动态变化。Riedmiller [14]首次将RL引入动态调度领域，使智能体能够自适应地选择最有利于调度目标的优先级调度规则（priority dispatching rule, PDR）。这种复合调度规则方法在一定程度上提高了PDR的调度性能，但减少了复杂生产调度问题的解空间，并且无法得到更优的调度方案。

本研究旨在研究兼顾生产力和能耗目标的DHHBFSP，并开发一种基于端到端深度强化学习（deep reinforcement learning, DRL）的调度方法，以实现实时决策。因此，本研究构建了一个DHHBFSP的多目标马尔可夫决策过程（multi-objective Markov decision process, MOMDP）模型，并提出了一种改进的近端策略优化（improved proximal policy optimization, IPPO)方法来解决这个问题。本研究的创新之处和主要贡献如下：

（1）通过定义状态特征、基于向量的奖励函数和端到端动作空间，为DHHBFSP建立了MOMDP模型。

（2）在分布式制造环境中采用多智能体深度强化学习（multi-agent DRL, MADRL）框架，每个混合流水车间被看作一个工厂智能体（factory agent, FA），多个FA异步工作，以选择未调度的工件。

（3）提出了一种IPPO方法来训练单个智能体，该智能体结合了两个具有不同权重的近端策略优化（proximal policy optimization, PPO）网络，以探索更多Pareto解。

（4）开发了一种两阶段训练策略，通过使用单策略和双策略数据同时进行训练来提高数据利用率。

本文其余部分安排如下：第2部分提供了相关研究的文献综述；第3部分描述问题并建立多目标DHHBFSP的数学模型；第4部分详细阐述了MOMDP模型设计和提出的IPPO方法；第5部分展示了数值实验结果与分析；最后，第6部分总结了本文研究内容并探讨了未来的研究方向。

2. 相关文献综述

2.1. DBFSP

针对DBFSP，Ribas等[12]在2016年构建了混合整数规划模型，以最小化最大完工时间为目标，通过混合整数规划方法实现了小规模算例的精确求解。为了求解大规模实例问题，他们设计了一种带有可变邻域的混合迭代贪婪算法和迭代局部搜索算法。Zhang等[15]提出了两种不同的数学模型来计算完工时间，并提出了一种混合离散差分进化算法来求解DBFSP。对于以总流动时间为目标的DBFSP，Chen等[16]利用基于种群的搜索方法和迭代贪婪算法来产生后代解并增强局部开发能力。Shao等[5]首次尝试使用果蝇优化算法来解决DBFSP，并提出了一种混合增强离散果蝇优化算法，包括基于气味和视觉觅食的有效初始化方案。同年，Shao等[13]研究了一个考虑阻塞的分布式模糊流水车间调度问题，并提出了两种基于问题特定知识和Nawaz-Enscore-Ham (NEH) 的启发式算法。随后，他们提出了一个混合整数线性规划模型来构建异构DBFSP，并提出了一个基于学习的超启发式选择框架[17]。Qin等[6]提出了DHHBFSP的数学模型，并设计了一种协作迭代贪婪算法。Zinn等[18]将深度Q学习方法应用于阻塞流水车间的案例研究中进行决策。Ren等[19]使用纳什Q学习研究了DPFSP，并获得了比元启发式算法更好的结果。Yang等[20]使用深度强化学习来选择规则，研究了具有动态工件到达的DPFSP。

大多数以能源为优化目标的流水车间调度研究都将能耗作为生产率相关指标的附加目标，并采用多目标优化框架进行权衡优化。Chen等[21]考虑机器处理速度对能耗的影响，提出了一种针对分布式无空闲流水车间调度问题的协同优化算法。Zhang等[22]针对DBFSP，研究了提高生

产能力和能源效率的生产调度决策方法，提出了基于贝叶斯推理的概率模型和特定的速度调整算子，以获得针对这两个目标的更好的搜索空间。Mou等[23]提出了一种基于机器学习的知识驱动求解策略，求解具有能耗目标的分布式流水车间调度问题，并设计了一种混合协同算法和双种群协同搜索机制，实现了全局探索与局部发展之间的平衡。Zhao等[24]提出了一种结合Q学习的元启发式算法来解决节能的DBFSP，其中Q学习用于选择合适的预先设计的低级启发式算法。Shao等[25]提出了一些通过Q学习选择的局部搜索方法来解决节能的考虑阻塞约束的分布式模糊混合流水车间调度问题。Zhao等[26]将邻域扰动算子和Q学习算法相结合，在搜索过程中选择合适的扰动算子。Bao等[27]开发了一个顶层Q学习模型，通过从四个序列相关操作中寻找调度策略来提高机器利用率，以及一个底层Q学习模型，通过学习最优速度调节策略来提高能源效率。大多数研究工作都致力于通过Q学习选择算子或调度规则，而端到端DRL在DBFSP中的应用有限。

2.2. 多目标强化学习

RL是在马尔可夫决策过程（Markov decision process, MDP）的基础上发展起来的。MORL通过融合多目标优化的权衡分析框架与强化学习的动态策略优化机制，为复杂决策场景中需同步平衡多个目标的优化问题提供了一种求解范式[28]。在MORL中，奖励是对多个优化目标的衡量，因此包含了多个值是一个向量而不是标量。Gábor等[39]假定不同目标之间有固定的排序，将问题的子目标视为约束条件，主目标的值越优则代表策略越好。Feinberg和Schwartz [30]提出了将多奖励问题表述为优化多种奖励类型的总奖励折扣加权和。Russell和Zimdars [31]则将奖励函数分解为多个独立学习的部分（采用单一策略）。Barrett和Narayanan [32]将凸壳定义为最小的凸集，其中包含位于该凸集边界上的所有点，这些点包含了在某个方向上最大的点。这与帕累托曲线有些类似，因为两者都是线性域中权衡的最优值。他们没有更新加权目标总和的单一Q值，而是更新了凸壳中的Q值集。Moffaert和Nowé [33]提出了一种帕累托Q学习算法。他们提出了三种机制，允许根据Q向量集的内容选择动作，并分别存储观察到的平均即时奖励和非支配Q向量集，允许它们分别收敛。

随着深度学习的发展，越来越多的学者关注将深度学习应用于MORL（multi-objective DRL, MODRL）。Mossalam等[34]提出了一种深度优化线性支持学习方法（optimistic linear support, OLS），用于解决目标间相对重要性先验未知的高维多目标决策问题。这是首次将深度学习引入MORL。OLS采用的是一种外循环方法，即通过求解一系列不同线性标量化权重向量的MDP来逐步构建凸覆盖集。Abels等[35]提出了一种条件网络，即通过增强Q网络来输出与权重相关的多目标Q值向量。为有效训练该网络，他们使用了动态权重设置更新规则和多样化经验重放，以提高样本效率并减少重放缓冲偏差。Nguyen等[36]基于深度Q网络提出了一种可扩展的MODRL框架，它既支持单一策略和多策略，也支持线性和非线性动作选择方法。Siddique等[37]使用非线性函数来平衡多个优化目标，而不是使用通常的线性加权方法来选择动作。He等[38]利用基于深度Q网络（deep Q network, DQN）的多智能体强化学习（multi-agent reinforcement learning, MARL）系统解决了纺织品生产过程的多目标优化问题。他们提出了一种基于DQN的自适应MARL框架，其中，m_{A} ($m_{\mathrm{A}} \in \mathbf{N}^{+}$)个优化目标被表述为$m_{\mathrm{A}}$个DQN智能体，这些智能体通过基于马尔可夫博弈构建的自适应过程进行训练。Yang等[39]使用向量化值函数并执行包络更新，利用前沿的凸包络更新参数。Luo等[40]设计了一种两层DQN，用于解决有新工件插入的多目标柔性作业车间问题。上层DQN是一个控制器，用于确定下层智能体的优化目标，下层作为执行器输出各调度规则的Q值。该方法未能同时考虑多个目标，因此需要探索一种有效的MODRL调度方法。

2.3. 研究不足

综上所述，可以得出以下结论：①DRL方法通常用于选择元启发式算法中的局部搜索算子，无法实时响应随机的工件到达；②虽然可以训练DRL智能体选择调度规则以获得更好的响应速度，但会减少解空间，得到的解决方案不能达到最优；③端到端的DRL在调度问题复杂状态空间的表示上显示出了巨大的潜力。据已有研究，端到端DRL尚未应用于考虑阻塞约束的分布式混合流水车间调度问题中。由于DHHBFSP包含更复杂的特性并需要实时决策，本研究考虑开发一种基于端到端DRL的在线调度方法，该方法根据智能体获得的概率分布直接选择未调度的工件。

从策略网络的角度来看，多目标深度强化学习方法可分为单策略方法和多策略方法。单策略方法将权重分配给多个目标，并将它们合并为一个目标进行评估优化。多策略方法可以充分探索小规模连续优化问题的凸包，并且也适用于探索离散优化问题（如调度问题）中的帕累托前沿。然而，由于其状态空间的复杂性，训练时间呈指数级增长。因此，本研究采用单策略框架，将两个PPO网络集成到一个智能体中，并开发了一个基于向量的奖励函数来解决设置不同目标权重的棘手问题。

3. DHHBFSP多目标在线问题描述

针对DHHBFSP，工件随机到达的在线调度问题可描述如下。

在已有的调度安排下，随机到达的工件需要分配到F个不同的加工工厂$H=\{H_1, H_2, ..., H_F\}$。所有工件都有K个操作，加工路线相同，但处理时间取决于分配的工厂。例如，第j个操作J_i的标准加工时间用$p_{i,j}\ (i, j \in \mathbf{N}^+)$表示。那么，当$J_i$被分配到$H_f\ (f \in \mathbf{N}^+)$并在第$j$个阶段以加工系数$\mu_{f,j}$进行加工时，实际加工时间应为$p_{i,j} \cdot \mu_{f,j}$。一个混合流程车间$H_f$包含$K$个加工阶段，至少有一个阶段配备了一台以上的可用机器，其中同一阶段的机器数量（用$M_{f,j}$表示）可能因工厂不同而不同。此外，在此生产环境中还考虑了阻塞约束，即只有在下一阶段有机器可用时，工件才能被释放。否则，工件应保留在当前机器上；这种阻塞状态也会消耗一定的能源。由于工件到达的随机性，DHHBFSP的工件总数是未知的，为了简化数学表述，假设在整个调度周期内有N个工件存在。其他常见的生产约束条件如下：

（1）所有机器在零时刻都可用；

（2）一个工件只能由一个工厂加工；

（3）一个工件一次只能进行一个操作；

（4）一台机器一次只能加工一个操作；

（5）所有准备时间和运输时间都被忽略或包含在加工时间内。

因此，DHHBFSP包括两个决策过程：将未调度工件分配给工厂和确定每个工厂的加工顺序。DHHBFSP的目标是同时最小化总拖期（total tardiness, TT）和总能耗（total energy consumption, TEC）。一旦给出DHHBFSP的可行计划，就可以根据完成时间和到期时间确定TT，而TEC则包括加工、空闲和阻塞状态下的能耗。以下是TT和TEC的计算方法：

$$\text{Minmize TT} = \sum_{i=1}^{N} \max\left(c_{i,K} - D_i\right) \tag{1}$$

$$\text{Minimize TEC} = E_{\text{p}} + E_{\text{i}} + E_{\text{b}} \tag{2}$$

$$E_{\text{p}} = \sum_{i=1}^{N}\sum_{j=1}^{K}\sum_{f=1}^{F} x_{i,f} \cdot p_{i,j} \cdot \mu_{f,j} \cdot e_{f,j} \tag{3}$$

$$E_{\text{i}} = \sum_{i=1}^{N}\sum_{j=1}^{K}\sum_{f=1}^{F} x_{i,f} \cdot \alpha_{i,j} \cdot e_{f,j} \cdot e_{\text{idle}} \tag{4}$$

$$E_{\text{b}} = \sum_{i=1}^{N}\sum_{j=1}^{K}\sum_{f=1}^{F} x_{i,f} \cdot \beta_{i,j} \cdot e_{f,j} \cdot e_{\text{block}} \tag{5}$$

式中，F为工厂总数；D_i为J_i的到期时间；$c_{i,K}$为J_i第K次操作的完成时间；$\alpha_{i,j}$为J_i第j次操作的空闲时间；$\beta_{i,j}$为J_i第j次操作的阻塞时间；$\mu_{f,j}$为工厂H_f第j阶段操作的过程时间系数；$e_{f,j}$为工厂H_f中第j个阶段的标准单位时间能耗；e_{idle}为机器空闲时间的单位能耗系数；e_{block}为机器堵塞时间的单位能耗系数。如果J_i被分配到H_f，则$x_{i,f}=1$，否则$x_{i,f}=0$。TT是总拖期，TEC是总能耗，E_{p}是加工状态下的能耗，E_{i}是空闲状态下的能耗，E_{b}是阻塞状态下的能耗。

本文还建立了一个数学模型来描述多目标DHHBFSP。该模型见附录A中的A部分。

4. 基于IPPO的DHHBFSP多目标在线调度方法

4.1. 强化学习基础

提出的IPPO旨在通过采用多智能体框架改进PPO训练方法，以解决多目标DHHBFSP问题。因此，本小节将介绍IPPO的一些基础概念，以帮助更好地理解该方法，包括MORL和MADRL。

4.1.1. 多目标强化学习

强化学习通常被建模为马尔可夫决策过程（Markov decision process, MDP），其中包含$M=\left(S, A, P, \gamma, R\right)$五个部分。$S$是环境中所有状态$s$的集合；$A$是智能体动作$a$的集合；$P$是状态转移函数$P(s'|s,a)$，表示智能体在状态$s$中执行动作$a$后发现自己处于下一个可能状态$s'$的概率；奖励函数$R(s,a)$表示在状态$s$中执行动作$a$后立即获得的奖励；$\gamma \in [0,1]$是折扣因子。在多目标优化中，目标空间由两个或多个维度组成，因此，一般的MDP被扩展为多目标MDP（multi-objective MDP, MOMDP）。MOMDP因为存在多个目标，需要根据目标维度提供奖励向量而不是标量奖励，即$\boldsymbol{R}(s_i, a_i) = (R_1(s_i, a_i), \ldots R_m(s_i, a_i))$，其中，$m$代表目标个数（$m \in \mathbf{N}^+$）。同时，状态$s$的状态值函数$\boldsymbol{V}$也是一个向量。

在多目标强化学习（MORL）领域，部分研究聚焦于基于单策略架构的优化框架。这类方法的核心思想在于通过设计动态权衡机制（如权重聚合或偏好引导函数），将多维目标空间中的交互关系映射为单维标量评估体系，从而在保留多目标特性的同时，利用单策略强化学习框架实现帕累托前沿的高效探索。标量化函数g是将向量$\boldsymbol{V}$投影到标量的函数：

$$\boldsymbol{V}_w = g(\boldsymbol{V}, \boldsymbol{w}) \tag{6}$$

式中，$\boldsymbol{w}$为权重向量。标量化函数可以有多种形式和偏好，但最常见的函数是本文使用的线性标量化函数。

4.1.2. 多智能体DRL

典型的MADRL算法，如多智能体深度确定性策略梯度下降[41]和多智能体PPO [42]，均采用集中训练和分布

式执行（centralized training and distributed execution, CTDE）的方式。CTDE 意味着所有智能体的网络都需要联合状态和动作进行训练（集中训练），但每个智能体可以独立决策并输出自己的动作（分布式执行）。由于每个智能体在决策时必须考虑其他智能体的动作和状态，因此智能体之间需要频繁的数据交互。当单个智能体进行集中训练时，其批判网络需要使用全局状态；因此，随着智能体数量的增加，学习的扩展难度会越来越大，这对算法的训练成本和收敛性也构成了挑战。同时，并非所有智能体都需要在同一时刻做出决策，因此，如果使用同步的决策测量，可能会产生大量无效数据，从而降低训练效率。

因此，本研究采用一种具有异步决策的多智能体深度强化学习方法，其中多个智能体协作并通过奖励相互影响。这些智能体分别进行训练并异步执行。当仿真环境运行到一个智能体当前工件第一阶段离开时间（决策时间）时，系统会从工件池中选择下一个工件，并将选择后产生的奖励添加到下一个决策点的状态中。因此，不同智能体在工件选择前后执行的操作会相互影响。因此，多个智能体在决策上是异步的，但智能体之间的协作可以通过决策前后的状态和奖励来实现。

4.2. MOMDP 建模

4.2.1. MOMDP 模型定义

在训练过程之前，先定义 MOMDP 模型中的状态、动作空间、状态转移概率和奖励函数。本研究构建的状态表征体系整合了待加工工件的动态属性集合，这些参数共同构成生产系统的实时全景映射。基于马尔可夫决策过程的无记忆特性，将各智能体的观测空间聚焦于未完成工件的实时生产数据。每个智能体都会根据本地观测结果（包括 $\boldsymbol{L}$ 个工件的状态向量）做出决策。每个向量包含 14 个特征，用于描述当前加工环境的状态。状态向量的详细信息如表 1 所示。图 1 显示了一个智能体的观察状态。多智能体环境通过 Python 的 SimPy 模拟库实现并执行状态转移。

4.2.2. 动作空间

MOMDP 中的动作是从新到达的工件池或当前工件池中抽取一个工件。为了提高训练效率，每一个训练轮次都

表 1 状态描述

States	Definitions
Process times	All process times on all stages
D-A	Time difference between the due date and current time
Blocking time	Estimated blocking time after choosing this job
Tardiness signal	Equal to 1 if no tardiness, otherwise equal to −1
Reward of last decision	Reward vector of last decision
Job type of last decision	Job from new arrival pool or not
Estimated immediate reward	The estimated immediate reward after choosing this job
Weights of all objectives	Weights between two objectives

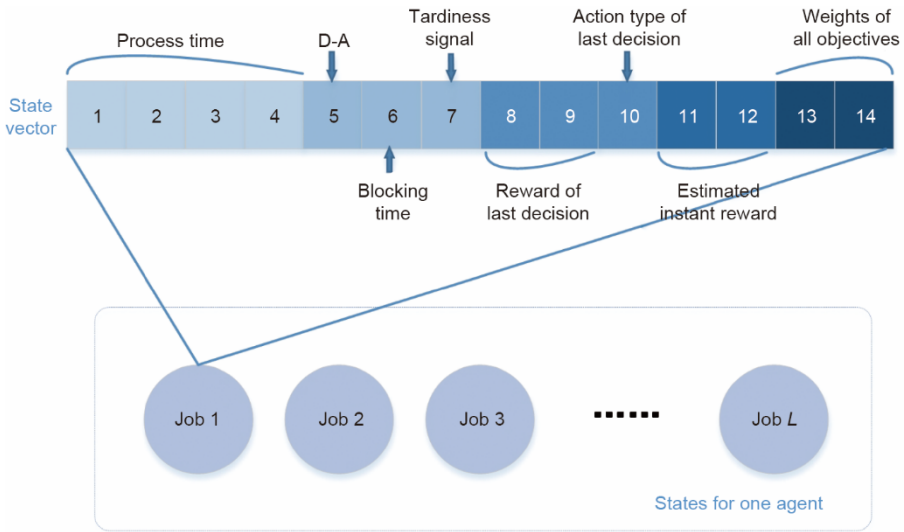

图 1. 智能体观察状态（$L \in \mathbf{N}^{+}$）。

有初始工件，并根据INSGA-Ⅱ生成调度计划，在加工过程中有工件随机到达。新工件不会自主选择工厂，而是与所有未加工的工件一起放入工作池，被选中的工件会被直接解码并安排到相应的工厂，并根据第一可用机器规则进行机器选择。每个阶段的离开时间在解码后进行更新，当模拟环境运行到一个智能体当前工件第一阶段的离开时间（决策时间）时，从工件池中选择下一个工件。在MOMDP中，动作就是从当前工件池中选择一个工件，工件池中的工件数量会随时间变化，为了固定输入规模，本研究使用最早到期（earliest due date, EDD）规则从所有工件中选择8个工件作为输入，最后由智能体得出8个工件的概率分布，并选择最大概率的工件作为下一步的动作。在多智能体环境下通过Python的Simpy仿真库进行状态转移，因此状态转移概率为1。

4.2.3. 奖励函数

本文中的奖励$\boldsymbol{R}$是一个向量，而不是标量，它与两个目标相对应。它由三部分组成，包括即时奖励$\overrightarrow{\mathbf{IR}}$、阶段奖励$\overrightarrow{\mathbf{ER}}$和先前奖励$\overrightarrow{\mathbf{PR}}$。为了避免在智能体的学习过程中以获得单步奖励为目标，本文将$\overrightarrow{\mathbf{IR}}$和$\overrightarrow{\mathbf{ER}}$函数分开设计。$\overrightarrow{\mathbf{IR}}$是选择工件时产生的拖期时间和能耗倒数的向量，如式（7），当拖期时间为0时，给与奖励为1。每个轮次结束后，所有工厂的实际平均拖期时间和能耗的倒数为$\overrightarrow{\mathbf{ER}}$，如式（8）所示，并在每个轮次结束时更新为所有智能体的奖励。考虑到之前的决策对当前状态的影响，会增加一个带有折扣系数的$\overrightarrow{\mathbf{PR}}$，如式（9）所示。为了避免奖励值过大或过小对状态值函数训练的负面影响，本文采用奖励缩放法[43]来调整奖励的尺度。

$$\overrightarrow{\mathbf{IR}} = \left(\frac{1}{T_t}, \frac{1}{\mathrm{EC}_t}\right) \tag{7}$$

$$\overrightarrow{\mathbf{PR}} = \left(\sum_{i=0}^{i=t-1} \frac{\gamma}{T_i},\ \sum_{i=0}^{i=t-1} \frac{\gamma}{\mathrm{EC}_i}\right) \tag{8}$$

$$\overrightarrow{\mathbf{ER}} = \left(\frac{1}{\mathrm{TT}\sum \mathrm{totalsteps}_f}, \frac{1}{\mathrm{TEC}\sum \mathrm{totalsteps}_f}\right) \tag{9}$$

$$\boldsymbol{R} = \overrightarrow{\mathbf{IR}} + \overrightarrow{\mathbf{PR}} + \overrightarrow{\mathbf{ER}} \tag{10}$$

式中，T_t为t时刻产生的拖期；EC_t为t时刻产生的能耗；γ为折扣因子；$\mathrm{totalsteps}_f$为工厂f在每一轮次中的步骤数。

4.3. 基于PPO的训练过程

4.3.1. 多智能体训练过程

在本节中，我们将介绍基于MADRL的拟议IPPO培训框架。在DHHBFSP环境中，每个工厂都被看作一个DRL智能体，并将其定义为用于工件分配的FA。多个FA以异步方式工作，没有直接交互，这表示一旦一个FA选择了一项工件并更新了计划，其他FA将观察到包含这些变化的更新状态，以避免冲突。提出的多智能体训练框架如图2所示。每个智能体都独立训练网络，但整个环境中前一步的行动和奖励都被视为下一步的状态。多智能体虚拟环境由固定数量的工作初始化，这些工作由之前研究中提出的HGA分配[44]。在新工件到达之前，所有智能体都会按照HGA生成的序列加工。一旦有新的工件到达，所有智能体都会进入学习过程，必须通过PPO网络在决策点选择下一个工件，直到没有工件选择为止。一个训练轮次完成后，虚拟环境将重新初始化，以开始新的训练轮次。当达到预定的训练轮次时，该过程结束。

基于这一训练框架，我们提出了MODRL的训练方法。在训练过程之前，环境参数和训练参数都已设定。在每一训练轮次（episode）中，虚拟环境都会重置，由HGA随机分配20个初始工件给不同的智能体加工。新工作到达后，当到达决策点时，agent i以EDD规则从工件池中选择8个工件作为输入，并获得观测值o_i和掩码m_i。如前所述，每个智能体都有两个PPO网络：$\mathrm{PPO}_{i,1}$和$\mathrm{PPO}_{i,2}$。如果训练轮次是奇数，agent i就会通过$\mathrm{PPO}_{i,1}$选择行动，接收奖励r_i、下一个观测值o_i'和下一个掩码m_i'，并将元组$(o_i, m_i, a_i, r_i, o_i', m_i')$存储到$\mathrm{memory}_{i,1}$中。否则，agent i将使用$\mathrm{PPO}_{i,2}$选择行动。在学习过程中，如果训练轮次达到策略训练迭代次数的一半（training iterations, TR），每个网络就会连续采样一批记忆数据，更新网络参数并保留记忆。如果训练轮次达到TR，则每个网络会连续采样一批记忆以更新网络参数，并清空记忆。当达到最大集数M时，训练过程结束。MODRL训练过程如算法1所示。

4.3.2. 单智能体PPO网络训练

本文使用PPO训练框架来训练深度学习网络[45]。一般来说，每个智能体只训练一个PPO网络，但针对多目标的单策略学习方法，只训练一个网络可能会缩小求解范围。因此，我们在一个智能体中训练两个目标权重分布不同的PPO网络。在训练过程中，这两个PPO网络按照奇数集和偶数集分别存储训练数据。训练示意图可见图3。

如上所述，这是一个多目标问题，奖励函数和状态值函数应该是表征两个目标的向量。在训练过程中，actor网络和critic网络的损失值应该是一个标量，以更新深度网络的参数。本文不使用不同的线性标量化权重向量来求解一系列标量MDP，而是随机抽样50组权重向量

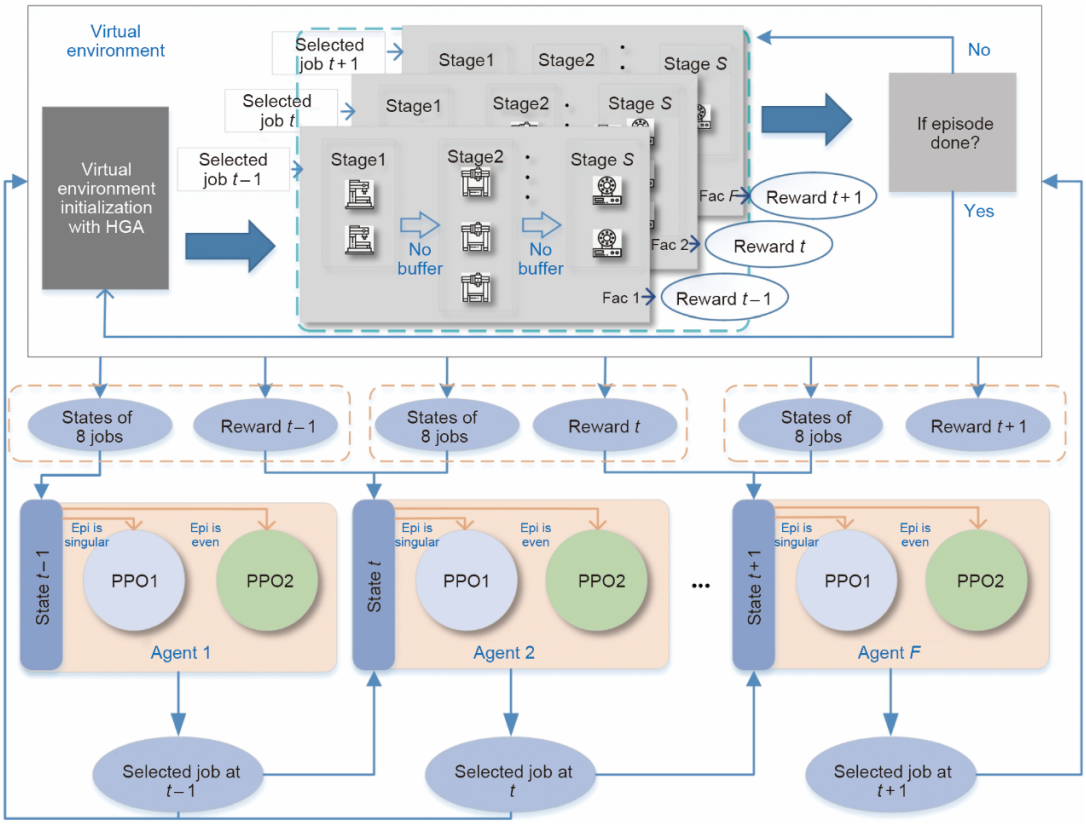

图2. 多智能体训练框架。Fac：工厂；Epi：训练集数。

$[w_{1,i}, w_{2,i}]$来计算平均标量化损失值，如式（11）所示。

$$\text{loss} = \sum_i w_{1,i} l_1 + w_{2,i} l_2 \tag{11}$$

PPO作为在线学习方法，使用单次相同的数据进行策略的多次更新，为了更有效地利用数据，本文提出了一种改进的学习策略，将训练过程分为两个部分：单策略数据训练和双策略数据训练。当replay buffer中的数据存储达到固定值的1/2时，在前TR/2次中，网络仅根据旧策略的数据进行更新。通过网络根据更新后的策略生成另一组训练数据，并在剩余的$TR/2$次中结合这两组数据更新当前的策略网络。

在本研究中，我们采用了参考文献[46]中提出的基于注意力的策略网络（attention-based policy network, APN），它主要由特征提取层、查询瞥见层（query glimpse layer）和指针层组成。APN的输入是建议的作业向量，输出是输入作业的概率分布。APN的详情见附录A中的B部分。

5. 数值实验

5.1. 实验设置

为了验证所提IPPO的有效性，本节进行了一系列数值实验。所有比较方法均使用Python进行编码，并在一台配备AMD Ryzen 5 6600H和Radeon Graphics CPU（主频为3.30 GHz，内存为16 GB，运行Windows 10 Standard 64位操作系统）的计算机上执行。在训练过程之前，多工厂环境的布局如表2所示。本文在GitHub上提供DHHBFSP运行环境的开放访问。

5.1.1. 算例设置

输入为8个工件的状态，每个状态包含8个14维向量。其中，actor的输出为8个动作，critic的输出为两个Q值。隐藏层与嵌入层的维度均设为128；多头注意力层（multi-head attention layer）包含8个注意力头（head）。参数优化采用Adam优化器，初始学习率设置为1×10^{-5}。对于新工件，根据排队理论，我们假设工件到达模式遵循到达率为λ（λ=20）的泊松分布。为了进行实验，我们随机生成了不同规模的实例。然后，工件总数设为$n \in \{10, 20, 50, 100, 200\}$。这些工件的到期日可通过不同交付时间紧度（due date tightness, DDT）$\in \{1, 1.5\}$的加工时间之和计算得出。每个工件数有5个实例，共50个实例。

5.1.2. 优化指标

在实验进行之前，本文选择了两个指标来评估多目标算法的性能，包括反世代距离（invert generational distance, IGD）和纯度（purity, P）。IGD是一个测量值，

算法1 IPPO训练过程

for episode = 1 to M do
  Reset the training environment with 20 initial jobs randomly
  Generate an initial schedule by the HGA
  for f = 1 to F do
    Combine all unscheduled jobs and select 8 jobs by the EDD
    The fth agent obtains the observation $o_i = [o_1, o_2, ..., o_S]$ and mask m_i
    if episode%2 = 1 **then**
      Choose action a_i for $PPO_{f,1}$
      Obtain reward r_i, the next observation o_i' and the next mask m_i'
      Decompose the action for the machine selection
      Store the tuple $(o_i, m_i, a_i, r_i, o_i', m_i')$ into $memory_{f,1}$
      if episode%TR = TR/2 **then**
        $PPO_{f,1}$ trained with a random B of $memory_{f,1}$
        Preserve the memory and update the network of $PPO_{f,1}$
      else if episode%TR = 0 **then**
        $PPO_{f,1}$ trained with a random B of $memory_{f,1}$
        Empty $memory_{f,1}$
      end if
      else if episode%2 = 0 **then**
      Choose action a_i for $PPO_{f,2}$
      Obtain reward r_i, the next observation o_i' and the next mask m_i'
      Decompose the action for the machine selection
      Store the tuple $(o_i, m_i, a_i, r_i, o_i', m_i')$ into $memory_{f,2}$
      if episode%TR = TR/2 **then**
        $PPO_{f,2}$ trained with a random B of $memory_{f,2}$
        Preserve the memory and update the network of $PPO_{f,2}$
      else if episode%TR = 0 **then**
        $PPO_{f,2}$ trained with a random B of $memory_{f,2}$
        Empty $memory_{f,2}$
      end if
    end if
  end for
end for

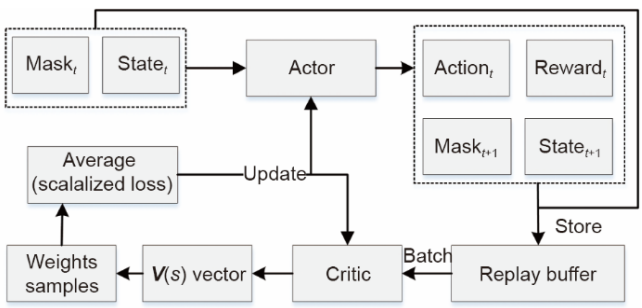

图3. 单智能体的训练过程。$V(s)$：s的状态依存值。

定义为问题的最优帕累托前沿（Pareto front*, PF*）中每个点与算法获得的帕累托前沿（Pareto front, PF）之间的距离。IGD是世代距离的改进，是一个更全面的指标，同时反映了收敛性和多样性。具体表现为

$$\mathrm{IGD} = \frac{\sqrt{\sum_{i=1}^{n} d_i^{*2}}}{n} \tag{12}$$

式中，d_i^*表示由PF*构成的每个点与PF最近的点之间的欧氏距离；n表示PF*的点数。因此，IGD值越小越好。由于真正的PF*可能是未知的或难以获得的，本研究通过合并每个实例下所有算法的运行结果来构建PF*。P是算法X产生的非支配解个数与最优的非支配解个数的比率。P指标反映了算法对其他方法的支配度。它的取值范围为[0, 1]，其中取值1意味着非支配集ND_x完全支配了其他算法生成的所有解，因此，P值越大越好。算法X的P值计算公式如下：

$$P_X = \frac{|\mathrm{ND}_X'|}{|\mathrm{ND}'|} \tag{13}$$

式中，$|\mathrm{ND}_X'|$是算法X生成的非支配解在最优非支配解集中的数量；$|\mathrm{ND}'|$是最优非支配解集的数量。

表2 多工厂环境布局

Setting	Values
The number of initial jobs	20
The number of factories	2
The number of stages	4
The number of machines in factory 1	[2, 3, 1, 4]
The number of machines in factory 2	[4, 3, 3, 2]
Time factors of factory 1	[1, 1.3, 0.7, 1]
Time factors of factory 2	[1.3, 1, 1, 0.7]
Energy consumption factors of factory 1	[1.3, 0.7, 1.3, 0.7]
Energy consumption factors of factory 2	[0.7, 1, 1, 0.7]
Energy consumption factors of blocking state	0.3
Energy consumption factors of idle state	0.2

5.1.3. 参数优化

为了发挥IPPO的优势，减少参数对不同实例的影响，通过参数优化，选出综合表现较好的一组参数值。本文提出的IPPO包含四个关键参数：批量大小（B）、折扣系数（γ）、先前奖励折扣系数（PR）和策略的训练迭代次数（TR）。田口方法是试验次数较少的最佳统计分析方法之一，本文对所有实例采用田口方法分析了参数不同水平的表现以及敏感度。

表3 参数水平值

Level	B	γ	PR	TR
1	256	0.85	0.7	40
2	512	0.90	0.8	60
3	1024	0.95	0.9	80

通过田口实验，本文设计了正交阵列L9(3^4)。表3列出了每个参数的值。每个组合将运行1000训练轮次，并独立求解一个随机实例30次。隐藏层和嵌入层的维数均设为128。参数优化采用Adam优化器，学习率设定为1×10^5。从图4可以看出，B、γ、PR和TR的最佳参数值分别为1024、0.95、0.9和40。批量大小B的值对训练过程最为敏感，γ对训练过程的影响最小。这些参数的结果详见附录A中的C部分。

5.2. IPPO改进策略对比实验

训练过程是DRL中最重要的部分之一。为了证明所提出的训练方法的有效性，本节将进行两个实验：首先，比较了单PPO网络与双PPO网络的训练过程；其次，将使用新学习策略的训练框架与不使用新学习策略的框架进行比较。每个实例均独立求解30次。详细结果见附录A中的D部分。

5.2.1. 单PPO与双PPO对比

对于本文提出的IPPO方法，两个PPO网络的目标是在不同的权重目标空间中搜索解。因此，我们为两个网络分配不同的权重作为输入。同时，为了简化训练过程，我们将训练数据分别放入两个缓冲区。相反，单个PPO网络（PPO_S）的训练过程输入无需分配权重集，只需要一个训练缓冲区。

图5显示了这两种方法在不同目标上的收敛过程。图中表明PPO_S可以在第4000次左右达到收敛状态，比IPPO在第6000次时更快。另外，在没有权重作为输入的情况下，PPO_S的训练过程比IPPO更稳定，这说明两个

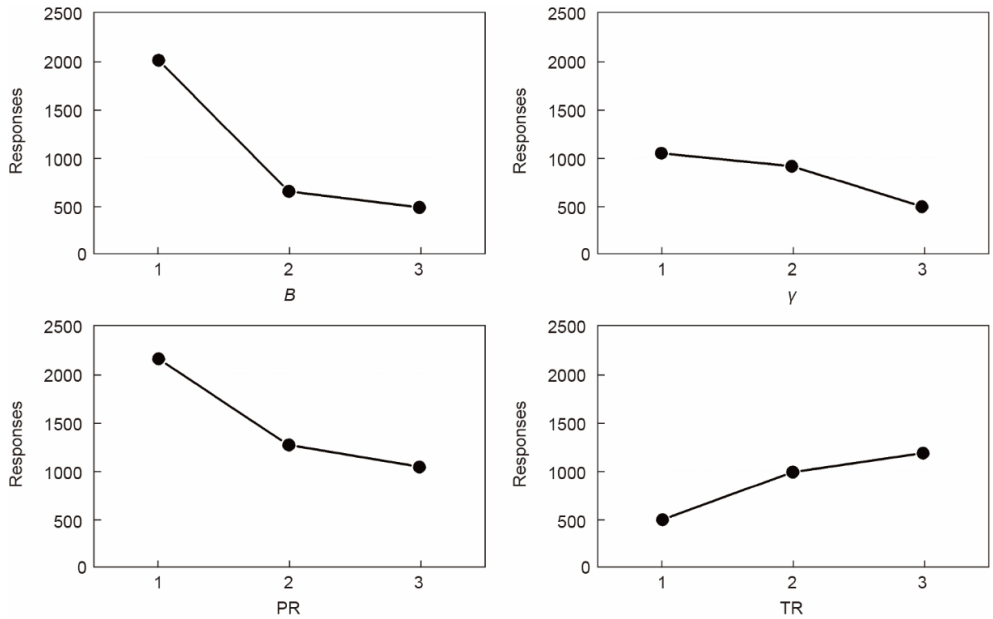

图4. 参数敏感性对比图。

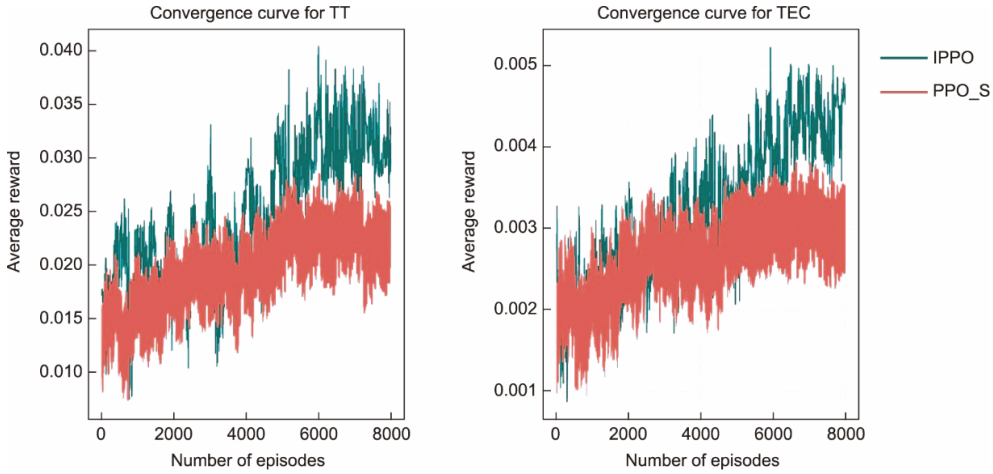

图5. IPPO和PPO_S收敛曲线。

目标的权重会使训练过程上下波动。但将权重作为输入可以使PPO模型在训练过程中获得更高的平均奖励。这会让PPO在每个决策点都做出更好的决策。为了证明这一观点，我们在25个包含两个DDT的实例上对IPPO和PPO_S进行了比较。从表4中可以看出，在DDT分别为1和1.5时，IPPO分别获得了24个和25个最佳IGD和P值。同时，为了验证双PPO网络在IPPO中的有效性，我们在图6中展示了随机选取的两个实例来说明IPPO和PPO_S的帕累托前沿。结果表明，在这些实例中，IPPO比PPO_S能获得更宽的帕累托前沿，而且IPPO的所有帕累托解均优于PPO_S。

5.2.2. 训练策略对比

为了更有效地利用训练数据，本文提出了一种新的学习策略。本实验展示IPPO与采用旧策略（PPO_OL）的PPO之间的比较。首先，从图7的IPPO和PPO_OL收敛曲线中可以看出，采用新学习策略的IPPO比PPO_OL可以提前1000个轮次收敛。这一结果表明，重复使用训练数据可以加快收敛过程。此外，它还使IPPO收敛到更高的平均奖励，相对于TT和TEC分别提升18%和27%。

为了验证IPPO模型的有效性，我们还在25个有两个DDT的实例上对IPPO和PPO_OL进行了比较。从表5中可以看出，在DDT分别为1和1.5的情况下，IPPO分别获

表4 IPPO和PPO_S的IGD值和P值

Instances	IGD (DDT = 1)		P (DDT = 1)		IGD (DDT = 1.5)		P (DDT = 1.5)	
	PPO_S	IPPO	PPO_S	IPPO	PPO_S	IPPO	PPO_S	IPPO
J10	5.61×10^3	$\mathbf{5.57 \times 10^2}$	0	**0.51**	5.50×10^3	$\mathbf{9.88 \times 10^1}$	0	**0.68**
J20	6.97×10^3	$\mathbf{3.04 \times 10^2}$	0	**0.75**	5.48×10^3	$\mathbf{4.67 \times 10^2}$	0	**0.72**
J50	4.38×10^3	$\mathbf{7.91 \times 10^2}$	0.03	**0.53**	8.98×10^3	$\mathbf{3.19 \times 10^2}$	0	**0.83**
J100	4.96×10^3	$\mathbf{2.85 \times 10^3}$	0.23	**0.36**	1.29×10^4	$\mathbf{2.67 \times 10^3}$	0	**0.72**
J200	2.19×10^4	$\mathbf{1.37 \times 10^4}$	0.18	**0.58**	9.62×10^4	$\mathbf{3.77 \times 10^3}$	0.05	**0.74**

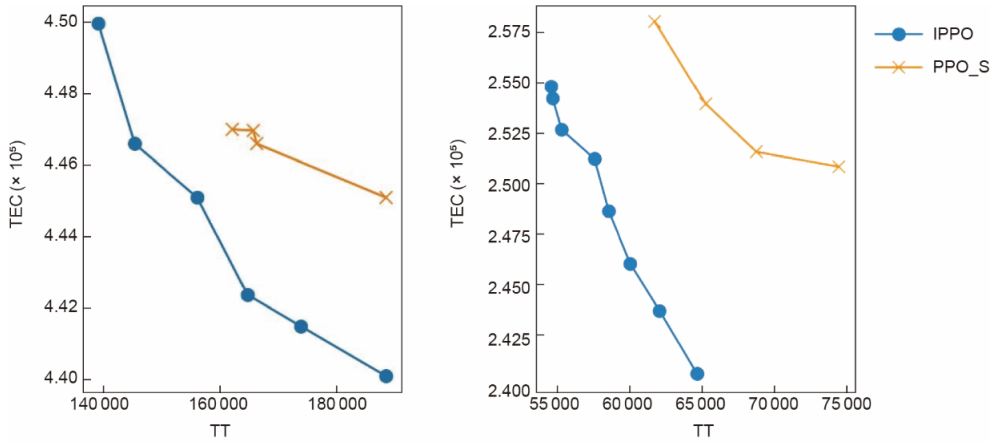

图6. IPPO和PPO_S的PF对比。

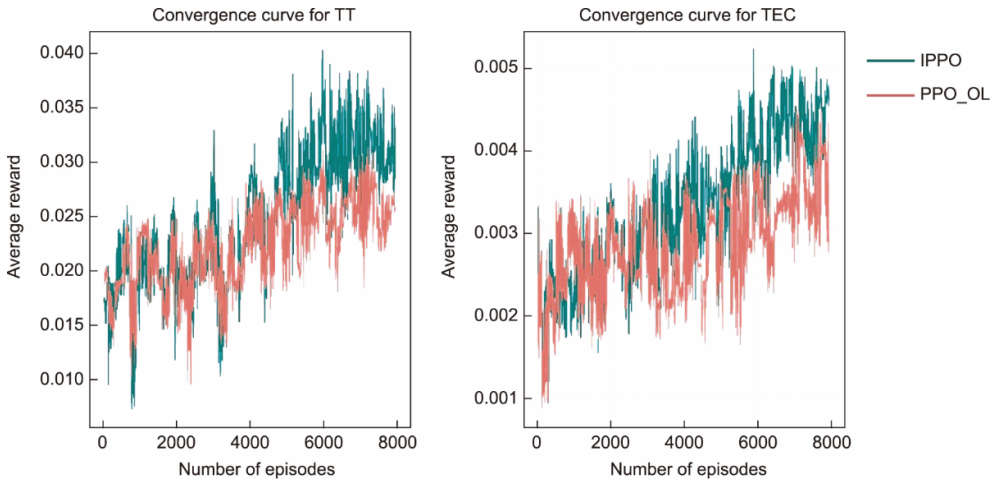

图7. IPPO和PPO_OL收敛曲线。

表5 IPPO和PPO_OL的IGD值和*P*值

Instances	IGD (DDT = 1.0)		*P* (DDT = 1.0)		IGD (DDT = 1.5)		*P* (DDT = 1.5)	
	PPO_OL	IPPO	PPO_OL	IPPO	PPO_OL	IPPO	PPO_OL	IPPO
J10	6.94×10^2	**5.57×10^2**	0.30	**0.51**	2.41×10^3	9.88×10^1	0.08	**0.68**
J20	2.00×10^3	**3.04×10^2**	0.06	**0.75**	2.29×10^3	4.67×10^2	0.16	**0.72**
J50	1.89×10^3	**7.91×10^2**	0.26	**0.53**	3.10×10^3	3.19×10^2	0.10	**0.83**
J100	5.26×10^3	**2.85×10^3**	0.17	**0.36**	7.15×10^3	2.67×10^3	0.15	**0.72**
J200	2.71×10^4	**1.37×10^4**	0.18	**0.58**	1.74×10^4	3.77×10^3	0.14	**0.74**

得了25个和23个最佳IGD值。在*P*方面，IPPO分别获得了24个和23个较好值。从结果来看，IPPO的双策略训练方法可以获得更好的帕累托前沿。IPPO的IGD越小，说明IPPO生成的非支配解集越接近真正的帕累托前沿。

5.2.3. 奖励更新对比

在我们提出的IPPO方法中，我们改进了奖励计算函数，增加了阶段奖励和先前奖励。为了证明其对IPPO的影响，在本部分中，我们比较了改进奖励的方法（IPPO）和未改进奖励的方法（PPO_U）。两种方法使用相同的训练框架，每个实例独立求解30次。从图8中IPPO和PPO_U的收敛曲线可以看出，对于TT和TEC，IPPO在6000次左右就达到了稳定的学习状态，而PPO_U则用了7000多次才达到收敛，这表明在相同的训练框架下，IPPO比PPO_U收敛得更快。但IPPO的奖励训练曲线并不平滑，在收敛前会有几次上升和下降。这说明奖励函数可以提高训练效率，但是因为考虑了多个因素，导致训练过程中给出的奖励不够均匀。相比之下，PPO_U的曲线相对平稳，整个趋势是逐渐上升的，但是优化趋势并不明显。充分表明了改进的奖励函数可以有效地提高训练过程的效率并获得更好的奖励。

表6显示了实例在不同DDT条件下的IGD值和*P*值。我们可以知道，在DDT等于1和1.5的情况下，IPPO分别获得了23个和24个最佳IGD，*P*值的结果也几乎相同。这表明，IPPO中更新的奖励提高了PPO的性能，包括加快收敛速度，并为这50个实例获得更好的解。IPPO与PPO_S、PPO_OL和PPO_U之间的详细分析见附录A中的E部分。

5.3. 与规则对比实验

在验证了IPPO中每个改进部分的有效性之后，本节将IPPO与几种常用且有效的调度规则（包括EDD、MDD、MST、SPT和LPT）以及一种常用的MODRL方法（即使用相同PPO网络结构的优化线性支持DRL(SW) [34]）进行对比。从表7和表8可以看出，与调度规则和SW相比，IPPO的IGD和*P*值表现更好。详细结果见附录A中的F部分。

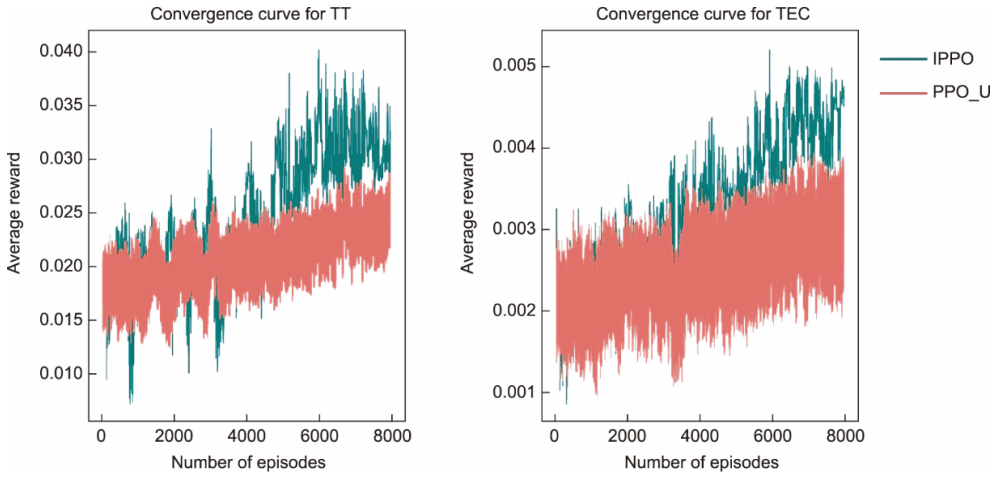

图8. IPPO和PPO_U的收敛曲线。

表6 IPPO与PPO_U的IGD值和*P*值

Instances	IGD (DDT = 1.0)		*P* (DDT = 1.0)		IGD (DDT = 1.5)		*P* (DDT = 1.5)	
	PPO_U	IPPO	PPO_U	IPPO	PPO_U	IPPO	PPO_U	IPPO
J10	1.32×10^3	**5.57×10^2**	0.19	**0.51**	1.72×10^3	**9.88×10^1**	0.25	**0.68**
J20	2.06×10^3	**3.04×10^2**	0.19	**0.75**	1.57×10^3	**4.67×10^2**	0.13	**0.72**
J50	2.71×10^3	**7.91×10^2**	0.18	**0.53**	3.93×10^3	**3.19×10^2**	0.07	**0.83**
J100	6.50×10^3	**2.85×10^3**	0.14	**0.36**	7.50×10^3	**2.67×10^3**	0.13	**0.72**
J200	2.83×10^4	**1.37×10^4**	0.07	**0.58**	4.91×10^4	**3.77×10^3**	0.06	**0.74**

表7 算法的IGD值

Instances	EDD	MDD	MST	LPT	SPT	SW	IPPO
J10	2.09×10^4	1.58×10^4	1.74×10^4	2.42×10^4	1.45×10^4	1.50×10^4	$\mathbf{3.27 \times 10^2}$
J20	2.41×10^4	1.19×10^4	1.72×10^4	2.12×10^4	1.74×10^4	1.33×10^4	**0**
J50	2.57×10^4	3.58×10^4	2.92×10^4	2.17×10^4	2.58×10^4	2.18×10^4	$\mathbf{6.53 \times 10^3}$
J100	1.29×10^5	9.11×10^4	7.53×10^4	8.65×10^4	8.97×10^4	4.35×10^4	$\mathbf{2.76 \times 10^4}$
J200	3.15×10^5	2.65×10^5	3.21×10^5	2.20×10^5	2.43×10^5	7.22×10^4	$\mathbf{4.64 \times 10^4}$

表8 算法的*P*值

Instances	EDD	MDD	MST	LPT	SPT	SW	IPPO
J10	0	0	0	0	0	0	**0.90**
J20	0	0	0	0	0	0	**1.00**
J50	0	0	0	0	0	0	**0.67**
J100	0	0	0	0	0	0	**0.53**
J200	0	0	0	0	0	0	**0.87**

表9 对比IGD值

Instances	MOEAD23	Pdjaya23	MAEQL	MAGPI	IPPO
J10	1.48×10^4	1.38×10^4	2.13×10^4	1.53×10^4	$\mathbf{3.27 \times 10^2}$
J20	1.38×10^4	1.47×10^4	2.80×10^4	1.20×10^4	**0**
J50	1.82×10^4	1.40×10^4	3.72×10^4	1.40×10^4	$\mathbf{6.53 \times 10^3}$
J100	7.20×10^4	7.02×10^4	4.66×10^4	3.37×10^4	$\mathbf{2.76 \times 10^4}$
J200	1.89×10^5	1.50×10^5	6.81×10^4	5.55×10^4	$\mathbf{4.64 \times 10^4}$

表10 对比*P*值

Instances	MOEAD23	Pdjaya23	MAEQL	MAGPI	IPPO
J10	0.05	0.05	0	0	**0.90**
J20	0	0	0	0	**1.00**
J50	0	0	0	0.33	**0.67**
J100	0.10	0	0	0.37	**0.53**
J200	0	0	0	0.13	**0.87**

5.4. 与元启发式方法和DRL方法进行对比

为了进一步研究拟议IPPO的性能，将其与近期研究中的元启发式算法和DRL方法进行了比较，包括基于分解的多目标进化算法（MOEAD23）[47]、基于帕累托的离散jaya算法（Pdjaya23）[48]和多智能体包络Q学习（MAEQL）[39]。此外，IPPO还与多智能体广义策略改进（MAGPI）方法进行了比较，后者采用了参考文献[49]中提出的广义策略改进（generalized policy improvement, GPI）。为了实现MAGPI，我们采用了GitHub上的GPI基本代码，并将GPI方法纳入了本研究的多智能体框架中。MAEQL和MAGPI的隐藏层大小与IPPO相同。

如表9和表10所示，与元启发式算法和两种MODRL相比，IPPO在IGD和*P*值方面表现更好，这说明学习策略和网络设计是有效的。详细结果见附录A中的F部分（表S7和表S8）。

图9和图10所示的帕累托前沿是不同算法在随机选取的两个实例上得到的帕累托近似值。可以看出，在二维坐标中，IPPO生成的帕累托前沿比其他算法生成的帕累托前沿更接近图的左下角，这表明IPPO在这两个实例中比其他算法获得了更宽的帕累托前沿。IPPO得到的非优势解的分布也更好。与其他两种DRL算法相比，结果表明所提出的损失函数计算方法能更好地优化两个目标。此外，结果也表明，由于在学习过程中考虑了全局信

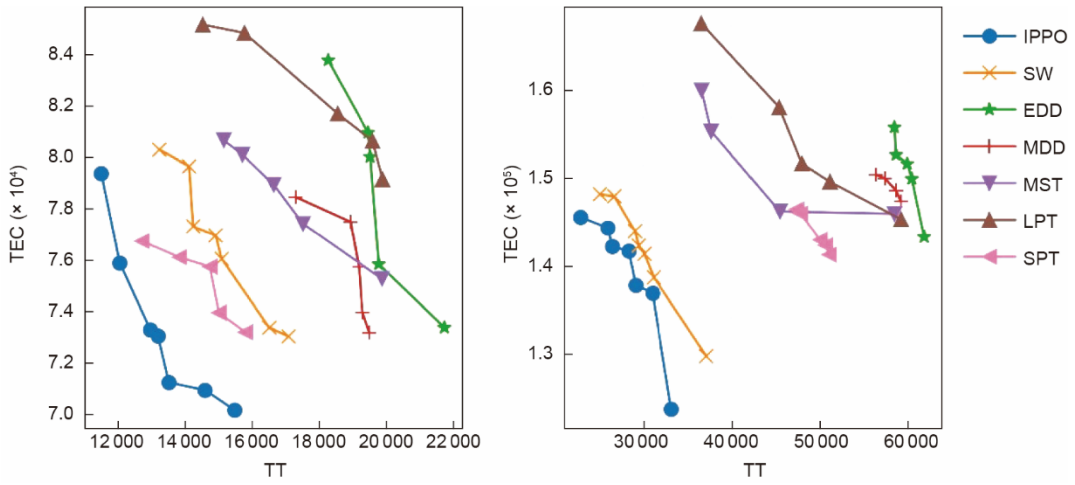

图9. IPPO与分派规则PF对比图。

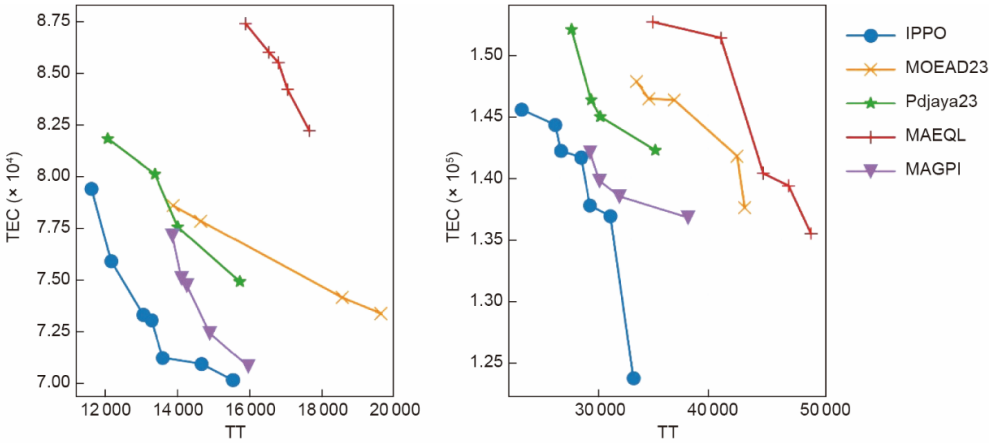

图10. IPPO与其他方法PF对比图。

息，IPPO和MAGPI比其他元启发式算法能获得更好的解。

5. 结论

本文研究了以总拖期和总能耗为优化目标的DHH-BFSP。为了有效求解这一复杂问题，本研究提出了一种基于PPO的多目标DRL方法，即IPPO。通过定义不同智能体的状态空间、动作空间和奖励函数，构建了MOMDP模型，并从三个方面改进了奖励函数。然后，在IPPO中引入新的学习策略，分别训练单策略数据和双策略数据，重复使用旧的策略数据。同时，针对框架中的智能体，在单个智能体中定义了两个结构相同、权重目标集不同的PPO网络，以拓宽帕累托前沿。实验表明，改进后的奖励函数和新的学习策略可以加快收敛过程，并在大多数情况下获得更好的模型。此外，虽然IPPO收敛过程不稳定且对训练设置敏感，但两个PPO网络能使非支配解的分布更加分散。通过与其他MODRL、元启发式算法和调度规则进行比较，证明了所提出的IPPO算法具有良好的求解能力。IPPO不仅能在所有实例中获得更好的非优势解，还能形成更宽的帕累托前沿。这表明，与其他方法相比，所提出的损失函数计算方法能更好地优化两个目标。

在未来的工作中，我们将优化训练设置，以确保IPPO在各种实例中表现相同。此外，我们可能会考虑将所提算法应用于不同类型的分布式调度问题，如分布式作业车间调度问题和分布式动态作业车间调度问题。最后，由于元启发式算法可以获得更好的解决方案，我们将研究一种结合元启发式算法的新DRL方法，以解决多目标问题。

致谢

本研究部分由科技部国家重点研发计划（2022YFE0114200）和国家自然科学基金（U20A6004）资助。

Compliance with ethics guidelines

Xueyan Sun, Weiming Shen, Jiaxin Fan, Birgit Vogel-Heuser, Fandi Bi, and Chunjiang Zhang declare that they have no conflict of interest or financial conflicts to disclose.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.eng.2024.11.033.

References

[1] Gao L, Shen W, Li X. New trends in intelligent manufacturing. Engineering 2019;5(4):619–20.
[2] Han W, Guo F, Su X. A reinforcement learning method for a hybrid flow-shop scheduling problem. Algorithms 2019;12(11):222.
[3] Martinez S, Dauzère-Pérès S, Guéret C, Mati Y, Sauer N. Complexity of flowshop scheduling problems with a new blocking constraint. Eur J Oper Res 2006;169 (3):855–64.
[4] Srai JS, Kumar M, Graham G, Phillips W, Tooze J, Ford S, et al. Distributed manufacturing: scope, challenges and opportunities. Int J Prod Res 2016;54(23): 6917–35.
[5] Shao Z, Pi D, Shao W. Hybrid enhanced discrete fruit fly optimization algorithm for scheduling blocking flow-shop in distributed environment. Expert Syst Appl 2020;145:113147.
[6] Qin HX, Han YY, Liu YP, Li JQ, Pan QK, Han X. A collaborative iterative greedy algorithm for the scheduling of distributed heterogeneous hybrid flow shop with blocking constraints. Expert Syst Appl 2022;201:117256.
[7] Qian F. Smart process manufacturing toward carbon neutrality: digital transformation in process manufacturing for achieving the goals of carbon peak and carbon neutrality. Engineering 2023;27(8):1–2.
[8] Wang R, Jiang L, Wang YD, Roskilly AP. Energy saving technologies and

masstherm al network optimization for decarbonized iron and steel industry: a review. J Clean Prod 2020;274:122997.
[9] He K, Wang L. A review of energy use and energy-efficient technologies for the iron and steel industry. Renew Sustain Energy Rev 2017;70:1022–39.
[10] Hernandez AG, Paoli L, Cullen JM. How resource-efficient is the global steel industry? Resour Conserv Recycling 2018;133:132-45.
[11] Gao Z, Geng Y, Wu R, Chen W, Wu F, Tian X. Analysis of energy-related CO_2 emissions in China's pharmaceutical industry and its driving forces. J Clean Prod 2019;223:94–108.
[12] Ribas I, Companys R, Tort-Martorell X. Efficient heuristics for the parallel blocking flow shop scheduling problem. Expert Syst Appl 2017;74:41–54.
[13] Shao Z, Shao W, Pi D. Effective heuristics and metaheuristics for the distributed fuzzy blocking flow-shop scheduling problem. Swarm Evol Comput 2020;59:100747.
[14] Riedmiller S, Riedmiller M. A neural reinforcement learning approach to learn local dispatching policies in production scheduling. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence; 1999 Jul 31 – Aug 6; Stockholm, Sweden. San Francisco: Morgan Kaufmann Publishers Inc.; 1999. p. 764–71.
[15] Zhang G, Xing K, Cao F. Discrete differential evolution algorithm for distributed blocking flowshop scheduling with makespan criterion. Eng Appl Artif Intell 2018;76:96–107.
[16] Chen S, Pan QK, Gao L, Sang HY. A population-based iterated greedy algorithm to minimize total flowtime for the distributed blocking flowshop scheduling problem. Eng Appl Artif Intell 2021;104:104375.
[17] Shao Z, Shao W, Pi D. LS-HH: a learning-based selection hyper-heuristic for distributed heterogeneous hybrid blocking flow-shop scheduling. IEEE Trans Emerg Top Comput Intell 2023;7(1):111–27.
[18] Zinn J, Ockier P, Vogel-Heuser B. Deep Q-learning for the control of specialpurpose automated production systems. In: Proceedings of the 2020 IEEE 16th International Conference on Automation Science and Engineering (CASE); 2020 Aug 20-21; Hong Kong, China. Piscataway: IEEE; 2020. p. 1434–40.
[19] Ren J, Ye C, Li Y. A new solution to distributed permutation flow shop scheduling problem based on NASH Q-Learning. Adv Prod Eng Manag 2021; 16(3):269–84.
[20] Yang S, Wang J, Xu Z. Real-time scheduling for distributed permutation flowshops with dynamic job arrivals using deep reinforcement learning. Adv Eng Inform 2022;54:101776.
[21] Chen JF, Wang L, Peng ZP. A collaborative optimization algorithm for energyefficient multi-objective distributed no-idle flow-shop scheduling. Swarm Evol Comput 2019;50:100557.
[22] Zhang X, Liu X, Cichon A, Królczyk G, Li Z. Scheduling of energy-efficient distributed blocking flowshop using pareto-based estimation of distribution algorithm. Expert Syst Appl 2022;200:116910.
[23] Mou J, Duan P, Gao L, Liu X, Li J. An effective hybrid collaborative algorithm for energy-efficient distributed permutation flow-shop inverse scheduling. Future Gener Comput Syst 2022;128:521–37.
[24] Zhao F, Di S, Wang L. A hyperheuristic with Q-learning for the multiobjective energy-efficient distributed blocking flow shop scheduling problem. IEEE Trans Cybern 2022;53(5):3337–50.
[25] Shao Z, Shao W, Chen J, Pi D. MQL-MM: a meta-Q-learning-based multiobjective metaheuristic for energy-efficient distributed fuzzy hybrid blocking flow-shop scheduling problem. IEEE Trans Evol Comput 2024:1–1.
[26] Zhao F, Zhou G, Xu T, Zhu N, Jonrinaldi. A knowledge-driven cooperative scatter search algorithm with reinforcement learning for the distributed blocking flow shop scheduling problem. Expert Syst Appl 2023;230:120571.
[27] Bao H, Pan Q, Ruiz R, Gao L. A collaborative iterated greedy algorithm with reinforcement learning for energy-aware distributed blocking flow-shop scheduling. Swarm Evolut Comput 2023;83:101399.
[28] Liu C, Xu X, Hu D. Multiobjective reinforcement learning: a comprehensive overview. IEEE Trans Syst Man Cybern 2014;45(3):385–98.
[29] Gábor Z, Kalmár Z, Szepesvári C. Multi-criteria reinforcement learning. In: Proceedings of the Fifteenth International Conference on Machine Learning; 1998 Jul 24-27; Madison, WI, USA. San Francisco: Morgan Kaufmann Publishers; 1998. p. 197–205.
[30] Feinberg EA, Shwartz A. Constrained Markov decision models with weighted discounted rewards. Math Oper Res 1995;20(2):302–20.
[31] Russell SJ, Zimdars A. Q-decomposition for reinforcement learning agents. In: Proceedings of the Twentieth International Conference on International Conference on Machine Learning; 2003 Aug 21 – 24; Washington, DC, USA. Palo Alto: AAAI Press; 2003. p. 656–63.
[32] Barrett L, Narayanan S. Learning all optimal policies with multiple criteria. In: Proceedings of the 25th international conference on Machine learning; 2008 Jul 5-9; Helsinki, Finland. New York: ACM; 2008. p. 41–7.
[33] Van Moffaert K, Nowé A. Multi-objective reinforcement learning using sets of pareto dominating policies. J Mach Learn Res 2014;15(1):3483–512.
[34] Mossalam H, Assael YM, Roijers DM, Whiteson S. Multi-objective deep reinforcement learning. 2016. arXiv:1610.02707.
[35] Abels A, Roijers D, Lenaerts T, Nowé A, Steckelmacher D. Dynamic weights in multi-objective deep reinforcement learning. 2018. arXiv:1809.07803.
[36] Nguyen TT, Nguyen ND, Vamplew P, Nahavandi S, Dazeley R, Lim CP. A multiobjective deep reinforcement learning framework. Eng Appl Artif Intell 2020;96:103915.
[37] Siddique U, Weng P, Zimmer M. Learning fair policies in multi-objective (deep) reinforcement learning with average and discounted rewards. In: Proceedings of the 37th International Conference on Machine Learning; 2020 Jul 13-18; Vienna, Austria. Brookline: JMLR; 2020. p. 8905–15.
[38] He Z, Tran KP, Thomassey S, Zeng X, Xu J, Yi C. Multi-objective optimization of the textile manufacturing process using deep-Q-network based multi-agent reinforcement learning. J Manuf Syst 2022;62:939–49.
[39] Yang R, Sun X, Narasimhan K. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems; 2019 Dec 8–14; Vancouver, BC, Canada. New York: Curran Associates; 2019. p. 14636–47.
[40] Luo S, Zhang L, Fan Y. Dynamic multi-objective scheduling for flexible job shop by deep reinforcement learning. Comput Ind Eng 2021;159:107489.
[41] Lowe R, Wu Y, Tamar A, Harb J, Pieter P, Mordatch I. Multi-agent actor-critic for mixed cooperative-competitive environments. In: Proceedings of the 31st International Conference on Neural Information Processing Systems; 2017 Dec 4–9; LongBeach, CA, USA. New York: Curran Associates; 2017. p. 6382–93.
[42] Yu C, Velu A, Vinitsky E, Gao J, Wang Y, Bayen A, et al. The surprising effectiveness of PPO in cooperative multi-agent games. In: Proceedings of the 36th International Conference on Neural Information Processing Systems; 2022 Nov 28–Dec 9; New Orleans, LA, USA. New York: Curran Associates; 2024. p. 24611–24.
[43] Engstrom L, Ilyas A, Santurkar S, Tsipras D, Janoos F, Rudolph L, et al. Implementation matters in deep RL: a case study on PPO and TROP. In: Proceedings of 8th International Conference on Learning Representations, 2020 April 26-30; Addis Ababa, Ethiopia. Appleton: ICLR; 2020. p. 12883–98.
[44] Sun X, Shen W, Vogel-Heuser B. A hybrid genetic algorithm for distributed hybrid blocking flowshop scheduling problem. J Manuf Syst 2023;71:390-405.
[45] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. 2017. arXiv:1707.06347.
[46] Zhao L, Fan J, Zhang C, Shen W, Zhuang J. A DRL-based reactive scheduling policy for flexible job shops with random job arrivals. IEEE Trans Autom Sci Eng 2024;21(3):2912–23.
[47] Zhao F, Zhang H, Wang L, Xu T, Zhu N, Jonrinaldi J. A multi-objective discrete differential evolution algorithm for energy-efficient distributed blocking flow shop scheduling problem. Int J Prod Res 2023;62 (12):4226–44.
[48] Zhao F, Zhang H, Wang L. A pareto-based discrete jaya algorithm for multiobjective carbon-efficient distributed blocking flow shop scheduling problem. IEEE Trans Industr Inform 2023;19(8): 8588–99.
[49] Alegre LN, Bazzan ALC, Roijers DM, Nowé A, da Silva BC. Sample-efficient multi-objective learning via generalized policy improvement prioritization. 2023. arXiv:2301.07784.