

DOI: 10.3901/JME.2023.12.078

基于深度强化学习的工序交互式智能体 Job shop 调度方法^{*}

陈睿奇¹ 黎雯馨^{1,2} 王传洋¹ 杨宏兵¹

(1. 苏州大学机电工程学院 苏州 215100;

2. 上海大学管理学院 上海 200230)

摘要: 针对作业车间调度问题(Job shop scheduling problem, JSSP)因 NP-难属性难以快速获得优质解, 以及生产场景随机扰动所导致的频繁重调度等求解难题, 基于深度强化学习提出一种新颖的交互式工序智能体(Interactive operation agent, IOA)调度模型框架。在分析工序间工艺路线和加工设备约束关系的基础上, 将 Job shop 的加工工序构建为工序智能体, 设计工序智能体间的交互机制, 智能体依据彼此关系进行特征交互并更新自身的特征向量, 并基于工序特征和最早加工时间设计拟合动作值函数的深度神经网络, 调度模型根据系统状态和工序智能体特征即可生成调度策略。采用 Double DQN 算法训练 IOA 调度模型, 引入经验回放机制消除序列训练样本间的相关性, 训练好的模型可以快速生成高质量的调度方案, 并在机器发生故障时能够有效执行重调度策略。试验结果表明所提出的 IOA 调度方法优于贪婪算法和启发式调度规则, 且具有良好鲁棒性和泛化能力。

关键词: Job shop 调度; 深度强化学习; 工序智能体; 机器故障; double DQN 算法

中图分类号: TH166

Interactive Operation Agent Scheduling Method for Job Shop Based on Deep Reinforcement Learning

CHEN Ruiqi¹ LI Wenxin^{1,2} WANG Chuanyang¹ YANG Hongbing¹

(1. School of Mechanical and Electric Engineering, Soochow University, Suzhou 215000;

2. School of Management, Shanghai University, Shanghai 200230)

Abstract: Job shop scheduling problem(JSSP) is difficult to obtain high-quality solution quickly due to NP hard attribute, and rescheduling occurs frequently due to the random disturbances of production scenarios. Based on deep reinforcement learning, a novel interactive operation agent(IOA) scheduling model framework is proposed. Through analysis of the constraint relationship between process route and processing equipment among operations, the processing processes in job shop are constructed as operation agents. The interaction mechanism between operation agents is designed, and each agent can interact with each other and update its own feature vector according to their relationship. Further, a deep neural network is constructed based on the operation characteristics and the earliest processing time to fit the action value function. As a result, the scheduling model can generate the scheduling strategy according to the system state and the characteristics of each operation agent. Double DQN algorithm is used to train IOA scheduling model, and the introduction of empirical playback mechanism effectively breaks the correlation between sequence training samples. The trained model can quickly generate high-quality scheduling scheme, and effectively execute rescheduling production strategy in case of machine failure. Experimental results show that the proposed IOA scheduling method is superior to greedy algorithm and heuristic scheduling rules, and has good robustness and generalization ability.

Key words: Job shop scheduling; deep reinforcement learning; operation agents; machine failure; double DQN

^{*} 国家自然科学基金资助项目(52075354)。20220702 收到初稿, 20230206 收到修改稿

0 前言

伴随着社会经济的发展和智能制造技术的推广, 产品的客户个性化定制水平正朝着广度和深度两个维度呈加速发展态势, 与之对应的多品种小批量生产也被越来越多的企业所采用, Job shop 作为一种典型的多品种小批量生产方式, 广泛存在于各类离散制造行业。由于复杂的生产任务和制造流程, 作业车间调度问题 (Job shop scheduling problem, JSSP) 已被证明为 NP 难组合优化问题, 一直以来倍受学术界的关注^[1-2]。JSSP 可分为静态调度和动态调度两类, 静态调度通常假定作业环境是确定且不变的, 而动态调度则需要考虑调度环境和任务不可预测的扰动对生产的影响。求解 JSSP 问题的方法主要分为精确算法和近似算法两大类。其中精确求解算法主要为数学规划方法等; 近似算法主要包括调度规则、人工智能^[3-4]和元启发式算法^[5-6]等。大部分的精确算法只适合求解小规模问题, 当问题规模扩大时, 它们则无法在有效的时间范围内获得高质量的解。20 世纪八十年代以来, 为提高求解速度, 研究者们对作业车间生产调度方法研究的重心由精确算法转移到了近似求解算法。WERNER 等^[7]结合插入算法和定向搜索算法, 考虑不同的插入顺序, 获得了较好的排产结果。ADAMS 等^[8]运用移动瓶颈法, 将作业车间调度问题分解为多个单机调度子问题, 然后分别单独求解。ZHANG 等^[9]针对多目标优化的柔性车间调度问题提出了一种改进的多种群遗传算法, 算法着重考虑了最短加工时间和机器的平衡利用。赵诗奎^[10]提出了一种融合问题领域知识的路径重连和可持续回溯禁忌搜索混合算法, 对作业车间调度的最大完工时间问题进行了研究。同样以最小化最大完工时间为优化目标, 孟磊磊等^[11]引入变邻域搜索算法提出一种混合蛙跳算法用于解决分布式柔性作业车间调度问题。MUHAMMAD 等^[12]综述了近年来基于遗传算法求解柔性作业车间调度问题的研究进展。为使得调度算法能够适应现实生产场景中的动态变化, SHAKHLEVICH 等^[13]设计了一种基于析取图的自适应算法, 他们在调度的学习和测试阶段使用了冲突消除策略, 从而使算法能够自适应多种实际约束。LEE 等^[14]基于模糊规则提出了一种用于动态调度的自适应算法。

传感器及智能设备的广泛应用, 使得制造过程中越来越多的状态信息和数据可以被捕捉和追溯, 生产过程变得更加透明, 为机器学习算法有效应用

于生产调度领域提供了丰富的数据支撑。强化学习凭借其独特的目标导向式无监督学习能力引起了广泛的关注, 部分学者尝试将强化学习应用到生产调度领域并取得了一定的研究进展^[15-16]。然而面对海量高维生产数据时, 强化学习即暴露了其采样效率低和状态维数灾的弱点。融合了深度学习的感知能力和强化学习的决策能力, MNIH 等^[17]首次提出了一种影响深远的深度强化学习算法——深度 Q 网络, 它能够有效处理高维的输入状态。依赖于深度神经网络强大的非线性拟合性能, 深度强化学习深度强化学习 (Deep reinforcement learning, DRL) 方法可以将状态值或状态-动作值直接输入深度神经网络中, 将其映射为输出的价值或具体的动作, 使得智能体可以展现出超越人类顶级专家的智能, 其中最著名的 DRL 应用当属 SILVER 等^[18]设计的围棋智能体 AlphaGo, 该模型将蒙特卡洛树搜索融入到 DRL 中, 以实现围棋比赛远期的推断。该方法的改进版本 AlphaGo Zero^[19]甚至可以不借助任何人类的专家经验和领域知识, 就能达到更强的围棋博弈水平。在作业车间调度应用方面, LIU 等^[20]提出了一种求解 JSSP 的演员评论家 (Actor-Critic) 模型, 演员和评论家的两套神经网络模型均为卷积神经网络和全连接网络的组合, 在训练时采用了多智能体方案以及 DDPG 算法。肖鹏飞等^[21]以启发式算法或分配规则作为调度决策候选行为, 设计了一种基于时序差分法的深度强化学习调度算法。王凌等^[22]设计了一种新的编码网络对车间调度问题进行建模, 提出了一种基于深度强化学习与迭代贪婪算法的调度框架。深度强化学习算法可以迅捷的响应动态事件, 被认为是解决动态调度问题最理想的方法之一。PALOMBARINI 等^[23]将实时调度的任务以闭环控制问题的方式建模, 为了实时响应紧急订单插入、机器故障及原材料延误到达三种意外的事件和扰动, 他们训练了一个深度 Q 网络来选择智能体的动作以修复调度计划。HAN 等^[24]训练了一个改进的 Dueling Double DQN 网络模型来求解自适应作业车间调度问题, 在静态调度中, 该算法性能远远优于遗传算法等启发式算法, 而在加工时间不确定的动态调度中, 其效果仍优于排产规则。为求解考虑新工件插入的动态柔性作业车间调度问题, LUO^[25]使用 DQN 算法将大量专家经验融于智能体对调度环境的观察和执行的动作之中。ZHANG 等^[26]提出了一种基于自组织机制和自学习策略的多智能体 DRL 模型, 该模型可有效应对新订单插入以及不可预知的机器故障。LIU 等^[27]将动态柔性作业车间调

度拆解为顺序决策和路径决策两个阶段,并基于 Double DQN 算法构建多智能体,使用代理奖励函数设计来提升训练效率,从而实现了在新订单持续到达状态下的实时调度。

上述研究大都采用将加工机器建模为智能体的建模方案,此类建模方案可以借鉴较多的专家经验,但是却往往很难应对机器故障这类随机扰动的影响。因为若以机器作为智能体,则机器就是调度问题建模的核心,需要从机器的视角来观察调度状态(如机器利用率等),然后做出调度动作(如选择启发式调度规则等)。一旦机器发生故障,这类方法往往也会失效,或是造成调度策略的巨大扰动。与传统的机器智能体建模不同,本文以 job shop 生产车间为应用场景,提出一种基于交互式工序智能体(Interactive operation agent, IOA)的调度框架,并采用离线学习的深度强化学习算法 Double DQN,对调度模型进行训练,充分利用深度强化学习端到端的学习优势,能更好的利用问题的原始信息,无需依赖于专家经验。该方法可以有效应对加工机器故障类意外事件对排产方案的扰动,拥有良好的柔性和鲁棒性。此外,IOA 调度模型拥有良好的泛化能力,单个经过训练的模型即可求解任意规模的 Job shop 调度问题,而不必为特定规模的调度问题储备模型或是重新训练。

1 交互式工序智能体调度模型

1.1 Job shop 调度问题描述

Job shop 调度本质上就是分派不同工件 J_i 在机

器 M_h 上的加工顺序问题, $i \in \{1, 2, \dots, n\}$, $h \in \{1, 2, \dots, m\}$, 工件 J_i 包含一系列具有固定加工顺序加工的工序 $o_{i,j}$, $j \in \{1, 2, \dots, m\}$, 工序 $o_{i,j}$ 的加工时间为 $p_{i,j}$ 。本文的调度目标是最大化所有工件的最大完工时间 C_{\max} 。JSSP 需要满足如下四个约束: ① 一个工件一次只能在一台机器上加工; ② 一台机器一次只能加工一个工件; ③ 一个工件的某道工序只有在该工序的先前工序都完成加工后才能加工; ④ 一道工序一旦开始加工就不能中断,直至加工完成。除此之外,本文的动态 Job shop 调度问题还需考虑机器随机故障的发生,假设用于工件加工的 m 台机器均有可能发生故障,任何一台机器发生故障的概率都服从参数为 λ 的指数分布

$$f(x; \lambda) = \begin{cases} \lambda \exp(-\lambda x) & x \geq 0 \\ 0 & x < 0 \end{cases} \quad (1)$$

式中, x 为机器连续加工的时间。假设机器故障的修复时间服从数学期望为 μ , 方差为 σ^2 的正态分布 $N(\mu, \sigma^2)$, 且不同机器间发生故障的概率和故障恢复时间都相互独立。在机器发生故障后生产系统需要执行重调度,修复后的排产计划仍需追求最小化最大完工时间 C_{\max} 。

1.2 IOA 调度模型运行框架

在基于交互式工序智能体的调度模型中,每一道加工工序都为独立的智能体,智能体以交互的方式观察当前的生产环境状态,并以此确定指派相应工序对缩短 C_{\max} 能够做出多大贡献,其运行框架如图 1 所示。在每一时间步 t , 模型依照该时刻的调度

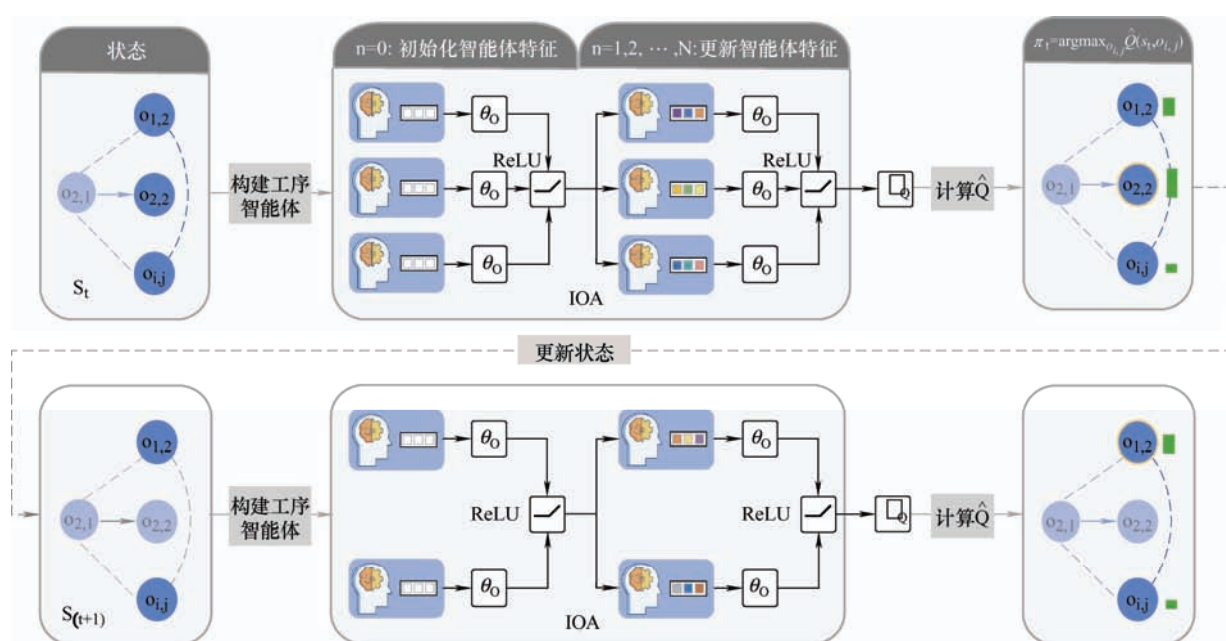


图1 IOA 调度模型运行框架

系统状态构建工件工序智能体，所有工序智能体在 n 次迭代中相互作用并更新各自的特征向量 $\mu_{i,j}^n$ 。模型通过综合分析各个工序智能体的特征，评估工序的动作价值 \hat{Q} ，动作价值最大的工序将被指派，并被添加至解序列 $\pi = \{\pi_1, \pi_2, \dots, \pi_T\}$ 中。随后，调

度系统将转移至下一调度状态 s_{t+1} 。

1.3 交互式工序智能体构建

工序智能体依据彼此间的相互关系经过交互后形成各自的特征向量 $\mu_{i,j}^n$ ，在 Job shop 生产系统中，两道工序间存在的直接关系有且仅有两种(图 2)。

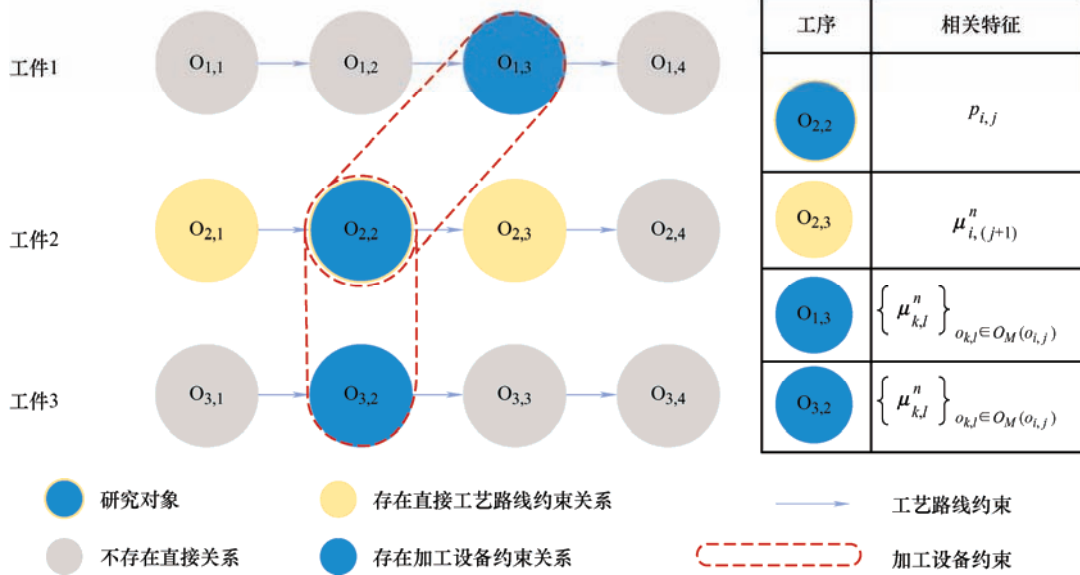


图 2 工序间的直接关系及关键工序所对应的特征

(1) 工艺路线约束关系，即一道工序是另一道工序紧前工序或紧后工序。

(2) 加工设备约束关系，即两道工序需由同一台机器加工，当其中一道工序被加工时另一道必须等待。

除了直接关系，两道工序间也可能存在间接关系，如图 2 中的工序 $o_{2,2}$ 和工序 $o_{1,4}$ 存在间接关系，在使用相同机器加工的工序 $o_{2,2}$ 和 $o_{1,3}$ 间若指派工序 $o_{2,2}$ 先加工，则 $o_{1,4}$ 的最早开始时间将进一步被延后。间接关系形成的根本原因是直接关系的传导，如上述的间接关系是由工序 $o_{2,2}$ 与 $o_{1,3}$ 间的加工设备约束关系和工序 $o_{1,3}$ 与 $o_{1,4}$ 间的工艺路线约束关系构成的。IOA 调度模型旨在通过构建如式(2)所示的映射关系，使工序智能体和与其存在直接关系的其他智能体进行交互，从而共同更新彼此的特征向量 $\mu_{i,j}$ 。经过多次迭代后，工序智能体将间接地与更远距离的其他智能体实现交互。

$$\mu_{i,j}^{(n+1)} \leftarrow F(p_{i,j}, \mu_{i,(j+1)}^n, \{\mu_{k,l}^n\}_{o_{k,l} \in O_M(o_{i,j})}; \theta_o) \quad (2)$$

式中， n 是迭代次数， $p_{i,j}$ 是工序 $o_{i,j}$ 的加工时间， $O_M(o_{i,j})$ 是所有与 $o_{i,j}$ 使用同一机器加工的工序集合， θ_o 是工序智能体的神经网络参数。在映射 F 中影响工序智能体特征的因素可分为静态的固有特征和动态的结构性特征两类。其中静态特征是工序的

加工时间 $p_{i,j}$ ，它反映工序的内在属性，是影响调度结果最重要的因素之一，它不随调度状态 s_t 的改变而变动，在特征迭代的过程中也不会发生变化。 $o_{i,j}$ 的后道工序以及 $o_{i,j}$ 使用同一机器加工的工序所对应的智能体特征分别为 $\mu_{i,(j+1)}^n$ 和 $\{\mu_{k,l}^n\}_{o_{k,l} \in O_M(o_{i,j})}$ ，它们属于工序的结构性特征，反映当前的调度状态 s_t ，随时间步 t 的改变而变动(例如工序 $o_{k,l}$ 被指派后，它将从 $O_M(o_{i,j})$ 中被剔除，不再影响特征 $\mu_{i,j}$ 的形成)。同时，在每一轮特征的迭代更新后，结构性特征也将随之发生改变。在分析某一道工序的特征时，应着重考虑其后续的工序，而非其先前的工序，因为先前的工序不会对当前工序的调度造成影响，且模型运行实质上是马尔科夫决策过程，该过程应体现无后效性，故在构建映射 F 时前道工序的特征 $\mu_{i,(j-1)}$ 不应被纳入考虑。图 2 以研究对象 $o_{2,2}$ 为例，展示了上述特征所对应的工序。

在多次迭代后，工序智能体的特征可沿着工序间的关系网络传播和扩散，从而使得远距离的工序智能体之间也能够相互关联。当迭代次数 $N = 2$ 时， $\mu_{i,j}$ 可包含与其存在直接关系的工序特征；当迭代次数 $N \geq 2$ 后，任意两道存在间接关系的工序特征都会相互关联。为实现式(2)中 F 的映射关系，本文

将 F 参数化为

$$\mu_{i,j}^{(n+1)} \leftarrow \text{Norm}(\text{ReLU}(\theta_1 p_{i,j} + \theta_2 \mu_{i,(j+1)}^n + \theta_3 \frac{1}{N_M} \sum_{o_{k,l} \in O_M(o_{i,j})} \mu_{k,l}^n)) \quad (3)$$

式中, 工序智能体特征向量 $\mu_{i,j}$ 的维数为 d_{model} , $\theta_o = \{\theta_1, \theta_2, \theta_3\}$ 为工序智能体的可学习参数, $\theta_1 \in \mathbb{R}^{1 \times d_{\text{model}}}$, $\theta_2, \theta_3 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$, N_m 为 $O_M(o_{i,j})$ 中的工序数量, ReLU 为线性整流函数, Norm 为归一化函数。

在 IOA 模型中, 所有工序智能体共用一套神经网络参数 θ_o , 但是各自的特征向量不同。为使模型具有良好的泛化性, 以适应 $O_M(o_{i,j})$ 中工序数量的变化, 基于 DAI 等^[28]的方法, 本文将总体特征以特征加和的方式聚合。聚合后的特征将由 ReLU 函数激活, 以此发挥神经网络的非线性拟合能力。在初次迭代 $n=0$ 时, $\mu_{i,j}^0$ 将被初始化为 $\bar{\theta}$ 而非随机向量, 保证了每次迭代的起点都相同, 初次迭代中唯一影响 $\mu_{i,j}^1$ 特征形成的只有工序 $o_{i,j}$ 的加工时间 $p_{i,j}$, 而式(3)中其余特征向量均为 $\bar{\theta}$ 。在后续的迭代中, $\mu_{i,j}^{(n+1)}$ 的形成只与前一时刻其他工序智能体的特征相关, 与自身前一时刻的特征 $\mu_{i,j}^n$ 无关, 由于在每一次迭代中, 工序智能体都会向其特征中加入工序加工时间信息, 切断 $\mu_{i,j}^n$ 和 $\mu_{i,j}^{(n+1)}$ 的联系可以确保 $\mu_{i,j}^{(n+1)}$ 中不会存在冗余的工序加工时间信息。

1.4 动作值函数拟合

只有在调度结束后, 才能依据最终调度方案求得最大完工时间 C_{\max} 。在调度的过程中, 最大完工时间无法被探知。所以, 如果直接以工件的最大完工时间作为值函数, 模型就只能进行回合更新, 而无法实现单步更新。由于机器空闲时间与最大完工时间 C_{\max} 存在如下关系

$$\begin{aligned} \frac{1}{m} \sum_{t=1}^T \text{idle}_t &= \frac{1}{m} \sum_{j=1}^m \text{idle}_m = \\ \frac{1}{m} \sum_{j=1}^m (C_{\max} - \sum_{i=1}^n p_{i,j}) &= \\ C_{\max} - \frac{1}{m} \sum_{j=1}^m \sum_{i=1}^n p_{i,j} \end{aligned} \quad (4)$$

式中, $\frac{1}{m} \sum_{j=1}^m \sum_{i=1}^n p_{i,j}$ 是定值, 在整个调度过程中累积的机器平均空闲时间 $\frac{1}{m} \sum_{t=1}^T \text{idle}_t$ 越大, 最大完工时间 C_{\max} 就越大, 因此, 在某一状态 s_t 下指派工序 $o_{i,j}$ 后,

以所有机器的平均空闲时间(自 t 时刻之后)的相反数衡量动作值函数的 Q 值, 以提高模型的训练效率。本文使用参数为 θ_Q 的神经网络来估计近似的 Q 值: $Q(s_t, o_{i,j}) \approx \hat{Q}(s_t, o_{i,j}, \theta_Q)$, 通过对参数 θ_Q 的不断更新, 可使 \hat{Q} 更接近 Q , 从而使模型更准确地判断调度某道工序对调度计划整体价值的影响。拟合 \hat{Q} 的公式为

$$\hat{Q}(s_t, o_{i,j}; \theta_Q) = \theta_4 \left(\text{ReLU} \left(\left[\theta_5 \sum_{o_{k,l} \in O} \mu_{k,l}^N, \theta_6 \mu_{i,j}^N, \theta_7 ES_{i,j} \right] \right) \right) \quad (5)$$

式中, $\theta_Q = \{\theta_4, \theta_5, \theta_6, \theta_7\}$ 为 \hat{Q} 网络的权重参数, $\theta_4 \in \mathbb{R}^{1 \times 3d_{\text{model}}}$, $\theta_5, \theta_6 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$, $\theta_7 \in \mathbb{R}^{1 \times d_{\text{model}}}$, $ES_{i,j}$ 是工序 $o_{i,j}$ 的最早开始时间, $[\cdot, \cdot, \cdot]$ 是向量的水平连接操作符。模型由三方面特征共同确定在状态 s_t 下选择工件 $o_{i,j}$ 的动作价值。首先所有工序智能体的特征将被求和为 $\sum_{o_{k,l} \in O} \mu_{k,l}^N$, 用以表征当前状态下所有工序智能体的总体聚合特征, 是 \hat{Q} 网络分析当前整体调度状态的依据。 $\mu_{i,j}^N$ 是候选工序 $o_{i,j}$ 的特征向量, 包含 $o_{i,j}$ 对应的工序智能体与其他智能体交互的信息, 是 \hat{Q} 网络衡量动作 $\pi_t = o_{i,j}$ 潜在价值的途径。工序 $o_{i,j}$ 的最早开始时间 $ES_{i,j}$ 直接影响状态从 s_t 转移至 s_{t+1} 时机器的空闲时间 idle_t , 选择在状态 s_t 下能够更早开始加工的工件, 可使 idle_t 更小, 从而获得更多的即时回报。 $ES_{i,j}$ 是 \hat{Q} 网络衡量动作 $\pi_t = o_{i,j}$ 直接价值的途径, 它将被参数 θ_7 在经过线性变换后投射至高维向量, 使其在被 ReLU 函数激活后包含更多的非线性特征。

仅考虑即时回报影响因素 $ES_{i,j}$ 的模型是短视的, 它无法在长期的决策过程中取得稳定的回报; 仅考虑整体因素 $\sum_{o_{k,l} \in O} \mu_{k,l}^N$ 的模型是无效率的, 它在训练时无法做出那些更可能带来高回报的决策动作, 而无法有效的探索高质量的策略, 使得模型不易收敛。所以, 本文设计的 \hat{Q} 网络同时观察了 $\sum_{o_{k,l} \in O} \mu_{k,l}^N$, $\mu_{i,j}^N$ 和 $ES_{i,j}$ 三方面特征, 从而更全面地分析当前的调度状态。拼接后的整体特征向量先经由 ReLU 函数进行非线性化处理, 再通过权重参数 θ_4 映射为输出的 \hat{Q} 值。经试验发现, 若使用较大的模型规模(如 $d_{\text{model}} \geq 32$), 则可将 θ_4 扩展为多层神经元数量逐层递减的隐藏层, 最后叠加单神经元的

输出层, 这样可使 \hat{Q} 网络在调度单个 JSSP 实例时效果更佳。但若隐藏层太深, 则模型容易过拟合, 从而使得模型在调度其他 JSSP 实例时性能下降, 可在隐藏层间添加 dropout 层可以缓解这一现象。

1.5 机器故障的重调度策略

在仿真试验中, 系统以离散事件仿真形式驱动 Job shop 生产环境。每一道工序的加工事件开始时, 系统以指数分布生成机器的正常运行时间, 若该时间长于工序加工时间, 则不触发机器故障事件, 否则在正常运行时间结束后触发机器故障并进行重调度。以正态分布随机生成故障机器的修复时间, 在修复结束后再次触发重调度。

IOA 调度模型主要依据工序间相互关系形成的特征进行决策, 仅有 \hat{Q} 网络的部分工序最早开始时间 $ES_{i,j}$ 易受到机器故障事件的影响(同时也能表征该影响), 所以动态事件的发生对系统整体的扰动相对较小, 保证调度模型具有较好的鲁棒性。IOA 调度模型以简单的方式处理机器故障事件, 仅需要在故障发生时更新所有受故障机器影响的候选工序的

最早开始时间 $ES_{i,j}$, 然后进行重调度即可。对于受到故障机器影响的工序 $o_{i,j}$, 其修正后的最早开始时间 $ES'_{i,j}$ 为

$$ES'_{i,j} = \max(End_{o_{i,j-1}}, End_{machine(o_{i,j})} + R_{machine(o_{i,j})}) \quad (6)$$

式中, $End_{o_{i,j-1}}$ 是工序 $o_{i,j}$ 前道工序的加工结束时间(如果 $j=1$, $End_{o_{i,j-1}}=0$), $End_{machine(o_{i,j})}$ 为处理工序 $o_{i,j}$ 的机器在原调度计划中的加工结束时间,

$R_{machine(o_{i,j})}$ 是预期的设备修理时间。机器故障发生时 IOA 重调度策略如图 3 所示。在每一时间步 t , 智能体特征 $\mu_{i,j}^0$ 先被初始化为 $\bar{\theta}$, 还未被指派的工序智能体经 N 次交互后形成当前情况下的特征向量(第 3~5 行)。对于所有处于候选工序集合 O_C 中的工序, 若其所用机器正常运转, 则该工序最早可在前道工序加工完成并且所用机器空闲时进行加工; 否则依照式(6)修正最早开始时间, 随后即可根据式(5)估计该工序的 Q 值(第 7~14 行)。 Q 值最大的工序将被安排加工, 随后系统转移到新状态(第 15~17 行)。

重调度策略(机器发生故障IOA动态调度)

输入: 当前的初始调度状态 s_1 , 剩余的工序数量 T , 工序特征更新迭代次数 I

输出: 调度策略 π

```

1: 初始化  $\pi$  为空
2: for  $t=1 \rightarrow T$  do
3: 将  $\mu_{i,j}^0$  初始化为 0 向量
4:   for  $n=0 \rightarrow N$  do
5:     对于所有未被指派的工序智能体  $o_{i,j} \in O_u$ ,
       计算  $\mu_{i,j}^{n+1} \leftarrow F\left(p_{i,j}, \mu_{i,(j+1)}^n, \left\{\mu_{k,l}^n\right\}_{o_{k,l} \in O_M(o_{i,j})}\right); \Theta_O$ 
6:   end for
7:   for  $o_{i,j} \in$  候选的工序集合  $O_C$  do
8:     if  $o_{i,j}$  所有机器损坏 then
9:        $ES'_{i,j} = \max(End_{o_{i,j-1}}, End_{machine(o_{i,j})} + R_{machine(o_{i,j})})$ 
10:    else
11:       $ES'_{i,j} = \max(End_{o_{i,j-1}}, End_{machine(o_{i,j})})$ 
12:    end if
13:    计算  $\hat{Q}(s_t, o_{i,j}; \Theta_Q)$ 
14:  end for
15:  $\pi_t = \arg\max_{o_{i,j} \in O_C} \hat{Q}(s_t, o_{i,j}; \Theta_Q)$ 
16: 将  $\pi_t$  添加至  $\pi$  中
17: 执行调度策略  $\pi_t$ , 更新调度状态  $s_{t+1}$ 
18: end for
19: return  $\pi$ 

```

图 3 机器故障时 IOA 重调度策略

2 IOA 调度模型训练

本文使用带有经验回放机制的 Double DQN 算法训练 IOA 调度模型中的权重参数 $\theta = \{\theta_o, \theta_Q\} = \{\theta_1, \theta_2, \dots, \theta_7\}$, 训练流程如图 4 所示。基于 DQN 算法原理, 训练的过程中使用了两套结构完全一致的神经网络, 一套主要应用在探索和决策的过程中, 通过分析当前的调度状态估计各调度动作的价值, 以此选择动作并获得训练的样本, 下文称为在线 Q 网络; 另一套主要应用在模型学习的

过程中, 用于估计动作的目标价值, 以此来指导模型的学习, 下文称为目标 \hat{Q} 网络。由于用于更新在线 Q 网络参数的目标 \hat{Q} 值在计算时也要依赖在线 Q 网络来估计下一状态的 \hat{Q} 值, 两套独立网络的设计可以避免这种相互循环依赖的关系。在训练开始时, 在线 Q 网络的参数将被随机初始化, 然后复制给目标 \hat{Q} 网络。在随后训练的过程中, 在线 Q 网络的参数每一步都将更新一次, 而目标 \hat{Q} 网络参数的更新则采用延时更新策略, 每隔 T_{update} 步更新一次, 更新方式为直接复制在线 Q 网络的参数。

算法: 使用 Double DQN 算法训练 IOA 调度模型

输入: 经验缓存区大小 N_{memory} , 训练回合数 E , 工序数量 T , 经验回放批量大小 B

输出: 神经网络参数 θ

```

1: 随机初始化在线  $Q$  网络参数  $\theta$ 
2: 初始化目标  $\hat{Q}$  网络参数为  $\theta' = \theta$ 
3: 初始化容量为  $N_{\text{memory}}$  的经验缓存区  $M$ 
4: for  $episode = 1 \rightarrow E$  do
5:   生成调度实例  $S$ , 获取初始调度状态  $s_1$ 
6:   for  $t = 1 \rightarrow T$  do
7:     计算所有未被指派的工序智能体的特征表示  $\mu_{i,j}^I$ 
8:     选择动作  $a_t = \begin{cases} \text{随机候选工序 } o_{i,j} \in O_C, \text{ w.p.c} \\ \arg\max_a Q(s_t, a_t; \theta) & \text{其他} \end{cases}$ 
9:     执行调度动作  $a_t$ , 更新调度状态  $s_{t+1}$ , 计算机器空闲时间  $idle_t$ 
10:    储存调度经验  $(s_t, a_t, idle_t, s_{t+1})$  至经验缓存区  $M$  中
11:    从经验缓存区  $M$  中随机抽取一批批量大小为  $B$  的回放经验  $(s_t, a_t, idle_t, s_{t+1})$ 
12:    设置  $y_r = \begin{cases} -idle_r & s_{r+1} \text{ 时调度结束} \\ -idle_r + \gamma Q'(s_{r+1}, \arg\max_a Q(s_{r+1}, a; \theta'); \theta') & \text{其他} \end{cases}$ 
13:    使用 SGD 以均方误差  $\frac{1}{B} \sum_r (y_r - Q(s_t, a_t; \theta))^2$  作为损失函数优化在线  $Q$  网络参数  $\theta$ 
14:    每隔  $T_{update}$  步更新目标  $\hat{Q}$  网络参数  $\theta' = \theta$ 
15:   end for
16: end for
17: return  $\theta$ 

```

图 4 IOA 调度模型训练流程

模型将经历 E 个回合的训练, 每个回合开始时, 系统随机生成调度实例并获取初始调度状态。在每一时间步 t , 计算所有未被指派的工序智能体的特征表示。为更好地探索解空间, 算法采用 ε -greedy 策略指导模型选择动作, 模型以概率 ε 指派随机候选工序, 否则指派 Q 值最大的工序。之后系统转移到新状态, 并计算各个机器的空闲时间。为训练在线 Q 网络, 算法采用了经验回放机制来打破序列训练样本间存在的相关性, 使训练数据保持独立同分布,

选择动作后得到的训练样本将会先被储存至经验缓存区中, 之后每步从经验缓存区中抽取批量大小为 B 的经验用于学习。在计算 y_r 时, 若调度已结束, 则以及时奖励 $-idle_r$ 作为 y_r 值, 否则以即时奖励和对下一时刻 \hat{Q} 值的预测来计算 y_r 值。算法采用随机梯度下降 (SGD) 方法, 以均方误差作为损失函数优化在线 Q 网络参数。经典的 DQN 算法经常会遇到一个不可避免的问题, 即 \max 算子使用相同的值来选择和计算一个动作, 容易导致对 \hat{Q} 值的估计过于乐

观^[29]。为了避免这一问题，HASSELT 等^[30]提出了一种 Double DQN 的改进方法，可以有效地解决网络过估计的问题，它在计算目标 \hat{Q} 值时使用的下一状态的动作来源于在线 Q 网络而非目标 \hat{Q} 网络(图 4，第 12 行)，这使得学习过程更加稳定和可靠。

3 仿真试验

试验测试分为两部分：静态试验和动态试验。静态试验旨在测试 IOA 调度模型在正常调度时的性能，该试验不考虑机器故障动态事件的发生，调度模型从 $t=1$ 时刻一直连续运行至调度结束；动态试验在调度过程中随机加入机器故障事件，故障发生后模型需修复调度策略后运行至结束。试验部分的调度案例均来自于国际标准案例库 OR-Library^[31]，涵盖大小从 15×15 到 100×20 (工件数量 \times 机器数量)的实例。试验挑选了 15×15 、 20×15 和 30×15 三种规模大小的部分实例用于训练，训练好的模型将在同等大小或是更大规模的实例中进行测试。试验条件为：Tensorflow2.0 平台，Linux 操作系统，英特尔 i7-8750H 的计算机设备。

3.1 静态调度试验

深度神经网络是黑箱模型，难以从其内部运行逻辑来分析它对于所解决的问题是否有效，故需要一个基准来判断深度神经网络学习的有效性。IOA 调度模型依赖于其 \hat{Q} 网络拟合动作价值进而选择动作。分析拟合 \hat{Q} 值的式(5)可以发现，容易导致模型失效的原因是工序智能体特征 $\mu_{i,j}^N$ 可能是无效特征，即 IOA 模型可能无法有效地提取 Job shop 环境

特征。但是即使 $\mu_{i,j}^N$ 失效， \hat{Q} 网络依然能够收敛，原因是 $\mu_{i,j}^N$ 失效后的 \hat{Q} 网络在训练的过程中，权重参数 θ_5 和 θ_6 会沿梯度被更新至趋近于 0，即因变量 \hat{Q} 值对自变量 $\sum_{o_{k,l} \in O} \mu_{k,l}^N$ 和 $\mu_{i,j}^N$ 的变化不敏感。此时式(5)实质上会成为从 $ES_{i,j}$ 到 $Q(s_t, o_{i,j})$ 的映射函数，即

$$Q(s_t, o_{i,j}) \approx \hat{Q}(s_t, o_{i,j}; \theta_Q) = f(ES_{i,j}) \quad (7)$$

此时模型一定收敛，因为若 $ES_{i,j}$ 为唯一可探知的自变量，则更小的 $ES_{i,j}$ 大概率会使得 $\hat{Q}(s_t, o_{i,j}; \theta_Q)$ 越大。也就是说，若智能体只凭借 $ES_{i,j}$ 选择动作，则应贪婪的选择 $ES_{i,j}$ 最小的工序，此时模型实质上已演变为一种贪婪算法模型。故试验以最小化 $ES_{i,j}$ 的贪婪算法作为基准进行比较，若 IOA 调度模型的性能更佳，则可证明所提出的 IOA 构建方法有效。贪婪算法的运行流程为：在每一决策时间点 t ，在所有候选的工序中选取最早开始时间最小的工件 $o_{i,j} = \arg \min(ES_{i,j})$ 进行加工，然后将 $o_{i,j}$ 加入至解序列 π 中。

对比试验结果如图 5 所示，试验使用了“Taillard”实例中 80 个不同的调度问题，涵盖 8 种不同的问题规模大小，取 10 次重复试验的均值作为结果，并以调度分的形式进行比较(调度分为 C_{LB}/C_{alg} ，此处 C_{LB} 为实例最优解的下界， C_{alg} 为算法的 C_{max} ，调度分越高则算法性能越好)。试验结果显示，在 80 个问题实例中仅有 6 个实例 IOA 方法的性能略逊于贪婪算法。IOA 方法的平均调度分为 0.792，贪婪算法的平均调度分为 0.717，IOA 方法的静态调度性能比贪婪算法平均提高了 10.34%，优势明显。

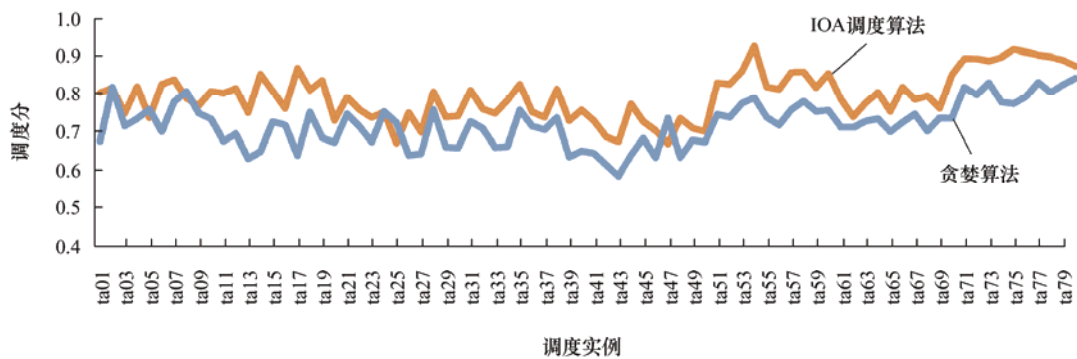


图 5 IOA 调度方法和贪婪算法的静态调度对比

启发式调度规则已被广泛应用于现实的 Job shop 调度场景中，是一种成熟且易用的调度方法，

可以在极短的时间内获得质量可接受的排产方案，表 1 中列举了 16 种常见的调度规则(含组合规则)。

为进一步验证 IOA 方法的静态调度性能, 试验将 IOA 调度方法与这 16 种启发式调度规则结果^[24]进行了比较, 结果如表 2 所示, 评价指标为 C_{\max} (越小越好), 最优的结果被加粗显示。表 2 试验结果显示,

在所有的测试实例中, IOA 调度算法的性能均优于参与对比的 16 种启发式调度规则, 且相较于在每个实例中表现最好的启发式调度规则, IOA 调度算法性能平均提升了 6.38%。

表 1 常见的启发式调度规则

规则	描述
SPT	选择加工时间最短的工件
LPT	选择加工时间最长的工件
SRM	选择剩余加工时间最短的工件 (不包括当前工序的时间)
SRPT	选择剩余加工时间最短的工件
SSO	选择后继工序加工时间最短的工件
LSO	选择后继工序加工时间最长的工件
LPT+LSO	选择当前工序与后继工序加工时间之和最长的工件
LPT*TWK	选择工序加工时间与总加工时间乘积最大的工件
LPT/TWK	选择工序加工时间与总加工时间比值最大的工件
LPT*TWKR	选择工序加工时间与剩余加工时间乘积最大的工件
LPT/TWKR	选择工序加工时间与剩余加工时间比值最大的工件
SPT+SSO	选择当前工序与后继工序加工时间之和最短的工件
SPT*TWK	选择工序加工时间与总加工时间乘积最小的工件
SPT/TWK	选择工序加工时间与总加工时间乘积最大的工件
SPT*TWKR	选择工序加工时间与剩余加工时间乘积最小的工件
SPT/TWKR	选择工序加工时间与剩余加工时间比值最小的工件

表 2 IOA 方法和启发式调度规则的性能比较

	IOA	SPT	LPT	LSO	SRM	SRPT	SSO	LPT+LSO	LPT*TWK	LPT/TWK	LPT/TWKR	LPT*TWKR	SPT+SSO	SPT*TWK	SPT/TWK	SPT*TWKR	SPT/TWKR
ta01	1 539	1 872	1 812	1 957	2 163	2 148	2 148	1 892	1 762	1 973	1 984	1 860	2 199	1 926	1 647	2 204	1 664
ta02	1 530	1 913	1 562	1 759	1 814	2 114	1 905	1 950	1 598	1 761	1 764	1 757	2 104	1 933	1 778	1 957	1 538
ta11	1 699	2 273	2 117	2 216	2 353	2 442	2 343	2 471	2 073	2 163	2 237	1 930	2 045	2 358	2 037	2 184	1 886
Ta12	1 684	2 527	2 213	2 187	2 459	2 160	2 253	2 174	2 142	2 179	2 207	1 908	2 176	2 406	2 160	2 556	1 969
ta21	2 084	2 488	2 691	2 647	3 071	2 955	2 610	2 443	2 648	2 770	2 732	2 802	2 898	3 019	2 338	2 789	2 206
ta22	2 067	2 510	2 515	2 522	2 796	2 726	2 636	2 500	2 422	2 613	2 545	2 348	2 678	2 666	2 788	2 822	2 111
ta31	2 186	2 993	2 589	2 478	3 101	3 156	2 916	2 670	2 622	2 565	2 636	2 491	2 976	3 001	2 619	3 154	2 435
ta32	2 345	3 050	2 624	2 634	3 166	3 272	2 890	2 773	2 681	2 509	2 649	2 562	2 847	3 271	2 663	2 981	2 512
ta41	2 625	3 105	3 155	2 873	3 482	3 232	3 058	3 129	3 158	3 222	3 315	3 014	3 148	3 362	3 266	3 269	2 898
ta42	2 757	3 772	3 356	3 096	3 641	3 624	3 528	3 348	3 199	3 194	3 043	2 954	3 358	3 738	3 361	3 965	2 813
ta51	3 338	4 456	3 881	3 844	4 174	4 443	4 418	3 873	3 902	3 859	3 956	3 833	4 099	4 460	4 058	4 590	3 768
ta52	3 348	4 179	3 891	3 715	4 588	4 371	4 059	3 878	3 900	4 023	3 976	3 678	4 187	4 556	3 724	4 694	3 588
ta61	3 663	4 500	4 467	4 188	5 024	5 041	4 520	4 432	4 409	4 258	4 283	3 943	4 523	4 813	4 174	4 933	3 752
ta62	3 896	4 933	4 416	4 217	4 764	4 821	4 757	4 551	4 365	4 485	4 494	4 000	4 699	4 868	4 323	5 310	3 925
ta71	6 129	7 830	6 949	6 754	7 916	8 118	7 594	7 065	6 874	7 067	7 264	6 819	7 638	8 186	7 445	7 912	6 705
ta72	5 813	7 611	6 675	6 674	7 607	7 639	7 077	6 689	6 758	7 199	7 011	6 471	7 601	7 640	6 910	7 643	6 351

3.2 动态调度试验

在动态调度试验中, 使用“Taillard”实例将 IOA 调度方法与贪婪算法进行比较, 在调度过程中随机加入机器故障事件。使用指数分布的参数 λ 控制机器故障事件发生的概率, 使用正态分布的参数 μ 控

制单次故障的持续时间。在 $\lambda = 0.000\ 2$, $\sigma^2 = 10$ 的条件下, 试验对比了不同的单次故障持续时间期望值 μ 对 IOA 方法和贪婪算法调度性能的影响, 结果如图 6 所示; 在 $\mu = 100$, $\sigma^2 = 10$ 的条件下, 试验对比了机器故障率参数 λ 对 IOA 方法和贪婪算法调

度性能的影响,如图 7 所示。试验结果表明,在机器故障概率变大,故障持续时间变长后,IOA 方法和贪婪算法的性能都会受到影响,调度分分别有不同程度的下降。但是在受到机器故障因素扰动后,IOA 调度方法的性能依然优于贪婪算法,并且调度结果受到的扰动更小,证明了 IOA 调度方法拥有较好的鲁棒性和更佳的算法性能。

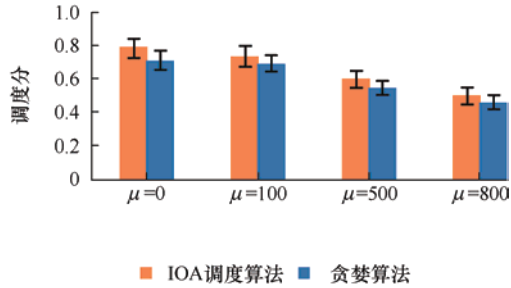


图 6 故障时间期望值 μ 对 IOA 方法和贪婪算法的影响

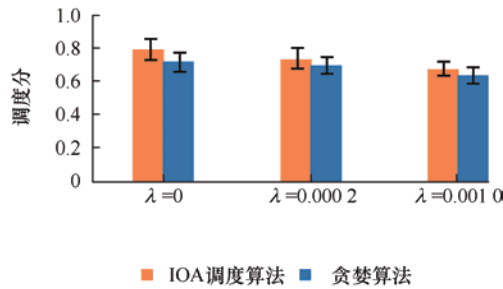


图 7 故障率参数 λ 对 IOA 方法和贪婪算法的影响

4 结论

论文提出了一种基于交互式工序智能体 IOA 的调度方法用于解决 Job shop 调度问题,构建的工序智能体可以有效地利用调度问题的原始信息,根据彼此间的关系进行特征交互,从而达到协同更新自身的特征向量。IOA 调度模型通过综合分析各工序智能体的特征,使用 \hat{Q} 网络近似估计不同状态下选择加工工序的动作价值,并将具有最大价值的工序添加至解序列中生成调度方案。设计了一种带有经验回放机制的深度强化学习 Double DQN 算法训练 IOA 调度模型中的可学习参数,经过训练的模型可以求解不同规模的 Job shop 调度问题,验证了 IOA 调度模型具有良好的泛化能力。大量的试验测试结果表明,无论在静态调度还是动态调度中,IOA 调度方法都能获得较高质量的解,且在面对机器故障动态事件的扰动时,可以快速进行重调度,及时有效地修复生产调度策略,保证生产方案不会因机器故障扰动而失效。

论文仅考虑了机器故障的随机扰动,研究订单插入和变更等动态事件对 IOA 调度模型的影响将是今后的主要工作之一。此外,完善工序智能体的交互方式,优化神经网络结构以及尝试不同的深度强化学习算法以提升 IOA 方法的性能也有待于进一步深入研究和探索。

参 考 文 献

- [1] ZHANG J, DING G, ZOU Y, et al. Review of job shop scheduling research and its new perspectives under Industry 4.0[J]. Journal of Intelligent Manufacturing, 2017, 30: 1809-1830.
- [2] 肖世昌, 吴自高, 孙树栋, 等. 双资源约束的鲁棒 Job Shop 调度问题研究[J]. 机械工程学报, 2021, 57(4): 227-239.
XIAO Shichang, WU Zigao, SUN Shudong, et al. Research on the dual-resource constrained robust job shop scheduling problems[J]. Journal of Mechanical Engineering, 2021, 57(4): 227-239.
- [3] XANTHOPOULOS A, KOULOURIOTIS D E. Cluster analysis and neural network-based metamodeling of priority rules for dynamic sequencing[J]. Journal of Intelligent Manufacturing, 2018, 29(1): 69-91.
- [4] WANG C, JIANG P. Manifold learning based rescheduling decision mechanism for recessive disturbances in RFID-driven job shops[J]. Journal of Intelligent Manufacturing, 2018, 29(7): 1485-1500.
- [5] PENG B, LÜ Z, CHENG T. A tabu search/path relinking algorithm to solve the job shop scheduling problem[J]. Computers & Operations Research, 2015, 53: 154-164.
- [6] CROCE F, TADEI R, VOLTA G. A genetic algorithm for the job shop problem[J]. Computers & Operations Research, 1995, 22(1): 15-24.
- [7] WERNER F, WINKLER A. Insertion techniques for the heuristic solution of the job shop problem[J]. Discrete Applied Mathematics, 1995, 58(2): 191-211.
- [8] ADAMS J, BALAS E, ZAWACK D. The shifting bottleneck procedure for job shop scheduling[J]. Management Science, 1988, 34(3): 391-401.
- [9] ZHANG W, WEN J B, ZHU Y C, et al. Multi-objective scheduling simulation of flexible job-shop based on multi-population genetic algorithm[J]. International Journal of Simulation Modelling, 2017, 16(2): 313-321.
- [10] 赵诗奎. Job Shop 基于无延迟调度路径重连与回溯禁忌搜索算法研究[J]. 机械工程学报, 2021, 57(14): 291-303.
ZHAO Shikui. Research on path relinking based on non-delay scheduling and backtracking tabu search algorithm of job shop scheduling problem[J]. Journal of

- Mechanical Engineering, 2021, 57(14): 291-303.
- [11] 孟磊磊, 张彪, 任亚平, 等. 求解分布式柔性作业车间调度的混合蛙跳算法[J]. 机械工程学报, 2021, 57(17): 263-272.
- MENG Leilei, ZHANG Biao, REN Yaping, et al. Hybrid shuffled frog-leaping algorithm for distributed flexible job shop scheduling[J]. Journal of Mechanical Engineering, 2021, 57(17): 263-272.
- [12] MUHAMMAD K A, SHAHID I B, RUBEENA K, et al. Recent research trends in genetic algorithm based flexible job shop scheduling problems[J]. Mathematical Problems in Engineering, 2018(8): 1-3.
- [13] SHAKHLEVICH N, SOTSKOV Y N, WERNER F. Adaptive scheduling algorithm based on mixed graph model[J]. IEE Proceedings-Control Theory and Applications, 1996, 143(1): 9-16.
- [14] LEE K K. Fuzzy rule generation for adaptive scheduling in a dynamic manufacturing environment[J]. Applied Soft Computing, 2008, 8(4): 1295-1304.
- [15] WANG L, PAN Z X, WANG J J. A review of reinforcement learning based intelligent optimization for manufacturing scheduling[J]. Complex System Modeling and Simulation, 2021, 1(4): 257-270.
- [16] DRUGAN M M. Reinforcement learning versus evolutionary computation : A survey on hybrid algorithms[J]. Swarm and Evolutionary Computation, 2019, 44: 228-246.
- [17] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.
- [18] SILVER D, HUANG A, MADDISON C J, et al. Mastering the game of go with deep neural networks and tree search[J]. Nature, 2016, 529(7587): 484-489.
- [19] SILVER D, SCHRITTWIESER J, SIMONYAN K, et al. Mastering the game of go without human knowledge[J]. Nature, 2017, 550(7676): 354-359.
- [20] LIU C L, CHANG C C, TSENG C J. Actor-critic deep reinforcement learning for solving job shop scheduling problems[J]. IEEE Access, 2020, 8: 71752-71762.
- [21] 肖鹏飞, 张超勇, 孟磊磊, 等. 基于深度强化学习的非置换流水车间调度问题[J]. 计算机集成制造系统, 2021, 27(1): 192-205.
- XIAO Pengfei, ZHANG Chaoyong, MENG Leilei, et al. Non-permutation flow shop scheduling problem based on deep reinforcement learning[J]. Computer Integrated Manufacturing System, 2021, 27(1): 192-205.
- [22] 王凌, 潘子肖. 基于深度强化学习与迭代贪婪的流水车间调度优化[J]. 控制与决策, 2021, 36(11): 2609-2617.
- WANG Ling, PAN Zixiao. Scheduling optimization for flow-shop based on deep reinforcement learning and iterative greedy method[J]. Control and Decision, 2021, 36(11): 2609-2617.
- [23] PALOMBARINI J A, MARTÍNEZ E C. Closed-loop rescheduling using deep reinforcement learning[J]. IFAC-PapersOnLine, 2019, 52(1): 231-236.
- [24] HAN B A, YANG J J. Research on adaptive job shop scheduling problems based on dueling double DQN[J]. IEEE Access, 2020, 8: 186474-186495.
- [25] LUO S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning[J]. Applied Soft Computing, 2020, 91: 106208.
- [26] ZHANG Y, ZHU H, TANG D, et al. Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems[J]. Robotics and Computer-Integrated Manufacturing, 2022, 78: 102412.
- [27] LIU R, PIPLANI R, TORO C. Deep reinforcement learning for dynamic scheduling of a flexible job shop[J]. International Journal of Production Research, 2022, 60(13): 4049-4069.
- [28] DAI H, KHALIL E B, ZHANG Y, et al. Learning combinatorial optimization algorithms over graphs[C]//Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach, California, USA: 2017: 6351-6361.
- [29] HASSELT H. Double Q-learning[J]. Advances in Neural Information Processing Systems, 2010, 23: 2613-2621.
- [30] HASSELT H V, GUEZ A, SILVER D. Deep reinforcement learning with double q-learning[C]//Proceedings of the AAAI Conference on Artificial Intelligence. Phoenix, Arizona, USA: 2016: 2094-2100.
- [31] BEASLEY J E. OR-Library: Distributing test problems by electronic mail[J]. Journal of the Operational Research Society, 1990, 41(11): 1069-1072.

作者简介: 陈睿奇, 男, 1999 年出生。主要研究方向为生产调度系统, 组合优化问题, 深度强化学习。

E-mail: Rui.IE@outlook.com

黎雯馨, 女, 1999 年出生。主要研究方向为生产/物流调度, 深度强化学习。

E-mail: liwenxinn@gmail.com

王传洋, 男, 1972 年出生, 博士, 教授, 博士研究生导师。主要研究方向为智能制造, 先进制造技术。

E-mail: cywang@suda.edu.cn

杨宏兵(通信作者), 男, 1977 年出生, 博士, 副教授。主要研究方向为制造系统建模与分析, 生产调度与优化方法。

E-mail: yanghongbing@suda.edu.cn