计算机科学
COMPUTER SCIENCE

一种基于修正机制和强化学习的作业车间调度问题的优化算法

苗宽, 李崇寿

引用本文

苗宽, 李崇寿. 一种基于修正机制和强化学习的作业车间调度问题的优化算法[J]. 计算机科学, 2023, 50(6): 274-282.

MIAO Kuan, LI Chongshou. Optimization Algorithms for Job Shop Scheduling Problems Based on Correction Mechanisms and Reinforcement Learning [J]. Computer Science, 2023, 50(6): 274-282.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220900112

一种基于修正机制和强化学习的作业车间调度问题的优化算法

苗　宽[1]　李崇寿[1,2]
1 西南交通大学计算机与人工智能学院　成都 610097
2 西南交通大学利兹学院　成都 610097
(1084083934@qq.com)

摘　要　近年来,使用深度强化学习解决作业车间调度问题的研究主要集中于构造法,通过将作业车间调度问题视为顺序决策问题,逐步选择调度节点从而得到完整的解。尽管这种算法思想已经取得了不小的成果,但仍面临奖励构造困难、解决方案质量不高的问题,因此这一方法的发展受到制约。针对这些问题,设计了一种基于图神经网络和近端策略优化算法的强化学习构造框架。同时,针对因训练与测试数据分布不一致而带来的次优解问题,还设计了一种修正交换算子,以保证解的质量。最后,为了证明算法的有效性,在公开数据集和生成的数据集上进行了实验。实验结果表明,所提算法在中小规模实例上的结果优于目前最好的强化学习框架,不仅充分发挥了构造式强化学习框架求解迅速的优势,还通过修正机制有效缓解了次优选择问题,缩短了实例的最大完成时间。

关键词:调度;作业车间调度问题;强化学习;修正搜索算法
中图法分类号　TP181

Optimization Algorithms for Job Shop Scheduling Problems Based on Correction Mechanisms and Reinforcement Learning

MIAO Kuan[1] and LI Chongshou[1,2]
1 School of Artificial Intelligence and Computing,Southwest Jiaotong University,Chengdu 610097,China
2 SWJTU-Leeds Joint School,Southwest Jiaotong University,Chengdu 610097,China

Abstract　In recent years,research on using deep reinforcement learning to solve job shop scheduling problems has concentrated on construction techniques,which model the scheduling problem as sequential choice problems and gradually select scheduling nodes for a complete solution. Although this algorithmic theory has produced impressive results,it still suffers from complicated reward formulation and poor solution quality,which prevents its future development. In this study,we design a reinforcement learning construction framework based on graphical neural networks and proximal policy optimisation algorithms,and an innovative and efficient search correction mechanism with a modified swap operator is proposed to enhance the solution quality. It searches the area around a known solution using a Monte Carlo tree,correcting the issue of suboptimal solution selection caused by the discrepancy between training and testing data. The proposed algorithm is comprehensively investigated on public and synthetic datasets. Experimental results demonstrate that the algorithm outperforms the state-of-the-art reinforcement learning framework on both small and medium-sized examples. It not only fully exploits the advantages of rapid solution of constructive reinforcement learning framework,but also effectively corrects the sub-optimal choice through the correction mechanism,reducing the maximum completion time in worst cases.

Keywords　Scheduling,Job shop scheduling problems,Reinforcement learning,Modified search algorithms

到稿日期:2022-09-13　返修日期:2022-11-09
基金项目:国家自然科学基金(62202395);四川省自然科学基金(2022NSFSC0930);中央高校基本科研业务费专项资金(2682022CX067);四川省重点研发项目(2022YFG0028)
This work was supported by the National Natural Science Foundation of China(62202395),Natural Science Foundation of Sichuan Province,China(2022NSFSC0930),Fundamental Research Funds for the Central Universities of Ministry of Education of China(2682022CX067) and Key R&D Project of Sichuan Province,China(2022YFG0028).
通信作者:李崇寿(lics@swjtu.edu.cn)

1　引言

由于可行解的数量随问题规模呈指数型增长，组合优化问题具有 NP-Hard 难度，近年来引起了大量研究人员的关注。组合优化问题难以进行精确求解，且无法在多项式时间内穷举得到问题的最优解，因此，研究人员往往通过设计启发式方法来得到实例的解决方案[1]。近年来，随着强化学习(Reinforcement Learning，RL)在 AlphaGo[2]、游戏[3-4]等领域不断取得突破性进展以及组合优化问题天然的顺序决策结构，研究人员开始尝试使用强化学习对组合优化问题进行求解。

作为一种经典的组合优化问题，作业车间调度问题(Jop-Shop Scheduling Problem，JSSP)同样具有 NP-hard 复杂度[5]。该问题由 n 个工作和 m 个机器组成，每项作业需要经过 m 道在不同机器上加工的工序，工序之间需要依次完成。调度员通过合理地调度工序，使得工作在尽可能短的时间内完成。优先调度规则(Priority Dispatching Rule，PDR)[6-8]是一种启发式规则，它根据设定的规则对作业进行评级，选择优先级最高的作业进行调度。由于规则简单，PDR 可以实现对作业车间调度问题的快速求解。尽管调度规则依赖于专家知识，但其忽略了调度问题的顺序性，因此所提供的解决方案质量偏低，这也促使研究人员开发新的算法以寻找质量更高的解决方案。近年来，随着强化学习在路由问题[9]上不断取得突破，研究人员开始探索强化学习在作业车间调度问题中的应用。由于车间调度天然的顺序性，多数研究将其视为顺序决策问题[10-13]，通过训练好的策略调度模型逐步进行调度，为问题实例提供了质量较好、可解释性强的解决方案。然而，构造式强化学习算法存在扩展性差、泛化性不足的缺点[14]。形成这些缺点的原因是多方面的。一方面，构造式强化学习训练算法往往需要指定训练数据，从而在标准数据集上取得较好的结果。训练集和测试集的分布不一致会导致解的质量较低[15]。另一方面，在进行候选工序选择时，动作选择器根据策略网络生成的候选工序概率进行工序选择。现阶段的动作选择器为贪婪和采样方法[16]，这两种方法并不能保证最优决策。相对而言，贪婪方法对策略网络更为敏感，当策略决策失误时，结果无任何改进。而采样方法可能选择概率低的工序，以寻找质量更好的解。但是采样方法也存在耗时长的缺点。此外，当最优工序的概率接近 0 时，采样只能选择次优工序，这也导致采样出的解的质量存在上限。最后，构造强化学习框架在奖励函数构造时往往依赖于专家经验，主观性较强，从而导致策略学习不够充分、次优工序选择概率增大等问题。

针对构造强化学习的上述缺陷及特点，本文设计了一种基于强化学习和图神经网络的构造框架，用于提供初始解，该框架将作业车间调度问题视为顺序决策问题，对其进行马尔可夫建模并使用图神经网络进行策略网络搭建，使用近端策略优化算法(Proximal Policy Optimzaion，PPO)[17]训练调度策略。此外，针对训练数据和测试数据分布不一致和奖励函数设置主观、策略网络学习不充分所导致的选择次优工序的问题，本文提出一种新颖有效的交换算子，使用蒙特卡洛树搜索算法对解的邻域进行进一步修正搜索，以提高解的质量。实验表明，所提算法通过修正策略网络，有效地提高了解的质量，在中小规模实例上的结果明显优于当前主流的强化学习框架。在训练完成后，模型无需进一步训练即可泛化到其他实例中。本文的主要贡献总结如下：

(1)设计了一个构造式强化学习框架，该算法基于图神经网络(Graph Neural Networks，GNN)的节点嵌入并以端到端的形式进行强化学习的训练调度，从而准确、高效地为实例提供高质量的解决方案。

(2)构建了一种有效的交换修正算子，并使用蒙特卡洛树算法对构造强化学习框架所提供的初始解进行修正搜索，有效提高了解的质量。

(3)实验表明，所提算法框架在各种中小规模车间调度问题实例上的性能优于广泛使用的调度算法，为解决 JSSP 问题提供了一种新的高质量方法。

2　相关工作

随着强化学习的不断发展，研究人员开始探索使用强化学习来解决具有较高复杂度的组合优化问题[18]。在初始阶段，研究人员主要研究路由问题[16-21]、图优化问题[22-23]等。近年来，研究人员开始对调度问题进行探索。

在现阶段，已经有一些工作对作业车间调度问题进行了研究。利用调度问题具有图结构的特点，针对结构相同而数据多样的问题，Khalil 等[24]对最大割等问题进行图嵌入编码，提出了基于强化学习和图神经网络的算法。基于此，Zhang 等[9]将工序作为节点，根据同一机器上加工的工序具备时间关系，建立图关系，并使用图同构网络[25]区分不同的实例结构，使用马尔可夫决策公式对 JSSP 问题进行建模并使用近端策略优化算法进行训练，取得了较好的效果。但是使用工序处理时间、机器作为状态的方法忽略了当前工序对后续的影响，为此，Park 等[10]构建了统计特征作为状态表示，如操作状态、处理时间、完成度、后续操作数和等待时间等。在图表示编码方面，将节点分为先行节点、后继节点、分离节点和目标节点，然后进行图神经网络编码，进而实现性能的提升。同时，上述方法的奖励是在每个调度完成后根据奖励函数计算所得，往往与完成时间具有高度的相关性，因此策略可以快速收敛并取得不错的效果。然而这种奖励函数需要精心设计，存在时间开销大的问题。针对这一问题，Park 等[11]根据车间调度实例存在的最大完成时间($C_{\max}$)的自然属性，在训练结束后给予 $-C_{\max}$ 的奖励，基于类型图注意力搭建策略网络，从而取得了远超前两者的效果。尽管马尔可夫建模过程并不相同，但这些研究的核心目的都是训练出有足够泛化能力的策略调度器，从而使得网络在测试数据集的实例上取得较好的结果。Tassel 等[12]则构建一个 JSSP 环境，通过在线训练的方式进行强化学习调度，最终得到接近谷歌公开求解器 OR-Tools[26]的结果。

此外，现有基于强化学习研究的调度问题研究也有其他思路。Zeng 等[27]首先将一系列 PDR 调度规则(如最短处理时间、最大剩余处理时间、先来先做、后来先做等 8 种)作为动作，在调度工序时，先根据策略选择调度规则，再根据调度规则完成调度。Monaci 等[28]提出使用双层长短期记忆网络[29]

(Long Short Term Memory, LSTM)缓解作业和工序的可变性。受迭代贪心(Iterated Greedy, IG)算法的影响,Ni 等[30]将强化学习与搜索框架相结合,先根据启发式算法(如 NEH[31])生成初始解,再设计 3 种随机算子,分别从全部作业、总空闲时间前 50%的作业以及总等待时间前 50%的作业中随机选择,不断迭代以生成新的解,训练得到接近于下限的结果。针对在线与离线训练的差距问题,Oren 等[32]提出将蒙特卡洛树搜索和图卷积搜索相结合,提升整体性能。Wang 等[33]沿袭这一思想,将遗传算法与强化学习相结合,构建双层框架。底层通过遗传算法进行启发式求解,顶层则通过修改边等措施进行图结构的优化从而达到总体优化,同时缓解了奖励稀疏的问题。

研究策略大多集中于构造法[9-11],这些方法能够在极短的时间内获取问题的解。但在模型训练阶段,往往会面临奖励稀疏、解决方案质量不高等问题。不仅如此,在测试阶段,由于训练数据和测试数据分布不一致、策略学习不到位等问题,常给出次优选择,从而影响后续工序的开始时间,最终推迟完成时间。针对这些问题,本文提出了一种修正搜索机制,与构造式强化学习框架相结合,提高解的质量。

3 问题定义

标准的车间调度问题由一组作业$|J|$和一组机器$|M|$组成。每项作业$J_i \in J(1 \leqslant i \leqslant |J|)$由$|M|$道工序组成,这些工序必须在特定的机器上完成。在标准问题中,工序各不相同。工序必须依次完成且机器在操作某一道工序时,不能同时加工其他工序。作业 J_i 的第 j 个工序处理时间$p_{ij} \in \mathbb{N}(1 \leqslant j \leqslant |M|)$通常为指定参数。JSSP 问题研究如何通过合理的调度来使得完成时间最短。若每个工序的开始时间为$s_{ij} \in S$,则优化目标可表示为最小化$C_{\max} = \max\limits_{i,j} s_{ij} + p_{ij}$。具体符号定义如表 1 所列。

表 1 问题符号定义

Table 1 Definition of problem symbols

Symbols	Implications
J	作业集合
M	机器集合
A	动作集合
S	开始处理时间集合
J_i	第 i 项作业
p_{ij}	第 i 项作业的第 j 道工序的处理时间
s_{ij}	第 i 项作业的第 j 道工序开始处理的时间
C_{ij}	第 i 项作业的第 j 道工序加工完成的时间
$C_{\max}$	最大完成时间
$\|J\|$ 或 n	工作数量
$\|M\|$ 或 m	工序/机器数量

4 算法

本节将介绍方法的基本原理。图 1 给出了整体算法框架。在解决问题实例时,先通过训练得到的代理调度模型生成初始解,再将初始解进行蒙特卡洛树搜索进行更新,当迭代步数达到限制或迭代时间达到上限后,搜索终止,输出问题实例的最终解决方案。

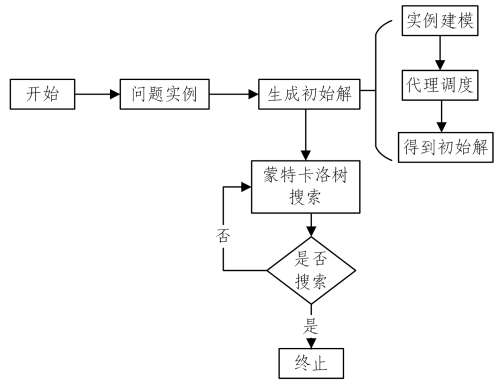

图 1 整体算法框架

Fig. 1 Overall algorithm framework

4.1 构造式强化学习算法

本节将调度过程视为顺序决策,构造强化学习框架算法。首先将调度过程进行马尔可夫决策建模,然后构建策略网络。

4.1.1 强化学习建模

(1)开始时间。在调度过程中,不仅需要选择好下一步调度的工序,也需要确定好工序的开始时间。调度工序的开始时间需要满足前一道工序已完成和对应机器处于空闲状态的限制,为了尽可能缩短完成时间,该工序的开始时间设置为其工作的前一道工序的完成时间和对应机器的空闲时间的最大值。

本文算法从甘特图中提取有效特征作为状态,状态输入到设计的图神经网络中通过强化学习进行训练,以决定下一步调度的工作,不断迭代从而得到完整的解。我们建立的马尔可夫决策过程(Markov Decision Processes, MDP)描述如下。

(2)状态。状态反应当前 JSSP 实例的完成程度及未来状态。基于 JSSP 解的甘特图表示,本文使用$|J| \times 8$的矩阵表示当前状态:1)一个二维量表示工作是否可被选择;2)当前工作已完成的最后一道工序的操作时间,初始状态设置为每项工作第一道工序的操作时间;3)工作的工序完成率;4)工作的下一道工序可被操作的时间(已经操作的工序的最小完成时间);5)工作距离全部完成的剩余时间;6)已完成的工序时间除以剩余的时间;7)该工作的调度所造成的总的空闲时间;8)该工作到目前为止的最后一道工序调度所产生的空闲时间。需要注意的是,调度完成时,6)设置为 0。4),7),8)在工作完成后仍为最后一道工序调度完成后的状态,不做改变。

(3)动作。本文将 JSSP 视为单代理问题。在整个调度过程中,代理选择要处理的工作,由于工序必须依次完成,因此,当确定动作时,便可确定将要调度的工序,动作集合 $A = J_0, \cdots, J_{|J|-1}$。

(4)奖励。在调度过程中,我们的目标是获得调度的最小完成时间。同时,当调度完成时间最小时,甘特图的整体空洞率最低。基于文献[12],本文设置奖励如下:

$$R_{s,a} = p_{a,j} - jobSumEmpty_a \tag{1}$$

在实验中,我们发现通过使用工作数量对$R_{s,a}$进行放缩后效果更好,其中 $JobSumEmpty_a$ 指工作 a 到目前为止因调度而产生的空闲时间(空洞面积)。

4.1.2　策略网络

本文基于甘特图进行特征统计作为状态，为充分挖掘内部信息，使用具有强区分能力的图神经网络对原始状态进行表征。

(1)图表示建模。在解决路由问题时，Xin 等[34]提出在访问节点后，该节点不可再被访问，同时，未来的选择也不再与这个节点的信息相关。基于此，我们进行图关系建模。假设每项作业为图中的一个节点，边为作业之间的时间关系集。这种建模形式为编码不同的解决方案和实例属性提供了高度的灵活性。建模的核心理念主要有以下两点：1)作业的状态与当前和未来所要调度的目标有关，与过去无关[34]；2)将作业视为节点，节点所代表的状态信息会随着工序的调度而更新，节点之间的关系也会随之改变。在动作的调度过程中，伴随着边的增加和删除。当调度动作 i 时，由于节点状态改变，删除关系集中所有指向 i 的关系。同时，假设调度完成后，动作 j 在动作 i 之前，因此添加从节点 j 指向节点 i 的关系。下面用实例进行详细阐述。图 2 中红色字体表示调度的顺序，框内(a,b)表示为 a 工作的第 b 道工序。假设存在一 3×3 的 JSSP 问题，目前已经调度了(3,1)，(2,1)两道工序，接下来需要调度(2,2)和(3,2)两道工序。当调度(2,2)工序时，此时关系集中没有涉及节点 e_2 的边，因此不需要删除，在 M_2 机器中，节点 e_3 和节点 e_2 形成时间上的连续关系，因此向边集中添加(e_3,e_2)的有向边。当调度(3,2)工序时，此时节点 e_3 的状态发生变化，关系集中有涉及节点 e_3 的边，因此删除所有含 e_3 的边。同时，在机器 M_1 上，节点 e_2 在前，因此，向关系图中添加(e_2,e_3)的有向边。

需要注意的是，在边关系的动态增删中，由于工序所在机器的前一道工序是固定的并且在增加涉及工作 i 的边时，需要将前一条指向 i 的边或 i 指向的边删除。因此，边关系的数量上限为 $n-1$。而 n 的值一般较小，因此，边关系的动态增减并不会带来巨大的计算开销。

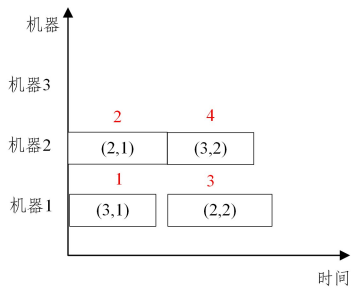

图 2　图表示建模示例(电子版为彩图)

Fig. 2　Example diagram of graph modeling

(2)图神经网络。本文使用图卷积神经网络(Graph Convolutional Network，GCN)[35]和图注意力网络(Graph Attention Network，GAT)[36]搭建图神经网络。图卷积网络可以线性扩展图边的数量，并学习编码局部图结构和节点特征的隐藏层表示。图注意力网络可以学习编码各个节点之间的关系，从而获取更准确的节点特征的表示。

策略网络的整体框架如图 3 所示。策略网络通过两个完全相同的图注意力网络对状态中的不同信息进行关注，为避免参数过多，使用 Dropout 层进行部分失活，进而通过图卷积网络对信息进行聚合，从而得到最终的编码信息。状态通过图神经网络后，由于原始状态中本身也包含了很多信息，为加强搜索效果，因此在演员网络前加入残差结构[37]。

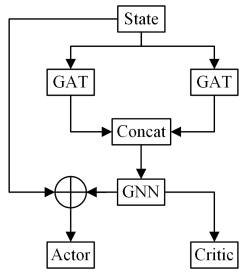

图 3　策略网络的整体框架

Fig. 3　Overall framework of strategy network

4.1.3　训练算法

本文使用近端策略优化算法[17]训练策略网络。PPO 算法是信任区域策略优化[38]的简化版本，在算法上有所简化。此外，我们也使用了文献[39]中提及的奖励归一化、放缩等措施。PPO 算法是一种 actor-critic 算法，actor 指策略网络 π_θ，critic 指状态评估网络 v_φ，actor 与 critic 共享图神经网络。训练算法的详细信息如算法 1 所示。

算法 1　构造强化学习训练算法

输入：actor 网络 π_θ 和行动参与者网络 $\pi_{\theta_{old}}$，评论家网络 v_φ，训练轮数 U，更新轮数 K，策略损失函数 c_p，价值损失函数 c_v，熵函数损失系数 c_e，裁剪率

输出：actor 网络网络 π_θ，critic 网络 v_φ

1. 初始化网络参数 π_θ，v_φ，令 $\theta_{old}=\theta$

2. for each u=1,2,…,U

3. 随机生成实例大小 m～[5,18]，n～[2,max(8,m)]

4. 根据分布 $\mathbb{D}$～(m,n,1,99)生成 m * n 的加工时间在[1,99]的实例

5. 根据实例初始化环境得到状态 s_0

6. for t=0,1,2,…,n×m−1

7. 　根据 $\pi_{\theta_{old}}(a_t|s_t)$ 和状态 s_t 采样得到 a_t

8. 　将 a_t 送入环境得到 r_t 和下一个状态 s_{t+1}

9. $A_t=r_t-\dfrac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$

10. end

11. $L^{CLIP}(\theta)=\sum_{t=0}^{n\times m-1}\min(r_tA_t,\mathrm{clip}(r_t,1-\varepsilon,1+\varepsilon)A_t))$

12. $L^{VF}(\varphi)=\sum_{t=0}^{n\times m-1}(v_\varphi(s_t)-A_t)^2$

13. $L^S(\theta)=\sum_{t=0}^{m\times n-1}S(\pi_\theta(a_t|s_t))$

14. 计算总的损失函数 $L(\theta,\varphi)=c_pL^{CLIP}(\theta)-c_vL^{VF}(\phi)+c_eL^s(\theta)$

15. for k=1,2,…,K

16. 　更新 actor 网络 π_θ 和 critic 网络 v_ϕ

17. 　$\theta,\phi=\mathrm{argmax}(L(\theta,\phi))$

18. end

19. 　若满足迭代轮数，令 $\theta_{old}=\theta$

20. End

4.2　基于蒙特卡洛树的搜索算法

本小节介绍基于蒙特卡洛树的邻域搜索算法，首先介绍交换修正算子，然后阐述蒙特卡洛树的搜索过程。

4.2.1　交换修正算子

在构造式求解过程中，当面对一组即将被调度的工序，

调度器可以调度比较合适的工序。然而，由于训练数据和奖励设置问题，调度器往往选择的是次优工序，造成解决方案质量较差。在传统的启发式算法中，作业车间调度的搜索算子基于路径问题的各种算子（如 2-opt，swap 等）进行设计，这些算子在路由问题中具备明确的物理意义，但在作业车间调度中则不具备实际的物理意义，且搜索空间稀疏。因此，现有的研究仅使用该算子对较小规模数据进行实验并取得了较好的效果，但难以泛化到较大规模问题实例中。

针对上述问题，本文设计了紧凑有效且具备物理意义的交换修正算子。在调度过程中，由于不同工序需要在不同机器上进行调度，在交换操作顺序时，存在难以评估对解的质量的影响问题。因此，本文将交换修正算子定义为将当前工序与在其后的、最近的同一机器上加工的工序的顺序交换。这两道工序均为当前调度的候选工序，因此，这种修正算子可以被看作是对策略次优选择的修正，具备物理意义。

假设存在一个 3×3 实例，策略调度器初始化后的解决方案可以编码表示为式(2)的形式。

$$s_t=[0,2,1,1,2,0,2,0,1] \tag{2}$$

这种编码形式可以有效避免交换时出现的工序约束冲突问题，当对第 3 道调度工序执行交换算子操作时，我们向后寻找，直至找到在同一机器上加工的工序。假设此时为第 5 个工序，则可执行交换操作，解更替为式(3)。

$$s_{t+1}=[0,2,2,1,1,0,2,0,1] \tag{3}$$

算法本质上即为多次重复调用这一过程对解进行次优选择的更换。在第 5 节中，我们将绘制甘特图对更换前后算子所做的工作进行可视化。

4.2.2 蒙特卡洛树搜索算法

蒙特卡洛树[40-43]（见图 4）包括 4 个部分：初始化、模拟、选择、反向传播，分别设计如下。

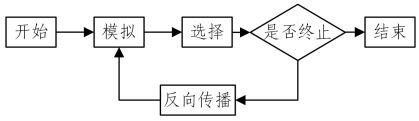

图 4 蒙特卡洛树搜索算法的整体过程

Fig. 4 Overall process of Monte Carlo tree search algorithm

(1)初始化。本文使用一系列工作序号作为解，初始解由 RL 策略调度得到。针对 $n\times m$ 的实例，我们定义两个 $n\times m$ 的矩阵，即奖励反馈矩阵 $\boldsymbol{W}$，其元素 W_{ij}（初始化为 1）表示工序(i,j)的权重；访问矩阵 $\boldsymbol{N}$，其元素 N_{ij}（初始化为 0）表示在模拟期间选择工序(i,j)的次数。此外，变量 L（初始化为 0）用于记录已经模拟的动作总数。整个初始化过程仅在开始时执行一次。

(2)模拟。给定状态 s_t，根据势矩阵选择目标工序，动作即为 4.1 节中所描述的交换算子，当选定工序时，工序的调度序号与对应机器上后一道加工的工序进行交换。我们使用式(4)计算对应工序的势（势越大，工序被选中的机会越大）。

$$Z_{i,j}=\frac{W_{i,j}}{N_{i,j}}+c\sqrt{\frac{\log(L+1)}{N_{i,j}+1}} \tag{4}$$

其中，左边表示工序(i,j)的平均权重，代表已知最好的探索；右边增强选择次数较少工序被选中的可能性；c 为超参数，用于实现探索和利用的平衡；“+1”是为了避免分子或分母为 0。

(3)选择。在上述模拟过程中，为了减小状态空间，只有当改进动作不会增加当前目标函数值时，该动作才应用于当前状态以获取新的状态 s_{t+1}。这种策略的有效性在实验中得到充分证明。

(4)反向传播。$\boldsymbol{L},\boldsymbol{W},\boldsymbol{N}$ 矩阵中的元素通过如下的反向传播进行更新，每选择一道工序(i,j)，L 就加 1，N_{ij} 也加 1。奖励反馈矩阵的更新公式如下（无论动作导致什么结果，均进行反向传播）：

$$W_{i,j}\leftarrow W_{i,j}+\beta\left[\exp\left(\frac{C^{s_t}_{\max}+C^{s'_{t+1}}_{\max}}{C^{s_t}_{\max}}\right)-1\right] \tag{5}$$

其中，$C^{s_t}_{\max}$表示 s_t 状态下的最大完成时间，s'_{t+1}指 s_t 采取动作 a_t 后的新解。无论动作结果好坏，均进行反向传播。由于我们不接受更差的解，因此当结果变差时，再次实施该动作，即可返回原状态。β 为超参数，用于控制增幅比率。实验中设置 $c=1$，$\beta=5$。

5 实验

本节讨论在生成的实例和在公开 JSSP 数据集上的实验结果。

本文按照 Taillard 方法生成不同大小的 100 个数据实例，包括 6×6，10×10，15×15，20×20，30×15，30×20，50×20，100×20。此外，本文还使用了公开的 JSSP 数据实例如 Taillard[44]，ABZ[45]，FT[46]，Lawrence[47]，SWV[48]，YN[49]，Orb[50]来评估所提方法的有效性。

模型所使用的训练集大小 $n\sim[5,18]$，$m\sim[2,\max(8,n)]$。构造算法中，验证集为 Taillard 30×20 的公开实例。模型策略中的图神经网络均为 2 层。动作选择网络 MLP_{θ_π} 和状态动作值预测网络 MLP_{θ_ϕ} 均为 3 层，隐藏层维度为 512。对于 PPO，设置每 20 个轮次更新一次网络，裁剪系数为 0.2；损失函数由策略损失、值函数损失和熵损失 3 部分构成，权重分别为 0.1，1，0.1。在训练过程中，折扣系数 γ 设置为 0.95，使用 Adam 优化器，学习率为 2.5×10^{-4}。我们使用 Pytorch Genmetric 库[51]中的代码在 Pytorch 中进行了实验。训练使用的机器 CPU 为 Inter Xeon E5-2620，一张 Nvidia Geforce Titian XP。

为研究本文所提框架对各种问题的泛化能力，首先在生成数据集上对所提算法进行评估。每个数据集中含有 100 个实例。为验证所提方法的有效性，本文选取以下方法作为基线。

CP-SAT[26]：具有 1 h 限制的谷歌公开求解器。

MWKR：剩余时间最长的作业（Most Work Remaining，MWKR）。

MOR：剩余操作数量最多（Most Operations Remaining，MOR）的作业。

FDD/MWKR：作业流开始的时间与已经调度工序的操作时间和剩余加工时间之和的比率最小的作业（Minimum radio of Flow Due Date to Most Work Remaining，FDD/MWKR）。

除上述方法外，本文还考虑了强化学习（RL）和 MWKR 两种初始化（Init）方法以及不同的搜索（Search）步数约束

(1 000，3 000，5 000 和 10 000)对结果的影响。本文根据初始化方法和搜索步数的不同进行对比实验，$Init_{\omega}+Search_{T}$ 表示使用 ω 方法得到初始解并基于初始解使用蒙特卡洛搜索机制搜索 T 次。此外，为了验证图关系编码的有效性，我们将边关系置空进行相同的训练和测试，使用修正算法修正 5 000 次，结果用 $Our_{\text{no edges}}$ 表示。表 2 列出了所有方法在生成的不同规模数据集 100 个实例上的最大完成时间的均值。

为验证模型有效性，将本文方法与传统的 PDR 规则如 MWKR，FDD/MWKR，MOR 以及 3 种精心设计的构造强化学习算法[9-11]进行对比验证，表 3－表 5 列出了不同算法在公开数据集上的结果。这里我们同样去除图关系进行对比实验。opt 为目前已知的最优解的完成时间[1)]。

表 2　不同算法在不同规模生成数据实例上的平均完成时间

Table 2　Average completion time of different algorithms on different sizes of generated data instances

Algorithm	6×6	10×10	15×15	20×20	30×15	30×20	50×20	100×20
CP-SAT	487.95	817.50	1 181.95	1 568.07	1 786.36	1 946.57	2 871.45	5 496.38
MWKR	643.47	1188.77	1 820.8	2 513.96	2 909.47	3 249.80	4 649.44	8 020.85
MOR	596.93	1073.41	1 651.60	2 283.50	2 527.02	2 864.27	3 999.05	6 686.03
FDD/MWKR	606.32	1 127.20	1 738.40	2 385.14	2 776.41	3 106.25	4 504.58	7 799.47
$Init_{\text{MWKR}}+Search_{1\,000}$	510.63	919.95	1 447.31	2 031.64	2 332.79	2 632.90	3 905.64	7 029.41
$Init_{\text{MWKR}}+Search_{3\,000}$	508.53	901.51	1 393.54	1 927.29	2 213.22	2 527.36	3 718.01	6 736.62
$Init_{\text{MWKR}}+Search_{5\,000}$	504.71	892.44	1 371.85	1 901.1	2 174.79	2 460.09	3 626.68	6 615.61
$Init_{\text{MWKR}}+Search_{10\,000}$	505.78	886.47	1 359.87	1 862.89	2 126.01	2 396.75	3 519.01	6 430.50
$Init_{\text{RL}}$	536.11	965.92	1 426.11	1 917.07	2 175.27	2 395.19	3 393.26	5 932.60
$Init_{\text{RL}}+Search_{1\,000}$	505.10	904.09	1 366.27	1 856.94	2 126.17	2 355.14	3 363.28	5 914.29
$Init_{\text{RL}}+Search_{3\,000}$	502.18	890.86	1 347.35	1 831.19	2 102.00	2 326.45	3 342.18	5 899.89
$Init_{\text{RL}}+Search_{5\,000}$	501.87	882.92	1 339.87	1 824.11	2 085.89	2 310.76	3 328.20	5 892.94
$Ours_{\text{no edges}}$	506.07	891.71	1 364.26	1 854.72	2 113.39	2 355.48	3 369.03	5 924.59
$Init_{\text{RL}}+Search_{10\,000}$	**501.04**	**879.68**	**1 330.48**	**1 809.18**	**2 072.21**	**2 292.69**	**3 309.24**	**5 881.80**

表 3　不同算法在 Taillard 数据实例上的平均完成时间

Table 3　Average completion time of different algorithms in Taillard data instances

Algorithm	15×15	20×15	20×20	30×15	30×20	50×15	50×20	100×20
Zhang 等[9]	1 547.4	1 774.7	2 128.1	2 378.8	2 603.9	3 393.8	3 593.9	6 097.6
Park 等[10]	1 476.2	1 703.7	2 090.3	2 230.4	2 570.2	3 214.6	3 452.2	5 862.8
Park 等[11]	1 417.2	1 630.0	1 896.0	**2 129.5**	2 411.3	**3 157.4**	**3 229.4**	**5 723.6**
MWKR	1 896.1	2 178,9	2 633.9	2 985.9	3 265.7	4 444.9	4 677.4	8 040.3
FDD/MWKR	1 808.0	2 109.6	2 508.0	2 918,8	3 235,0	4 294.0	4 477.4	7 836.6
MOR	1 735.7	1 998.5	2 334.8	2 567.6	3 010.4	3 714.1	4 123.6	6 728.8
$Init_{\text{RL}}$	1 473.2	1 702.4	1 980.3	2 227.3	2 454.4	3 341.3	3 434.2	6 009.6
$Init_{\text{RL}}+Search_{1\,000}$	1 422.5	1 653.6	1 929.4	2 201.0	2 429.3	3 287.7	3 424.1	5 967.4
$Init_{\text{RL}}+Search_{3\,000}$	1 412.2	1 632.2	1 903.1	2 183.9	2 413.1	3 276.2	3 399.6	5 950.1
$Ours_{\text{no edges}}$	1 420.1	1 645.4	1 924.8	2 207.4	2 440.3	3 263.9	3 447.2	6 025.6
$Init_{\text{RL}}+Search_{5\,000}$	**1 389.6**	**1 614.0**	**1 887.2**	2 162.2	**2 396.8**	3 256.0	3 377.6	5 945.6
opt	1 228.0	1 364.6	1 588.0	1 788.0	1 948.0	2 773.8	2 843.9	5 351.2

表 4　不同算法在 ABZ，Ft，Orb，Yn 数据实例上的平均完成时间

Table 4　Average completion time of different algorithms in ABZ，Ft，Orb，Yn data instances

Algorithm	Abz10×10	Abz20×15	Ft6×6	Ft10×10	Ft20×5	Orb10×10	Yn20×20	Swv20×15	Swv50×10
Park 等[10]	1 198.0	859.33	71	1 143	**1 338**	1 100.9	1 139.00	**2 174.4**	**3 415.1**
Park 等[11]	1 158.5	**803.33**	59	1 111	1 498	1 087.0	1 081.50	2 196.6	3 669.8
MWKR	1 369.5	1 047.67	74	1 289	2 020	1 446.2	1 408.25	3 141.0	5 163.5
FDD/MWKR	1 356.5	967.67	70	1 422	1 983	1 475.2	1 336.00	2 970.6	4 887.1
MOR	1 322.0	987.00	60	1 319	1 672	1 391.6	1 296.00	2 738.4	4 340.4
$Init_{\text{RL}}+Search_{5\,000}$	**1 138.0**	807.33	**57**	**1 099**	1 516	**1 052.7**	**1 051.50**	2 268.2	3 773.0
Opt	1 088.5	666.33	55	930	1 165	902.8	912.00	1 683.0	2 915.3

最后，由于在不同机器上消耗的时间不同，无法进行更精确的时间对比，本文仅对步数进行限制，以达到限制运行时间的目的。

在生成数据集上，为了观察初始解对最终解决方案质量的影响程度，我们使用 3 种 PDR 规则中质量最差的 MWKR 提供初始解。从表 2 可以看到，尽管修正算法对给定解决方案都有明显的修正提升效果，但解的质量和步数的限制依旧对整体性能影响较大。此外，为了验证本文提出的图关系建模的有效性，我们将图关系删除，使用相同配置进行训练和实例预测调度，由表 2 和表 3 可以看到，去除图关系后，结果较原来有明显的下降。这也证明了图表示建模的有效性。

由表 2－表 5 可以看出，在各种公开数据实例上，本文算法在大部分中小规模实例上的效果超越了目前最好的构造强化学习框架。然而，在其他规模实例上，其效果较差，可能的原因有以下 3 个方面：1）初始解的质量太差，次优选择更多，

1) http://jobshop.jjvh.nl/

因此需要更多步数的修正;2)实例越大,越可能陷入局部最优解,单纯的蒙特卡洛可能无法预测到实际应修改的位置,需要更多的试错和训练;3)修正算子交换机器上相邻工序的操作顺序,具体请见 4.2.1 小节的交换过程。实际上,对于操作顺序位置所导致的问题,只能通过接受相同操作质量的解进行一定的缓解,不能得到根本解决。最后,从不同步数的修正结果来看,我们的修正搜索机制可以采用一种较为简单的方式稳步提高小规模实例解的质量。

表 5 不同算法在 La 数据实例上的平均完成时间

Table 5 Average completion time of different algorithms in La data instances

Algorithm	La10×5	La15×5	La20×5	La10×10	La15×10	La20×10	La30×10	La15×15
Park 等[10]	719.4	927.4	1207.8	1011.6	1197.8	1569.2	1905.0	1531.6
Park 等[11]	694.6	941.2	1226.0	966.8	1127.6	1428.2	**1848.0**	1466.6
MWKR	865.0	1206.2	1414.4	1260.0	1537.2	1867.6	2597.6	1936.6
FDD/MWKR	854.4	1190.0	1365.6	1168.8	1413.8	1740.4	2743.0	1807.8
MOR	804.0	1039.0	1306.8	1116.0	1407.2	1730.4	2311.4	1869.4
$Init_{RL}+Search_{5000}$	**674.2**	**930.0**	**1199.0**	**930.6**	**1087.0**	**1420.2**	1898.4	**1436.6**
Opt	620.0	917.6	1182.0	864.2	983.4	1235.2	1782.4	1263.2

表 6 交换算子有效性证明实验结果

Table 6 Experimental results for proving validity of exchange operator

	Ta15×15	Ta20×15	Ta20×20	Ta30×15	Ta30×20
Zhang 等[9]	1530.5	1772.4	2081.8	2318.3	2598.4
Zhang 等[9]+$Search_{5000}$	1461.8	1685.4	2016.1	2216.9	2519.7
Ours	**1389.6**	**1614.0**	**1887.2**	**2162.2**	**2396.8**

为说明蒙特卡洛修正机制对构造式强化学习框架的修正作用,我们使用文献[9]中公开的代码和模型在数据集为 Taillard15×15,20×15,20×20,30×15,30×20 的 50 个实例上进行实验,修正次数为 5000。从表 6 可以看到,文献[9]中提出的构造框架确实存在较多的选择次优现象,导致其完成时间较长,在进行次优选择修正后,完成时间明显缩短。

为进一步说明本文提出的交换算子的有效性,我们对交换过程进行可视化展示,如图 5 所示。图中呈现了一个 JSSP 5×5 实例在交换前后的状态。具体地,矩形内数字为所属作业,黑色虚线方框内部分为交换的工序。从图 5(a)和图 5(b)可以看出,交换操作可以有效修正策略学习中的次优选择,使得相关的工序开始时间发生变化,完成时间缩短。需要注意的是,即使是替换前后完成时间不变的交换,也是一种有效修正,它使得当前解缓慢逃离局部最优解。

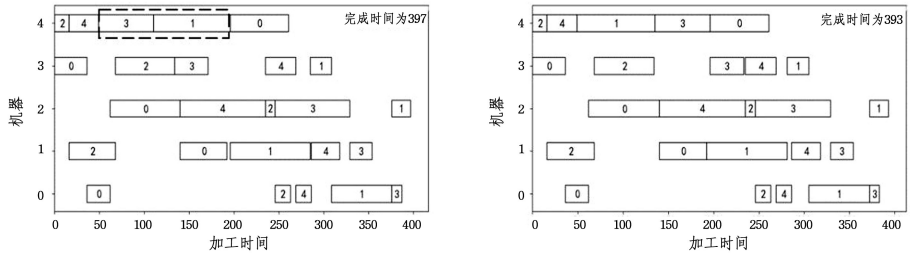

图 5 蒙特卡洛交换算子可视化

Fig. 5 Visualisation of Monte Carlo exchange operators

结束语 本文提出了一种基于强化学习的调度器和一种有效的搜索修正机制,以提高 JSSP 问题解的质量。在生成的数据集和公开数据集上进行了性能评估,实验结果表明,本文所建立的框架是一种有效的通用调度方法,可以采用一种相对有效的方式来解决 JSSP 问题,在中小规模实例上取得了较好的结果。本文算法在较大规模实例上表现一般,在未来的研究工作中,将对这方面进行深入研究。

参 考 文 献

[1] GLOVER F, LANGUNA M. Tabu search[M]// Handbook of combinatorial optimization. Boston: Springer, 1998: 2093-2229.

[2] SILVER D, HUBERT T, SCHRITTWIESER J, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play[J]. Science, 2018, 362(6419): 1140-1144.

[3] VINYALS O, BABUSCHKIN I, CZARNECKI W M, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning[J]. Nature, 2019, 575(7782): 350-354.

[4] BERNER C, BROCKMAN G, CHAN B, et al. Dota 2 with large scale deep reinforcement learning[J]. arXiv: 1912.06680, 2019.

[5] GAREY M R, JOHNSON D S, SETHI R. The complexity of flowshop and jobshop scheduling[J]. Mathematics of Operations Research, 1976, 1(2): 117-129.

[6] GIFFLER B, THOMPSON G L. Algorithms for solving production-scheduling problems[J]. Operations Research, 1960, 8(4): 487-503.

[7] HAUPT R. A survey of priority rule-based scheduling [J]. Operations-Research-Spektrum, 1989, 11(1): 3-16.

[8] GEIGER C D, UZSOY R, AYTUĞ H. Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach[J]. Journal of Scheduling, 2006, 9(1): 7-34.

[9] ZHANG C, SONG W, CAO Z, et al. Learning to dispatch for job shop scheduling via deep reinforcement learning[J]. Advances in Neural Information Processing Systems, 2020, 33: 1621-1632.

[10] PARK J，CHUN J，KIM S H，et al. Learning to schedule job-shop problems：representation and policy learning using graph neural network and reinforcement learning[J]. International Journal of Production Research，2021，59(11)：3360-3377.

[11] PARK J，BAKHTIYAR S，PARK J. Schedu-leNet：Learn to solve multi-agent scheduling problems with reinforcement learning[J]. arXiv：2106.03051，2021.

[12] TASSEL P，GEBSER M，SCHEKOTIHIN K. A reinforcement learning environment for job-shop scheduling[J]. arXiv：2104.03760，2021.

[13] MONACI M，AGASUCCI V，GRANI G. An actor-critic algorithm with deep double recurrent agents to solve the job shop scheduling problem[J]. arXiv：2110.09076，2021.

[14] BOGYRBAYEVA A，MERALIYEV M，MUS-TAKHOV T，et al. Learning to Solve Vehicle Routing Problems：A Survey[J]. arXiv：2205.02453，2022.

[15] XIN L，SONG W，CAO Z，et al. Generative Adversarial Training for Neural Combinatorial Optimization Models[J/OL]. https：//xs2. zidianzhan. net/scholar? hl＝zh-CN&as_sdt＝0%2C5&q＝Generative＋Adversarial＋Training＋for＋Neural＋Combinatorial＋Optimization＋Models&btnG＝.

[16] KOOL W，VAN HOOF H，WELLING M. Att-ention，learn to solve routing problems! [J]. arXiv：1803.08475，2018.

[17] SCHULMAN J，WOLSKI F，DHARIWAL P，et al. Proximal policy optimization algorithms[J]. arXiv：1707.06347，2017.

[18] BENGIO Y，LODI A，PROUVOST A. Machine learning for combinatorial optimization：a methodological tour d'horizon[J]. European Journal of Operational Research，2021，290(2)：405-421.

[19] JOSHI C K，LAURENT T，BRESSON X. An efficient graph convolutional network technique for the travelling salesman problem[J]. arXiv：1906.01227，2019.

[20] LU H，ZHANG X，YANG S. A learning-based iterative method for solving vehicle routing problems[C]// International Conference on Learning Representations. 2019.

[21] ROY A，SAFFAR M，VASWANI A，et al. Efficient content-based sparse attention with routing transformers[J]. Transactions of the Association for Computational Linguistics，2021，9：53-68.

[22] ZHOU J，CUI G，HU S，et al. Graph neural networks：A review of methods and applications[J]. AI Open，2020，1：57-81.

[23] DWIVEDI V P，JOSHI C K，LAURENT T，et al. Benchmarking graph neural networks[J]. arXiv：2003.00982，2020.

[24] KHALIL E，DAI H，ZHANG Y，et al. Learning combinatorial optimization algorithms over graphs[J]. Advances inNeural Information Processing Systems，2017，30：6348-6358.

[25] XU K，HU W，LESKOVEC J，et al. How powerful are graph neural networks? [J]. arXiv：1810.00826，2018.

[26] PERRON L，FURNON V. Or-tools. UR-L [OL]. https：//developers. google. com/optimization.

[27] ZENG Y，LIAO Z，DAI Y，et al. Hybrid intelligence for dynamic job-shop scheduling with deep reinforcement learning and attention mecha-nism[J]. arXiv：2201.00548，2022.

[28] MONACI M，AGASUCCI V，GRANI G. An actor-critic algorithm with deep double recurrent agents to solve the job shop scheduling problem[J]. arXiv：2110.09076，2021.

[29] HOCHREITER S，SCHMIDHUBER J. Long short-term memory[J]. Neural computation，1997，9(8)：1735-1780.

[30] NI F，HAO J，LU J，et al. A Multi-Graph Attributed Reinforcement Learning Based Optimization Algorithm for Large-Scale Hybrid Flow Shop Scheduling Problem[C]// Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2021：3441-3451.

[31] NAWAZ M，ENSCORE JR E E，HAM I. A heuristic algorithm for the m-machine，n-job flow-shop sequencing problem[J]. Omega，1983，11(1)：91-95.

[32] OREN J，ROSS C，LEFAROV M，et al. SOLO：search online，learn offline for combinatorial optimization problems[C]// Proceedings of the International Symposium on Combinatorial Search. 2021：97-105.

[33] WANG R，HUA Z，LIU G，et al. A bi-level framework for learning to solve combinatorial optimization on graphs[J]. Advances in Neural Information Processing Systems，2021，34：21453-21466.

[34] XIN L，SONG W，CAO Z，et al. Step-wise deep learning models for solving routing problems[J]. IEEE Transactions on Industrial Informatics，2020，17(7)：4861-4871.

[35] KIPF T N，WELLING M. Semi-supervised classification with graph convolutional networks[J]. arXiv：1609.02907，2016.

[36] VELIČKOVIĆ P，CUCURULL G，CASA-NOVA A，et al. Graph attention networks[J]. arXiv：1710.10903，2017.

[37] HE K，GKIOXARI G，DOLLÁR P，et al. Mask r-cnn[C]// Proceedings of the IEEEInternational Conference on Computer Vision. 2017：2961-2969.

[38] SCHULMAN J，LEVINE S，ABBEEL P，et al. Trust region policy optimization[C]// International Conference on Machine Learning. PMLR，2015：1889-1897.

[39] ENGSTROM L，ILYAS A，SANTURKAR S，et al. Implementation matters in deep policy gradients：A case study on PPO and TRPO[J]. arXiv：2005.12729，2020.

[40] BROWNE C B，POWLEY E，WHITEHOUSE D，et al. A survey of monte carlo tree search methods[J]. IEEE Transactions on Computational Intelligence and AI in games，2012，4(1)：1-43.

[41] COULOM R. Efficient selectivity and backup operators in Monte-Carlo tree search[C]// International Conference on Computers and Games. Berlin：Springer，2006：72-83.

[42] 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.

[43] SILVER D，SCHRITTWIESER J，SIMONYAN K，et al. Mastering the game of go without human knowledge[J]. Nature，2017，550(7676)：354-359.

[44] TAILLARD E. Benchmarks for basic scheduling problems[J]. European Jurnal of Operational Research，1993，64(2)：278-285.

[45] ADAMS J, BALAS E, ZAWACK D. The shifting bottleneck procedure for job shop scheduling[J]. Management Science, 1988, 34(3): 391-401.

[46] FISHER H. Probabilistic learning combinations of local job-shop scheduling rules[J/OL]. https://xs2.zidianzhan.net/scholar?hl=zh-CN&as_sdt=0%2C5&q=Probabilistic+learning+combinations+of+local+job-shop+scheduling+rules&btnG=#d=gs_cit&t=1681371135544&u=%2Fscholar%3Fq%3Dinfo%3ATfo-0odqynIJ%3Ascholar.google.com%2F%26output%3Dcite%26scirp%3D0%26hl%3Dzh-CN.

[47] LAWRENCE S. Resouce constrained project scheduling: An experimental investigation of heuristic scheduling techniques (Supplem-ent)[J/OL]. https://xs2.zidianzhan.net/scholar?hl=zh-CN&as_sdt=0%2C5&q=Resouce+constrained+project+scheduling%3A+An+experimental+investigation+of+heuristic+scheduling+techniques+&btnG=#d=gs_cit&t=1681373202282&u=%2Fscholar%3Fq%3Dinfo%3AXvOT-Lo5cCigJ%3Ascholar.google.com%2F%26output%3Dcite%26scirp%3D0%26hl%3Dzh-CN.

[48] STORER R H, WU S D, VACCARI R. New search spaces for sequencing problems with application to job shop scheduling[J]. Manag-ement Science, 1992, 38(10): 1495-1509.

[49] YAMADA T, NAKANO R. A genetic algorithm applicable to large-scale job-shop problems[C]//PPSN. 1992: 281-290.

[50] APPLEGATE D, COOK W. A computational study of the job-shop scheduling problem[J]. ORSA Journal on Computing, 1991, 3(2): 149-156.

[51] FEY M, LENSSEN J E. Fast Graph Representation Learning with PyTorch Geometric[J]. arXiv: 1903.02428, 2019.

MIAO Kuan, born in 1998, postgraduate, is a member of China Computer Federation. His main research interests include reinforcement learning and combinatorial optimisation.

LI Chongshou, born in 1988, Ph.D, associate professor, is a member of China Computer Federation. His main research interests include multi-scale data intelligence, AI and applied optimization.

（责任编辑：杨雪敏）