

控制与决策

Control and Decision



基于深度强化学习与迭代贪婪的流水车间调度优化

王凌, 潘子肖

引用本文:

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.

在线阅读 View online: <https://doi.org/10.13195/j.kzyjc.2020.0608>

您可能感兴趣的其他文章

Articles you may be interested in

基于多班教学优化的多目标分布式混合流水车间调度

Multi-class teaching-learning-based optimization for multi-objective distributed hybrid flow shop scheduling

控制与决策. 2021, 36(2): 303–313 <https://doi.org/10.13195/j.kzyjc.2020.0549>

区间数可重入混合流水车间调度与预维护协同优化

Collaborative optimization of interval number reentrant hybrid flow shop scheduling and preventive maintenance

控制与决策. 2021, 36(11): 2599–2608 <https://doi.org/10.13195/j.kzyjc.2020.0973>

带峰值能耗约束流水线调度的协同群智能优化

Cooperative memetic optimization for flowshop scheduling with peak power consumption constraint

控制与决策. 2021, 36(10): 2350–2358 <https://doi.org/10.13195/j.kzyjc.2020.0429>

基于机床超低待机状态的流水车间能耗调度

Energy consumption scheduling in flow shop based on ultra-low idle state of numerical control machine tools

控制与决策. 2021, 36(1): 143–151 <https://doi.org/10.13195/j.kzyjc.2019.0433>

基于改进蛙跳算法的分布式两阶段混合流水车间调度

An improved shuffled frog leaping algorithm for the distributed two-stage hybrid flow shop scheduling

控制与决策. 2021, 36(1): 241–248 <https://doi.org/10.13195/j.kzyjc.2019.0472>

基于深度强化学习与迭代贪婪的流水车间调度优化

王 凌[†], 潘子肖

(清华大学 自动化系, 北京 100084)

摘 要: 流水车间调度是应用背景最为广泛的调度问题,其智能算法研究具有重要的学术意义和应用价值. 以最小化最大完工时间为目标,提出求解流水车间调度的一种基于深度强化学习与迭代贪婪算法的框架. 首先,设计一种新的编码网络对问题进行建模,解决了传统模型受问题规模影响而难以扩展的缺陷,并利用强化学习训练模型以获取优良输出结果;然后,提出一种带反馈机制的迭代贪婪算法,以网络的输出结果为初始解,协同利用多种局部操作提高搜索能力,并根据性能反馈调节各操作的使用,进而获得最终的调度解. 仿真结果和统计对比表明,所提出的深度强化学习与迭代贪婪融合的算法能够取得更好的性能.

关键词: 流水车间调度; 深度强化学习; 迭代贪婪算法; 反馈协同机制

中图分类号: TP18

文献标志码: A

DOI: 10.13195/j.kzyjc.2020.0608

开放科学(资源服务)标识码(OSID):



引用格式: 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.

Scheduling optimization for flow-shop based on deep reinforcement learning and iterative greedy method

WANG Ling[†], PAN Zi-xiao

(Department of Automation, Tsinghua University, Beijing 100084, China)

Abstract: As the scheduling problem with wide application backgrounds, the research of intelligent algorithms for flow-shop scheduling is of important academic significance and application value. With the criterion of minimizing the maximum completion time, a framework is proposed based on deep reinforcement learning and the iterative greedy method for solving the permutation flow-shop scheduling. Firstly, a new encoding network is designed to model the problem to avoid the defect in generalizing the classic model affected by problem scale, and the reinforcement learning is used to train the model to yield good output result. Then, an iterative greedy algorithm with feedback mechanism is proposed by using the output result of the trained model as the initial solution. Multiple local search operators are conducted in a collaborative way and adjusted their utilizations according to the feedback of performances for obtaining the final schedule. Simulation results and statistical comparisons show that the proposed algorithm fusing deep reinforcement learning and the iterative greedy method is able to achieve better performances.

Keywords: flow-shop scheduling; deep reinforcement learning; iterative greedy method; feedback and collaboration mechanism

0 引 言

流水车间调度是最为经典的生产调度问题,应用背景极为广泛,其高效求解算法的研究一直是相关领域的重要课题^[1-2]. 2015年,国务院发布《中国制造2025》阐明智能制造是主攻方向,强调人工智能技术在制造业中的推广应用^[3]. 因此,如何利用人工智能技术提出高效的生产调度算法具有重要的现实意义.

目前,流水车间调度的求解方法主要分为两大类^[4-5]: 精确算法,譬如分支定界、数学规划;近似算

法,譬如启发式算法和智能优化算法. 由于流水车间调度问题的NP-难特性,精确算法难以求解大规模问题. 启发式算法虽能快速构造调度方案,但缺乏优化的全局性. 同时,对于复杂工况下的调度问题,在缺失问题性质的挖掘时也很难设计出高效的启发式算法. 代表性启发式算法有CDS (campbell-dudek-smith) 算法^[6]、Gupta 算法^[7]和NEH (nawaz-enscore-ham) 算法^[1]等,其中NEH被公认为是最有效和最常用的启发式算法.

收稿日期: 2020-05-22; 修回日期: 2020-07-08.

基金项目: 国家杰出青年科学基金项目(61525304); 国家自然科学基金项目(61873328).

[†]通讯作者. E-mail: wangling@tsinghua.edu.cn.

鉴于精确算法和启发式算法的不足,近些年智能优化算法得到了发展. 针对置换流水车间调度问题(permutation flow-shop scheduling problem, PFSP), Liu等^[8]设计了粒子群优化与分布估计的混合算法以最小化最大完成时间; Pan等^[9]和Liu等^[10]分别设计了离散差分进化和人工蜂群算法; 郑晓龙等^[11]设计了混合离散果蝇算法, 通过四阶段迭代搜索取得优良的性能; Ruiz等^[12]设计了一种融合NEH的迭代贪婪算法(IG_{RS}), 通过与多种算法的对比验证了IG_{RS}的优良性能; 秦旋等^[13]结合群智能算法与局部搜索策略, 设计了一种混合共生生物搜索算法(hybrid symbiotic organisms search, HSOS). 对于PFSP的诸多扩展问题, 智能优化算法也取得了很好的效果. 针对分布式流水线调度问题, 王凌等^[14]提出了一种混合离散果蝇优化算法, 采用多操作协同的方式平衡全局探索与局部开发能力. 针对无等待流水线调度问题, 宋存利等^[15]设计了一种邻域迭代搜索算法. 针对低碳车间调度问题, 雷德明^[16]提出了一种新型教学优化算法以同时优化总碳排放与平均延迟时间. 针对混合流水车间调度问题, 王圣尧等^[17]设计了基于概率模型采样的分布估计算法.

尽管智能优化算法在诸多调度问题上取得了满意的性能, 但其迭代搜索过程通常比较耗时, 而且许多算法很少利用历史信息来调整搜索行为, 因此在大规模问题上的性能仍有很大提升空间. 针对特定的调度问题, 挖掘问题的性质, 提出相应的初始化启发式方法, 设计调整搜索行为的反馈策略, 合理利用历史信息自适应调整算法的搜索模式, 都是提升算法性能的可行途径.

随着人工智能的发展, 近些年深度学习在自然语言处理^[18]、模式识别^[19]等领域得到了成功应用, 但在组合优化领域的研究还很少^[20]. Vinyals等^[21]提出了Pointer网络, 利用循环神经网络(recurrent neural network, RNN)和Attention机制, 解决序列到序列的建模问题, 并用于求解旅行商问题(traveling salesman problem, TSP). Ling等^[22]通过建立全卷积网络, 实现从问题到最优解的映射, 在中小规模问题上取得了不错的结果. 遗憾的是, Pointer网络和全卷积网络都需要通过有监督的方式训练模型, 训练效果严重依赖于标签的质量.

区别于有监督学习, 强化学习(reinforcement learning, RL)具有很强的决策优化能力, 不需要预先设定任何标签, 通过接受环境对动作的反馈获得学习信息并更新模型参数. Bello等^[23]将Pointer网

络与强化学习技术结合, 克服了Pointer网络对高质量标签数据的依赖性. 针对车辆路径问题(vehicle routing problem, VRP), Nazari等^[24]改进Pointer网络, 采用策略梯度算法进行参数优化, 在中规模问题上取得优于OR-Tools的结果. Kool等^[25]则采用Attention机制和强化学习技术取得了较好的性能. 因此, 将深度学习与强化学习结合形成深度强化学习(deep reinforcement learning, DRL), 可自主挖掘问题的特征, 积累问题信息并进行决策优化, 有助于设计面向特定问题的有效优化方法.

考虑到Pointer网络所建模型受问题规模影响而难以扩展, 本文针对PFSP设计一种新的编码网络对问题建模, 采用DRL训练模型获得初始调度解, 进而提出一种带反馈协同机制的迭代贪婪算法(iterated greedy, IG)得到最终的优良调度方案. 仿真结果和统计对比表明, 所提出的深度强化学习与迭代贪婪融合的算法能够取得更好的性能.

1 调度问题描述

PFSP^[1]考虑有 n 个工件 $J = \{J_1, J_2, \dots, J_n\}$ 在 m 台机器 $M = \{M_1, M_2, \dots, M_m\}$ 上加工过程, 各工件 J_i 均需以相同的顺序通过 m 个机器完成相应的操作 $\{O_{i1}, O_{i2}, \dots, O_{im}\}$, 其中 O_{ij} 表示工件 J_i 在机器 M_j 上的加工操作. 已知工件 J_i 在机器 M_j 上的加工时间 p_{ij} , 需确定所有工件的合理加工顺序 $\pi = (\pi_1, \pi_2, \dots, \pi_n)$, 使得所有工件的最大完成时间 $C_{\max}$ 最小.

通常, PFSP约定如下假设^[1]: 1) 所有工件相互独立且在零时刻均可加工; 2) 每台机器在同一时刻只能加工一个工件; 3) 每个工件均需在每台机器上加工且只加工一次; 4) 所有工件在各台机器上的加工顺序相同; 5) 在每台机器上工件一旦开始加工则不能中断; 6) 工件在不同机器间的运输和准备时间忽略或包含在工件的加工时间内.

调度指标 $C_{\max}$ 的计算过程如下:

$$C_{\max} = \max\{C(\pi_i, j)\},$$

$$i = 1, 2, \dots, n, j = 1, 2, \dots, m; \quad (1)$$

$$C(\pi_1, 1) = p_{\pi_1 1}, \quad (2)$$

$$C(\pi_i, 1) = C(\pi_{i-1}, 1) + p_{\pi_i 1}, i = 2, 3, \dots, n; \quad (3)$$

$$C(\pi_1, j) = C(\pi_1, j-1) + p_{\pi_1 j},$$

$$j = 2, 3, \dots, m; \quad (4)$$

$$C(\pi_i, j) = \max\{C(\pi_{i-1}, j), C(\pi_i, j-1)\} + p_{\pi_i j},$$

$$i = 2, 3, \dots, n, j = 2, 3, \dots, m. \quad (5)$$

其中 $C(\pi_i, j)$ 表示工件 π_i 在机器 j 上的完成时间。

作为数据驱动的求解技术,智能优化算法并不直接利用问题的数学模型进行求解,而是采用一定的方式表征解,并通过迭代搜索的方式得到优良的解。对于智能算法,表征解的方式称为编码,生成具体调度方案并计算调度指标的过程称为解码。在此,采用调度串 $\pi = (\pi_1, \pi_2, \dots, \pi_n)$ 进行编码,其中 π_i 表示工件序号, $\pi_i \in \{J_1, J_2, \dots, J_n\}, \forall i \neq j, \pi_i \neq \pi_j$ 。

至于解码,在此采用两种方式,在不同的环节采用不同的解码方式。解码方式1直接按照调度串中的工件排列顺序安排加工;解码方式2参考NEH方法,根据调度串中工件的顺序依次选取各个工件并插入到合适位置,进而确定使得最大完成时间最小的调度方案。

2 基于DRL与IG的求解框架

融合深度强化学习和迭代贪婪方法的算法框架(DRL_IG)如图1所示。

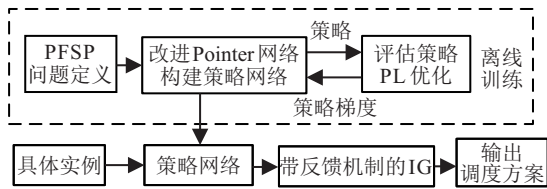


图1 DRL_IG算法框架

首先,根据PFSP的定义,采用改进的Pointer网络对问题进行建模,构建策略网络;然后,评估策略网络生成的策略,并利用基于策略梯度的RL指导策略网络参数的优化。上述训练均采用离线方式,在求解具体问题时,采用训练好的策略网络产生工件的初始排序,再利用带反馈机制的IG通过迭代搜索获得最终的调度解。下面逐一介绍基于DRL的建模与参数优化以及带反馈机制的IG。

2.1 基于DRL的建模与参数优化

深度学习用于构建策略网络,输入置换流水车间问题 s 的数据,输出一个表征调度方案的工件序列;强化学习用于优化网络参数。

2.1.1 基于深度学习的网络模型

Vinyals等^[21]结合Attention机制设计了求解二维TSP的Pointer网络,编码网络输入为一个二维向量。调度问题规模由工件数与机器数决定,编码网络的输入维数依赖于机器数,因问题不同而不同,导致Pointer网络不能直接应用于调度问题。考虑到工件在不同机器上的加工时间直接影响调度过程,采用双RNN结构搭建编码网络。

编码网络主要分为两部分,第1部分是针对加工

时间 p_{ij} 的建模,将工件 i 在所有机器上的加工时间编码转换成一个固定维数的向量 p_i ;第2部分是对所有 p_i 进行建模, $i = 1, 2, \dots, n$,进而编码转换成向量 P 。

第1部分针对所有工件的 $p_{ij} (j = 1, 2, \dots, m)$ 的建模流程如下。其中: W_B, h_{ij}, h_0 为 k 维向量, h_0, U, W, b 和 W_B 为网络参数, $f(\cdot)$ 为RNN输入到隐层输出的映射。另外,RNN之间共享网络参数,使其能够处理输入长度可变的问题。

step 1: p_{ij} 经过嵌入层可得 $y_{ij} = W_B p_{ij}$ 。

step 2: 将 y_{i1} 与 h_0 输入第1个RNN,得到隐层输出 $h_{i1} = f(y_{i1}, h_0; U, W, b)$ 。

step 3: 依次将 $h_{i,j-1}$ 与 y_{ij} 输入第1个RNN,得到隐层输出 $h_{ij} = f(y_{ij}, h_{i,j-1}; U, W, b), j = 2, 3, \dots, m$,最终获得 $p_i = h_{im}$ 。

第2部分针对所有 $p_i (i = 1, 2, \dots, n)$ 的建模流程如下。其中 $\tilde{h}_i$ 为 d 维向量, $\tilde{h}_0, \tilde{U}, \tilde{W}, \tilde{b}$ 为网络参数。

step 1: 将 p_1 与 $\tilde{h}_0$ 输入第2个RNN,得到隐层输出 $\tilde{h}_1 = f(p_1, \tilde{h}_0; \tilde{U}, \tilde{W}, \tilde{b})$ 。

step 2: 将 $\tilde{h}_{i-1}$ 与 p_i 输入第2个RNN,得到隐层输出 $\tilde{h}_i = f(p_i, \tilde{h}_{i-1}; \tilde{U}, \tilde{W}, \tilde{b}), i = 2, 3, \dots, n$,最终获得 $P = \tilde{h}_n$ 。

采用RNN和Attention机制构建解码网络,生成工件序列 $\pi = (\pi_1, \pi_2, \dots, \pi_n)$,流程如下,其中 $\langle g \rangle, \tilde{U}, \tilde{W}, \tilde{b}, v, W_1, W_2$ 为待训练参数。

step 1: 将 $\langle g \rangle$ 和 P 输入解码网络RNN,得到隐层输出 $d_1 = f(\langle g \rangle, P; \tilde{U}, \tilde{W}, \tilde{b})$,其中 $\langle g \rangle$ 为 k 维参数向量,表示解码网络开始的输入。

step 2: 计算 $p(\pi_1 | \pi_{<1}, s) = \text{soft max}(u^1), u_j^1 = v^T(W_1 \tilde{h}_j + W_2 d_1), j = 1, 2, \dots, n, \tilde{h}_j$ 为第2个编码RNN的隐层输出,采用轮盘赌方式确定 π_1 。

step 3: 将 p_{π_1} 和 d_1 输入解码RNN,得到隐层输出 $d_2 = f(p_{\pi_1}, P; \tilde{U}, \tilde{W}, \tilde{b})$,类似step 2 计算 $p(\pi_2 | \pi_{<2}, s)$,其中 $u_{\pi_1}^2 = -\infty$ 。然后采用轮盘赌方式确定 π_2 。

step 4: 重复上述过程,直至产生一个完整的工件序列 π 。

以5个工件、 m 台机器的PFSP为例,图2显示了搭建的网络结构。首先,对5个工件的加工信息依次建模,得到5个 k 维向量 $p_i (i = 1, 2, \dots, 5)$,进而将 $p_i (i = 1, 2, \dots, 5)$ 依次输入第2个RNN得到 P ;然后,将 P 与 $\langle g \rangle$ 一起输入解码网络,得到 d_1 和 $p(\pi_1 | \pi_{<1}, s)$,采用轮盘赌方式确定 π_1 (假设 $\pi_1 = 4$);接着,将 p_4 与 d_1 一起输入得到 d_2 和 $p(\pi_2 | \pi_{<2}, s)$,并令 $p(\pi_2 = 4 | \pi_1 = 4, s) = 0$,采用轮盘赌方式确定 π_2 (假设 $\pi_2 = 5$);依据上述方式,最终获得工件序列 $\pi = (4, 5, 1, 2, 3)$,即一个编码后的调度方案。

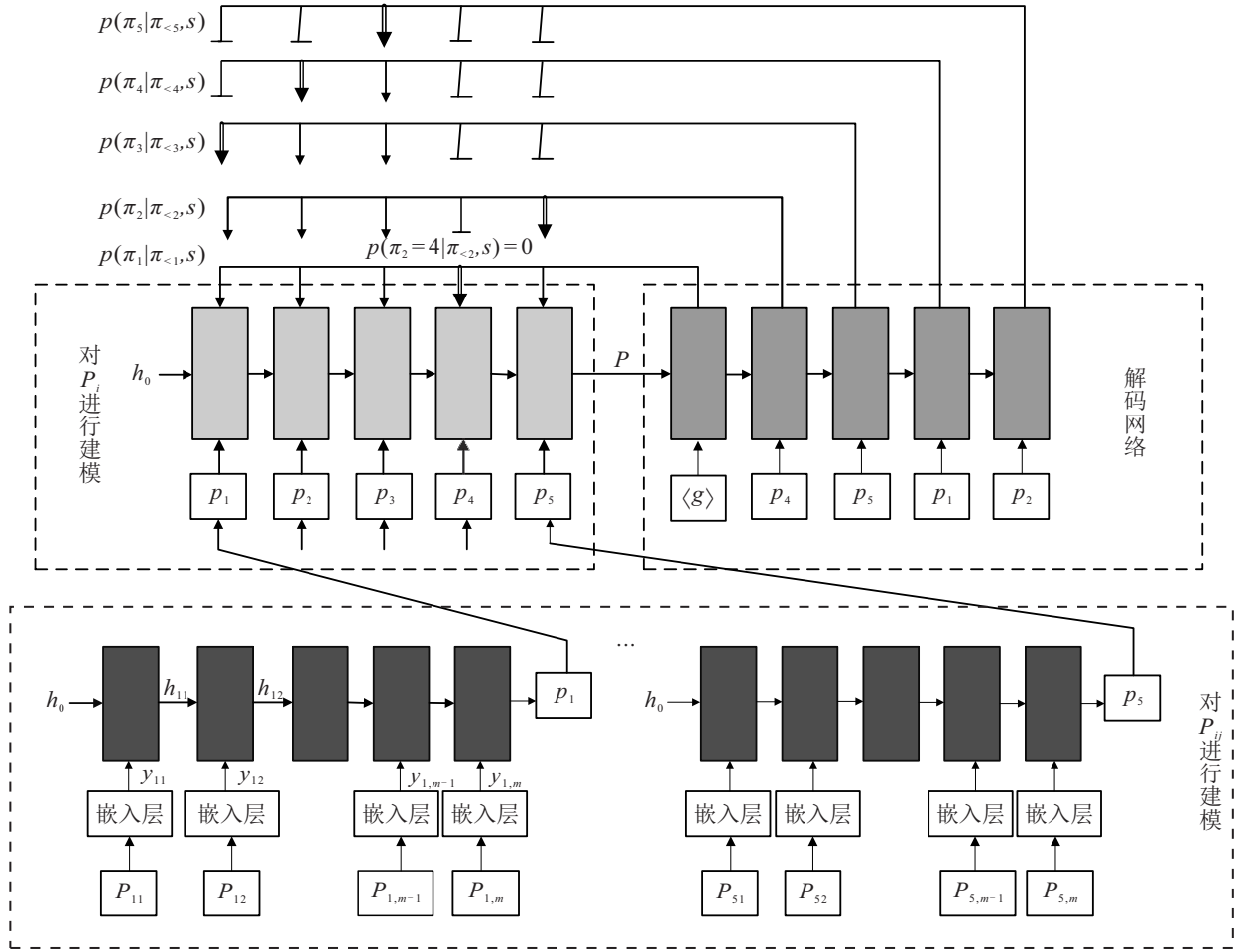


图2 网络结构

2.1.2 基于强化学习的参数优化

为了避免对高质量标签的依赖性,采用强化学习的策略梯度算法训练策略网络的所有参数 θ .

假设某调度问题 s 的加工时间服从分布 S ,记为 $s \sim S$,策略网络生成的工件序列 π 服从分布 $\pi \sim p(\cdot|s)$,欲最小化的训练目标函数 $J(\theta)$ 为

$$J(\theta) = E_{s \sim S} J(\theta|s). \quad (6)$$

对于调度问题 s ,网络输出的加工序列 $C_{\max}$ 的期望为

$$J(\theta|s) = E_{\pi \sim p_{\theta}(\cdot|s)} C_{\max}(\pi|s). \quad (7)$$

其中 $C_{\max}(\pi|s)$ 为针对调度问题 s 网络输出的加工序列 π 采用解码方式2得到的 $C_{\max}$.

根据策略梯度,式(7)的梯度为

$$\nabla_{\theta} J(\theta|s) = E_{\pi \sim p(\cdot|s)} [(C_{\max}(\pi|s) - b(s)) \nabla_{\theta} \log p_{\theta}(\pi|s)]. \quad (8)$$

其中: $p_{\theta}(\pi|s) = \prod_{i=1}^n p_{\theta}(\pi_i | \pi_{<i}, s)$; $b(s)$ 为一个不随 π 变化的基准线,表示对问题 s 的 $C_{\max}$ 的估计,用于减小训练方差.

假设每次迭代从分布 S 中采样生成 B 个独立同

分布的 s_i ,即 $s_1, s_2, \dots, s_B \sim S$,针对 s_i 采样生成工件加工序列 $\pi^i \sim p_{\theta}(\cdot|s_i)$,则可获得目标函数关于 θ 的近似梯度为

$$\nabla_{\theta} J(\theta) \approx \frac{1}{B} \sum_{i=1}^B (C_{\max}(\pi^i|s_i) - b(s)) \nabla_{\theta} \log p_{\theta}(\pi^i|s_i), \quad (9)$$

其中基准线 $b(s) = \frac{1}{B} \sum_{i=1}^B C_{\max}(\pi^i|s_i)$. 进而,由式(9)给出的梯度采用Adam优化算法^[26]对网络参数进行迭代优化.

2.2 带反馈协同机制的IG算法

作为一种简单的迭代搜索算法,IG首先对解进行初始化,然后不断通过解构、重构、局部搜索等操作改进解,并利用贪婪策略接受新解,一旦达到最大迭代次数则输出结果^[12]. 对于PFSP,现有的IG算法常以NEH的结果作为初始解,并且通常只采用一种局部搜索操作^[12,27]. 在此,一方面考虑采用DRL训练后的网络输出结果作为初始调度,另一方面考虑IG算法中多种搜索操作的协同运作并根据性能反馈调节各操作的使用.

2.2.1 算法初始化与解构重构

NEH是求解PFSP最有效的启发式方法之一,用来产生IG的初始解^[12,27]. NEH构造调度的原理根据每个工件在所有机器上的加工时间之和的降序而确定,但该机制并不能对所有问题都获得优良调度. 本文考虑利用DRL训练好的策略网络产生工件序列,为IG算法提供初始解 x^* ,同时采用解码方式2获得调度并评价其性能.

IG算法解构环节的实现方式如下:从解 x^* 中随机抽取 d 个工件,余下 $n-d$ 个工件保持相对顺序不变组成序列 π_D ,然后将抽取出的 d 个工件随机排序组成序列 π_R .

重构环节的实现方式如下:依次从序列 π_R 中取出工件并插入序列 π_D 中的合适位置,使得插入后调度子序列的最大完成时间最小;一旦序列 π_R 中所有工件都插入序列 π_D 则完成重构,得到新解 x^{new} .

2.2.2 局部搜索与接受准则

文献[28]表明,多种搜索操作的协同有利于提升优化性能. 为了丰富IG的局部搜索行为,并提升求解不同问题时的搜索能力,在此考虑3种局部搜索操作协同运作模式,同时根据操作的性能反馈调节其使用概率. 3种局部搜索操作如下:

1) 插入操作(LS1). 在新解 x^{new} 中,随机选择一个工件并将其插入到余下工件序列的任意可能位置,从中选择调度指标最优的方案.

2) 交换操作(LS2). 在新解 x^{new} 中,随机选择两个工件并交换其位置.

3) 逆序操作(LS3). 在新解 x^{new} 中,随机选择两个工件并将两者之间的工件序列逆序.

每一代执行局部搜索时,采用基于概率选择的方式确定局部搜索操作. 设 $p_{1,g}$ 、 $p_{2,g}$ 、 $p_{3,g}$ 分别为第 g 代搜索选择插入、交换、逆序操作的概率,在每一代搜索后按下式进行更新:

$$p_{1,g} = \frac{\text{num}_1}{\text{num}_1 + \text{num}_2 + \text{num}_3}, \quad (10)$$

$$p_{2,g} = \frac{\text{num}_2}{\text{num}_1 + \text{num}_2 + \text{num}_3}, \quad (11)$$

$$p_{3,g} = 1 - p_{1,g} - p_{2,g}, \quad (12)$$

其中 num_1 、 num_2 、 num_3 为各操作相应的计数变量,一开始均设为1.

假设第 g 次选择LS1,若产生的新解性能有提高,则令 $\text{num}_1 = \text{num}_1 + 1$,而 num_2 和 num_3 保持不变;若产生的新解性能没有提高,则令 $\text{num}_i = \text{num}_i + 0.5$, $i = 2, 3$, num_1 保持不变. 如果选择其他操作,则计数变量的更新方式类似. 按照上述方式在算法后

期总迭代步数很大,如果某个操作长期占优势将导致其 num_i 很大,则计数变量较小的变化很难改变相应的选择概率. 为了避免上述现象,每次计算完 num_i 后进行判断,若存在 $\text{num}_i \geq 100$,则将所有计数量减半,即 $\text{num}_i = 0.5 \times \text{num}_i$, $i = 1, 2, 3$.

另外,为了避免搜索过程陷入局部最优解,采用概率突跳接受准则^[12]. 如果局部搜索产生的新解 x^{new} 优于旧解 x^{old} ,则直接替换,否则通过判断 $\text{random} \leq \exp\{-(C(x^{\text{new}}) - C(x^{\text{old}}))/\text{tem}\}$ 决定是否替换,其中 random 为 $(0, 1)$ 之间均匀分布的随机数, tem 按下式计算, T 为算法参数:

$$\text{tem} = T \times \frac{\sum_{i=1}^m \sum_{j=1}^n p_{ij}}{n \times m \times 10}. \quad (13)$$

3 仿真测试和性能对比

3.1 实验设置

采用python实现DRL环节并离线训练模型,采用Matlab实现IG环节并迭代搜索优良调度,运行环境为Intel Core i7 CPU/2.8 GHz主频.

DRL环节采用51 200个规模为 $n_m = 50_30$ 的问题训练模型,即50个工件、30台机器,加工时间为服从 $(0, 1)$ 之间均匀分布的随机数,模型离线训练48 h. 然后采用国际标准Taillard集进行调度优化算法的性能测试,考虑20_5、20_10、20_20、50_5、50_10、50_20、100_5、100_10、100_20、200_10、200_20和500_20共12种不同规模. 与文献[13]相同,在Taillard测试集里选取每一规模下10个算例中的最后一个算例进行性能测试,测试时需将所有加工数据归一化后输入策略网络. 同时,采用相对百分偏差(relative percentage deviation,RPD)作为算法的评价指标,有

$$\text{RPD} = (C_{\max}^{\text{alg}} - C_{\max}^*) / C_{\max}^* \times 100. \quad (14)$$

其中: $C_{\max}^*$ 为算例已知最佳 $C_{\max}$, $C_{\max}^{\text{alg}}$ 为算法所得 $C_{\max}$.

3.2 参数设置

在模型训练环节,DRL每次采样数(batch size)设为128,RNN采用长短时记忆(long short-term memory,LSTM)网络,网络隐层特征维数和嵌入层参数维数均设为256,参数优化采用Adam算法^[26],学习率设为 1×10^{-5} ,网络初始参数为 $[-0.08, 0.08]$ 之间均匀分布的随机数.

在IG搜索环节,本文一律设置最大迭代次数为8 000. 首先考察两个核心参数对算法性能的影响,包括解构过程抽取工件数 d 和接收准则参数 T . 采用50_20规模算例进行正交实验,每个参数设置3个水

平值,如表1所示. 根据正交表 $L_9(3^2)$,在每组参数组合下独立运行算法10次,算法所得RPD的均值作为响应值(response value, RV),结果如表2所示.

表1 参数的水平取值

参数	水平		
	1	2	3
d	4	6	8
T	0.2	0.3	0.4

表2 实验方案及结果

实验编号	d	T	RV
1	1	1	2.386
2	1	2	2.348
3	1	3	2.223
4	2	1	1.994
5	2	2	1.906
6	2	3	2.252
7	3	1	2.103
8	3	2	1.941
9	3	3	2.207

计算各参数在不同水平下的平均响应值,进而确定参数对性能影响的等级,如表3所示. 各参数的影响趋势如图3所示. 可见, d 对算法性能的影响较 T 大. 根据实验结果的对比,最佳参数组合为 $d = 6, T = 0.3$.

表3 各参数平均RV

水平	d	T
1	2.319	2.161
2	2.051	2.065
3	2.084	2.228
极差	0.268	0.162
等级	1	2

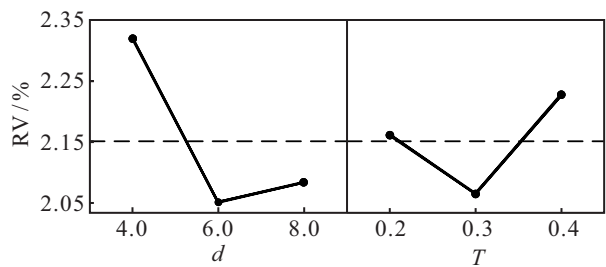


图3 各参数对算法性能影响趋势

3.3 初始化策略和反馈协同机制的有效性检验

DRL_IG利用DRL训练策略网络的输出获得初始解,然后采用多种操作进行协同搜索并根据性能反馈调节各操作的使用情况. 为了验证初始化策略、反馈协同机制的有效性,构造DRL_IG的两种变体进行性能对比,变体1(记作NEH_IG)采用NEH为IG提供初始解,变体2(DRL_IG2)采用等概率方式选择局部操作进行IG搜索. 3种算法参数设置相同,对每个实

例均运行10次,表4为3种算法所得RPD的均值和方差. 同时,按95%的置信度对3种算法所得结果进行非参数配对样本检验,统计分析结果如表5所示.

表4 DRL_IG与两种变体RPD的均值和方差

n_m	$C_{\max}^*$	DRL_IG		NEH_IG		DRL_IG2	
		均值	方差	均值	方差	均值	方差
20_5	1 108	0.000	0.000	0.000	0.000	0.000	0.000
20_10	1 591	0.603	0.111	0.629	0.051	0.880	0.133
20_20	2 178	0.230	0.050	0.298	0.050	0.294	0.025
50_5	2 782	0.000	0.000	0.000	0.000	0.000	0.000
50_10	3 065	1.155	0.086	1.308	0.222	1.223	0.149
50_20	3 756	1.816	0.184	2.396	0.413	2.210	0.096
100_5	5 322	0.000	0.000	0.011	0.001	0.060	0.003
100_10	5 845	0.332	0.102	0.546	0.168	0.765	0.103
100_20	6 434	1.728	0.091	1.932	0.064	2.031	0.066
200_10	10 675	0.383	0.026	0.408	0.028	0.408	0.028
200_20	11 288	2.041	0.044	2.519	0.033	2.173	0.066
500_20	26 457	0.919	0.006	0.865	0.003	0.976	0.014
平均统计		0.767	0.058	0.909	0.086	0.918	0.057

表5 统计分析结果

配对检验	p 值	配对检验	p 值
(DRL_IG, NEH_IG)	0.017	(DRL_IG, IG _{RS})	0.007
(DRL_IG, DRL_IG2)	0.005	(DRL_IG, HSOS)	0.003
(DRL_IG, IG _{all})	0.005		

由表4结果可见,相比NEH_IG, DRL_IG对12个算例中的9个算例取得了优于NEH_IG的结果,对2个算例的结果相同. 由于最后一个算例的规模远大于训练规模,这可能是导致DRL_IG的结果稍差于NEH_IG的原因,未来将围绕模型训练开展进一步的改进研究. 另外,方差数据的对比表明, DRL_IG多次运行结果的一致性优于NEH_IG. 因此,利用策略网络的输出作为IG的初始解,能够取得更有效、更一致的调度解.

由表4结果可见,相比DRL_IG2, DRL_IG对12个算例中的10个算例取得了优于DRL_IG2的结果,对2个算例的结果相同. 同时,方差数据的对比表明DRL_IG多次运行结果的一致性与DRL_IG2几乎一样. 因此,利用反馈协同机制执行IG搜索,能够取得更有效的调度解.

由表5统计分析结果可见,相应 p 值小于0.05,这表明DRL_IG的性能显著优于NEH_IG和DRL_IG2,再次验证了初始化策略和反馈协同策略的有效性. 另外,图4显示了DRL_IG求解20_5、50_20、100_20、200_20算例时各局部搜索操作的选择概率的变化过程. 可见,在小规模算例上LS1的使用相对一直占优,而在中大规模算例上不同操作的使用变化很大,清晰地显示了各操作的使用随性能变化而自适应调节的过程.

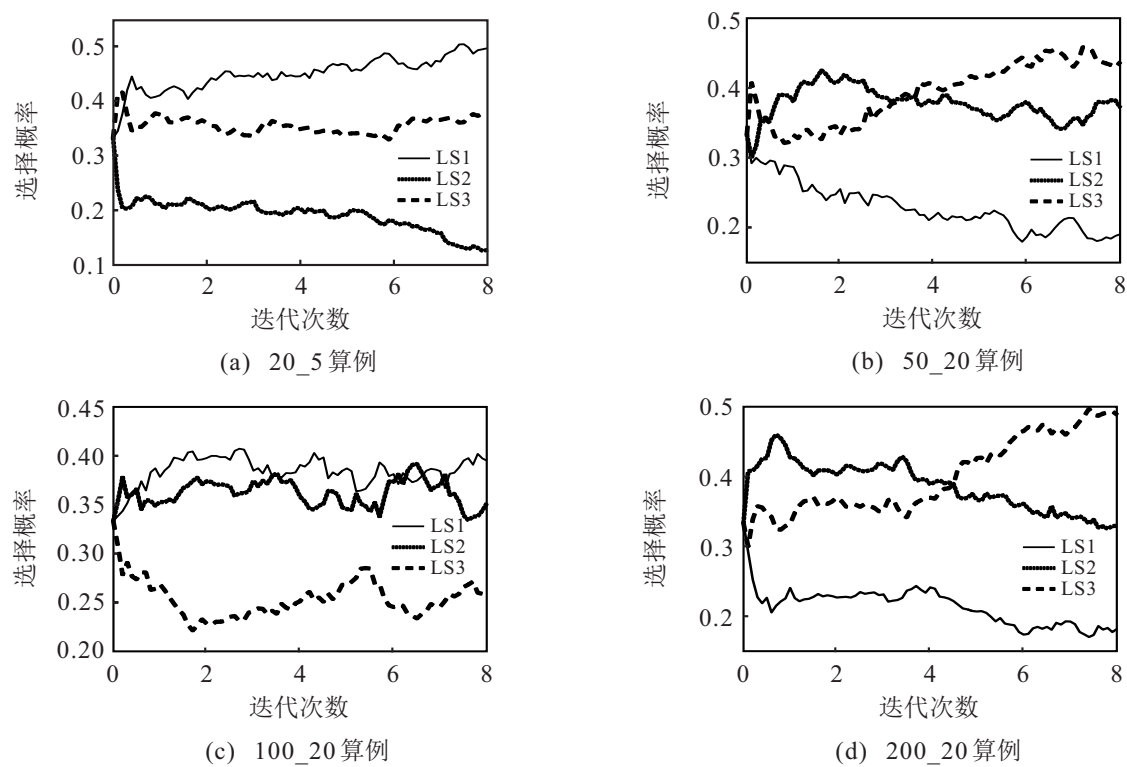


图 4 各局部搜索操作选择概率的变化曲线

3.4 与其他算法的性能对比

下面将带反馈协同机制的IG算法与求解PFSP的经典算法IG_{RS}^[12]、IG_{all}^[27]和HSOS^[13]进行性能对比,其中IG_{all}为近期提出的改进IG算法,HSOS为一种融合局部搜索的群智能优化算法。IG_{all}和IG_{RS}的参数设置与相应文献一致,DRL_IG、IG_{RS}和IG_{all}

均以8000次迭代为终止条件;HSOS的参数设置和终止准则与原文一致,即种群规模50,进化代数3000。每种算法对每个算例均运行10次,4种算法求解12个算例所得RPD的均值和方差如表6所示。按95%的置信度采用非参数配对样本检验结果见表5,4种算法的平均运行时间如表7所示。

表 6 4种算法RPD的均值和方差

<i>n_m</i>	DRL_IG		IG _{all}		IG _{RS}		HSOS	
	均值	方差	均值	方差	均值	方差	均值	方差
20_5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
20_10	0.603	0.111	0.647	0.044	0.597	0.088	2.307	0.324
20_20	0.230	0.050	0.404	0.081	0.546	0.089	4.077	0.709
50_5	0.000	0.000	0.000	0.000	0.007	0.001	0.252	0.000
50_10	1.155	0.086	2.157	0.067	2.254	0.020	4.992	0.505
50_20	1.816	0.184	2.431	0.128	2.785	0.257	6.171	0.310
100_5	0.000	0.000	0.038	0.006	0.000	0.000	0.357	0.000
100_10	0.332	0.102	0.635	0.114	0.992	0.000	0.992	0.000
100_20	1.728	0.091	2.069	0.026	2.070	0.066	3.371	0.060
200_10	0.383	0.026	0.521	0.019	0.448	0.015	1.099	0.002
200_20	2.041	0.044	2.426	0.043	2.554	0.054	4.912	0.015
500_20	0.919	0.006	1.094	0.013	1.138	0.013	1.906	0.002
平均统计	0.767	0.058	1.035	0.045	1.116	0.050	2.536	0.161

表7 4种算法的平均运行时间 s

<i>n</i> _{<i>m</i>}	DRL_IG	IG _{all}	IG _{RS}	HSOS
20_5	2.25	2.56	2.26	10.27
20_10	2.65	3.11	2.63	14.69
20_20	3.43	3.98	3.48	19.60
50_5	7.27	8.35	7.05	19.18
50_10	9.43	11.12	9.29	26.83
50_20	14.38	16.86	14.23	41.30
100_5	19.23	22.22	18.75	35.67
100_10	27.45	33.72	27.12	54.19
100_20	46.94	55.95	45.95	91.55
200_10	87.61	108.01	87.30	135.42
200_20	172.12	191.63	167.79	226.00
500_20	972.28	1070.14	960.19	1 024.00
平均统计	113.75	127.30	112.27	141.56

由表6可见,对12个算例DRL_IG在8个算例上取得了比3种对比算法都好的结果,在所有算例上的性能都好于IG_{all}和HSOS. 由于IG_{RS}仅采用LS1,而LS1对小规模算例效率较高,DRL_IG对20_10算例略劣于IG_{RS}. 同时,方差数据表明,DRL_IG多次运行结果的一致性优于HSOS,与IG_{all}和IG_{RS}相近. 由表5的配对样本检验结果可见,DRL_IG的性能显著优于其余3种算法. 另外,由表7的运行时间对比可见,DRL_IG的平均运行时间与IG_{all}和IG_{RS}相近,因此反馈协同机制对算法运行时间的影响很小. 同时,DRL_IG的平均运行时间比群智能HSOS算法要小很多,相比文献中的群智能算法可以在更短的时间内获得更好的调度解.

4 结 论

本文提出了求解流水线调度问题的一种基于深度强化学习与迭代贪婪算法的框架,通过数值仿真和性能对比验证了所提出算法的有效性. 研究工作的创新之处在于:基于深度强化学习设计了一种新的编码网络,可适合不同规模的调度问题;采用网络输出提供初始解,有利于获得大规模调度问题更好的性能;提出了一种带反馈协同机制的迭代贪婪算法,有利于充分利用多种局部搜索操作并根据性能变化动态调节各操作的使用. 未来的研究工作将侧重于模型的训练以及利用深度强化学习求解其他类型的调度问题,并将深度强化学习与群体智能优化算法合理融合探讨提升性能的高效学习与反馈机制.

参考文献(References)

[1] Nawaz M, Enscore E E, Ham I J. A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem[J]. Omega, 1983, 11(1): 91-95.

[2] Liu W B, Jin Y, Price M. A new improved NEH heuristic for permutation flowshop scheduling problems[J]. International Journal of Production Economics, 2017, 193(1): 21-30.

[3] 周济. 智能制造是“中国制造2025”主攻方向[J]. 企业观察家, 2019, 11(1): 54-55.
(Zhou J. Intelligent manufacturing is the main direction of “made in China 2025” [J]. Enterprise Observer, 2019, 11(1): 54-55.)

[4] Framinan J M, Gupta J N D, Leisten R. A review and classification of heuristics for permutation flow-shop scheduling with makespan objective[J]. Journal of Operational Research Society, 2004, 55(12): 1243-1255.

[5] Fernandez-Viagas V, Ruiz R, Framinan J M. A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation[J]. European Journal of Operational Research, 2017, 257(3): 707-721.

[6] Campbell H G, Dudek R A, Smith M L. A heuristic algorithm for the *n* job, *m* machine sequencing problem[J]. Management Science, 1970, 16(10): 630-637.

[7] Gupta J N D. A functional heuristic algorithm for the flow-shop scheduling problem[J]. Journal of the Operational Research Society, 1971, 22(1): 39-47.

[8] Liu H C, Gao L, Pan Q K. A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem[J]. Expert Systems with Applications, 2011, 38(4): 4348-4360.

[9] Pan Q K, Tasgetiren M F, Liang Y C. A discrete differential evolution algorithm for the permutation flowshop scheduling problem[J]. Computers & Industrial Engineering, 2008, 55(4): 795-816.

[10] Liu Y F, Liu S Y. A hybrid discrete artificial bee colony algorithm for permutation flowshop scheduling problem[J]. Applied Soft Computing, 2013, 13(3): 1459-1463.

[11] 郑晓龙, 王凌, 王圣尧. 求解置换流水线调度问题的混合离散果蝇算法[J]. 控制理论与应用, 2014, 31(2): 159-164.
(Zheng X L, Wang L, Wang S Y. A hybrid discrete fruit fly optimization algorithm for solving permutation flow-shop scheduling problem[J]. Control Theory & Applications, 2014, 31(2): 159-164.)

[12] Ruiz R, Stützle, T. A simple and effective iterated greedy

- algorithm for the permutation flowshop scheduling problem[J]. *European Journal of Operational Research*, 2007, 177(3): 2033-2049.
- [13] 秦旋, 房子涵, 张赵鑫. 混合共生生物搜索算法求解置换流水车间调度问题[J]. *浙江大学学报: 工学版*, 2020, 54(4): 712-721.
(Qin X, Fang Z H, Zhang Z X. Hybrid symbiotic organisms search algorithm for permutation flow shop scheduling problem[J]. *Journal of Zhejiang University: Engineering Science*, 2020, 54(4): 712-721.)
- [14] 王凌, 郑洁, 王晶晶. 求解区间数分布式流水线调度的混合离散果蝇优化算法[J]. *控制与决策*, 2020, 35(4): 930-936.
(Wang L, Zheng J, Wang J J. A hybrid discrete fruit fly optimization algorithm for distributed permutation flowshop scheduling with interval data[J]. *Control and Decision*, 2020, 35(4): 930-936.)
- [15] 宋存利, 刘晓冰, 王伟. 大规模无等待流水调度问题的邻域迭代搜索算法[J]. *控制与决策*, 2011, 26(4): 535-539.
(Song C L, Liu X B, Wang W. An iterative neighborhood search algorithm for large scale no-wait flow-shop[J]. *Control and Decision*, 2011, 26(4): 535-539.)
- [16] 雷德明. 基于新型教学优化算法的低碳柔性作业车间调度[J]. *控制与决策*, 2017, 32(9): 1621-1627.
(Lei D M. Novel teaching-learning-based optimization algorithm for low carbon scheduling of flexible job shop[J]. *Control and Decision*, 2017, 32(9): 1621-1627.)
- [17] 王圣尧, 王凌, 许烨, 等. 求解混合流水车间调度问题的分布估计算法[J]. *自动化学报*, 2012, 38(3): 437-443.
(Wang S Y, Wang L, Xu Y, et al. An estimation of distribution algorithm for solving hybrid flow-shop scheduling problem[J]. *Acta Automatica Sinica*, 2012, 38(3): 437-443.)
- [18] Tom Y, Devamanyu H, Soujanya P, et al. Recent trends in deep learning based natural language processing[J]. *IEEE Computational Intelligence Magazine*, 2018, 13(3): 55-75.
- [19] Lateef F, Ruichek Y. Survey on semantic segmentation using deep learning techniques[J]. *Neurocomputing*, 2019, 338(1): 321-348.
- [20] Mirshekarian S, Sormaz D. Machine learning approaches to learning heuristics for combinatorial optimization problems[J]. *Procedia Manufacturing*, 2018, 17(1): 102-109.
- [21] Vinyals O, Fortunato M, Jaitly N. Pointer networks[C]. *The 29th Conference on Neural Information Processing Systems*. Montreal: IEEE, 2015: 2692-2700.
- [22] Ling Z X, Tao X Y, Zhang Y, et al. Solving optimization problems through fully convolutional networks: An application to the traveling salesman problem[J]. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019, 10: 1-11.
- [23] Bello I, Pham H, Le Q V, et al. Neural combinatorial optimization with reinforcement learning[C]. *The 5th International Conference on Learning Representations*. Toulon: IEEE, 2017: 1-13.
- [24] Nazari M, Oroojlooy A, Taká M, et al. Reinforcement learning for solving the vehicle routing problem[C]. *The 32nd Conference on Neural Information Processing Systems*. Montreal: IEEE, 2018: 9839-9849.
- [25] Kool W, Van Hoof H, Welling M. Attention, learn to solve routing problems![C]. *The 7th International Conference on Learning Representations*. New Orleans: IEEE, 2019: 1-12.
- [26] Kingma D P, Ba J L. Adam: A method for stochastic optimization[C]. *The 3rd International Conference on Learning Representations*. San Diego: IEEE, 2015: 1-11.
- [27] Dubois-Lacoste J, Pagnozzi F, Stützle T. An iterated greedy algorithm with optimization of partial solutions for the makespan permutation flowshop problem[J]. *Computers & Operations Research*, 2017, 81(1): 160-166.
- [28] Zheng X L, Wang L. A collaborative multiobjective fruit fly optimization algorithm for the resource constrained unrelated parallel machine green scheduling problem[J]. *IEEE Transactions on System, Man, and Cybernetics: Systems*, 2018, 48(5): 790-800.

作者简介

王凌(1972—), 男, 教授, 博士生导师, 从事智能优化调度理论方法、复杂生产过程建模优化与调度等研究, E-mail: wangling@tsinghua.edu.cn;

潘子肖(1997—), 男, 博士生, 从事智能优化调度方法的研究, E-mail: pzx19@mails.tsinghua.edu.cn.

(责任编辑: 郑晓蕾)