计算机科学
COMPUTER SCIENCE

基于双层注意力网络的强化学习方法求解柔性作业车间调度问题

王皓焱, 李崇寿, 李天瑞

引用本文

王皓焱, 李崇寿, 李天瑞. 基于双层注意力网络的强化学习方法求解柔性作业车间调度问题[J]. 计算机科学, 2026, 53(1): 231-240.

WANG Haoyan, LI Chongshou, LI Tianrui. Reinforcement Learning Method for Solving Flexible Job Shop Scheduling Problem Based onDouble Layer Attention Network [J]. Computer Science, 2026, 53(1): 231-240.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.250100088

基于双层注意力网络的强化学习方法求解柔性作业车间调度问题

王皓焱　李崇寿　李天瑞
西南交通大学计算机与人工智能学院　成都 611756
(19983459395@163.com)

摘　要　柔性作业车间调度问题作为作业车间调度问题的一种变体,因其广泛的适用性成为现代制造业智能化转型中的重要研究内容。近年来,深度强化学习被用于求解柔性作业车间调度问题,但允许将操作分配给具有不同处理时间的多台兼容机器的特点给决策和状态表示带来了额外的复杂性。为此,提出了一种基于改进的注意力机制和近端策略优化算法的端到端深度强化学习框架,用于解决柔性作业车间调度问题。基于异构析取图结构的特点,设计了一种基于分层注意力思想的双层注意力网络,包括节点级注意力层与类型级注意力层,充分提取操作与机器间的复杂信息,以支持高质量的调度决策。在合成数据集和公开数据集上的实验结果表明,所提方法在保持高效率的同时,性能和泛化能力均优于传统的优先调度规则方法和目前先进的深度强化学习方法。

关键词:柔性作业车间调度问题;深度强化学习;图注意力网络;注意力机制
中图分类号　TP181

Reinforcement Learning Method for Solving Flexible Job Shop Scheduling Problem Based on Double Layer Attention Network

WANG Haoyan,LI Chongshou and LI Tianrui
School of Computing and Artificial Intelligence,Southwest Jiaotong University,Chengdu 611756,China

Abstract　Flexible job shop scheduling problem(FJSP),as a variant of the job shop scheduling problem,has become an important research topic in the intelligent transformation of modern manufacturing industry due to the wide applicability. In recent years,deep reinforcement learning(DRL) has been applied to solve flexible job shop scheduling problems. However,the characteristic that operations can be assigned to multiple compatible machines with different processing times brings additional complexity to decision making and state representation. This paper proposes an end-to-end deep reinforcement learning framework based on an improved attention mechanism and proximal policy optimization algorithm to solve the FJSP. Considering the characteristics of heterogeneous disjunction graph structure,it designs a double-layer attention network based on hierarchical attention,including node-level attention layers and type-level attention layers,to fully extract the complex information between operations and machines to support high-quality scheduling decisions. Experimental results on synthetic and public datasets show that the proposed method outperforms traditional priority dispatching rules and currently state-of-the-art DRL methods in both of performance and generalization ability while maintaining high efficiency.

Keywords　Flexible job shop scheduling problem,Deep reinforcement learning,Graph attention network,Attention mechanism

1　引言

人工智能、大数据、云计算、物联网等新一代信息技术的不断发展和应用,推动了工业的第四次革命,即工业4.0。这引入了一种新的制造范式——智能制造。在制造业的智能化转型浪潮中,作为提升生产效率、优化资源配置的关键环节,作业车间调度问题(Job-Shop Scheduling Problem,JSSP)正日益受到学术界和工业界的广泛关注。在现代制造环境中,由于需求的多样化和生产过程中的不确定性,调度灵活性成为关键。柔性作业车间调度问题(Flexible Job-Shop Scheduling Problem,FJSP)在JSSP问题的基础上,允许在多台不同的机器上处理操作,更适合处理新制造模式中灵活多样的任务-资源关系[1],同时也意味着相比于JSSP问题,FJSP问题更具挑战,拥有更复杂的拓扑结构和更大的解决方案空间[2]。

JSSP问题是经典的NP-Hard组合优化问题,FJSP问题比JSSP问题更加复杂,因为属于作业的操作可以分配给具有不同处理时间的多台兼容机器。数学规划和约束规划等精确算法[3-5]是求解组合优化问题的经典方法。它在整个解空

到稿日期:2025-01-14　返修日期:2025-03-29
基金项目:国家自然科学基金(62202395,62176221)
This work was supported by the National Natural Science Foundation of China(62202395,62176221).
通信作者:李崇寿(lics@swjtu.edu.cn)

间中寻找最优解，但当问题规模扩大时，往往需要大量的计算成本，因此难以在合理时间内解决大型调度问题。在实际生产中，常使用启发式方法和元启发式方法来解决现实问题。启发式算法是一类基于直觉和经验的建设性算法[6]，优先调度规则(Priority Dispatching Rule，PDR)[7-9]作为典型的启发式算法，由于计算复杂度较低且易于实现，被广泛应用于实时调度系统中，它通过定义优先级规则，迭代地选择操作和机器。然而，设计有效的 PDR 通常需要大量的专业知识和反复实验[10]，并且由于启发式规则的短视性，往往难以保证最优解[11]，同时也缺乏适应不同调度问题和应用场景的能力。元启发式算法，如群体智能算法(Swarm Intelligence，SI)[12-15]、进化算法(Evolutionary Algorithms，EA)[16-18]等，作为另一种流行的启发式范式，也被广泛应用于调度问题。这些方法采用复杂的搜索过程来探索解空间，以找到高质量的解决方案[19]；但其并不适用于实时调度环境，因为当底层算法需要大量迭代时，它们可能需要无法预测的极长计算时间才能获得满意的解决方案[20]。

近年来，作为人工智能领域的一项重要突破，深度强化学习(Deep Reinforcement Learning，DRL)在解决复杂决策和优化问题方面展现出了巨大的潜力。DRL 通过结合深度学习的特征提取能力和强化学习的策略优化能力，能够在高维状态空间和复杂约束条件下，实现策略的自适应学习和优化，非常适用于解决 FJSP 这类需要实时决策和优化的问题。一些方法将 DRL 技术与启发式方法相结合[21]，采用不同的强化学习算法进行训练，自适应地选择调度策略。这通常可以在较短的时间内得到调度结果，并且优于普通启发式规则。然而，这种类型的强化学习本质上是在启发式搜索空间中运行，而非在解搜索空间中运行，因此解的质量仍然取决于所选择的启发式规则[11]。最近的一些工作以给定的问题实例为输入，使用训练好的深度神经网络直接输出解决方案，形成端到端的学习方法[1,2,11,20]。尽管这些基于 DRL 的方法已取得不错的成果，但如何建模马尔可夫决策过程表达，以及如何设计特征表示方案和模型架构以充分从原始调度状态中提取操作和机器的有效信息，仍然是提升方法有效性和效率的关键问题。

为了解决以上问题，本文针对标准 FJSP 问题，提出了一种新的端到端强化学习框架，旨在最大限度地缩短完成时间。首先，提出了一种适用于 FJSP 问题的马尔可夫决策过程(Markov Decision Process，MDP)公式，使用改进的异构析取图[1]来表示机器与节点间的关系，并设计了充分表征与决策相关的操作和机器相关信息的状态表示，将操作选择和机器选择视为一个整体，使操作选择和机器分配决策可以同时进行。其次，基于异构图注意力网络的思想[22]，提出了双层注意力网络(Double Layer Attention Networks，DLANs)模型，以充分提取操作与机器的复杂信息。这是一种基于分层注意力的网络架构，包括节点级注意力层和类型级注意力层。节点级注意力分别学习操作节点和机器节点与其基于不同类型的邻居节点之间的重要性，类型级注意力学习不同类型的重要性。在合成实例和公开基准上进行了广泛的实验，结果表明，在保持较高计算效率的同时，本文方法优于传统的 PDR 方法与先进的 DRL 方法，并能有效地推广到大规模实例和分布外实例中。本文的主要贡献如下：

1)构建了一种新的端到端强化学习框架，用于解决 FJSP 问题，该方法不受问题尺寸限制，可以处理不同大小的 FJSP 实例；

2)引入了适用于 FJSP 问题的 MDP 公式，使用改进的异构析取图表示机器与节点间的关系，并设计了能充分表征操作和机器信息的状态表示；

3)提出了基于分层注意力的 DLANs 模型，使用动态的注意力机制，从异构析取图中充分提取操作和机器特征，用于进行高质量的调度决策；

4)在合成数据集和公开基准数据集上进行大量实验结果表明，本文方法在保持高效率的同时，性能和泛化能力均优于传统的 PDR 方法和先进的 DRL 方法。

2 相关工作

柔性作业车间调度 FJSP 问题指在具有多个作业和资源约束的柔性制造系统中，合理安排作业任务的顺序和时间，以最大化生产效率和资源利用率的问题。该问题在工业生产中具有重要的实际意义，因此在学术界和工业界都受到广泛关注。

求解 FJSP 的传统方法可分为精确方法、启发式方法和元启发式方法[23]。基于精确求解的方法通过建立数学模型，利用优化算法求解最优解。常用的求解方法包括混合整数规划、约束规划等。Özgüven 等[4]开发了两个混合整数线性规划模型，用于解决包括路由和排序子问题的 FJSP 问题以及包括工艺计划选择子问题的具有工艺计划灵活性的 FJSP 问题。Müller 等[5]研究了 5 个约束规划求解器在 FJSP 问题上的相对性能差异，并利用由此产生的性能互补性提出了基于实例特征或参数来预测给定问题实例的最佳求解器的算法选择方法。这些基于精确求解的方法具有找到最优调度解的理论保证，但需要指数级长的计算时间。

启发式算法则是一种通过启发式规则和经验知识来指导搜索过程的算法，由于其易于实现、计算效率高，能快速构造接近最优解的解决方案，被学者广泛研究。优先调度规则 PDR 算法是一种典型的 FJSP 启发式方法。PDR 根据一些规定的规则反复选择优先级最高的操作或机器，直到生成完整的计划。Sels 等[8]提出了大量用于 JSP 的 PDR。Ortız 等[9]提出了一个新的模型用于对数学化后的 FJSP 问题进行排序。Sobeyko 等[24]则为 FJSP 问题提供了一种能够快速找到高质量解决方案的迭代局部搜索启发式方法，可以有效地应用于大型问题实例。然而，启发式方法通常不能保证找到最优解，特别是对于复杂情况下的调度问题，如果缺乏对问题本质的深入挖掘，则很难设计出高效的启发式算法。

元启发式方法是受自然界中的自然现象或生物的某些规律启发而产生的算法，由于能在合理时间内取得较优解，已成为解决 FJSP 问题的重要手段，包括遗传算法(Genetic Algorithms，GA)、禁忌搜索(Tabu Search，TS)、蚁群优化(Ant Colony Optimization，ACO)、人工蜂群(Artificial Bee Colony，ABC)等。Li 等[13]通过将遗传算法与禁忌搜索相结合，提出

了一种以最大完工时间最小化为目标的高效混合算法。Defersha等[15]开发了一种两阶段遗传算法来求解具有序列相关设置、机器发布日期和滞后时间的综合FJSP。针对多目标FJSP问题，Wang等[18]提出了一种定制的多目标进化算法，使用智能初始化方法来丰富第一代种群，并提出各种交叉算子来创造更好的后代多样性，同时采用不同的局部搜索策略来探索邻域，以获得更好的解决方案。

尽管元启发式算法在诸多车间调度问题上可以取得近似较优解，但由于在大规模问题上收敛到局部最优点的迭代搜索过程非常耗时，且算法的性能严重依赖于设计者的经验，泛化性能较差，难以实现算法的直接迁移，因此其在大规模问题上的应用还有很大的提升空间。近年来，相当多的学者利用深度强化学习DRL方法来解决复杂的调度问题，他们将调度过程建模为马尔可夫决策过程。现有的基于DRL方法的调度问题研究工作可大体分为两类：1)将DRL技术与传统的优化研究方法相结合，如启发式调度规则、元启发式算法；2)端到端的DRL学习方法，参数化神经网络模型接收关于生产环境的信息作为状态，并直接输出调度解决方案。例如，Luo等[21]针对动态FJSP问题，开发了1个DQN代理，每个重新调度点从所提出的6个复合调度规则中选择最合适的1个。然而，设计有效的调度规则是一项复杂的任务，因为这需要大量的专业知识和实验，并且针对不同FJSP问题，其性能大不相同。此外，一些研究采用DRL技术来提高FJSP问题的元启发式算法的性能。Chen等[25]使用DRL方法，在迭代过程中动态调整遗传算法的关键参数。Long等[26]针对人工蜂群算法收敛速度慢、达到局部最优的问题，提出了一种基于强化学习的改进自学习人工蜂群算法，根据RL算法智能地选择ABC算法的更新维数，提高了收敛速度和精度。

端到端的方法则基于从作业和机器的特征中提取信息性知识来学习解决FJSP实例的策略，无需手动设计调度规则，且避免了元启发式方法的大量耗时。Han等[11]提出了一种基于三维析取图的端到端深度强化学习框架，该框架由一组编码器、解码器和递归神经网络组成。然而，他们设计的基于注意力的策略网络只处理原始特征，而不考虑图结构，因此无法充分提取有效信息。Lei等[20]将FJSP问题公式化为多马尔可夫决策过程，分别使用析取图和多层感知机(Multilayer Perceptron，MLP)来学习操作和机器特征嵌入，并引入多近端策略优化算法来训练神经网络。机器的优先级仅针对选定的操作进行计算，可能会导致训练期间的探索有限。Song等[1]设计了一种异构图结构来表示调度状态，将操作和机器视为异构节点，并提出一个两阶段图神经网络来捕捉操作和机器之间的关系。Wang等[2]结合用于深度特征提取的自注意模型和用于可扩展决策的DRL的优点，提出了一种由多个相互关联的操作消息注意块和机器消息注意块组成的双注意网络来提取操作和机器特征。

综上所述，柔性车间调度问题的灵活性对有效学习机制的设计提出了两大挑战。首先，FJSP中的决策更为复杂，不仅涉及操作选择，还涉及机器分配。其次，由于操作和机器之间的复杂关系，使用神经网络对调度状态进行有效信息提取更加困难。针对这些问题，本文构建了一种新的端到端强化学习框架，提出了基于分层注意力的DLANs模型，用于充分提取操作与机器的复杂信息。

3 问题定义

标准的柔性作业车间调度问题的表述如下。给定n个作业的集合$J=\{J_1,J_2,\cdots,J_n\}$和m台机器的集合$M=\{M_1,M_2,\cdots,M_m\}$。每个作业$J_i\in J$由具有优先约束的n_i个连续操作$O_i=\{O_{i1},O_{i2},\cdots,O_{in_i}\}$组成。由$O_{ij}$表示的操作可以在满足条件的机器$M_{ij}\subseteq M$上进行处理。操作$O_{ij}$在机器$M_k$上的加工处理时间表示为$p_{ijk}$。每个操作一旦启动就不能中断，并且一台机器一次只能处理一个操作。FJSP问题研究如何通过合理的调度，将每个操作分配给一台兼容的机器，并确定其开始处理时间S_{ij}，以最小化所有作业的最大完成时间(Makespan)$C_{\max}=\max\limits_{i,j}\{C_{ij}\}$，其中$C_{ij}$表示操作$O_{ij}$的完工时间。FJSP实例的大小用$|J|\times|M|$表示。

析取图被广泛用于表示JSP问题和FJSP问题。在FJSP问题中，析取图可表示为$G=(O,C,D)$。如图1所示，$O=\{O_{ij}\mid\forall i,j\}\cup\{S,E\}$是操作节点集，包含所有操作节点；$S$和$E$分别表示生产开始和结束的虚拟节点；$C$是连接弧集合，表示来自同一作业的各个操作的加工顺序；D是无向的析取弧集合，连接可以由同一机器处理的一组操作。为了更好地表示机器与操作间的关系，本文使用改进的异构析取图$H=(O,M,C,E)$[1]来表示FJSP问题。

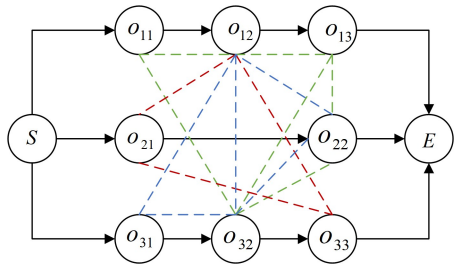
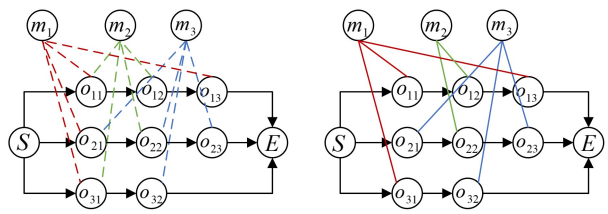


图1 FJSP的析取图表示

Fig. 1 Disjunction graph for FJSP

如图2所示，异构析取图保留操作节点集O和连接弧集合C，增加了机器节点集M和操作-机器(O-M)弧集合E。其中，每个元素$e_{ijk}\in E$表示一个连接操作节点O_{ij}与可兼容的机器节点M_k的无向弧。图2(a)和图2(b)分别给出了一个3×3FJSP实例的异构析取图示例和可行解。图中黑色实线表示连接弧，彩色线表示O-M弧，与JSP不同的是，FJSP中的一个操作可以兼容多个机器，因此一个操作可以连接到多台机器。



(a)一个FJSP实例 (b)一个可行解

图2 FJSP的异构析取图表示(电子版为彩图)

Fig. 2 Heterogeneous disjunction graph of FJSP

4 本文方法

FJSP 问题可以看作一个序列决策过程，通过迭代地执行调度动作，在每个决策步将一个操作分配给一台兼容的机器，直到所有操作都被调度。本文方法的总体框架如图 3 所示，共包含 3 个模块：1)将调度状态转换为异构析取图结构模块；2)用于提取操作和机器特征嵌入的 DLANs；3)使用行动者-评论家(Actor-Critic)架构的决策网络，该网络将操作选择和机器选择视为一个整体动作，生成动作概率分布并采样调度动作。

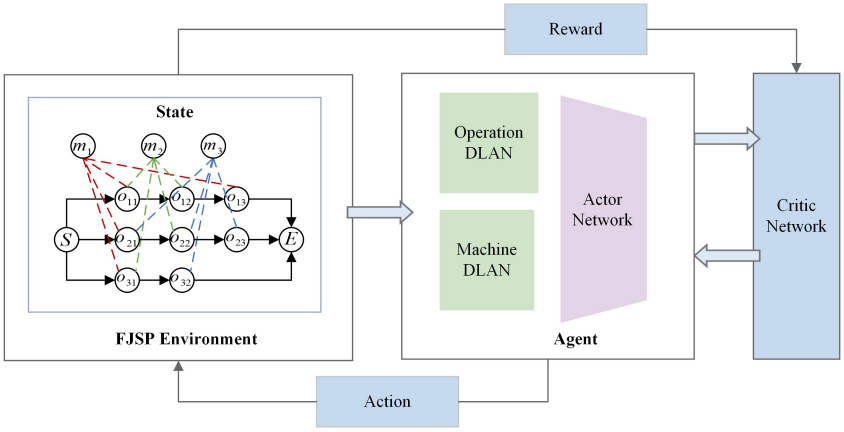


图 3 本文深度强化学习方法总体框架

Fig. 3 Overall framework of the proposed DRL method

4.1 马尔可夫决策建模

1)状态。状态s_t是一组表示所有与决策相关的操作信息及机器信息的特征向量集合。

对于每个操作O_{ij}，操作原始特征$h_{O_{ij}} \in \mathbb{R}^{10}$记录了如下信息：(1)一个指示操作是否已调度的二维量；(2)所有兼容机器的最短处理时间；(3)所有兼容机器的平均处理时间；(4)所有兼容机器的处理时间跨度；(5)可以处理O_{ij}的机器比例；(6)O_{ij}预计完成时间的下限；(7)作业J_i中的剩余操作数；(8)作业J_i的剩余工作量，即未调度操作的平均处理时间和；(9)O_{ij}的等待时间；(10)O_{ij}的剩余处理时间。

对于每个机器M_k，机器原始特征$h_{M_k} \in \mathbb{R}^9$记录了如下信息：(1)一个指示机器是否工作的二维量；(2)所有兼容操作中的最短处理时间；(3)所有兼容操作的平均处理时间；(4)M_k兼容的未调度操作数；(5)M_k兼容的候选操作数；(6)M_k的空闲时刻；(7)M_k的等待时间，即从空闲时刻到T_s的时间；(8)M_k的剩余处理时间；(9)M_k的利用率。

对于每个兼容的操作-机器对，其特征$h_{(O_{ij},M_k)} \in \mathbb{R}^8$记录了如下信息：(1)处理时间$p_{ijk}$；(2)$p_{ijk}$与$O_{ij}$最大处理时间的比率；(3)$p_{ijk}$与$M_k$兼容的候选操作最大处理时间的比率；(4)$p_{ijk}$与未调度操作的最大处理时间的比率；(5)$p_{ijk}$与$M_k$可处理的未调度操作的最大处理时间的比率；(6)$p_{ijk}$与兼容对的最大处理时间的比率；(7)$p_{ijk}$与$J_i$的剩余工作量的比率；(8)$O_{ij}$和$M_k$的等待时间和。

2)动作。本文将操作选择和机器选择视为一个整体，动作$a_t \in A_t$为在决策步t的一个可行的操作-机器对(O_{ij}, M_k)。

3)奖励。本文的目标是在调度过程中最小化完工时间，奖励r_t应设计为可指导代理选择有助于减少任务中所有操作的最大完成时间的动作。因此，在决策步t估计每个操作的完成时间为$C_{\max}(s_t)$，本文将r_t定义为决策步t和$t+1$的估计完工时间值之间的差：

$$r_t = C_{\max}(s_t) - C_{\max}(s_{t+1}) \tag{1}$$

当折现因子$\gamma=1$时，可以得到累积奖励为$\sum_{t=0}^{|O|} r_t = C_{\max}(s_0) - QUOTEC_{\max}$。对于特定的问题实例，$C_{max}(s_0)$是一个常量值，表示初始状态的估计 makeapan，这意味着最大化累积奖励等同于最小化完工时间。

4.2 双层注意力网络 DLANs 模型

传统的图神经网络通常使用固定的聚合方式来聚合邻居节点的信息，无法区分不同邻居的重要性。而注意力机制动态计算节点之间的权重，能够根据任务需求自适应地关注重要节点，在处理针对 FJSP 问题的异构图时，注意力网络能更灵活地捕捉关键信息。因此，本文考虑使用注意力机制来提取操作和机器特征，捕获不同操作和机器之间的相关性，从而找到重要程度高的操作和机器用于决策选择。FJSP 的异构析取图包含了两种类型的节点——操作节点和机器节点，操作与操作之间有前继、后继的连接关系，操作与机器之间有兼容关系，机器与机器之间有兼容同一操作的竞争关系。因此，本文将操作的邻居节点定义为两种类型：1)其直接前继、直接后继的操作节点；2)与其兼容的机器节点。将机器的邻居节点也定义为两种类型：1)与其兼容的操作节点；2)与其有竞争关系的机器节点。而现有的方法通常未充分考虑到节点之间的多种关系，也未考虑到不同关系对于优先级建模的不同重要程度。因此，本文提出了一种针对 FJSP 问题特有的双层注意力网络 DLANs 模型，这是一种基于分层注意力的图神经网络，包括节点级注意力(Node Attention)和类型级注意力(Type Attention)。节点级注意力分别学习操作节点和机器节点与其基于不同类型的邻居节点之间的重要性，类型级注意力学习不同类型的重要性。图 4 展示了用于提取操作特征的 DLANs 结构，操作节点级注意力分别学习操作与邻居操作、操作与邻居机器之间的重要性，再通过类型级注意力学习两种类型的重要性，最后聚合特征生成最终的操作特征。机器特征的提取与上述方式类似。

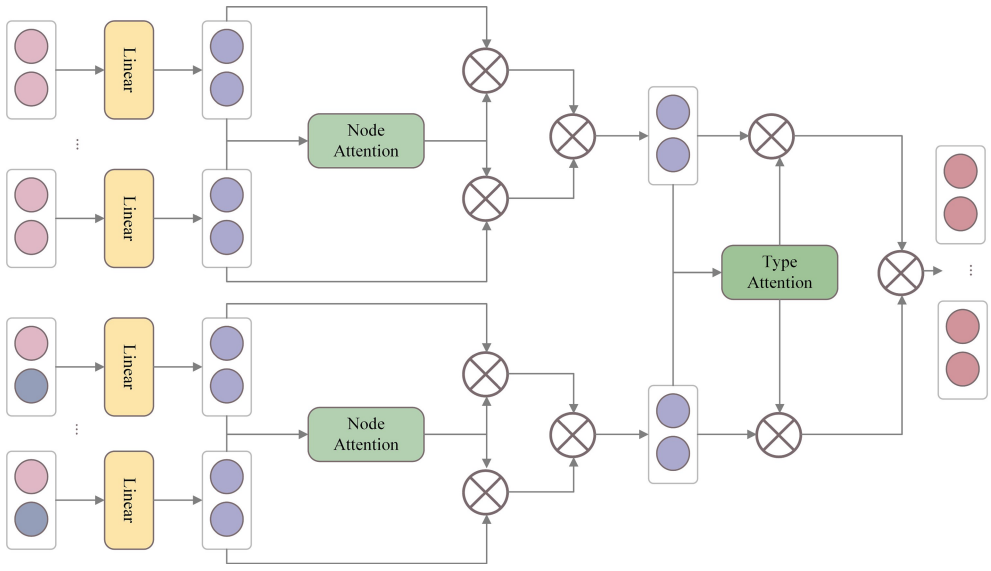


图 4 DLANs 结构

Fig. 4 Architecture of DLANs

4.2.1 节点级注意力

1)操作特征提取

操作O_{ij}的邻居节点分为两种类型:(1)同一作业中的直接前继操作节点$O_{i(j-1)}$和直接后继操作节点$O_{i(j+1)}$;(2)与其兼容的机器节点$M_k\in\mathcal{N}(O_{ij})$。

对于邻居操作节点,首先通过关联同一作业中的操作,学习不同操作节点的重要性。具体来说,给定输入特征为$h_{O_{ij}}\in\mathbb{R}^{10}$的操作节点$O_{ij}$,计算其与每个邻居操作节点的注意力系数。

$$e(h_{O_{ij}},h_{O_{ip}})=\boldsymbol{a}^{\mathrm{T}}LeakyReLU(\boldsymbol{W}\cdot[h_{O_{ij}}\parallel h_{O_{ip}}]),\quad |p-j|\leqslant 1 \tag{2}$$

其中,$\boldsymbol{a}^{\mathrm{T}}$和$\boldsymbol{W}$是线性变换,$\parallel$表示向量拼接。与原始图注意力网络(Graph Attention Network,GAT)[27]计算注意力系数的方式不同,这里使用了动态的图形注意力网络 GATv2[28]的计算方式,核心是将 LeakyReLU 移至两个线性操作$\boldsymbol{a}^{\mathrm{T}}$与$\boldsymbol{W}$之间,可以提高性能,降低时间复杂度。接着,用 softmax 函数将注意力系数在邻域内归一化:

$$\alpha_{i,j,p}=\mathrm{softmax}(e(h_{O_{ij}},h_{O_{ip}}))=\frac{\exp(e(h_{O_{ij}},h_{O_{ip}}))}{\sum_{p'=j-1}^{j+1}\exp(e(h_{O_{ij}},h\boldsymbol{W}_{O_{ip'}}))} \tag{3}$$

最后,通过计算邻居节点线性变换后的特征的加权平均值,得到操作特征向量一:

$$h^1_{O_{ij}}=\sigma\Big(\sum_{p=j-1}^{j+1}\alpha_{i,j,p}\boldsymbol{W}h_{O_{ip}}\Big) \tag{4}$$

其中,σ是一个非线性激活函数。

对于邻居机器节点,操作O_{ij}的相邻机器为兼容机器$M_k\in\mathcal{N}(O_{ij})$,为了表征相邻机器对操作的重要性,本文首先将每个机器节点$M_k\in\mathcal{N}(O_{ij})$的原始特征$h_{M_k}\in\mathbb{R}^9$与异构析取图中最重要的弧特征即兼容操作-机器对的处理时间拼接起来,得到特征向量$\boldsymbol{h}'_{M_k}\in\mathbb{R}^{10}$。然后计算对于操作$O_{ij}$,每个相邻机器的注意力系数:

$$e(h_{O_{ij}},h_{M_k})=LeakyReLU(\boldsymbol{b}^{\mathrm{T}}[\boldsymbol{W}^{O}h_{O_{ij}}\parallel\boldsymbol{W}^{M}\boldsymbol{h}'_{M_k}]) \tag{5}$$

其中,对操作节点和机器节点分别使用了不同的线性变换$\boldsymbol{W}^{O}$和$\boldsymbol{W}^{M}$。使用 softmax 函数将注意力系数在邻域内进行归一化后得到归一化注意力系数$\alpha_{ij,k}$,最后将相邻节点变换后的特征加权平均作为操作特征向量二:

$$h^2_{O_{ij}}=\sigma\Big(\sum_{M_k\in\mathcal{N}(O_{ij})}\alpha_{ij,k}\boldsymbol{W}^{M}\boldsymbol{h}'_{M_k}\Big) \tag{6}$$

2)机器特征提取

机器M_k的邻居节点也分为两种类型:1)可以处理同一个操作的竞争机器M_s;2)其可处理的操作节点$O_{ij}\in\mathcal{N}(M_k)$。

对于有竞争关系的邻居机器节点,沿用了文献[2]中用于度量机器M_k和机器M_s竞争强度的c_{ks},具体定义可参见原文。给定输入特征为$h_{M_k}\in\mathbb{R}^9$的机器节点M_k,使用c_{ks}计算注意力系数:

$$e(h_{M_k},h_{M_s})=\boldsymbol{c}^{\mathrm{T}}LeakyReLU(\boldsymbol{W}\cdot[h_{M_k}\parallel h_{M_s}]\parallel\boldsymbol{W}^{c}c_{ks}) \tag{7}$$

经过 softmax 函数归一化后,得到归一化的注意力系数α_{ks},最后通过σ激活函数加权平均得到最后的机器特征向量$\boldsymbol{h}^1_{M_k}$。

对于可处理的相邻操作节点$O_{ij}\in\mathcal{N}(M_k)$,首先将每个操作节点$O_{ij}\in\mathcal{N}(M_k)$的原始特征$h_{O_{ij}}\in\mathbb{R}^{10}$与异构析取图中最重要的弧特征即兼容操作-机器对的处理时间拼接起来,得到特征向量$\boldsymbol{h}'_{O_{ij}}\in\mathbb{R}^{11}$。然后对于机器$M_k$,计算其每个相邻操作的注意力系数:

$$e(h_{M_k},h_{O_{ij}})=LeakyReLU(\boldsymbol{d}^{\mathrm{T}}[\boldsymbol{W}^{M'}h_{M_k}\parallel\boldsymbol{W}^{O'}]) \tag{8}$$

其中,对机器节点和操作节点分别使用了不同的线性变换$\boldsymbol{W}^{M'}$和$\boldsymbol{W}^{O'}$。使用 softmax 函数将这些注意力系数在邻域内进行归一化后得到归一化注意力系数$\alpha_{k,ij}$。最后,将相邻节点变换后的特征做加权平均得到机器特征向量二$\boldsymbol{h}^2_{M_k}$。

4.2.2 类型级注意力

操作节点和机器节点都包含两种不同类型的邻居信息,为了学习更全面的节点嵌入特征,需要融合多个类型的信息。本文将从节点级注意力学习的 N 个特征h_{φ_n}作为输入,每个类型的学习权值表示为β_{φ_n}。为了得到不同类型邻居的重要程度,首先通过非线性函数转换节点级注意力网络特定于某

个类型的输出特征向量。然后，用转换后的特征向量与类型级关注向量 $\boldsymbol{q}$ 的相似度来衡量类型特定嵌入的重要性。进一步，将所有类型特定节点嵌入的重要性进行平均。各类型的重要度表示为 w_{φ_n}：

$$w_{\varphi_n}=\frac{1}{|N|}\sum_{n\in N}\boldsymbol{q}^{\mathrm{T}}\cdot\tanh(\boldsymbol{W}\cdot h_{\varphi_n}+\boldsymbol{b}) \tag{9}$$

其中，$\boldsymbol{W}$ 为权重矩阵，$\boldsymbol{b}$ 为偏置向量，$\boldsymbol{q}$ 为类型级关注向量。在得到每个类型的重要性后，通过 softmax 函数对它们进行归一化：

$$\beta_{\varphi_n}=\frac{\exp(w_{\varphi_n})}{\sum_{n=1}^{N}\exp(w_{\varphi_n})} \tag{10}$$

最终得到最后的嵌入 h：

$$h=\sum_{n=1}^{N}\beta_{\varphi_n}\cdot h_{\varphi_n} \tag{11}$$

对于操作节点，有两种类型的邻居节点，$N=2$。将 $h_{O_{ij}}^1$ 和 $h_{O_{ij}}^2$ 输入类型级注意力网络得到最终的操作嵌入特征 $\boldsymbol{h}'_{O_{ij}}$。对于机器节点，同理，将 $h_{M_k}^1$ 和 $h_{M_k}^2$ 输入类型级注意力网络得到最终的机器嵌入特征 h'_{M_k}。

4.2.3 池化

最后分别对操作特征和机器特征进行平均池化，将所得结果相连接形成 FJSP 实例的全局特征，即：

$$h_G=\left[\frac{1}{|O|}\sum_{O_{ij}\in O}\boldsymbol{h}'_{O_{ij}}\ \|\ \frac{1}{|M|}\sum_{M_k\in M}h'_{M_k}\right] \tag{12}$$

4.3 决策网络

本文设计了基于行动者-评论家架构的强化学习决策网络，该网络保持了注意力模型大小不可知的特性。使用两个 MLP 网络分别作为行动者和评论家，其参数分别用 θ 和 ϕ 表示。如图 5 所示，首先将 DLAN 网络提取的操作特征、机器特征、全局特征及操作-机器对特征输入行动者网络。

$$\mu(a_t\mid s_t)=MLP_\theta[\boldsymbol{h}'_{O_{ij}}\ \|\ h'_{M_k}\ \|\ h_G\ \|\ h_{(O_{ij},M_k)}] \tag{13}$$

接着，通过对所有 $\mu(a_t\mid s_t)$ 应用 softmax 函数来计算选择每个动作 a_t 的概率：

$$\pi_\theta(a_t\mid s_t)=\frac{\exp(\mu(a_t\mid s_t))}{\sum_{a_t'\in A(t)}\exp(\mu(a_t'\mid s_t))} \tag{14}$$

在训练期间，根据策略 π 对动作进行采样，以探索更多的轨迹。在测试期间，使用两种行动选择策略。一种是贪婪策略，它总是选择具有最高概率 $\pi_\theta(a_t\mid s_t)$ 的行动，另一种是与训练时一样的动作采样策略。

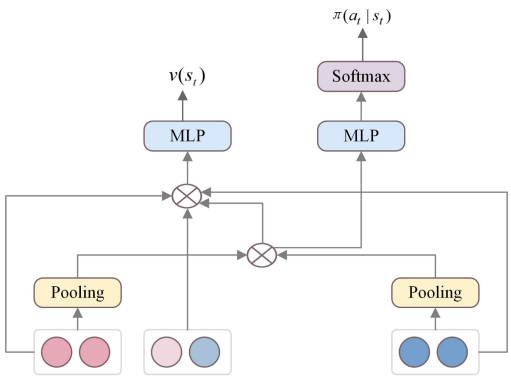


图 5 决策网络的结构

Fig. 5 Structure of decision network

4.4 训练

本文使用带有剪切代理目标的近端策略优化（PPO-Clip）[29]算法来训练所提出的调度模型。PPO 算法旨在通过裁剪机制限制策略更新的幅度，从而保证训练的稳定性。

训练算法的详细信息如算法 1 所示。损失函数由 3 部分组成，第一部分是策略损失（第 11 行），也是 PPO 的核心部分，用于更新策略参数，它基于重要性采样来估计新策略与旧策略之间的差异，并通过裁剪机制限制更新幅度。第二部分是价值函数损失（第 12 行），通过最小化价值函数损失，模型可以更准确地估计状态值，为策略优化提供更可靠的优势估计。第三部分是熵正则化项（第 13 行），用于鼓励策略的探索性，防止策略过早收敛，这也是平衡探索与利用的关键部分，确保策略能尝试更多可能性。

训练共进行 U 次迭代，在此过程中，DRL 代理并行求解一批 B 个实例（每 20 次迭代替换一次）（第 3－15 行），并收集转换数据，更新模型参数。在训练期间，每迭代 10 次在一组独立的验证实例上验证策略。

算法 1 强化学习训练算法

输入：行动者网络 π_θ，行动参与者网络 $\pi_{\theta\mathrm{old}}$，评论家网络 v_φ，训练轮数 I，批实例数 B，更新轮数 K，裁剪率 ε，策略损失系数 c_p，价值损失系数 c_v，熵损失系数 c_e

输出：行动者网络 π_θ，评论家网络 v_φ

1. 初始化网络参数 θ 和 φ，采样 B 个 FJSP 实例
2. for i=1,2,…,I do
3. for b=1,2,…,B do //并行
4. for t=0,1,2… do
5. 根据 $\pi_{\theta\mathrm{old}}(a_t\mid s_t)$ 采样动作得到 a_t；
6. 得到奖励 r_t 和下一个状态 s_{t+1}；
7. 收集转换 (s_t,a_t,r_t,s_{t+1})；
8. $s_t\leftarrow s_{t+1}$；
9. end for
10. 根据收集的转换集合计算每一步的广义优势估计 $\hat{A}_t$
11. $L_b^{CLIP}(\theta)=\sum_0^t\min(r_t\hat{A}_t,\mathrm{clip}(r_t,1-\varepsilon,1+\varepsilon)\hat{A}_t)$；
12. $L_b^{VF}(\varphi)=\sum_0^t(v_\varphi(s_t)-\hat{A}_t)^2$；
13. $L_b^{S}(\theta)=\sum_0^t S(\pi_\theta(a_t\mid s_t))$；
14. 计算总损失函数 $L_b(\theta,\varphi)=c_pL^{CLIP}(\theta)-c_vL^{VF}(\varphi)+c_eL^{S}(\theta)$；
15. end for
16. for k=1,2,…,K do
17. 更新网络参数 θ 和 ϕ
18. if i mod 10=0 do
19. 验证策略
20. if i mod 20=0 do
21. 采样新一批 FJSP 实例；
22. end for

5 实验

本章通过实验得到了在合成数据集和公开 FJSP 数据集上的实验结果，以验证所提方法的性能。

5.1 数据集

本文使用两种不同分布的合成数据来检验所提方法的学

习性能和泛化性能。分别按照文献[1]和文献[2]的方法(分别用 SD1 和 SD2 表示)生成不同大小的训练集、验证集和测试集，包括 10×5，15×5，20×5，15×10，20×10，30×10，40×10，40×20。其中训练集实例在训练过程中实时生成，验证集和测试集实例预先生成，每个集合均包含 100 个实例。SD1 数据集允许作业具有不同数量的操作，操作随机处理时间范围更小；SD2 数据集允许每个作业具有相同数量的操作，操作随机处理时间范围更大。本文在 5 个较小的尺寸上进行训练，并使用最大的 3 个尺寸来测试训练策略的泛化性能。除了合成实例外，还使用了 4 组公开的 FJSP 基准来探索模型的泛化能力，1 组 10 个 mk 实例[30]和 3 组 la 实例(rdata，edata 和 vdata，每组 40 个实例)[31]，这些实例的大小各不相同，与训练数据的分布也大不相同，因此在这些基准数据上进行测试，可以有效验证所提方法在分布外实例任务上的能力。

5.2 实验设置

本文首先在 SD1 数据集 10×5 尺寸上，对机器操作嵌入维度、MLP 网络层数以及隐藏维度进行了实验。将模型嵌入维度设置为 4，8，16；MLP 网络设置为 2，3，4；隐藏维度设置为 32，64，128。实验发现，网络层数与隐藏维度的数量对调度结果的影响很小，64 个隐藏层节点能保证更快地收敛到最佳结果。因此，本文将 DLAN 设为单层，机器嵌入和操作嵌入维度设为 8。动作选择网络 MLP_θ 和状态动作值评估网络 MLP_ϕ 均为 3 层，隐藏层维度为 64。对于 PPO，损失函数中的策略、值和熵系数分别设置为 1，0.5 和 0.01。裁剪参数 ε 和折扣系数 γ 分别设置为 0.2 和 1。训练期间，网络更新周期设为 $K=4$，使用 Adam 优化器更新网络，学习率 $lr=3\times10^{-4}$。对于测试，分别使用贪婪策略和采样策略，对训练模型进行测试。所有实验均在具有 24GB 显存的 Nvidia GeForce RTX 3090 GPU 工作站上进行。

5.3 对比方法

首先，本文选择与在实践中显示出良好性能的 4 种著名 PDRs 方法进行比较，包括先进先出(First in First Out，FIFO)、剩余操作最多(Most Operations Remaining，MOR)、最短加工时间(Shortest Processing Time，SPT)及剩余平均处理时间最多(Most Work Remaining，MWKR)。其次，选择 OR-Tools 的结果作为参考线。OR-Tools 作为一种权威约束规划求解器，耗时长但性能高，因此设置 1 800 s 为时限生成(近)最优解。最后，将本文方法与最近提出的先进的 DRL 方法[1-2]进行了比较。模型的性能根据平均完工时间以及完工时间与知名的解决方案之间的相对差距进行评估，该解决方案是合成实例的 OR-Tools 求解结果或者最佳结果。

5.4 实验结果与分析

5.4.1 合成数据上的性能

表 1 展示了每个模型在测试实例上的平均 makespan 和与 OR-Tools 解决方案的相对差距。

表 1 在两个小到中等训练规模的合成数据上的表现

Table 1 Performance on two synthetic data of small to medium training size

	Size		PDRs				Greedy strategy			Sampling strategy			OR-Tools
			FIFO	MOR	SPT	MWKR	HGNN[1]	DANIEL[2]	Ours	HGNN[1]	DANIEL[2]	Ours	
SD1	10×5	Objective	119.4	115.38	129.82	113.23	111.67	106.71	**106.54**	105.59	101.67	**101.16**	
		Gap	24.06%	19.87%	34.76%	17.58%	16.03%	10.87%	**10.6%**	9.66%	5.57%	**5.05%**	96.32
		Time/s	0.10	0.09	0.09	0.10	0.46	0.38	0.33	1.20	0.81	0.73	
	15×5	Objective	167.95	164.43	181.77	161.09	160.48	150.95	**149.97**	158.72	146.61	**145.92**	
		Gap	17.77%	15.32%	27.4%	12.94%	12.55%	5.82%	**5.16%**	11.29%	2.78%	**2.29%**	142.65
		Time/s	0.14	0.14	0.13	0.14	0.58	0.61	0.50	1.53	1.40	1.29	
	20×5	Objective	216.08	214.16	230.48	209.78	211.22	197.56	**195.97**	207.53	192.78	**192.31**	
		Gap	14.87%	13.85%	22.56%	11.51%	12.27%	5.03%	**4.17%**	10.31%	2.46%	**2.22%**	188.15
		Time/s	0.20	0.19	0.18	0.20	0.92	0.95	0.65	2.42	2.12	1.85	
	15×10	Objective	184.55	173.15	198.33	171.25	166.92	161.28	**159.38**	160.86	153.22	**151.21**	
		Gap	28.65%	20.68%	38.22%	19.41%	16.33%	12.42%	**11.05%**	12.13%	6.79%	**5.38%**	143.53
		Time/s	0.30	0.30	0.31	0.19	1.52	1.34	1.00	3.72	3.89	3.86	
	20×10	Objective	233.48	219.80	255.17	216.11	215.78	198.5	**196.45**	214.81	193.91	**191.66**	
		Gap	19.22%	12.20%	30.25%	10.30%	10.15%	1.31%	**0.27%**	9.64%	−1.03%	**−2.18%**	195.98
		Time/s	0.45	0.43	0.44	0.44	1.98	1.86	1.52	6.73	6.51	6.24	
SD2	10×5	Objective	569.41	557.48	514.39	549.28	553.61	408.40	**404.86**	483.90	366.74	**362.64**	
		Gap	76.47%	72.52%	57.96%	70.01%	71.42%	25.68%	**24.26%**	49.71%	12.57%	**11.21%**	326.24
		Time /s	0.10	0.10	0.10	0.10	0.42	0.39	0.36	0.87	0.78	0.71	
	15×5	Objective	795.39	790.57	653.79	776.81	783.63	528.74	**522.80**	714.37	485.8	**481.23**	
		Gap	75.68%	74.62%	43.61%	71.59%	72.98%	16.21%	**14.94%**	57.73%	6.77%	**5.76%**	454.98
		Time/s	0.12	0.12	0.12	0.12	0.56	0.58	0.49	1.57	1.37	1.23	
	20×5	Objective	1045.83	1045.94	835.94	1026.03	1059.04	671.03	**663.80**	962.90	629.94	**624.04**	
		Gap	74.59%	74.58%	38.91%	71.31%	76.79%	11.52%	**10.26%**	60.70%	4.66%	**3.68%**	602.04
		Time/s	0.20	0.20	0.20	0.20	0.95	0.82	0.65	2.32	2.07	1.91	
	15×10	Objective	871.14	845.16	703.07	830.53	807.47	591.21	**585.93**	756.07	521.83	**515.20**	
		Gap	132.23%	125.32%	86.74%	121.45%	115.26%	57.16%	**55.99%**	101.52%	38.70%	**36.96%**	377.17
		Time/s	0.32	0.32	0.31	0.31	1.46	1.26	1.22	4.16	4.08	3.84	
	20×10	Objective	1088.05	1059.68	829.14	1040.69	1045.82	610.16	**597.34**	990.37	552.64	**545.81**	
		Gap	135.27%	129.09%	78.82%	124.98%	126.12%	31.58%	**28.85%**	114.15%	19.13%	**17.65%**	464.10
		Time /s	0.44	0.41	0.43	0.42	1.94	1.59	1.34	7.22	6.40	6.40	

这些测试实例是使用与训练实例相同的分布，从相同的规模尺度中生成的。可以看出，在 SD1 和 SD2 数据集所有规

模的问题上，本文方法不仅明显优于 4 种基于 PDR 的方法，并且相对于文献[1]和文献[2]中的先进 DRL 方法，两种行动选择策略都表现出了更优的性能且运行时间更短。在 SD1 数据集的 20×10 规模上，本文方法甚至比 OR-Tools 高出 2.18%。本文方法的优势在 SD2 数据集上更加明显，当机器处理时间范围变大时，PDR 的性能受到影响，与 OR-Tools 存在相当大的差距，但本文方法仍然保持了良好的性能。值得注意的是，随着作业数量的增加，本文方法与 OR-Tools 解决方案的相对差距逐渐缩小，这表明大量的作业促使模型更好地捕捉操作与机器之间的复杂关系。

此外，分别使用 30×10，40×10 和 40×20 的测试实例，检验了在 10×5 和 20×10 任务上训练的模型在两种数据上的泛化性能，实验结果如表 2 和表 3 所列。结果表明，本文模型在中小型实例上学习到的策略在求解大型实例时仍然表现出色，特别是随着实例大小的增加，性能更加突出，甚至表现出远超 OR-Tools 的性能。

表 2 SD1 数据集中训练的模型在两个较大尺寸的合成数据上的表现

Table 2 Performance of models trained on the SD1 dataset on two synthetic data of larger size datasets

Strategy	Method	SD1 30×10			SD1 40×10			SD1 40×20		
		Objective	Gap/%	Times/s	Objective	Gap/%	Times/s	Objective	Gap/%	Times/s
Greedy strategy	HGNN[1] 10×5	314.71	14.61	2.86	417.87	14.21	3.82	422.32	0.31	7.58
	HGNN 20×10	313.04	14.01	2.84	416.18	13.75	3.81	421.77	0.18	7.31
	DANIEL[2] 10×5	288.61	5.10	2.60	379.28	3.65	3.58	384.76	−8.61	7.91
	DANIEL 20×10	281.49	2.50	2.73	371.45	1.52	3.60	370.47	−12.01	7.58
	Ours 10×5	286.86	4.46	2.07	377.80	3.25	2.84	377.85	−10.26	6.45
	Ours 20×10	**280.16**	**2.02**	2.42	**369.98**	**1.12**	2.96	**366.92**	**−12.85**	6.43
Sampling strategy	HGNN[1] 10×5	308.55	12.36	12.79	410.76	12.26	24.54	417.64	−0.80	63.84
	HGNN 20×10	312.59	13.49	12.80	415.25	13.49	24.40	425.89	1.16	62.27
	DANIEL[2] 10×5	286.83	4.43	12.78	379.56	3.77	20.01	383.96	−8.8	132.10
	DANIEL 20×10	279.20	1.67	13.84	370.48	1.14	21.87	372.50	−11.53	137.81
	Ours 10×5	285.01	3.79	11.78	377.89	3.28	18.73	378.53	−10.09	126.12
	Ours 20×10	**276.79**	**0.79**	11.52	**368.22**	**0.63**	19.73	**363.63**	**−13.63**	120.18
OR-Tools			274.67			365.96			421.24	

表 3 SD2 数据集中训练的模型在两个较大尺寸的合成数据上的表现

Table 3 Performance of models trained on the SD2 dataset on two synthetic data of larger size datasets

Strategy	Method	SD1 30×10			SD1 40×10			SD1 40×20		
		Objective	Gap/%	Times/s	Objective	Gap/%	Times/s	Objective	Gap/%	Times/s
Greedy strategy	HGNN[1] 10×5	1564.57	123.55	2.93	2048.96	109.87	3.87	2143.71	166.97	7.74
	HGNN 20×10	1543.69	123.57	2.93	2032.54	108.12	3.92	2041.59	154.29	7.71
	DANIEL[2] 10×5	794.62	14.85	2.82	983.37	0.52	3.56	899.19	11.92	7.12
	DANIEL 20×10	774.56	11.95	2.75	962.58	−1.67	3.54	912.15	13.55	6.98
	Ours 10×5	780.06	12.71	2.10	964.41	−1.45	2.93	860.28	6.99	6.55
	Ours 20×10	**764.61**	**10.5**	2.35	**949.22**	**−3.00**	2.27	**835.38**	**3.92**	6.71
Sampling strategy	HGNN[1] 10×5	1486.58	115.21	12.88	1976.25	102.45	24.55	2041.59	156.79	65.29
	HGNN 20×10	1461.16	111.51	12.75	1945.33	99.26	24.50	2050.74	155.39	61.97
	DANIEL[2] 10×5	757.48	9.47	12.48	951.21	−2.74	22.36	860.34	7.03	130.28
	DANIEL 20×10	725.27	4.80	12.17	914.02	−6.60	21.75	828.99	3.13	126.12
	Ours 10×5	739.41	6.85	11.76	936.51	−4.24	20.80	814.18	1.28	124.65
	Ours 20×10	**720.89**	**4.15**	12.08	**913.66**	**−6.62**	20.14	**783.37**	**−2.56**	122.76
OR-Tools			692.26			998.39			805.67	

5.4.2 公共数据集上的性能

现实世界中的问题可能来自未知的分布，因此模型在分布外实例的泛化性能尤为重要。本文进一步探索了模型在 4 组公共基准上的泛化性能，这些基准实例的分布与训练实例完全不同。为了进一步验证本文方法的有效性，增加了机器利用率作为评价指标，其中机器利用率等于机器工作时间与机器工作加空闲总时间的比率。本文在 4 个不同的基准数据集上与先进的 DRL 方法[1-2]进行比较。表 4 和表 5 分别列出了在 SD1 和 SD2 数据集上训练的模型在公开基准数据集上的表现，表中展示了每个方法最好结果的对比。与先进的 DRL 方法相比，本文方法无论是在贪婪策略还是采样策略上都表现出了最优性能，这表明通过本文方法学习到的策略可以很好地泛化到这些分布外实例。此外，在机器利用率上，本文方法也展现了良好的性能，这也表明优化 makespan 隐式关联了机器利用率指标，缩短 makespan 通常会缩短空闲时间，从而提升机器利用率。注意到，在作业数量更多的规模上训练的模型整体性能更优，这可能是因为对于更复杂的调度任务，本文模型能更好地学习有效信息，从而使训练更加充分。

最后，本文进一步与先进的元启发式算法进行对比，以观察 DRL 方法与元启发式算法在 FJSP 问题上的表现。分别与遗传算法 GA[32]、禁忌搜索 TS[33]、人工蜂群 ABC[33] 3 种元启发式算法进行对比，结果如表 6 所列。其中 GA 是在 30 个 la(vdata)实例(la01-la30)上计算的，文献[32]给出了 Rooyani 和 Defersha 的结果。可以发现，遗传算法能找到高质量的解决方案，然而这些基于搜索的方法的运行时间比基于 DRL 中的方法长得多。

表 4 在 SD1 数据集上训练的模型在公共基准上的表现

Table 4 Performance of models trained on SD1 data on public benchmark

Dataset		Greedy strategy			Sampling strategy		
		HGNN[1]	DANIEL[2]	Ours	HGNN[1]	DANIEL[2]	Ours
mk	Objective	201.40	185.30	**183.40**	191.60	179.7	**178.60**
	Machine utilization	69.68%	74.87%	**77.43%**	74.21%	78.10%	**79.65%**
	Time/s	1.18	1.16	0.98	4.12	4.59	4.11
la(rdata)	Objective	1030.83	1031.63	**1024.53**	995.73	987.33	**979.93**
	Machine utilization	84.64%	84.59%	**85.35%**	87.78%	88.47%	**89.11%**
	Time /s	1.23	1.35	1.08	4.81	5.21	4.67
la(edata)	Objective	1187.48	1194.98	**1171.10**	1121.95	1124.88	**1115.55**
	Machine utilization	73.67%	72.81%	**74.85%**	77.77%	77.52%	**78.54%**
	Time/s	1.22	1.34	1.03	4.91	5.47	4.77
la(vdata)	Objective	954.33	942.18	**942.05**	932.95	927.00	**924.48**
	Machineutilization	91.02%	92.04%	**92.26%**	92.98%	93.81%	**94.15%**
	Time/s	1.46	1.34	1.08	4.72	5.14	4.69

表 5 在 SD2 数据集上训练的模型在公共基准上的表现

Table 5 Performance of models trained on SD2 data on public benchmark

Dataset		Greedy strategy			Sampling strategy		
		HGNN[1]	DANIEL[2]	Ours	HGNN[1]	DANIEL[2]	Ours
mk	Objective	196.10	183.10	**183.00**	192.70	180.40	**178.70**
	Machine utilization	71.87%	74.69%	**74.93%**	74.06%	77.26%	**78.03%**
	Time/s	1.13	1.19	0.96	4.32	4.82	4.26
la(rdata)	Objective	1031.35	1040.18	**1025.40**	1001.43	991.68	**978.28**
	Machine utilization	84.16%	83.82%	**85.26%**	87.32%	88.05%	**89.50%**
	Time/s	1.31	1.33	1.08	5.06	5.02	4.72
la(edata)	Objective	1200.28	1190.33	**1169.05**	1125.30	1119.35	**1103.05**
	Machine utilization	73.07%	73.73%	**75.27%**	77.95%	78.23%	**79.83%**
	Time/s	1.22	1.32	1.04	4.53	5.27	4.70
la(vdata)	Objective	953.35	982.68	**950.85**	937.90	938.63	**927.80**
	Machine utilization	91.14%	88.31%	**91.48%**	92.69%	92.57%	**93.91%**
	Time/s	1.22	1.36	1.08	4.84	5.08	4.76

表 6 在公共基准上与元启发式算法的性能比较

Table 6 Performance comparison of DRL and metaheuristic algorithms on public benchmark

Method		mk		la(rdata)		la(edata)		la(vdata)	
		Objective	Time/s	Objective	Time/s	Objective	Time/s	Objective	Time/s
Metaheuristic	GA[32]	183.00	280.10	—	—	—	—	**836.13**	191.40
	TS[33]	**176.76**	91.34	952.96	104.85	1070.31	108.12	919.65	103.41
	ABC[33]	177.38	99.84	**944.53**	116.46	**1057.59**	126.84	928.64	110.59
DRL	Ours(SD1)	178.6	4.11	979.93	4.67	1115.55	4.77	924.48	4.69
	Ours(SD2)	178.7	4.26	978.28	4.72	1103.05	4.70	927.80	4.76

本文模型可以通过少量微调在未知分布的现实问题上表现得更好,这将在未来的研究中进一步验证。

结束语 本文针对 FJSP 问题构建了一种新的端到端强化学习框架,提出了基于分层注意力的 DLANs 模型以充分提取异构析取图中操作与机器的复杂信息。本文方法结合了动态的注意力机制与深度强化学习,在合成数据集和公共数据集上的大量实验表明,本文方法在保持高效率的同时,性能和泛化能力均优于传统的 PDR 方法和先进的 DRL 方法。在未来的工作中,将研究具有更多不确定性的动态 FJSP 问题,如作业随机到达、机器故障等复杂场景下的调度研究。

参 考 文 献

[1] SONG W,CHEN X Y,LI Q,et al. Flexible Job-Shop Scheduling via Graph Neural Network and Deep Reinforcement Learning[J]. IEEE Transactions on Industrial Informatics,2023,19(2):1600-1610.

[2] WANG R Q,WANG G,SUN J,et al. Flexible Job Shop Scheduling via Dual Attention Network-Based Reinforcement Learning[J]. IEEE Transactions on Neural Networks and Learning Systems,2024,35(3):3091-3102.

[3] MENG L L,ZHANG C Y,REN Y P,et al. Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling[J]. Computers & Industrial Engineering,2020,142:106347.

[4] ÖZGÜVEN C,ÖZBAKR L,YAVUZ Y. Mathematical Models for Jobshop Scheduling Problems with Routing and Process Plan Flexibility[J]. Applied Mathematical Modelling,2010,34(6):1539-1548.

[5] MÜLLER D,MÜLLER M G,KRESS D,et al. An Algorithm Selection Approach for the Flexible Job Shop Scheduling Problem:Choosing Constraint Programming Solvers through Machine Learning[J]. EuropeanJournal of Operational Research,2022,302(3):874-891.

[6] MATI Y,REZG N,XIE X L. An integrated greedy heuristic for a flexible job shop scheduling problem[C]// Proceedings of the IEEE International Conference on Systems,Man and Cybernetics. 2001:2534-2539.

[7] HAUPTR. A Survey of Priority Rule-based Scheduling[J]. Operations Research Spektrum, 1989, 11(1): 3-16.
[8] SELS V, GHEYSEN N, VANHOUCK M. A Comparison of Priority Rules for the Job Shop Scheduling Problem under Different Flow Time-and Tardiness-related Objective Functions[J]. International Journal of Production Research, 2012, 50(15): 4255-4270.
[9] ORTIZ M A, BEATANCOURT L E, NEGRETE K P, et al. Dispatching Algorithm for Production Programming of Flexible Job-shop Systems in the Smart Factory Industry[J]. Annals of Operations Research, 2018, 264(1): 409-433.
[10] ZHANG C, SONG W, CAO Z G, et al. Learning to Dispatch for Job Shop Scheduling via Deep Reinforcement Learning[C]// Proceedings of the 34th International Conference on Neural Information Processing Systems. Red Hook, NY: Curran Associates Inc., 2020: 1621-1632.
[11] HAN B A, YANG J J. A Deep Reinforcement Learning Based Solution for Flexible Job Shop Scheduling Problem[J]. International Journal of Simulation Modelling, 2021, 20(2): 375-386.
[12] HUANG R H, YANG C L, CHENG W C. Flexible job shop scheduling with due window a two-pheromone ant colony approach[J]. International Journal of Production Economics, 2013, 141(2): 685-697.
[13] LI X, LIANG G. An Effective Hybrid Genetic Algorithm and Tabu Search for Flexible Job Shop Scheduling Problem[J]. International Journal of Production Economics, 2016, 174: 93-110.
[14] MANOSIJ G, RITAM G, SARKAR R, et al. A Wrapper-filter Feature Selection Technique Based on Ant Colony Optimization [J]. Neural Computing and Applications, 2020, 32(12): 7839-7857.
[15] DEFERSHA F M, ROOYANI D. An Efficient Two-stage Genetic Algorithm for Flexible Job-shop Scheduling Problem with Sequence Dependent Attached/Detached Setup, Machine Release Date and Lag-Time[J]. Computers and Industrial Engineering, 2020, 147: 106605.
[16] WU X L, WU S M. An elitist quantum-inspired evolutionary algorithm for the flexible job-shop scheduling problem[J]. Journal of Intelligent Manufacturing, 2017, 28(6): 1441-1457.
[17] ZARROUK R, BENNOUR I E, JEMAI A. A two-level particle swarm optimization algorithm for the flexible job shop scheduling problem[J]. Swarm Intelligence, 2019, 13(2): 145-168.
[18] WANG Y L, STEIN B V, BACK T, et al. A Tailored NSGA-III Instantiation for Flexible Job Shop Scheduling[C]// 2020 IEEE Symposium Series on Computational Intelligence. 2020: 2746-2753.
[19] GAO K Z, CAO Z G, ZHANG L, et al. A Review on Swarm Intelligence and Evolutionary Algorithms for Solving Flexible Job Shop Scheduling Problems[J]. IEEE-CAA Journal of Automatica Sinica, 2019, 6(4): 904-916.
[20] LEI K, GUO P, ZHAO W C, et al. A multi-action deep reinforcement learning framework for flexible Job-shop scheduling problem[J]. Expert Systems with Applications, 2022, 205: 117796.
[21] LUO S, ZHANG L, FAN Y. Real-time Scheduling for Dynamic Partialno-wait Multiobjective Flexible Job Shop by Deep Reinforcement Learning[J]. IEEE Transactions on Automation Science and Engineering, 2022, 19(4): 3020-3038.
[22] WANG X, JI H Y, SHI C, et al. Heterogeneous Graph Attention Network[C]// Proceedings of the World Wide Web Conference. 2019: 2022-2032.
[23] XIE J, GAO L, PENG K K, et al. Review on Flexible Job Shop Scheduling[J]. IET Collaborative Intelligent Manufacturing, 2019, 1(3): 67-77.
[24] SOBEYKO O, MONCH L. Heuristic Approaches for Scheduling Jobs in Large-scale Flexible Job Shops[J]. Computers and Operations Research, 2016, 68: 97-109.
[25] CHEN R H, YANG B, LI S, et al. A Self-learning Genetic Algorithm Based on Reinforcement Learning for Flexible Job-Shop Scheduling Problem[J]. Computers and Industrial Engineering, 2020, 149: 106778.
[26] LONG X J, ZHANG J T, QI X, et al. A Self-learning Artificial Bee Colony Algorithm Based on Reinforcement Learning for a Flexible Job-shop Scheduling Problem[J]. Concurrency and Computation: Practice and Experience, 2022(4): 34.
[27] VELICKOVIC P, CUCURULL G, CASANOVA A, et al. Graph Attention Networks[C]// Proceedings of the International Conference on Learning Representations(ICLR). 2018.
[28] BRODY S, ALON U, YAHAV E. How Attentive are Graph Attention Network[C]// Proceedings of the International Conference on Learning Representations(ICLR). 2022.
[29] SCHULMAN J, WOLSKI F, DHARIWAL P, et al. Proximal policy optimization algorithms[J]. arXiv: 1707.06347, 2017.
[30] BRANDIMARTE P. Routing and scheduling in a flexible job shop by tabu search[J]. Annals of Operations Research, 1993, 41(3): 157-183.
[31] HURINK J, JURISCH B, THOLE M. Tabu search for the job-shop scheduling problem with multi-purpose machines[J]. OR Spektrum, 1994, 15(4): 205-215.
[32] ROOYANI D, DEFERSHA F M. An Efficient Two-Stage Genetic Algorithm for Flexible Job-Shop Scheduling[C]// Proceedings of 9th IFAC Conference on Manufacturing Modelling, Management and Control(IFAC MIM). 2019: 2519-2524.
[33] LI X Y, GAO L. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem[J]. International Journal of Production Economics, 2016, 174: 93-110.

WANG Haoyan, born in 2001, postgraduate. Her main research interests include deep reinforcement learning and job shop scheduling.

LI Chongshou, born in 1988, Ph.D, associate professor, is a member of CCF (No. J8308M). His main research interests include multi-scale data intelligence, AI and applied optimization.

(责任编辑:柯颖)