控制与决策
Control and Decision

学习驱动的迭代局部搜索算法求解分布式流水车间鲁棒调度问题

郭恒伟, 桑红燕, 潘全科

引用本文:
郭恒伟, 桑红燕, 潘全科. 学习驱动的迭代局部搜索算法求解分布式流水车间鲁棒调度问题[J]. 控制与决策, 2026, 41(4): 977-986.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2025.0275

学习驱动的迭代局部搜索算法求解分布式流水车间鲁棒调度问题

郭恒伟[1]，桑红燕[1†]，潘全科[2]

(1. 聊城大学 计算机学院，山东 聊城 252000；2. 上海大学 机电工程与自动化学院，上海 200444)

摘　要: 针对分布式流水车间中加工时间不确定性与序列相关准备时间耦合的鲁棒调度问题, 提出一种强化学习驱动的迭代局部搜索算法 (QILS). 首先, 构建以最大完工时间为目标的期望-风险鲁棒调度模型, 有效平衡调度方案的稳定性与最优性; 其次, 设计面向不确定环境的 NEHUPT 启发式方法, 基于场景分析确定工件的调度优先级, 结合微调策略提升初始解的质量; 另外, 构建Q-learning 与迭代局部搜索算法的协同优化框架, 利用强化学习以及动态衰减方法驱动扰动策略的动态选择, 平衡算法的搜索和开发能力; 最后, 提出一种基于鲁棒贡献度的局部搜索方法, 进一步提升解的质量. 通过系统性的仿真实验及与多种先进代表性算法的对比分析结果表明, 所提出的算法在求解分布式鲁棒车间调度问题方面具有显著优势.

关键词: 分布式置换流水车间调度；鲁棒调度；强化学习；迭代局部搜索；不确定性

中图分类号: TP391　　**文献标志码:** A

DOI: 10.13195/j.kzyjc.2025.0275

引用格式: 郭恒伟, 桑红燕, 潘全科. 学习驱动的迭代局部搜索算法求解分布式流水车间鲁棒调度问题 [J]. 控制与决策, 2026, 41(4): 977-986.

A learning-driven iterated local search algorithm for solving distributed flowshop robust scheduling problems

GUO Heng-wei[1]，*SANG Hong-yan*[1†]，*PAN Quan-ke*[2]

(1. School of Computer Science，Liaocheng University，Liaocheng 252000，China；2. School of Mechatronic Engineering and Automation，Shanghai University，Shanghai 200444，China)

Abstract: This study addresses the robust scheduling problem in distributed flowshop involving coupled uncertainties in processing times and sequence-dependent setup times. A reinforcement learning-driven iterated local search algorithm (QILS) is proposed. Firstly, an expectation-risk robust scheduling model is established with the objective of minimizing makespan to effectively balance the stability and optimality of the solution. Secondly, the NEHUPT heuristic is designed for uncertain environments, where scheduling priorities are generated based on scenario analysis, and a fine-tuning strategy is applied to enhance the initial solution quality. Additionally, a collaborative optimization framework integrating Q-learning and ILS is developed, where reinforcement learning and dynamic decay methods guide the adaptive selection of perturbation strategies to balance exploration and exploitation capabilities. Finally, a robustness contribution-based local search method is introduced to further enhance solution quality. Comprehensive simulation experiments and comparative analysis with multiple state-of-the-art algorithms demonstrate the superior performance of the proposed method in solving distributed robust flow shop scheduling problems.

Keywords: distributed permutation flowshop scheduling；robust scheduling；reinforcement learning；iterated local search；uncertainty

0　引　言

近年来, 多工厂生产调度模式凭借其高生产率、低交付成本和管理风险可控等优势, 在智能制造领域获得广泛应用[1-5]. 作为该模式的核心问题之一, 分布式置换流水车间调度问题 (DPFSP) 因其 NP-hard 特性与工程应用价值备受关注[6-8]. 在实际制造环境

收稿日期: 2025-03-15；**录用日期:** 2025-04-18.
基金项目: 国家自然科学基金项目 (62473186)；山东省自然科学基金项目 (ZR2024MF17).
责任编委: 王凌.
†通信作者. E-mail: sanghongyan@lcu-cs.com.

中, 存在两个关键特征需特别考量:

1) 序列相关准备时间 (SDST): 即相邻任务之间的准备时间受加工顺序的影响. 如印刷行业更换墨盒/纸张的耗时取决于相邻任务参数差异, 此类现象在化工原料切换、食品加工设备清洗等场景也普遍存在[9-10];

2) 加工时间不确定性: 源于设备故障、订单变更等随机事件, 可能导致传统确定性调度方案失效[6].

针对上述复杂调度环境, 现有研究在确定性DPFSP/SDST领域已取得一定进展. Guo等[9]设计了带有组合更新机制的果蝇优化算法求解DPFSP/SDST, 该算法设计了差异飞行策略, 避免了传统果蝇优化算法短视飞行的缺陷; Kouvelis等[11]提出了一种新颖的约束规划模型, 并基于自适应框架设计了一个进化策略方法来解决DPFSP/SDST; Chen等[12]针对大规模个性化制造中不同规模的订单, 建立了具有可变任务分割的分布式混合流水车间调度问题模型, 提出了基于订单模块化处理方法的任务分割方法对客户订单进行处理; Zhao等[13]为以最小化最大完成时间为目标的DPFSP/SDST设计了多策略果蝇算法, 并引入了一个正弦优化算法控制搜索范围, 提高了算法的全局搜索能力.

在不确定调度领域, 鲁棒优化已经被证明是一种有效的求解策略, 近年来在运筹学和决策辅助(OR-DA)领域受到越来越多的关注[14]. Kouvelis等[11]提出了一种调度鲁棒性的度量方法, 并开发了精确的启发式方法解决加工时间不确定的两机流水车间调度问题. Allahverdi等[15]设计了5种启发式方法解决具有有限处理时间的双机流水车间调度问题. Xu等[16]在区间场景下提出了Min-Max regret模型, 并开发了迭代松弛方法解决鲁棒并行机调度问题. Jing等[17]建立了具有不确定加工时间的DPFSP期望-标准差鲁棒模型, 提出了两种基于局部搜索的算法, 所提出的算法采用加速方法进一步节省了计算时间. Guo等[6]进一步结合序列相关准备时间约束, 研究了不确定的DPFSP/SDST, 并为果蝇优化算法设计了一种年龄机制避免局部最优. Wang等[18]提出了一种协同迭代贪心算法解决具有组约束的分布式鲁棒流水车间调度问题, 该算法整合了一种虚拟场景方法. 另外, Wang等[19]使用混合局部搜索方法研究了具有加工时间不确定的作业车间调度问题, 通过组合多个单场景邻域来增强混合局部搜索方法的性能. Lv等[20]针对绿色炼钢连铸调度问题中的不确定性环境设计了一种增强交叉熵方法. Li等[21]为不确定的集成规划和调度问题建立了鲁棒模型, 并通过整合一种新的区间排序方法和混合遗传算子改进了粒子群算法. 然而, 现有方法在面对复杂的DPFSP/SDST问题时, 仍然存在一些局限性, 如策略的静态化和缺乏历史优化信息的有效利用.

尽管已有研究在DPFSP/SDST和不确定调度领域取得了一定成果, 但当前研究仍存在不足:

1) 多数研究聚焦于单一不确定性因素, 未同时考虑加工时间不确定性和SDST耦合带来的复杂性;

2) 鲁棒模型多数关注均值性能, 对极端场景下调度稳定性的建模与提升不足.

迭代局部搜索算法 (ILS) 作为一种基于轨迹的元启发式算法, 其流程简单、可扩展性高、参数设定少且寻优能力强, 在调度领域得到广泛应用. 然而, 现有ILS在应对加工时间不确定的具有序列相关准备时间约束的分布式置换流水车间调度问题 (UDPFSP/SDST) 时仍存在以下局限:

1) 扰动策略静态化导致探索-开发能力失衡, 难以在复杂解空间中有效搜索;

2) 搜索过程缺乏知识引导机制, 历史优化信息未得到有效利用;

3) 现有鲁棒模型对极端场景的适应性不足.

针对UDPFSP/SDST以及ILS的局限性, 本文采用区间场景刻画加工时间的不确定性, 建立鲁棒调度数学模型, 并提出一种强化学习驱动的迭代局部搜索算法, 主要贡献如下:

1) 构建基于区间场景的期望-标准差鲁棒模型, 通过均值和标准差平衡解的鲁棒性与稳定性;

2) 设计强化学习驱动的扰动阶段, 利用Q-learning动态选择最优扰动策略, 增强算法在搜索过程中的适应性和灵活性;

3) 提出NEHUPT初始化方法和基于鲁棒贡献度的局部搜索方法, 实验表明QILS算法显著优于对比方法.

1 带序列相关准备时间的分布式鲁棒车间调度问题

1.1 问题描述

UDPFSP/SDST是一类考虑生产效率 (makespan) 的资源分配问题, 描述如下: n个工件$J=\{J_1,J_2,\ldots,J_n\}$分配到f个工厂$F=\{F_1,F_2,\ldots,F_f\}$加工. 每个工厂拥有相同的m台机器$M=\{M_1,M_2,\ldots,M_m\}$构成的流水线. 一旦工件J_i分配到工厂F_l, 则工件J_i的所有工序将在工厂F_l完成. 当两个工件J_i和J_j先后在机器M_k上加工时, 机器M_k存在一个

与工件加工顺序相关的准备时间S_{ij}. 此外, 本文假设 SDST 仅与工件加工顺序相关. 一个工件在同一时刻只能选择一台机器进行加工, 且每台机器只能同时加工一个工件. 任何时刻, 工件的加工过程都不能中断. 对于 UDPFSP/SDST, 由于不可避免的随机事件或信息不完全, 工件的加工时间是不确定的.

1.2 问题模型

UDPFSP/SDST 符号及模型如下:

n_l: 工厂F_l分配的工件数量;

$P_{i,k}$: 工件J_i在机器M_k的加工时间;

$S_{i,j}$: 工件J_j在工件J_i之后加工的机器准备时间;

π: 完整解决方案$\pi = \{\pi_1, \pi_2, \ldots, \pi_f\}$;

π_l: 工厂F_l的加工序列;

$\pi_{l,k}$: 序列π_l $(k = 1, 2, \ldots n_l)$的第k个加工工件;

Λ: 不确定加工时间的所有场景的集合;

$|\Lambda|$: 场景规模;

μ: 从场景集Λ中随机选择的场景规模 ($\mu \leqslant |\Lambda|$);

s: 场景索引;

$P^s_{i,k}$: 场景Λ_s下工件J_i在机器M_k的加工时间;

$S^s_{i,j}$: 场景Λ_s下工件J_j在工件J_i之后加工的机器准备时间;

$C_{l,s}$: 场景Λ_s下工厂F_l的最后一个加工工件的完成时间;

$C(\Lambda_s)$: 场景Λ_s下所有工件的最大完成时间;

$C^{\mu}(\Lambda)$: 所有场景的最大完成时间集合$C^{\mu}(\Lambda) = \{C(\Lambda_1), C(\Lambda_2), \ldots, C(\Lambda_{\mu})\}$;

$C^{\mu}_l(\Lambda)$: 工厂F_l的最后一个工件在所有场景的完成时间集合;

$E[C^{\mu}(\Lambda)]$: 平均完成时间集合;

$E[C^{\mu}_l(\Lambda)]$: 工厂F_l的平均完成时间集合;

$\mathrm{Std}[C^{\mu}(\Lambda)]$: μ个场景的最大完成时间标准差集合$\mathrm{Std}[C^{\mu}(\Lambda)] = \sqrt{\sum_{s=1}^{\mu} \left(C_l(\Lambda_s) - E[C^s(\Lambda)]\right)^2} \Big/ \mu$;

$\mathrm{Std}[C^{\mu}_l(\Lambda)]$: 工厂$F_l$的完成时间标准差集合$\mathrm{Std}[C^{\mu}_l(\Lambda)] = \sqrt{\sum_{s=1}^{\mu} \left(C(\Lambda_s) - E[C^s_l(\Lambda)]\right)^2} \Big/ \mu$;

$\mathrm{Obj}[C^{\mu}(\Lambda)]$: 鲁棒优化目标函数;

$\mathrm{Obj}[C^{\mu}_l(\Lambda)]$: 工厂$F_l$的鲁棒优化目标函数.

在 UDPFSP/SDST 中, 由于加工时间的不确定性, 采用区间场景方法进行描述. 每个场景代表一组独特的工件加工时间. 在此基础上, 构建期望-标准差模型作为鲁棒优化目标函数[17], 如下所示:

$$\min \mathrm{Obj}[C^{\mu}(\Lambda)] = r \times E[C^{\mu}(\Lambda)] + (1 - r) \times \mathrm{Std}[C^{\mu}(\Lambda)]. \quad (1)$$

其中: $E[C^{\mu}(\Lambda)]$表示不同场景的最大完工时间平均值, 反映了调度方案的平均性能. 然而, 不同场景的完工时间可能存在较大差异. 即便所有场景的平均完工时间较优, 某些特定场景的实际完工时间仍可能较差, 导致调度结果的不可预测性, 增加决策者面临的风险. 标准差$\mathrm{Std}[C^{\mu}(\Lambda)]$则用于衡量各个场景完工时间波动的幅度, 体现了调度方案的稳定性. r为偏好系数, 由决策者根据实际需要决定, 研究设定$r = 0.01$.

2 学习驱动的 ILS

2.1 解的表示

本文采用基于置换的 2 维向量$\pi = \{\pi_1, \pi_2, \ldots, \pi_f\}$表示一个完整的解决方案. 每个解由$f$个无重复的排列组成, 其中$\pi_k = \{\pi_{k1}, \pi_{k2}, \ldots, \pi_{kn_k}\}$表示分配给工厂$F_k$加工的工件序列, 其顺序表示在每台机器的加工顺序.

2.2 初始化阶段

启发式方法基于问题的先验知识, 在较短时间内快速获得可行解, 能为元启发式算法提供高质量的初始解. Nawaz-Enscore-Ham (NEH) 是一种被证明在解决流水车间调度问题时有效的启发式方法. 因此, 本文将其拓展到 UDPFSP/SDST 问题, 提出了 NEHUPT 启发式方法. 首先, 根据工件在所有场景下的总加工时间确定工件的调度优先级, 总加工时间由下式计算:

$$\mathrm{Total}P_i = \sum_{s=1}^{\mu} \sum_{k=1}^{m} P^s_{i,k},\ i = 1, 2, \ldots, n. \quad (2)$$

其中$\mathrm{Total}P_i$表示工件J_i的总加工时间, 其值越大, 表明工件的优先级越高, 在调度过程中会被优先分配至工厂进行加工. 在确定工件优先级后, 将按照优先级从高到低的顺序, 依次把工件插入到能使鲁棒目标值$\mathrm{Obj}[C^{\mu}(\Lambda)]$最小的位置. 由于 UDPFSP/SDST 存在加工时间不确定性等复杂因素, 为充分探索解空间, 提高初始解质量, NEHUPT 在将工件插入到当前最佳位置后, 随机选取该位置附近的工件进行微调, 尝试将其插入到其他能使鲁棒目标值最小的新位置. 重复上述过程直到所有工件被分配至最佳位置.

2.3 学习驱动的扰动阶段

2.3.1 强化学习方法 (Q-learning)

强化学习作为一种机器学习方法, 通过让智能体在环境中学习如何做出决策来实现目标. 在强化学习的过程中, 智能体根据所采取动作获得的奖励或惩罚, 动态调整其行为策略, 从而不断优化决策能力. Q-learning 是一种无模型的强化学习技术[22], 它维护着一个存储状态-动作对的Q表, 其值表示在某种状态下采取动作后得到的预期收益. Q-learning 通过不断地与环境交互来更新Q表, 从而逼近最优策略. Q值的更新公式如下所示:

$$Q(S,a)=(1-\alpha)Q(S,a)+\alpha(R+\gamma\max_{a'}Q(S',a')-Q(S,a)). \quad (3)$$

其中: S和a分别表示当前状态和选择的动作, S'表示执行动作a后所达到的状态, a'是状态S'中执行的动作, R、α和γ分别代表奖励值、学习率和折扣因子.

2.3.2 扰动策略集

基于 UDPFSP/SDST 的特点, 本文设计 6 种扰动策略, 作为 QILS 的扰动策略集. 这些扰动策略分别针对不同的解空间探索需求进行设计, 以增强算法在求解过程中的搜索能力.

SWAP_PERTURB (SP): 随机交换两个不同的工件, 包括工厂内交换和工厂之间交换. SP 能够在一定程度上改变解的结构, 实现小范围的快速探索.

INSERT_RBLOCK_PERTURB (IBP): 随机选择两个不同的工件, 并根据两个工件的平均加工时间从大到小进行排列, 组成一个工件块. 将工件块作为整体, 重新插入至产生最佳目标值的位置. IBP 通过对工件块的操作, 不仅考虑了工件的个体特性, 还能在一定范围内调整工件的加工顺序, 从而探索新的区域.

INSERT_BLOCK_InFAC_PERTURB (IBIP): 针对每个分布式加工车间, 随机选择长度为len的连续工件组成一个工件块. 为控制计算资源消耗, 避免完全插入搜索带来的过高成本, 随机生成IC个候选插入位置 (IC $\in[n_k/2,n_k]$). 将工件块依次尝试插入这些候选位置中, 最终选择具有最小鲁棒目标值的位置插入. IBIP 在保证解的多样性的同时, 有效降低了搜索的复杂度, 使得算法能够在有限的时间内探索更多的解.

REVERSE_BLOCK_InFAC_PERTURB (RBIP): 以与 IBIP 相同的方式确定规模为len的工件块, 然后对工件块执行反转操作. 不同于 IBIP, 在 RBIP 中, IC $\in[1,n_k/2]$, 这是因为反转操作往往会导致序列产生较大变化. RBIP 通过缩小插入范围, 在保证解的多样性的同时, 能够有效控制计算复杂度. 同时, 较大幅度的扰动有助于算法跳出局部最优, 增加全局探索能力.

REVERSE_BLOCK_ExtFAC_PERTURB (RBEP): 随机选择一个工厂, 然后从该工厂随机选择len个连续工件组成一个工件块. 对工件块执行反转操作后, 尝试将其插入到其他工厂的所有可能位置. 最终, 保留能够产生最佳目标值的插入方案作为扰动后的新解. RBEP 充分考虑了分布式生产环境的特点, 通过跨工厂的操作, 进一步增强了全局搜索能力.

RUIN_REPAIR_PERTURB (RRP): 基于破坏与修复的思想, 首先随机确定破坏强度D ($D\in[f,2f]$). 在破坏阶段, 从当前解中随机移除D个工件, 并存储在集合Θ. 在修复阶段, 依次从集合Θ选择一个工件, 尝试将其插入到所有工厂的可能位置, 并选择能够产生最佳鲁棒目标值的位置插入. 修复过程重复, 直至Θ的所有工件被重新分配至新位置, 生成修复后的新解.

2.3.3 学习驱动的扰动策略

学习机制被广泛引入元启发式算法中解决调度问题. 为了有效平衡算法的全局探索与局部开发能力, 本研究将学习机制引入迭代局部搜索算法 (ILS) 的扰动阶段, 构建基于Q-learning 驱动的扰动阶段.

在Q-learning 的框架下, 任何动作的执行都与特定的环境状态紧密相关. 在 QILS 中, 环境状态通过一组精心设计的二进制变量来表示, 这些变量直观反映了每次扰动操作的成功与否. 具体而言, 当状态$s=1$时, 意味着当前选择的扰动策略成功应用, 即该操作改进了当前解; 反之, 当$s=0$时, 则表明扰动操作未能对当前解进行优化, 此时, 算法会依据历史信息以及当前的环境状态, 智能地选择新的扰动策略, 以探索解空间中尚未开发的区域. 为了实现全局探索与局部开发的平衡, QILS 采用了动态ϵ-greedy 策略. 该策略使 QILS 既能够以概率ϵ随机选择动作, 进行广泛的全局探索, 避免陷入局部最优解, 又能基于历史经验, 以概率$1-\epsilon$选择当前最优策略, 提升解的质量. 随着搜索过程的推进, 探索概率ϵ基于下式逐渐衰减至较低的ϵ^{end}值:

$$\epsilon=\epsilon^0(\epsilon^0-\epsilon^{\text{end}})\frac{\mathcal{T}_{\text{cur}}}{\mathcal{T}_{\max}}. \quad (4)$$

其中: ϵ^0表示初始探索率; $\mathcal{T}_{\text{cur}}$和$\mathcal{T}_{\max}$分别表示当前CPU运行时间和算法最大CPU运行时间, 本文设置$\epsilon^{\text{end}}=0.15$. 在搜索初期, 保持较高的$\epsilon$值以增强探索能力, 使算法能够充分探索解空间. 随着搜索的深入, 逐渐降低ϵ值, 增加选择最优扰动策略的概率, 促使算法加速收敛.

在每一轮搜索结束后, QILS基于扰动策略的性能计算奖励值R, 并更新Q表, 奖励值计算如下所示:

$$R=\begin{cases}10+5(\text{Obj}-\text{Obj}'), & \text{Obj}^g>\text{Obj}';\\ 5(\text{Obj}-\text{Obj}'), & \text{Obj}>\text{Obj}';\\ -2|\text{Obj}-\text{Obj}'|, & \text{otherwise}.\end{cases}\tag{5}$$

其中: Obj和Obj′表示扰动前的鲁棒目标值和一轮搜索后的鲁棒目标值, Obj^g表示当前全局最优目标值.

2.4　局部搜索阶段

局部搜索阶段旨在进一步优化解决方案[23], 本文设计一种基于鲁棒贡献度的局部搜索方法. 工厂F_l的鲁棒性贡献度ρ_l定义为

$$\rho_l=\frac{f_l-f_{\min}}{f_{\max}-f_{\min}}+\frac{\sigma_l}{\sum_{k=1}^{f}\sigma_k}.\tag{6}$$

其中: f_l为工厂F_l的鲁棒目标值, 它综合反映了该工厂在不同场景下的调度性能; σ_l为其标准差, 用于衡量工厂在不同场景下完工时间的波动程度.

首先, 确定具有最大鲁棒贡献度的工厂为关键工厂 ($\arg\max_{1\leqslant l\leqslant f}\rho_l$). 然后, 随机选择关键工厂中的一个工件, 并尝试将其插入到所有工厂的所有可能位置. 若插入操作能够改进当前解, 即新的鲁棒目标值小于原目标值, 则接受该新解, 并重新确定关键工厂. 重复上述过程, 直到关键工厂中的所有工件都已被测试, 且不再出现能够改进解的插入操作, 局部搜索阶段结束.

2.5　接收准则

QILS采用模拟退火准则决定是否接受当前解进入下一次迭代. 基于UDPFSP/SDST的特点和确定性环境下的退温公式[17], 本文采用所下所示的温度公式:

$$T=\beta\frac{\sum_{s=1}^{\mu}\sum_{i=1}^{n}\sum_{k=1}^{m}P_{i,k}^{s}}{10\times\mu\times n\times m},\tag{7}$$

其中β是需要校准的参数.

2.6　QILS算法流程

基于上述组件设计, QILS的算法流程如图1所示. 算法执行步骤如下:

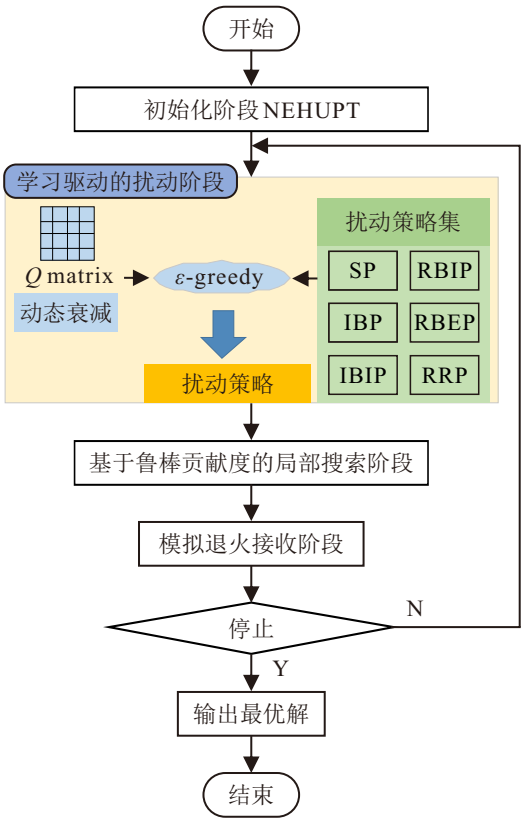

图1　QILS算法流程图

1) 初始化阶段: 采用改进的NEHUPT启发式方法生成初始解, 通过场景分析确定工件优先级, 并引入微调策略增强初始解质量;

2) 强化学习驱动的扰动阶段: 基于Q-learning构建策略选择机制, 通过动态ϵ-贪婪策略平衡探索与开发, 自适应选择最优扰动策略;

3) 局部搜索: 以鲁棒贡献度为导向, 通过跨工厂工件重排进一步提升解的质量;

4) 接受准则: 采用模拟退火机制决定是否接受新解, 确保算法在收敛性与多样性间取得平衡.

3　仿真实验与统计分析

3.1　实验设置和测试数据集

本文提出的QILS和所有对比算法均采用Visual Studio 2017编码, 运行环境为AMD Ryzen7-7840HS CPU@3.80 GHz和32 GB RAM的Windows 11操作系统. 为了消除偶然性并确保公平, 所有算法独立运行10次, 通过10次的平均结果验证算法的有效性. 本文采用相对百分比偏差RPI作为评估指标[23], 用于评估算法的性能, 如下所示:

$$\text{RPI}=\frac{\text{Obj}_{\text{ins}}^{\text{alg}}-\text{Obj}_{\text{ins}}^{\text{best}}}{\text{Obj}_{\text{ins}}^{\text{best}}}\times 100\,\%.\tag{8}$$

其中: $\text{Obj}_{\text{ins}}^{\text{alg}}$是算法alg求解算例ins的目标值, 而$\text{Obj}_{\text{ins}}^{\text{best}}$是所有对比算法求解算例ins得到的最佳目标

值.

基于 DPFSP/SDST 和 UDPFSP 问题的测试数据集[7, 17], 本文设计了 540 组数据作为 UDPFSP/SDST-测试数据集. 其中, 工厂规模$f=\{2,3,4\}$, 机器规模$m=\{2,4,6\}$, 工件规模$n=\{50,70,100,200,500\}$, 准备时间由随机分布$U[1,50]$产生, 工件的加工时间通过随机分布$U[\mathrm{pt}^1,\mathrm{pt}^2]$产生, 其中下界$\mathrm{pt}^1$由随机分布$U[10,\ 50\times\delta_1]$产生, 上界$\mathrm{pt}^2$由随机分布$U[\mathrm{pt}^1,\mathrm{pt}^1\times(1+\delta_2)]$产生. 参数$\delta_1\in\{0.4,0.8\}$, $\delta_2=\{1.0,1.5,2.5\}$. 对于每种组合$\{f,m,n,\delta_1,\delta_2\}$, 重复运行 2 次, 共生成 540 个测试算例, 构成 UDPFSP/SDST 数据集, 从中随机选择 270 个算例作为校准数据集, 用于算法性能调优. 由于加工时间的不确定性, 本文为每个算例生成 30 个场景 ($|\Lambda|=30$), 540 个算例共$540\times30=16\,200$个场景.

3.2 参数设置

QILS 的关键参数包括: 学习率、折扣因子、搜索率和退温系数. 每个参数的取值如下: 学习率 $=\{0.4, 0.5, 0.6, 0.7\}$; 折扣因子 $=\{0.6, 0.7, 0.8, 0.9\}$; 搜索率 $=\{0.6, 0.7, 0.8, 0.9\}$, 退温系数 $=\{0.05, 0.06, 0.07, 0.08\}$. 本文采用田口正交实验$L_{20}(5^1,4^3)$来确定最佳的参数组合. 基于 RPI 主效应图 (图 2), QILS 的学习率设置为 0.6, 折扣因子设置为 0.8, 搜索率设置为 0.8, 退温系数设置为 0.07.

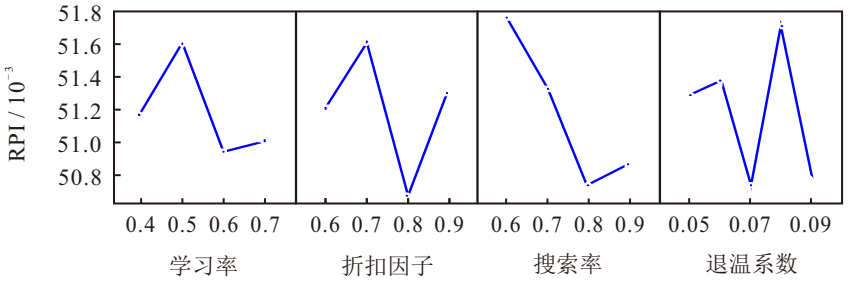

图2 关键参数主效应图

3.3 学习机制的有效性

为了验证学习驱动机制的有效性, 本文设计一组对比实验. 实验包括QILS和变体QILS_1, QILS在扰动阶段采用Q-learning 学习扰动策略, QILS_1采用随机机制选择扰动策略. QILS_1的参数设置与QILS相同, 并在所有数据集上独立运行 10 次, 其 ANOVA 分析结果见图 3. 由图 3 可知, QILS与QILS_1具有显著性差异, 且QILS明显优于QILS_1, 验证了采用学习机制驱动的扰动策略的有效性.

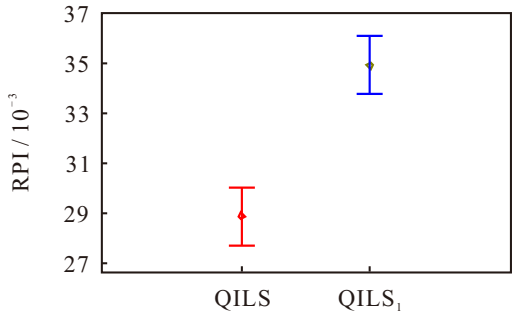

图3 学习机制有效性的主效应图

此外, 为了进一步评估Q-learning 在搜索过程中的作用, 分析不同问题实例下 QILS 的搜索过程, 并展示其扰动策略的选择和执行情况 (图 4 ～ 图 7). 图中包括收敛曲线、各搜索步骤所选择的动作以及不同动作的执行率.

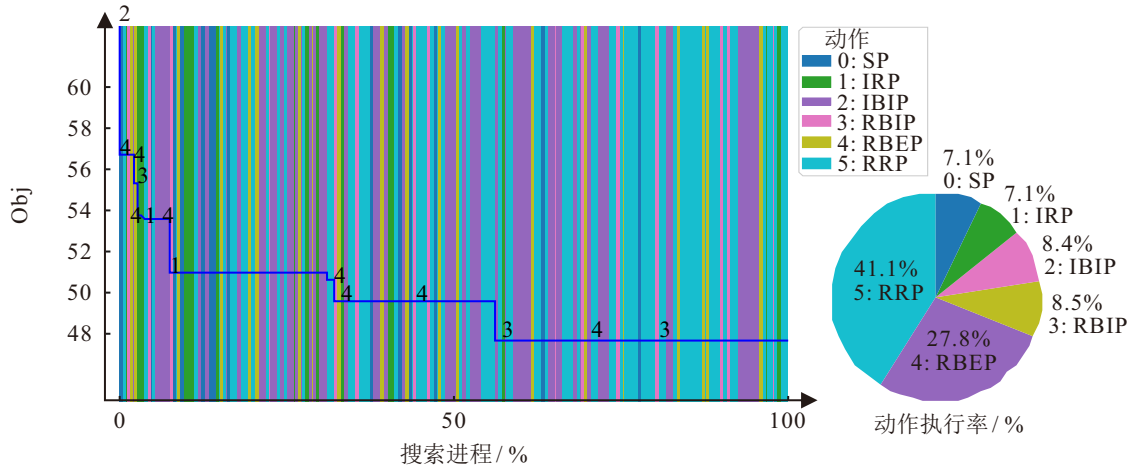

图4 算例动作执行图 ($f=2, n=20, m=2$)

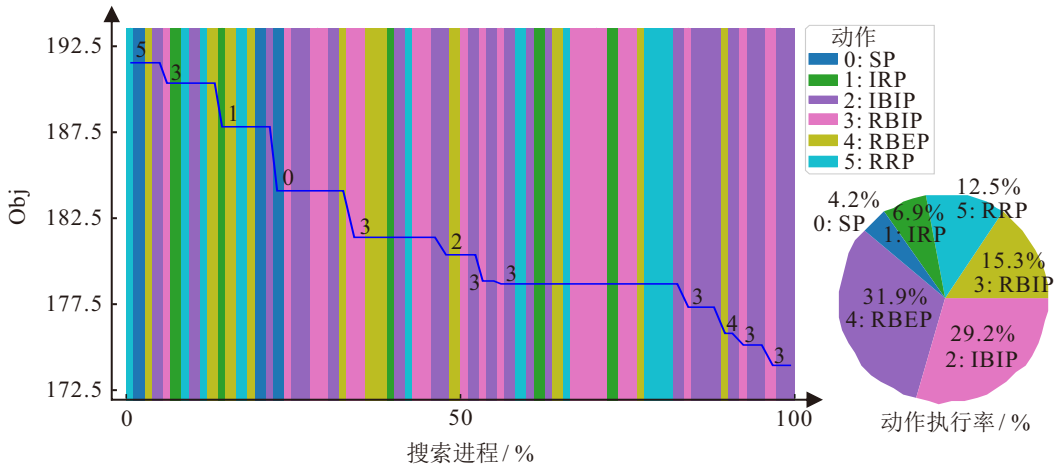

图5　算例动作执行图 ($f=2, n=100, m=4$)

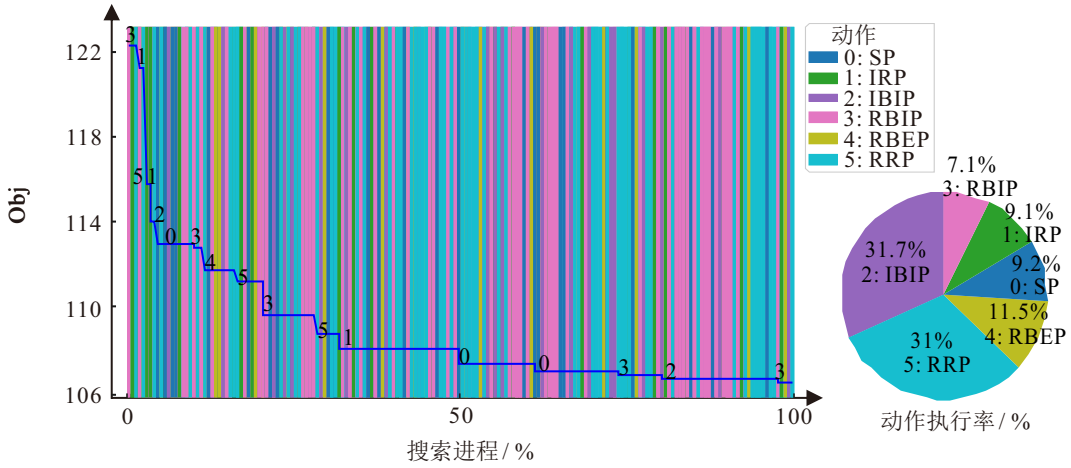

图6　算例动作执行图 ($f=3, n=60, m=6$)

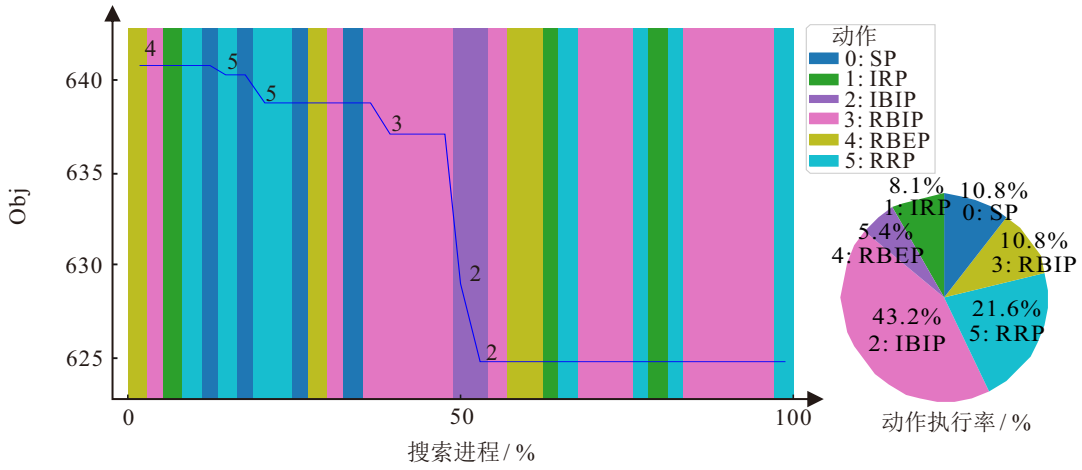

图7　算例动作执行图 ($f=4, n=200, m=2$)

从图 4 ～ 图 7 的饼状图可以观察到, 所有扰动策略都对目标值的改进做出了贡献, 但各策略的贡献程度因问题实例的不同而有所变化. 这表明, QILS 能够根据问题环境的不同, 自适应地调整搜索过程中的扰动策略, 以更好地适应不同问题实例的求解需求. 图 4 ～ 图 7 的收敛曲线进一步展现了算法的动态调整能力, 从中可以看出: 1) 在搜索过程中, QILS 采用了不同的扰动策略, 展现出较强的策略调整能力; 2) 当目标值在较长时间内未发生改进时, QILS 尝试采用更多的扰动策略, 以期望跳出当前局部最优区域, 从而提升搜索质量; 3) 在鲁棒目标值改进时, QILS 主要倾向于执行此前产生改进的优势策略, 以加速收敛. 综上分析, 不同问题实例下, 扰动策略的执行率存在较大差异, 并且导致全局最优解改进的主要扰动策略也有所不同. 这表明 QILS 能够根据具体的搜索状态和问题实例的特性, 动态调整扰

动策略的选择, 从而提高算法的适应性和求解能力. 这种自适应性使得 QILS 能够更有效地处理不同的问题实例, 展现出良好的优化能力.

3.4 对比实验

为了测试学习驱动 QILS 的有效性, 本文将 QILS 与几种先进的元启发式算法进行对比, 包括 FFO[6]、EA[8]、IGR[10] 和 sIG[17]. 为了确保公平, 所有对比算法均采用 C++编码, 并在相同的硬件和软件环境下运行. 当 CPU 最大运行时间达到$\mathcal{T}_{\max}=\omega\times m\times n$ 毫秒时, 算法停止运行.

图 8 和图 9 分别给出了不同停止时间下, 各个算法在 95% LSD 置信区间下的平均 RPI 值. 由图 8(a) 可见, $\omega=200$时, QILS 在整体表现上显著优于其他算法, 获得了最低的 RPI 值, 这表明 QILS 在解决 UDPFSP/SDST 问题时具备卓越的性能. 图 8(b) ~ 图 8(d) 分别展示了按照工件规模、工厂数量和机器数量分组的交互图. 从图 8(b) 和图 8(d) 可以看出, 无论工厂规模和机器规模如何变化, QILS 均表现出显著的优势, 且随着工厂数量和机器数量的增加, 其相对于其他算法的性能优势愈发明显. 而对于工件规模交互图 (图 8(c)), IGR 在小规模工件 ($n=20$)

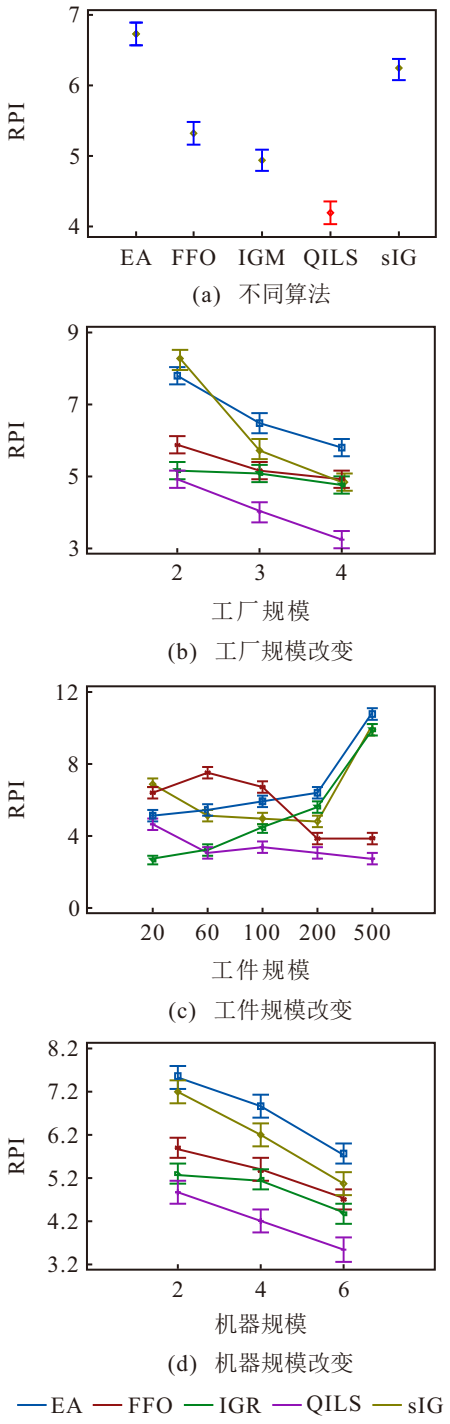

图8 ANOVA 分析结果 ($\omega=200$)

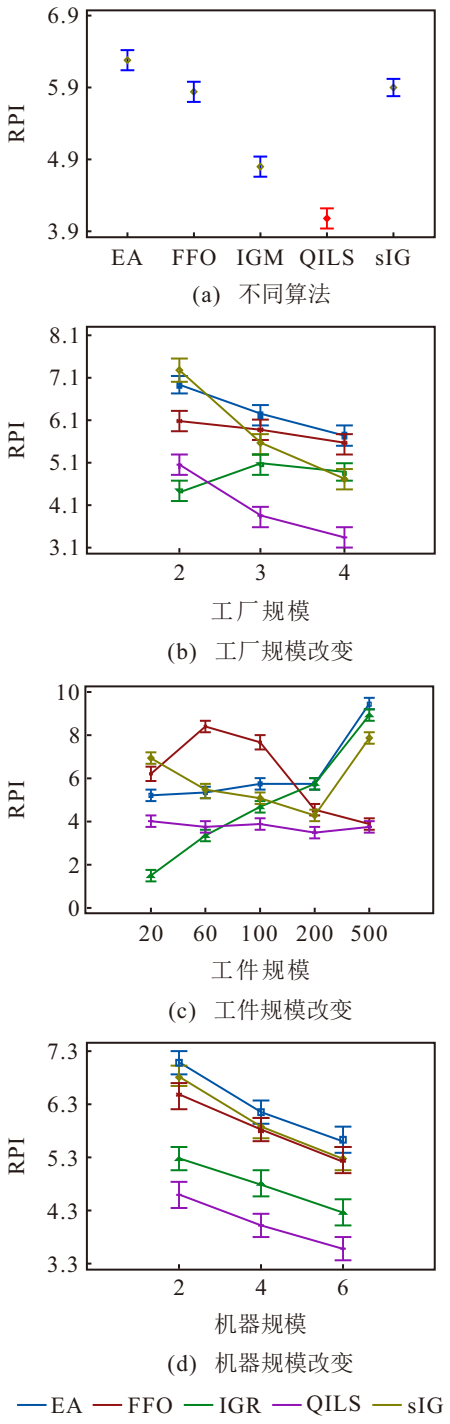

图9 ANOVA 分析结果 ($\omega=350$)

时的表现优于 QILS, 但随着工件规模的增大, IGR 的性能急剧下降, 而 QILS 则展现出稳定且不断提升的性能. 从整体趋势来看, QILS 的交互曲线波动幅度较小, 且始终维持较低的 RPI 值, 由此验证了 QILS 在解决 UDPFSP/SDST 问题上的优越性.

图 9 进一步验证了上述结论, 在$\omega = 350$时, QILS 的性能仍显著优于其他算法. 尽管在部分小规模工件 ($n = 20$和$n = 60$) 时, IGR 略有优势, 且在$n = 500$时, FFO 的 RPI 值与 QILS 接近, 但 QILS 在全局上的均衡性和稳定性更优, 进一步表明了 QILS 解决 UDPFSP/SDST 的有效性.

图 10 和图 11 展示了 QILS 与 4 种对比算法在 540 个测试数据集上的邓尼特 T3 检验结果. 在两种停止时间标准下, QILS 均显著优于 4 种对比算法, 验证了 QILS 解决 UDPFSP/SDST 的优越性能.

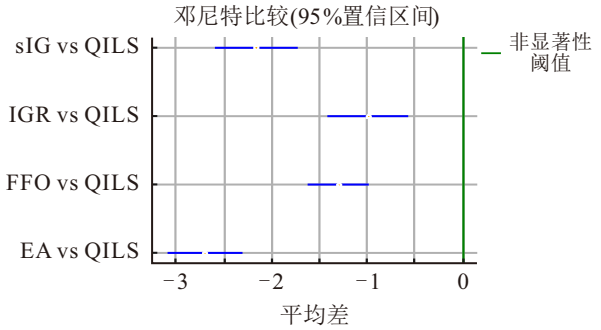

图10　邓尼特 T3 分析结果 ($\omega = 200$)

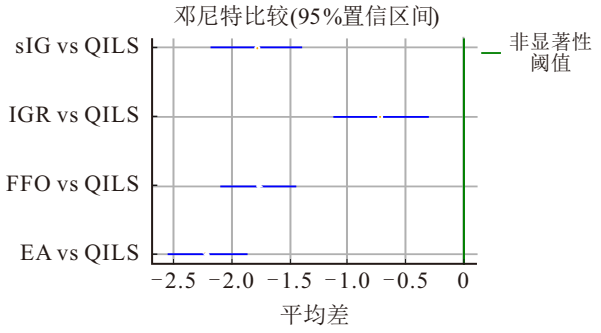

图11　邓尼特 T3 分析结果 ($\omega = 350$)

基于上述实验结果和分析, 可以得出结论: QILS 在解决 UDPFSP/SDST 问题时, 相较于其他先进算法, 展现出了更好的性能和稳定性.

4　结　论

为了应对分布式车间中的不确定性, 提高解的鲁棒性, 本文提出了强化学习驱动的迭代局部搜索算法 (QILS) 来解决具有序列相关准备时间的分布式车间鲁棒调度问题. QILS 通过 NEHUPT 启发式来产生高质量的初始解决方案, NEHUPT 基于不确定加工时间确定工件的调度优先级, 并采用微调策略进一步增强对解空间的搜索. 同时, QILS 引入Q-learning 算法, 实现扰动策略的自适应动态选择. QILS 通过学习历史搜索信息和当前状态, 智能评估并选择最优扰动策略, 显著提升了优化能力. 全面的仿真实验验证了Q-learning 驱动的扰动阶段的有效性, 它能够根据不同的问题实例和搜索状态, 自适应地调整扰动策略, 有效提升了算法的求解能力. 同时, 与其他先进算法的对比分析进一步表明了 QILS 解决 UDPFSP/SDST 的优越性和稳定性.

未来的研究工作可从以下几个方面进一步拓展: 在优化目标方面, 考虑多目标鲁棒调度模型, 联合优化生产效率、能源消耗、运营成本等综合指标, 更贴近实际生产要求[24]; 在问题建模方面, 引入工厂间资源异构性[25]、模糊加工[26–27] 等现实约束条件, 提升模型适应性; 在算法机制方面, 当前Q-learning 状态空间维度较低, 未来可探索深度强化学习或图神经网络等方法融合 ILS, 提高策略学习的泛化能力与长期决策效果.

参考文献 (References)

[1] Perez-Gonzalez P, Framinan J M. A review and classification on distributed permutation flowshop scheduling problems[J]. European Journal of Operational Research, 2024, 312(1): 1-21.

[2] 轩华, 李文婷, 李冰. 混合离散人工蜂群算法求解含不相关并行机的分布式柔性流水线调度[J]. 控制与决策, 2023, 38(3): 779-789.
(Xuan H, Li W T, Li B. Hybrid discrete artificial bee colony algorithm for distributed flexible flowline scheduling with unrelated parallel machines[J]. Control and Decision, 2023, 38(3): 779-789.)

[3] 雷德明, 苏斌. 基于多班教学优化的多目标分布式混合流水车间调度[J]. 控制与决策, 2021, 36(2): 303-313.
(Lei D M, Su B. Multi-class teaching-learning-based optimization for multi-objective distributed hybrid flow shop scheduling[J]. Control and Decision, 2021, 36(2): 303-313.)

[4] 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 problem[J]. Computers & Industrial Engineering, 2020, 142: 106347.

[5] Xiong F L, Chen S Y, Xiong N X, et al. Scheduling distributed heterogeneous non-permutation flowshop to minimize the total weighted tardiness[J]. Expert Systems with Applications, 2025, 272: 126713.

[6] Guo H W, Sang H Y, Sun X B, et al. An effective metaheuristic for the robust distributed flowshop scheduling problem[J]. Journal of Physics: Conference Series, 2022, 2258(1): 012029.

[7] Guo H W, Sang H Y, Zhang X J, et al. An effective fruit fly optimization algorithm for the distributed permutation flowshop scheduling problem with total

flowtime[J]. Engineering Applications of Artificial Intelligence, 2023, 123: 106347.

[8] Fernandez-Viagas V, Perez-Gonzalez P, Framinan J M. The distributed permutation flow shop to minimise the total flowtime[J]. Computers & Industrial Engineering, 2018, 118: 464-477.

[9] Guo H W, Sang H Y, Zhang B, et al. An effective metaheuristic with a differential flight strategy for the distributed permutation flowshop scheduling problem with sequence-dependent setup times[J]. Knowledge-Based Systems, 2022, 242: 108328.

[10] Huang J P, Pan Q K, Gao L. An effective iterated greedy method for the distributed permutation flowshop scheduling problem with sequence-dependent setup times[J]. Swarm and Evolutionary Computation, 2020, 59: 100742.

[11] Kouvelis P, Daniels R L, Vairaktarakis G. Robust scheduling of a two-machine flow shop with uncertain processing times[J]. IIE Transactions, 2000, 32(5): 421-432.

[12] Chen X, Li Y B, Wang K P, et al. Reinforcement learning for distributed hybrid flowshop scheduling problem with variable task splitting towards mass personalized manufacturing[J]. Journal of Manufacturing Systems, 2024, 76: 188-206.

[13] Zhao C, Wu L H. A multi-strategy fruit fly optimization algorithm for the distributed permutation flowshop scheduling problem with sequence-dependent setup times[J]. Applied Soft Computing, 2024, 167: 112436.

[14] Gorissen B L, Yanıkoğlu İ, den Hertog D. A practical guide to robust optimization[J]. Omega, 2015, 53: 124-137.

[15] Allahverdi A, Aydilek H. Heuristics for the two-machine flowshop scheduling problem to minimise makespan with bounded processing times[J]. International Journal of Production Research, 2010, 48(21): 6367-6385.

[16] Xu X Q, Cui W T, Lin J, et al. Robust makespan minimisation in identical parallel machine scheduling problem with interval data[J]. International Journal of Production Research, 2013, 51(12): 3532-3548.

[17] Jing X L, Pan Q K, Gao L. Local search-based metaheuristics for the robust distributed permutation flowshop problem[J]. Applied Soft Computing, 2021, 105: 107247.

[18] Wang Z Y, Pan Q K, Gao L, et al. A cooperative iterated greedy algorithm for the distributed flowshop group robust scheduling problem with uncertain processing times[J]. Swarm and Evolutionary Computation, 2023, 79: 101320.

[19] Wang B, Wang X Z, Lan F M, et al. A hybrid local-search algorithm for robust job-shop scheduling under scenarios[J]. Applied Soft Computing, 2018, 62: 259-271.

[20] Lv Y, Qian B, Hu R, et al. An enhanced cross-entropy algorithm for the green scheduling problem of steelmaking and continuous casting with uncertain processing time[J]. Computers & Industrial Engineering, 2022, 171: 108445.

[21] Li X P, Yang Z, Ruiz R, et al. An iterated greedy heuristic for no-wait flow shops with sequence dependent setup times, learning and forgetting effects[J]. Information Sciences, 2018, 453: 408-425.

[22] Miao Z H, Guo H W, Pan Q K, et al. A reinforcement learning-enhanced multi-objective iterated greedy algorithm for weeding-robot operation scheduling problems[J]. Expert Systems with Applications, 2025, 263: 125760.

[23] Guo H W, Miao Z H, Ji J, et al. An effective collaboration evolutionary algorithm for multi-robot task allocation and scheduling in a smart farm[J]. Knowledge-Based Systems, 2024, 289: 111474.

[24] Song H B, Lin J, Chen Y R. An effective two-stage heuristic for scheduling the distributed assembly flowshops with sequence dependent setup times[J]. Computers & Operations Research, 2025, 173: 106850.

[25] Meng L L, Zhang C Y, Zhang B, et al. MILP modeling and optimization of multi-objective flexible job shop scheduling problem with controllable processing times[J]. Swarm and Evolutionary Computation, 2023, 82: 101374.

[26] 郑小操, 龚文引. 改进人工蜂群算法求解模糊柔性车间调度问题[J]. 控制理论与应用, 2020, 37(6): 1284-1292.
(Zheng X C, Gong W Y. An improved artificial bee colony algorithm for fuzzy flexible job-shop scheduling problem[J]. Control Theory & Technology, 2020, 37(6): 1284-1292.)

[27] 李瑞, 龚文引. 改进 MOEA/D 算法求解双目标模糊柔性作业车间调度问题[J]. 控制理论与应用, 2022, 39(1): 31-41.
(Li R, Gong W Y. An improved multi-objective evolutionary algorithm based on decomposition for bi-objective fuzzy flexible job-shop scheduling problem[J]. Control Theory & Technology, 2022, 39(1): 31-41.)

作者简介

郭恒伟 (1996−), 男, 硕士生, 主要研究方向为智能优化算法、车间调度, E-mail: 15628837665@163.com;

桑红燕 (1981−), 女, 教授, 博士, 主要研究方向为智能计算、优化与应用, E-mail: sanghongyan@lcu-cs.com;

潘全科 (1971−), 男, 教授, 博士, 主要研究方向为智能计算、优化与应用, E-mail: panquanke@shu.edu.cn.