中国科学:信息科学 在审文章 onlinelatex

SCIENTIA SINICA Informationis

论文

基于深度强化学习和质量多样性优化的分布式柔性作业车间集成调度方法

秦浩翔[1], 向毅[1*], 韩玉艳[2], 王玉亭[2], 陈庆达[3,4], 周平[4]

1. 华南理工大学软件学院, 广州 510006
2. 聊城大学计算机学院, 聊城 252000
3. 内蒙古大学电子信息工程学院, 呼和浩特 010021
4. 东北大学 流程工业综合自动化国家重点实验室, 沈阳 110004
* 通信作者. E-mail: xiangyi@scut.edu.cn

广东省基础与应用基础研究基金 (批准号: 2024A1515030022)、贵州省教育厅自然科学研究项目 (批准号: QJJ2023061) 、中央高校基本科研业务费 (批准号:2024ZYGXZR097、93K172024K03)、国家自然科学基金 (批准号: 61906069、62203101)、山东省自然科学基金 (批准号:ZR2023MF022、ZR2024MF112)、聊城大学光岳青年学者创新团队 (批准号:LCUGYTD2022-03)

摘要 在智能制造领域, 自动导引车与机器的集成调度对完工时间与能源消耗具有重大影响. 然而, 现有调度方法难以高效协调自动导引车 (Automated Guided Vehicle, AGV) 运输与作业执行, 尤其在分布式柔性作业车间调度场景下, 该问题尤为突出. 为此, 本文提出一种深度强化学习增强的质量多样性优化算法, 以充分利用运输与机器的行为特征, 生成高质量且多样化的帕累托解集. 首先, 针对 AGV 运输特性, 设计知识辅助的协作启发式策略, 以优化作业、机器和工厂的调度, 提升解的整体质量. 其次, 针对搜索算子利用率低的问题, 提出基于深度强化学习的智能选择机制, 以克服随机选择的局限性, 提高搜索效率与优化质量. 仿真实验表明, 所提算法在完工时间与能耗优化方面均显著优于现有方法, 验证了其有效性.

关键词 分布式柔性作业车间调度, 自动导引车运输, 质量-多样性优化, 深度强化学习, 多目标优化

1 引言

随着智能制造的发展, 生产调度在企业运营中的作用日益突出[1]. 柔性作业车间调度问题（Flexible Job-Shop Scheduling Problem, FJSP）是一类 NP-hard 的离散组合优化问题, 其相较于传统调度问题更具复杂性[2]. 在传统车间调度问题中，每道工序的加工机器和加工顺序是预先固定的, 而 FJSP 允许每道工序在多个可选机器上进行加工, 并且不同机器的加工时间可能不同. 尽管该生产模式提升了灵活性，但也加大了求解难度. 自动导引车 (automated guided vehicle, AGV) 作为智能制造的重要组成部分, 已被广泛应用于 FJSP, 以提高调度的自动化水平[3,4]. 此外，随着制造需求的提

引用格式: 秦浩翔, 向毅, 韩玉艳, 等. 基于深度强化学习和质量多样性优化的分布式柔性作业车间集成调度方法. 中国科学: 信息科学, 在审文章
Qin H X, Xiang Y, Han Y Y, et al. Intelligent scheduling approach for distributed flexible job-shop with deep reinforcement learning and quality-diversity optimization (in Chinese). Sci Sin Inform, for review

升，分布式柔性作业车间调度 (distributed FJSP, DJFSP) 受到广泛关注. 相比 FJSP, DFJSP 涉及多个加工工厂，具备更高的调度灵活性, 能够快速响应紧急生产任务, 提升制造系统的响应能力 [5].

在 DFJSP-AGVs 中, AGV 调度与机器加工和作业顺序高度耦合. AGV 调度直接影响后续作业的开始时间, 而作业加工又反过来决定 AGV 运输调度时机 [6,7]. 因此, 调度系统必须在制定运输方案时评估其可行性. 若 AGV 调度不合理，不仅延长完工时间，还会显著增加能耗 [8]. 为此，设计高效的智能算法以实现 AGV 与生产调度的协同优化，对智能制造系统至关重要. 目前，大多数研究仍将 DFJSP 与 AGV 调度分开处理 [9]. 在 DFJSP 中, 运输时间常被简化处理，如并入加工时间或假设运输资源无限 [10]; 而 AGV 调度研究则主要聚焦于任务分配和路径规划 [11]. 显然, 上述两种方法均未充分反映生产调度与 AGV 运输之间复杂的耦合关系及协同作用. 尽管联合优化可有效提升生产效率, 但现有研究主要聚焦于作业调度, 对有限数量 AGV 设备的调度研究十分有限 [12~15].

此外，现有研究多集中于最小化完工时间，而对集成调度中的能耗优化关注较少. 在实际生产中, AGV 运输能耗在总能耗中占比较高，仅次于加工能耗，且对整体调度性能影响显著 [16]. 多个案例验证了其重要性：在南京的机械加工厂中, 优化 AGV 调度显著降低了能耗并且提升了生产效率 [16]. 另有案例显示，运输调度缺陷导致能源浪费，进而影响生产进度 [17]. 此外, 在电机设备制造厂中, 优化 AGV 策略有效减少了能耗与完工时间 [3]; 在安全监控机器人组件的生产中, AGV 调度亦对能耗与效率产生重要影响 [18]. 综上, 优化 AGV 运输能耗在集成调度中具有重要的理论与实践价值.

为求解 FJSP 和 DFJSP, 已有研究主要包括数学方法[19~21] 和元启发式算法[22~26], 但大多未涉及 AGV 运输. 在考虑 AGV 约束的 FJSP 研究中，主要采用元启发式方法[27~36]. 其中, 部分研究 [27~30] 未考虑 AGV 能耗, 另有研究忽略分布式调度环境 [31~33]. 尽管已有工作开始关注分布式场景中的运输优化[34,35], 但多聚焦于起重机而非 AGV. Luo 等 [36] 提出用于作业排序、工厂分配和机器分配的扰动策略, 但缺乏针对 AGV 分配的优化机制.

总体而言, 当前针对 DFJSP-AGVs 的研究仍较有限, 且普遍忽视 AGV 调度对能耗与生产周期的影响. 多数方法未能充分利用 AGV 的调度特性，可能导致在完工时间和能耗上的次优结果. 同时，现有基于随机选择的搜索机制缺乏对 AGV 状态的有效建模，易陷入局部最优，且搜索效率不高 [37~39]. 因此, 亟需开发高效、面向 AGV 特性的优化算法和搜索策略.

针对上述问题,本文提出一种融合深度 Q 网络 (Deep-Q-Network, DQN) 与质量多样性 (Quality-Diversity, QD) 优化的演化算法 DQN-QD. QD 已在理论上被证实能够有效规避局部最优, 并在多样性探索上优于多数传统演化算法[40]. 该方法最初主要应用于机器人与游戏领域, 并已在《Nature》期刊发表相关研究[41,42], 其坚实的理论基础与实践验证提供了可靠支撑. 此外，在 DFJSP-AGVs 中，机器加工与 AGV 运输高度耦合，共同影响工序的完工时间与能耗. QD 依赖明确的行为特征构建特征空间以存储帕累托解集，机器空闲与 AGV 运输可自然嵌入为关键行为特征，提供结构清晰、判别性强的特征基础. DQN 用于辅助 QD 优化，依托其通过深度神经网络建模环境的能力，可自适应学习调度策略 [32,43]，相比传统方法和 Q 学习等强化学习方法具备更强的策略学习与泛化能力. 鉴于 DFJSP-AGVs 的非线性与高耦合特性，DQN 通过累积回报机制有效引导搜索方向，提升搜索效率并降低无效探索. 此外，本文进一步设计了知识辅助的协作启发式搜索策略，有效提升了算法的收敛速度与解的质量. 实验结果表明, 所提算法在性能上较现有先进方法提升约 4% 至 15%. 综上所述, 本文的主要贡献如下:

- 本文针对 AGV 运输与 DFJSP 的集成调度问题展开研究, 旨在同时优化能耗与完工时间. 通过引入 AGV 运输特性作为关键特征, 促进算法在帕累托前沿上的高质量解集构建与维护. 基于 AGV 运输特性, DQN-QD 在特征空间内探索更广泛且多样化的帕累托解集, 进一步提升调度方案的优化

质量与鲁棒性.

• 本文设计了一种领域知识辅助的协作启发式搜索策略, 并协同优化作业、机器和工厂的调度过程, 以全面提升解的质量. 实验结果表明, 相较于传统搜索算子, 该搜索策略在优化过程中表现出更优收敛性, 验证了其在集成调度优化中的有效性.

• 为提升算法的搜索能力并规避搜索算子的低效利用, 本文提出一种基于 DQN 的自适应算子选择机制. 该机制将 AGV、机器、作业和工厂的状态向量作为网络输入, 以强化搜索决策的智能性. 通过动态学习与调整算子选择策略, 该方法有效规避了随机选择可能导致的无效搜索, 提升了优化过程的全局探索能力与局部开发效率.

2　问题描述

DFJSP-AGVs 涉及协同作业与 AGV 运输优化, 具有作业流程复杂、资源分配约束多、任务协同要求高、运输优化难度大等特点. 在 DFJSP-AGVs 中, 需在确保生产任务按计划完成的前提下, 优化机器与 AGV 的调度方案, 提高设备利用率, 从而降低能源消耗和完工时间.

2.1　带 AGV 运输的分布式柔性作业车间集成调度问题

针对 DFJSP-AGVs, 需要解决以下四个关键任务: 1) 将作业分配至各个工厂; 2) 确定工序顺序; 3) 将作业的工序分配至具体机器; 4) 确定 AGV 运输的作业. 令 $F = \{1, ...f, ...l\}$ 表示由 l 个工厂组成的集合, 每个工厂内部均为包含 m 台机器的 FJSP, 机器集合表示为 $M = \{1, ...k, ...m\}$. 考虑 n 个作业 $I = \{1, ...i, ...n\}$, 其中, 每个作业 i 包含 w_i 个工序 $J_i = \{1, ...j, ...w_i\}$. 所有作业需要通过 v 台 AGV 进行运输, AGV 集合表示为 $A = \{1, ...a, ...v\}$. 若作业的相邻任务在同一台机器上加工, 则无需 AGV 运输. 此外, 工序 $O_{i,j}$ 在工厂 f 的机器 k 上的加工时间表示为 $T_{i,j,f,k}$.

本文假设所有工厂、机器、作业及 AGV 在调度开始时均可用. 每个作业必须分配至唯一的工厂, 且在每个工厂内, 每台机器在任意时刻只能加工一道工序. 工序的加工必须在单台机器上完成, 一旦开始加工, 不允许被中断或提前终止. 此外, 每台 AGV 每次只能运输一个作业, 且每个作业的运输仅能由一台 AGV 运输.

为更直观地说明该问题, 本文给出一个示例, 其中，单个工厂内的作业数 $n = 5$, 机器数 $m = 4$, AGV 数量 $v = 2$, 最大工序数 $n_{max} = 18$. 表1列出了各工序的机器分配及其对应的加工时间, AGV 在任意两台机器之间的运输时间详见表2. 图1展示了单个工厂内可行的调度序列的甘特图, 其他工厂的调度示意方式类似. 在图1中, 同一作业的所有工序使用相同颜色表示, 不同颜色的虚线表示不同的 AGV. 当新的作业需要加工时, AGV 会前往存储站取作业, 并将其运输至指定的机器. 此外, 在完成作业运输后, AGV 会立即前往下一步任务所需的机器, 或在当前机器上等待 (若下一道工序仍需在同一台机器上进行).

由于篇幅限制, DFJSP-AGVs 的符号定义和约束条件已在补充材料中提供. 其目标函数定义如下:

本文的目标是同时优化最大完工时间 obj_1 和总能耗 obj_2. TPE 表示作业总加工能耗,TIE 表示机器总空闲能耗, TTE 表示 AGV 总运输能耗.

$$Minimize\ obj_1 = C_{max}, \tag{1}$$

https://www.sciengine.com/doi/10.1360/SSI-2025-0113

表 1　所有工序分配的机器和加工时间

Table 1　The machines and processing time of each operation

O_{ij}	M/T_{ij}	O_{ij}	M/T_{ij}	O_{ij}	M/T_{ij}
O_{11}	*M1/5*	O_{22}	*M1/4*	O_{41}	*M2/6*
O_{12}	*M2/15*	O_{23}	*M4/3*	O_{42}	*M1/11*
O_{13}	*M1/5*	O_{31}	*M3/7*	O_{43}	*M3/5*
O_{14}	*M2/7*	O_{32}	*M3/10*	O_{51}	*M4/11*
O_{15}	*M1/3*	O_{33}	*M1/6*	O_{52}	*M2/6*
O_{21}	*M1/5*	O_{34}	*M2/4*	O_{53}	*M1/8*

表 2　任意两台机器之间的运输时间

Table 2　The transportation time between any two machines

Machine	*Depot*	*M1*	*M2*	*M3*	*M4*
Depot	0	5	7	5	3
M1	1	0	4	5	9
M2	13	4	0	6	5
M3	5	6	7	0	3
M4	7	3	6	3	0

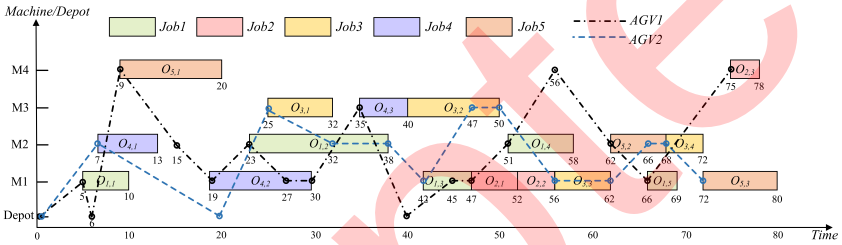


图 1　单个工厂的甘特图，其他工厂与此结构类似

Figure 1　Gantt chart of a single factory. Other factories are similar to this structure

$$Minimize\ obj_2 = TPE + TIE + TTE, \tag{2}$$

(1) 加工所有作业的总能耗计算如下，其中 PP_k 表示单位时间内机器 k 的加工效率，$TP_{f,k}$ 表示工厂 f 中机器 k 的总加工时间.

$$TPE = \sum_{f\in F}\sum_{k\in M}(PP_k \cdot TP_{f,k}), \tag{3}$$

(2) 机器空闲状态下处理所有机器的总能耗计算如下，其中 PI_k 表示机器闲置状态下单位时间内机器 k 的空闲功率. $TI_{f,k}$ 表示工厂 f 中机器 k 的总空闲时间.

$$TIE = \sum_{f\in F}\sum_{k\in M}(PI_k \cdot TI_{f,k}), \tag{4}$$

(3) AGV 运输的总能耗计算如下，其中 PT_a 表示 AGV a 在单位时间内的运输功率，$TO_{i,j,a}$ 表示 AGV a 将工序 $O_(i,j)$ 运输到机器 k 上加工所需的时间.

$$TTE = \sum_{a\in A}\left(PT_a \cdot \sum_{i'\in I}\sum_{j'\in J_{i'}} TO_{i,j,a}\right), \tag{5}$$

3　基于深度 Q 网络的质量-多样性优化算法

为求解 DFJSP-AGVs, 本文设计了一种基于深度 Q 网络的质量-多样性优化算法. 本文选择最著名 QD 算法之一: MAP-Elites[44], 作为基本的算法框架. MAP-Elites 将特征空间离散化为由多个单元组成的网格, 并在每个单元中存储当前最优解供调度决策者选择. 接下来, 本文将详细介绍该算法的实现过程, 包括算法框架、编码与解码方式、评估与更新机制、深度 Q 网络 (Deep Q-Networks) 以及协作启发式搜索策略的设计.

https://www.sciengine.com/doi/10.1360/SSI-2025-0113

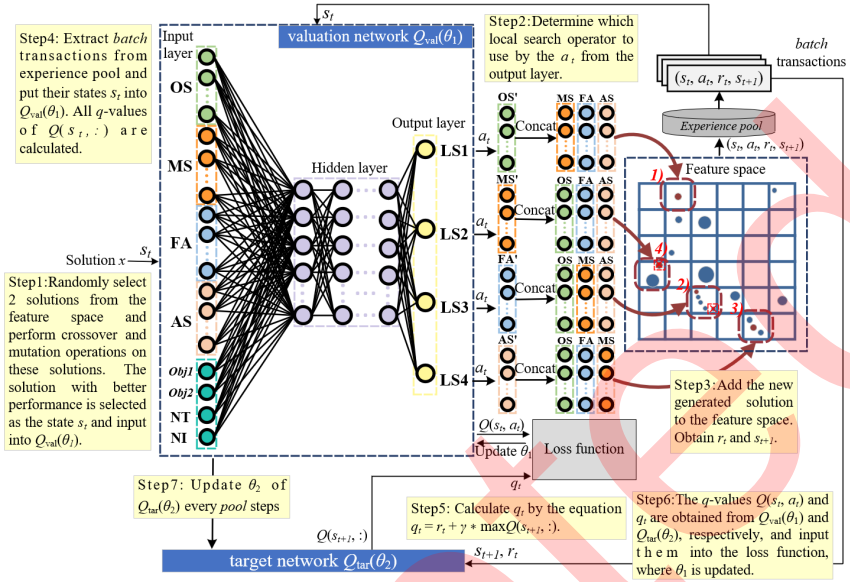


图 2　解的更新过程（步骤 1-3）及网络训练过程（步骤 4-7）示意图. 图中展示了启发式算子执行后，解在特征空间中更新的四种典型情形：1）若网格单元为空，则新生成的解直接加入；2）若网格单元已满，且存储的解互不支配，则在新解优于当前任一解的情况下，随机替换一个已有解；3）若单元尚未存满，且新解优于已有解，则将其直接加入；4）若新解不优于当前任何解，则直接丢弃.

Figure 2　Illustration of the solution update process (Steps 1–3) and the network training process (Steps 4–7). This figure outlines four typical scenarios for updating solutions in the feature space after applying heuristic search operators: 1) If the grid cell is empty, the new solution is directly inserted; 2) If the cell is full and all stored solutions are non-dominated, the new solution replaces a randomly selected one only if it demonstrates better performance; 3) If the cell is not full and the new solution dominates existing ones, it is directly added; 4) Otherwise, the new solution is discarded.

3.1　算法框架

在 DFJSP-AGVs 中, 影响完工时间和能耗的两个关键特征是 AGV 运输次数与机器空闲次数. 通过搜索不同特征组合的解, 可以进一步优化目标函数值. QD 方法能够充分利用上述两个问题特性, 以搜索高质量且多样化的解决方案. 为此, 本文引入 QD 框架, 并以 AGV 运输次数和机器空闲次数作为组成特征空间的两个维度. QD 通过将特征空间离散化为网格, 在每个单元中存储帕累托解集, 并在该特征空间内执行所有更新, 具体示意如图2所示. 此外, 本文在 QD 框架中集成 DQN, 以智能引导决策过程, 进一步增强算法的搜索能力与优化效率.

DQN-QD 框架如算法1所示, 其中输入包括网格维度 N, 解集大小 $batch$, 学习因子 α, 折扣因子 γ, 经验池 S_E, 贪心因子 ϵ 及 $Epoch$ 参数. 输出包括特征-性能网格, 其中 $\mathcal{P}(\boldsymbol{b}_x)$ 和 $\mathcal{X}(\boldsymbol{b}_x)$ 分别表示网格坐标 $\boldsymbol{b}_x$ 处的目标值存储和帕累托解集. 在算法1中, 第 2-4 行表示初始化 ps 个解并将其添加到特征空间. 其中, 函数 Add_to_Grid $(\mathcal{P}, \mathcal{X}, x)$ (算法3) 用于评估初始解, 其中 $'*'$ 表示占位符, 没有实际意义. 第 5 行表示使用参数 $batch$, α, γ, ϵ, S_E, $Epoch$ 创建估值网络 $Q_{val}(\theta_1)$ 和目标网络 $Q_{tar}(\theta_2)$. 第 8 行表示选择两个解 x 和 x' 进行改进. 其中, 解 x 从特征空间中的非支配解集 (所有单元内互不支配的解集合) 中选取, 而解 x' 则随机从特征空间或非支配解集中选取. 随后, 采用前序交叉算子、均匀交叉算子及变异策略[43] 对解 x 和 x' 进行扰动, 并保留质量较优的解. 若两个解互不支配, 则随机选择一个解作为 x'. 第 10 行表示将 x 作为当前状态 s_t 输入 DQN, 随后基于 q 值选择策略, 以执行动作 a_t. 动作 a_t 的选择受参数 ϵ 影响: 若随机数 $rand$ 小于 ϵ, 则从估值网络 $Q_{val}(\theta_1)$ 中选择 q 值最高的动作 a_t; 否则, 从所有动作集中随机选择一个 a_t (11 行). 执行 a_t 后生成新解 x' (12 行), 并将其作为下一个状态 s_{t+1}. 第 13 行将事务 (s_t, a_t, r_t, s_{t+1}) 存入经验池中. 随后, 训练估值网络 $Q_{val}(\theta_1)$ 并更新目标网络 $Q_{tar}(\theta_2)$. 详细过程可参考算法4.

https://www.sciengine.com/doi/10.1360/SSI-2025-0113

算法 1 基于深度 Q 网络的质量-多样性优化算法

输入: $N, batch, \alpha, \gamma, \epsilon, S_E, Epoch$

初始化解集 $\mathcal{X} \leftarrow \emptyset$ 和特征网格 $\mathcal{P}$, 生成 N 维网格
for $i = 1$ **to** ps **do**
    $\{\mathcal{P}, \mathcal{X}, *\} \leftarrow$ Add_to_grid($\mathcal{P}, \mathcal{X}$, random_solution()) // 算法 3
end for
初始化估值网络 $Q_{val}(\theta_1)$ 和目标网络 $Q_{tar}(\theta_2)$
while 终止条件未满足 **do**
    for $i = 1$ **to** $batch$ **do**
        选择两个解 x 和 x', 利用交叉变异策略生成新解 y 和 z, 并判断其支配关系，选择较优的解作为 x'
        $\{\mathcal{P}, \mathcal{X}, r_t\} \leftarrow$ Add_to_grid($\mathcal{P}, \mathcal{X}, x'$)
        $s_t \leftarrow x$, $s_{t+1} \leftarrow x'$
        $a_t \leftarrow \arg\max Q_{val}(\theta_1, s_t)$ 或随机选择
        执行 a_t, 生成新解 x'', 并更新 $\mathcal{P}, \mathcal{X}$
        存储 (s_t, a_t, r_t, s_{t+1}) 至经验池
        训练 $Q_{val}(\theta_1)$ 并更新 $Q_{tar}(\theta_2)$ // 算法 4
    end for
end while

输出: 特征-性能网格 ($\mathcal{X}$ 和 $\mathcal{P}$)

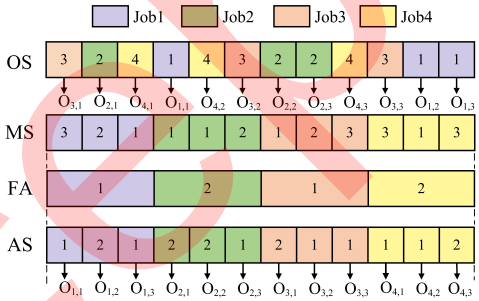


图 3　可行解的编码向量示意图

Figure 3　The encoding vectors of a feasible solution

3.2　评估与更新

在评估解决方案之前, 本文首先介绍 DFJSP-AGVs 的编码方案. 与文献 [45, 46] 相同, 本文采用整数编码方式来表示解. 该编码方案由四个向量组成: 1) 操作序列 (Operation Sequence, OS), 2) 机器选择 (Machine Selection, MS), 3) 工厂分配 (Factory Assignment, FA), 和 4) AGV 选择 (AGV Selection, AS). 如图3所示, 矩形的不同颜色代表不同的作业. FA 向量的长度等于作业总数 n, 而 OS, MS 和 AS 向量的长度均等于总工序数 n_{max}. 对于相同的工序 $O_{i,j}$, 其可选的机器集合相同, 但在不同工厂中的加工时间可能不同. 通过将这四个向量整合到解码方案中, 可对其进行评估.

如算法2所示, x 表示完整的解, 由 OS, MS, FA 和 AS 四个向量衔接而成. 待优化的两个目标完工时间和能源消耗分别用 obj_1 和 obj_2 表示. 其中, $\boldsymbol{b}_x$ 表示特征空间内与 NI 和 NT 对应的坐标, 其中 NI 表示机器空闲次数, NT 表示 AGV 运输次数. NI 和 NT 均为整数, 其取值范围分别为 0 至 n_{max} 以及 0 至 $2n_{max}$ (考虑 AGV 返回停靠点的情况). 因此工序数决定了特征空间的大小，一旦特征空间构建完成，其大小不变. 此外, 为方便阐述，本文额外定义五个符号: AT_a, AP_a, $P_{f,k}$, $U_{i,j}$ 和 $Start_{i,j}$. 其中, AT_a 表示 AGV a 在当前调度方案中的累计运输时间, AP_a 表示 AGV a 的当前位置, $P_{f,k}$ 表示工厂 f 中机器 k 的可用时间集合, $U_{i,j}$ 表示执行工序 $O_{i,j}$ 的机器, 而 $Start_{i,j}$ 表示工序 $O_{i,j}$ 的开始加工时间.

第 3 行描述了对工序 $O_{i,j}$ 所分配的工厂、机器和 AGV 的检索过程. 第 4 行表示指定 AGV 的初始运输时间为零, 并将其位置设为调度站 ($U_{i,j} = 0$ 表示调度站). 随后, 第 6 行表示计算 AGV 的运输时间 AT_a 并确定 AGV 完成初始作业后停留的机器 AP_a (第 7 行). 第 9 行计算非第一道工序的 AGV 的运输时间. 第 11 计算 AGV 的能耗和运输次数. 第 12-13 行计算机器 $U_{i,j}$ 的能耗和空闲时间. 随后确定工序 $O_{i,j}$ 的开始时间 $Start_{i,j}$ (14 行). 第 15

https://www.sciengine.com/doi/10.1360/SSI-2025-0113

算法 2 评估方法

输入: 解决方案 x 由 OS, MS, FA, AS 向量组成

1: 初始化 $NI \leftarrow 0, NT \leftarrow 0$
2: **for** $pos \leftarrow 1$ **to** n_{max} **do**
3: 获取 $O_{i,j}$, 其工厂 f、机器 $U_{i,j}$ 和 AGV a
4: 计算 $AT_a \leftarrow 0$, $AP_a \leftarrow 0$
5: **if** $j = 1$ **then**
6: $AT_a \leftarrow TR_{AP_a,0} + TR_{0,U_{i,j}}$
7: $AP_a \leftarrow U_{i,j}$
8: **else**
9: $AT_a \leftarrow \max(AT_a + TR_{AP_a,U_{i,j-1}}, C_{i,j-1}) + TR_{U_{i,j-1},U_{i,j}}$
10: **end if**
11: 计算 AGV 运输能耗 (公式 5) 并累加 NT
12: **if** $AT_a > P_{f,U_{i,j}}$ **then**
计算机器空闲能耗 (公式 4), 更新 NI
13: **end if**
14: 计算 $Start_{i,j} \leftarrow \max(AT_a, P_{f,U_{i,j}})$; $AP_a \leftarrow U_{i,j}$
15: $C_{i,j} \leftarrow Start_{i,j} + T_{i,j,f,U_{i,j}}$; $P_{f,U_{i,j}} \leftarrow C_{i,j}$
16: 计算加工能耗 (公式 3), 更新 $P_{f,U_{i,j}}$ 和 AP_a
17: **end for**
18: 计算目标值 obj_1 和 obj_2 (公式 1-2)
19: $\boldsymbol{b}_x \leftarrow (NT, NI)$

输出: $\{obj_1, obj_2, \boldsymbol{b}_x\}$

行表示计算 $O_{i,j}$ 的完工时间 $C_{i,j}$. 第 16 行计算了加工能耗, 以及更新了该机器后续作业的可用时间和 AGV 位置. 在完成评估后 (第 18-19 行), 可获得该解在特征空间内对应的帕累托前沿坐标 $\boldsymbol{b}_x$ 及其两个目标值.

随后, 需确定是否将该解添加到特征空间中. 具体的更新流程如算法 3 所示. 若单元 $\mathcal{P}(\boldsymbol{b}_x)$ 为空, 则将解 x 添加到特征空间中, 奖励 r_t 设置为 1 (第 3 行). 若 $\mathcal{P}(\boldsymbol{b}_x)$ 为非空, 则查看该单元内的解数量. 若 x 相对于所有解均占优 (记作 $\prec$) 或者非支配 (记作 $\prec_d$), 且单元 $\boldsymbol{b}_x$ 内解的数量小于 P, 则将 x 添加到特征空间, 并设定奖励为 1 (第 6 行). 若 $\boldsymbol{b}_x$ 内的解数量已达上限 P, 则随机替换该单元中的一个解 x, 并给予 0.8 的奖励 (第 9 行). 否则, 若 x 被单元 $\boldsymbol{b}_x$ 中的旧解支配, 则不进行更新, 并设定奖励为 0 (第 11 行). 解的更新过程如图 2 中步骤 3 所示.

算法 3 更新方法

输入: $\mathcal{P}, \mathcal{X}, x$

1: $\{obj_1, obj_2, \boldsymbol{b}_x\} \leftarrow evaluate(x)$ // 算法 2
2: **if** $\mathcal{P}(\boldsymbol{b}_x) = \emptyset$ **then**
3: $\mathcal{P}(\boldsymbol{b}_x) \leftarrow \{obj_1, obj_2\}$; $\mathcal{X}(\boldsymbol{b}_x) \leftarrow x$; $r_t \leftarrow 1$
4: **else**
5: **if** $(\mathcal{X}(\boldsymbol{b}_x) \prec x \ or \ \mathcal{X}(\boldsymbol{b}_x) \prec_d x) \ and \ |\mathcal{X}(\boldsymbol{b}_x)| < \delta$ **then**
6: $\mathcal{P}(\boldsymbol{b}_x) \leftarrow \{obj_1, obj_2\}$; $\mathcal{X}(\boldsymbol{b}_x) \leftarrow x$; $r_t \leftarrow 1$
7: **else if** $(\mathcal{X}(\boldsymbol{b}_x) \prec x \ or \ \mathcal{X}(\boldsymbol{b}_x) \prec_d x) \ and \ |\mathcal{X}(\boldsymbol{b}_x)| \geqslant \delta$ **then**
8: $\mathcal{P}^{[\kappa]}(\boldsymbol{b}_x) \leftarrow \{obj_1, obj_2\}$; // $[\kappa]$ 代表 $\mathcal{P}(\boldsymbol{b}_x)$ 中的一个随机位置
9: $\mathcal{X}^{[\kappa]}(\boldsymbol{b}_x) \leftarrow x$; $r_t \leftarrow 0.8$
10: **else**
11: $r_t \leftarrow 0$
12: **end if**
13: **end if**

输出: $\{\mathcal{P}, \mathcal{X}, r_t\}$

3.3 深度 Q 网络

DQN 的训练过程如下: 在 DQN 架构中, 事务 (s_t,a_t,r_t,s_{t+1}) 由当前状态 s_t、动作 a_t、奖励 r_t 和下一状态 s_{t+1} 组成. 在每个决策步 t 开始时, DQN 智能体感知当前状态 s_t 并启动决策过程, 该过程以未调度的 OS、MS、FA 和

https://www.sciengine.com/doi/10.1360/SSI-2025-0113

AS 向量作为输入, 并从当前时间步开始执行. 随后, 智能体选择适当的动作 a_t(即启发式搜索算子) 并执行该操作, 进而生成新的状态 s_{t+1}. 该状态对应一个新的解, 从而推动网络进入下一个决策步 $t+1$. 相关的决策过程如下所述:

状态 *(State):* 如图2的步骤 1 所示, 在决策步 t 时, 状态 s_t 由 OS、MS、FA 和 AS 向量构成. 在本研究中, 通过交叉变异策略生成的新解被视为状态 s_t, 并输入至 DQN 的输入层.

动作 *(Action):* 如图2的步骤 2 所示, DQN 选择的动作是启发式搜索算子 LS1-4, 其分别基于四种问题特性 (作业、机器、工厂和 AGV) 进行设计. 在每次迭代中, DQN 需选择一个针对某个启发式局部搜索算子, 以调整相应的调度序列. 关于 LS1-4 的具体细节将在下一小节详细介绍.

事务 *(Transition):* 当智能体完成动作 a_t 选择后, 该动作随即执行, 进而得到下一状态 s_{t+1}(即新的解). 此时, 启发式搜索算子被应用于优化解的质量.

反馈 *(Reward):* 根据特征空间中的解更新机制 (见算法3), 不同情况下的反馈奖励值 r_t 有所不同. 值得注意的是, 在生成初始解或使用交叉变异策略对特征空间内的解进行更新时, 无需计算 r_t.

网络结构 *(Network structure):* 参考文献 [43], 估值网络 $Q_{val}(\theta_1)$ 和目标网络 $Q_{tar}(\theta_2)$ 具有相同的结构. 两者均由 6 个全连接层组成. $n^{[k]}$ 表示第 k 层的输入维度, $m^{[k]}$ 表示第 k 层的输出维度. 网络结构如下: $n^{[1]}$: In, $m^{[1]}$: 128; $n^{[2]}$: 128, $m^{[2]}$: 256; $n^{[3]}$: 256, $m^{[3]}$: 128; $n^{[4]}$: 128, $m^{[4]}$: 64; $n^{[5]}$: 64, $m^{[5]}$: 32; $n^{[6]}$: 32, $m^{[6]}$: Out. 其中, In 表示 OS、MS、FA、AS 向量、两个优化目标及两个特征坐标的总长度,Out 表示启发式搜索算子的数量. 相邻层之间采用整流线性单元 (Rectified Linear Unit, $ReLU$) 作为激活函数.

算法 4 DQN 训练过程

输入: $batch$, $Q_{val}(\theta_1)$, $Q_{tar}(\theta_2)$, γ, S_E, $Epoch$

1: **for** $epoch \leftarrow 1$ **to** $Epoch$ **do**
2:　从经验池 S_E 随机选择 $batch$ 组事务
3:　提取事务中的 s_t, a_t, r_t, s_{t+1}
4:　通过 $Q_{tar}(\theta_2)$ 使用公式 (6) 计算目标 q 值 q_t
5:　通过 $Q_{val}(\theta_1)$ 计算预测的 $Q(s_t, a_t)$
6:　使用公式 (7) 计算损失函数
7:　通过 $Adam$ 优化器最小化损失函数, 更新网络参数 θ_1
8:　// 计算损失函数相对于 θ_1 的梯度, 并沿着降低损失的方向更新 θ_1
9:　**if** 学习步数大于 S_E **then**
10:　　$\theta_2 \leftarrow \theta_1$
11:　**end if**
12: **end for**

输出: $Q_{val}(\theta_1)$ 和 $Q_{tar}(\theta_2)$ 网络

DQN 的训练: 算法4 展示了 DQN 的训练过程. 输入包括解批量大小 $batch$、估值网络 $Q_{val}(\theta_1)$、目标网络 $Q_{tar}(\theta_2)$ 以及网络参数 γ、S_E 和 $Epoch$. 训练输出的是更新后的网络 $Q_{val}(\theta_1)$ 和 $Q_{tar}(\theta_2)$. 第 2-3 行表示从经验池 S_E 中随机选取 $batch$ 组事务 (s_t, a_t, r_t, s_{t+1}). 第 4 行表示目标网络 $Q_{tar}(\theta_2)$ 通过公式 6 计算目标 q 值 q_t:

$$q_t = r_t + \gamma * \max Q\left(s_{t+1}, :\right), \tag{6}$$

其中, q_t 表示时间步 t 处的目标 q 值, r_t 是时间步 t 处的即时奖励. 折扣因子 γ 用于平衡即时奖励与未来奖励, 以确定在当前决策中未来奖励的重要性. $\max Q\left(s_{t+1}, :\right)$ 代表在状态 s_{t+1} 下所有可能动作的最大期望回报. 随后, 估值网络 $Q_{val}(\theta_1)$ 计算 $Q(s_t, a_t)$ (第 5 行). 在获得 $Q(s_t, a_t)$ 和 q_t 后, 通过损失函数衡量它们之间的差异, 该损失函数定义如下 (公式 7):

$$J\left(\theta_1\right) = \frac{1}{batch} \sum_{t=1}^{batch} \left(Q\left(s_t, a_t\right) - q_t\right)^2, \tag{7}$$

其中, $J\left(\theta 1\right)$ 衡量在一个批量训练样本中, 预测 q 值与目标 q 值之间的平均平方误差. 训练的目标是最小化该损失函数. 其中, $Q(s_t, a_t)$ 由估值网络 $Q_{val}(\theta_1)$ 提供, 代表基于当前策略, 在状态 s_t 采取动作 a_t 的估计回报. q_t 则是由目标网络 $Q_{tar}(\theta_2)$ 计算得到的目标值.

https://www.sciengine.com/doi/10.1360/SSI-2025-0113

第 7 行表示计算损失后, 更新估值网络 $Q_{val}(\theta_1)$ 的参数 θ_1. 需要注意的是, 当学习步数超过 S_E 时, 目标网络的参数 θ_2 也需要使用 θ_1 进行更新 (第 9-11 行). 解决方案与网络更新的额外细节在图 2 中进一步说明.

3.4　知识辅助的协作启发式搜索策略

知识辅助的协作启发式搜索策略包括四种启发式局部搜索 (Local Search, LS) 算子. 搜索算子的应用意味着 DQN 执行某个动作 a_t. 其中, *LS1* 基于作业排序设计, *LS2* 针对作业的机器选择, *LS3* 作用于工厂分配, 而 *LS4* 则用于 AGV 运输调度. 通过协同应用上述四种算子, 可以高效调整 OS, MS, FA 和 AS 向量排序, 进而优化调度方案.

LS1: 在工厂 f 内随机选择两个不同作业的工序, 交换其在 OS 向量中的位置.

LS2: 首先从 OS 向量中随机选择一个工序 $O_{i,j}$, 并从 MS 向量中获取其分配的机器. 其次, 确定所有可加工该工序的机器集合 $M_{i,j,f}$. 若存在多台可选机器, 则重新分配该工序的加工机器. 最后, 在 MS 向量中更新该工序的机器编号.

LS3: 首先识别关键工厂 f (即完工时间最长的工厂). 随后, 从该工厂中选取一个作业, 并与一个非关键工厂中随机选择的作业进行交换. 该交换必须满足同一作业的所有工序必须在同一工厂内加工的约束. 交换完成时, 相应作业在 FA 向量中的工厂分配信息也随之更新.

LS4: 随机选择一个工序 $O_{i,j}$, 从 AS 向量中检索该工序对应的 AGV a. 重新调整 AS 向量中 $O_{i,j}$ 所分配的 AGV, 以生成新的调度方案.

上述算子通过扰动调度方案的不同决策维度形成结构性互补. 协作机制体现在两个层面：其一，在个体层面，算子之间共同调整调度方案的多个向量组成部分，实现对整体排序的多维重构，从而突破单一扰动局部收敛的限制；其二，在策略层面，DQN 基于当前调度状态自适应感知动态环境，学习各算子在不同情形下的贡献效应，从而有针对性地选择当下最优算子以最大化收益. 该协作机制融入 QD 框架中使算法在全局探索与局部开发之间实现平衡.

3.5　时间复杂度分析

所提 DQN-QD 算法的时间复杂度主要由父代选择、交叉与变异、解更新方法、知识协作启发式搜索策略、DQN 策略选择与训练决定. 设每轮执行次数为 $batch$，编码向量总长度为 L，特征空间大小为 $n_{\max} \times 2n_{\max}$，则遍历整个二维网格以选择父代解的时间复杂度为 $\mathcal{O}(batch \times n_{\max} \times 2n_{\max})$. 但由于该二维表格中存在大量空单元格，为节省时间成本，算法采用哈希索引记录非空格子位置，因此时间复杂度降为 $\mathcal{O}(batch)$. 交叉与变异操作中，每轮从特征空间中选择两个父代个体进行操作，其时间复杂度为 $\mathcal{O}(batch \times L)$. 解更新方法用于判断新解是否加入特征空间，包含支配关系判断与解集更新. 一般情况下，新解需与当前单元格中已有解 (最多 δ 个) 进行比较，时间复杂度为 $\mathcal{O}(\delta)$. 由于每次至多更新一个单元格，因此总时间复杂度为 $\mathcal{O}(batch \times \delta)$. 设最长向量长度为 N，执行知识协作启发式搜索策略的时间复杂度为 $\mathcal{O}(batch \times N)$. 设 DQN 输入层维度为 In，第一层输出维度为 $m^{[1]}$，由于除输入层外其余层维度均为固定值，且 In 为最大输入维度，因此策略选择的前向传播复杂度为 $\mathcal{O}(batch \times In \times m^{[1]})$. 设神经网络的总参数量为 Θ，每轮从经验池中采样 $batch$ 个元组进行梯度更新，则一次训练过程的复杂度为 $\mathcal{O}(batch \times \Theta)$. 若每轮训练进行 $Epoch$ 次，则训练复杂度为 $\mathcal{O}(batch \times \Theta \times Epoch)$. 设总迭代轮数为 T，则 DQN-QD 总时间复杂度为：$\mathcal{O}\left(T \times batch \times \left(1 + L + \delta + N + In \times m^{[1]} + \Theta \times Epoch\right)\right)$.

4　仿真实验

4.1　实验设置

与文献 [43] 保持一致, 本文在 15 个不同规模的测试实例上进行实验, 每个实例独立运行 20 次, 以确保实验结果的稳健性. 其中, 测试实例的作业数量设定为 $i \in \{10, 20, 30, 40, 50, 100\}$, 工厂数量设定为 $f \in \{2, 3, 4, 5, 6, 7\}$. 每个工厂均配备 5 台机器 ($l = 5$), 并配备 2 台 AGV($v = 2$). 其中, 每个作业 i 包含 $w_i = 5$ 道工序. 各工序的加工时间 $T_{i,j,f,k}$ 在区间 $[5, 20]$ 内服从均匀随机分布. 关于 AGV 在机器间的运输时间, 采用文献 [47] 提供的基准数据集, 以确保数据的合理性和可比性. 此外, 所有算法的终止条件均设定为总评估次数, 其计算方式为 $Max\_Iter = 50 \cdot \sum_1^n w_i$.

4.2　实验环境与评价指标

实验在 Windows 11 操作系统环境下进行, 计算平台采用 Intel Core i7-13790F @ 2.10 GHz 处理器和 16.0 GB 内存. 算法使用 Python 语言实现. 根据文献 [48], 本文采用三种多目标优化评价指标——世代距离 (Generational Distance, GD)、反世代距离 (Inverted Generational Distance, IGD) 和超体积 (Hypervolume, HV)——对各测试算法的优化性能进行量化评估. 由于完工时间与总能源消耗数值尺度存在较大差异, 需对目标值进行归一化处理, 以确保上述指标的合理性. 此外, 由于研究问题的真实帕累托解集未知, 本文采用所有算法所得的最优解集作为参考帕累托解集, 记作 P^*.

(1) *GD:* 该指标用于评估解集的收敛性, 其定义为算法生成的解集 P_g 中每个解 ζ 到参考解集 P^* 中最近解 ξ 之间的平均最小欧几里得距离. GD 值越小, 说明算法所得到的解集越接近真实帕累托前沿, 即收敛性越优.

$$GD_g\left(P_g, P^*\right)=\frac{\sqrt{\sum_{\zeta \in P_g} \min_{\xi \in P^*} dis\left(\zeta, \xi\right)^2}}{\left|P_g\right|}, \tag{8}$$

其中, $dis\left(\zeta, \xi\right)^2$ 表示解 ζ 到 ξ 的最小欧几里得距离的平方, $\left|P_g\right|$ 表示解集 P_g 中解的数量.

(2) *IGD:* 该指标综合衡量解集 P_g 的收敛性和分布性, 定义为参考解集 P^* 中每个解 ξ 到解集 P_g 中最近解 ζ 的平均欧几里得距离. IGD 值越小, 说明算法的分布性和收敛性越优.

$$IGD_g\left(P_g, P^*\right)=\frac{\sum_{\xi \in P^*} \min_{\zeta \in P_g} dis\left(\zeta, \xi\right)}{\left|P^*\right|}, \tag{9}$$

其中, $dis\left(\zeta, \xi\right)$ 表示解 ζ 到 ξ 的最小欧几里得距离, $\left|P^*\right|$ 表示帕累托解集 P^* 中的解的数量.

(3) *HV:* 该指标衡量解集 P_g 的收敛性和分布性, 其定义为算法 g 生成的非支配解集所覆盖的目标空间体积, 由该解集与参考点 re 共同限定. 本文设定 re 为 (1.1,1.1). HV 值越大, 说明算法 g 生成的解集整体性能越优.

$$HV_g\left(P_g, re\right)=\mathcal{L}\left(\bigcup_{\zeta \in P_g}\left\{\xi \mid \zeta \prec \xi \prec re\right\}\right), \tag{10}$$

其中, $\mathcal{L}(\cdot)$ 表示解集 P_g 的 Lebesgue 测度, ζ 为 P_g 中的解, $\bigcup_{\zeta \in P_g}\left\{\xi \mid \zeta \prec \xi \prec re\right\}$ 表示 ξ 严格大于 ζ, 且严格小于 re 的值.

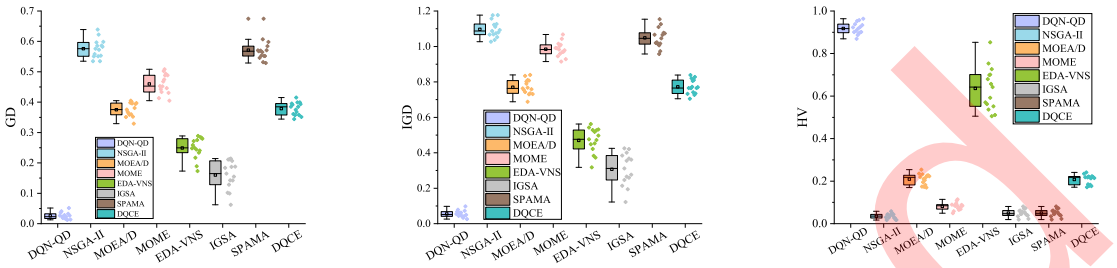
表 3　所有对比算法的 GD 结果. 加粗字体表示最佳值, † 表示该对比算法与 DQN-QD 存在显著性差异

Table 3　GD results of all comparison algorithms. Bold font represents the best value, † indicates that the comparison algorithm is significantly different from DQN-QD

Instance	GD							
J_F	DQN-QD	NSGA-II	MOEA/D	MOME	EDA-VNS	IGSA	SPAMA	DQCE
10_2	**0.017**	0.551†	0.360†	0.474†	0.275†	0.143†	0.532†	0.368†
20_2	**0.028**	0.586†	0.398†	0.468†	0.244†	0.129†	0.564†	0.415†
20_3	**0.023**	0.572†	0.396†	0.488†	0.289†	0.187†	0.552†	0.375†
30_2	**0.023**	0.597†	0.359†	0.508†	0.217†	0.102†	0.553†	0.399†
30_3	**0.014**	0.535†	0.352†	0.452†	0.234†	0.142†	0.552†	0.357†
40_2	**0.019**	0.576†	0.363†	0.493†	0.173†	0.062†	0.569†	0.385†
40_3	**0.012**	0.553†	0.329†	0.433†	0.189†	0.099†	0.529†	0.344†
40_4	**0.023**	0.558†	0.368†	0.446†	0.250†	0.165†	0.546†	0.358†
50_3	**0.033**	0.586†	0.381†	0.502†	0.241†	0.155†	0.571†	0.361†
50_4	**0.041**	0.551†	0.375†	0.443†	0.272†	0.214†	0.568†	0.399†
50_5	**0.052**	0.583†	0.394†	0.488†	0.284†	0.212†	0.584†	0.392†
100_4	**0.037**	0.622†	0.401†	0.453†	0.259†	0.184†	0.675†	0.393†
100_5	**0.026**	0.639†	0.405†	0.430†	0.282†	0.207†	0.607†	0.395†
100_6	**0.026**	0.535†	0.351†	0.413†	0.248†	0.188†	0.570†	0.351†
100_7	**0.014**	0.599†	0.397†	0.405†	0.280†	0.211†	0.598†	0.385†
Mean	**0.026**	0.576	0.375	0.460	0.249	0.160	0.571	0.379

4.3　对比实验与分析

为验证 DQN-QD 求解 DFJSP-AGVs 的优越性, 本文选取多种经典及前沿优化算法进行对比分析. 其中, 选取非支配排序遗传算法 II (NSGA-II) [49] 和基于分解的多目标进化算法 (MOEA/D) [50], 其选择基于二者在多目标优化领域的代表性. 为确保公平比较, NSGA-II 和 MOEA/D 均被调整为使用 DQN-QD 采用的搜索算子. 选取多目标



(a)　(b)　(c)

图 4　所有算法在不同实例上的 GD、IGD 和 HV 值

Figure 4　The GD, IGD, and HV values of all algorithms on different instances

MAP-Elites (MOME) [51] 的主要原因是其特征空间划分方式与 DQN-QD 保持一致. 此外, 本文选取迭代贪婪-模拟退火算法 (IGSA) [31] 和基于分布估计的变邻搜索算法 (EDA-VNS) [34] 作为对比算法, 上述两种算法均用于优化带运输约束的 FJSP 的完工时间和能耗. 其中, IGSA 的工厂分配原则与 DQN-QD 保持一致. 最后, 本文还对比了用于求解 FJSP 的先进机器学习优化算法: 基于 Surprisingly-Popular 机制的自适应忆忆算法 (SPAMA) [52] 和基于 DQN 的协同进化算法 (DQCE) [43]. 所有对比算法的参数配置均按照其原始文献设定, 具体包括: 种群规模 $ps = 100$, 邻域大小 $Ne = 10$, 以及批量大小 $batch = 10$.

表 4　所有对比算法的 IGD 结果. 加粗字体表示最佳值, † 表示该对比算法与 DQN-QD 存在显著性差异

Table 4　IGD results of all comparison algorithms. Bold font represents the best value, † indicates that the comparison algorithm is significantly different from DQN-QD

Instance	IGD							
J_F	DQN-QD	NSGA-II	MOEA/D	MOME	EDA-VNS	IGSA	SPAMA	DQCE
10_2	**0.044**	1.037†	0.767†	0.915†	0.518†	0.266†	0.973†	0.741†
20_2	**0.062**	1.068†	0.810†	0.959†	0.453†	0.246†	1.017†	0.806†
20_3	**0.040**	1.074†	0.790†	0.969†	0.528†	0.355†	1.007†	0.772†
30_2	**0.061**	1.087†	0.730†	0.983†	0.383†	0.194†	1.046†	0.726†
30_3	**0.033**	1.052†	0.743†	0.971†	0.422†	0.259†	1.014†	0.765†
40_2	**0.052**	1.111†	0.736†	1.009†	0.317†	0.122†	1.066†	0.735†
40_3	**0.042**	1.027†	0.688†	0.944†	0.373†	0.214†	0.957†	0.705†
40_4	**0.044**	1.079†	0.754†	1.002†	0.472†	0.312†	1.024†	0.765†
50_3	**0.073**	1.123†	0.759†	1.043†	0.448†	0.276†	1.077†	0.766†
50_4	**0.079**	1.067†	0.764†	1.016†	0.527†	0.401†	1.020†	0.770†
50_5	**0.098**	1.127†	0.802†	1.067†	0.563†	0.425†	1.074†	0.820†
100_4	**0.068**	1.176†	0.807†	0.979†	0.497†	0.367†	1.154†	0.810†
100_5	**0.050**	1.177†	0.834†	0.990†	0.532†	0.384†	1.109†	0.838†
100_6	**0.054**	1.092†	0.732†	0.929†	0.476†	0.361†	1.066†	0.732†
100_7	**0.026**	1.157†	0.839†	0.992†	0.543†	0.421†	1.126†	0.828†
Mean	**0.055**	1.097	0.770	0.985	0.470	0.307	1.049	0.772

表3, 4和5分别展示了所有对比算法在 GD、IGD 和 HV 指标上的实验结果. 与消融实验类似, 其中最优值以加粗字体表示, 而符号 † 用于标记其他算法与 DQN-QD 之间存在的显著性差异. 由实验结果可知, DQN-QD 在所有测试实例中均表现出最好的优化能力. 为进一步分析算法间的性能差异, 本文采用 Wilcoxon 秩和检验, 其显著性水平设定为 $\alpha = 0.05$. 由表 6 可知, DQN-QD 在整体性能排名中位列首位, 且 p 值远低于 0.05, 该结果从统计学角度验证了 DQN-QD 的优越性.

此外, 为深入探究不同算法在解集分布及其稳定性方面的表现, 本文绘制了 GD、IGD 和 HV 指标的箱线图, 以直观展现数据的分布特征. 如图4所示, DQN-QD 得到的解集表现出较强的稳定性和密集度, 且无明显离群值. 同时, 本文在图5中绘制了各算法所得的帕累托解集, 以进一步可视化其效果. 为更直观地分析不同算法的帕累托前沿分布, 本文对两个优化目标进行归一化处理. 结果表明, DQN-QD 在帕累托解集质量及整体收敛性方面均优于其他算法, 且更接近真实帕累托前沿, 体现出更强的搜索能力与优化性能.

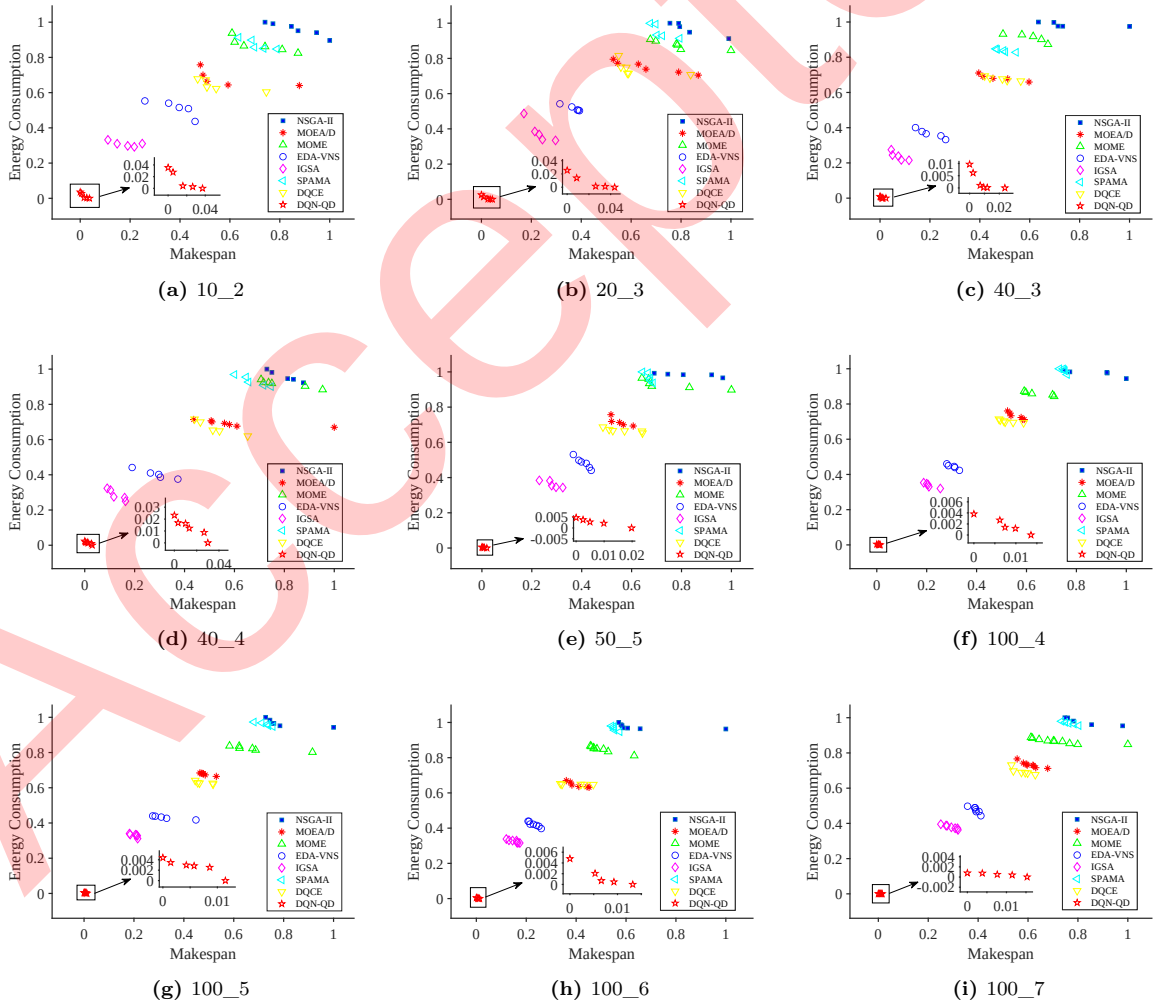
实验结果表明, DQN-QD 算法在求解 DFJSP-AGVs 时表现出显著优势, 其主要归因于:

在 DFJSP-AGVs 中，调度不仅取决于传统的目标值优化，更受到 AGV 运输与机器空闲状态交互影响的显著制约.DQN-QD 通过引入 AGV 运输次数与机器空闲次数作为行为特征，构建了结构清晰的二维特征空间，从而为解集提供了判别性强的分布依据. 此特征空间反映了解在“资源利用效率”维度上的差异，使得搜索过程在保证目标优化的同时提升了解的多样性，有效避免了传统算法中因目标导向易陷入局部最优的问题，并为多目标帕累托边界

表 5　所有对比算法的 **HV** 结果. 加粗字体表示最佳值, † 表示该对比算法与 **DQN-QD** 存在显著性差异

Table 5　HV results of all comparison algorithms. Bold font represents the best value, † indicates that the comparison algorithm is significantly different from DQN-QD

Instance	HV							
J_F	DQN-QD	NSGA-II	MOEA/D	MOME	EDA-VNS	IGSA	SPAMA	DQCE
10_2	**0.918**	0.058†	0.204†	0.114†	0.398†	0.657†	0.081†	0.218†
20_2	**0.895**	0.040†	0.173†	0.085†	0.462†	0.701†	0.055†	0.178†
20_3	**0.953**	0.050†	0.210†	0.096†	0.414†	0.592†	0.072†	0.222†
30_2	**0.887**	0.035†	0.218†	0.073†	0.521†	0.753†	0.047†	0.213†
30_3	**0.957**	0.042†	0.232†	0.083†	0.518†	0.699†	0.054†	0.218†
40_2	**0.907**	0.028†	0.226†	0.065†	0.612†	0.853†	0.045†	0.223†
40_3	**0.926**	0.044†	0.254†	0.086†	0.544†	0.726†	0.072†	0.241†
40_4	**0.939**	0.043†	0.228†	0.075†	0.473†	0.642†	0.061†	0.220†
50_3	**0.901**	0.027†	0.219†	0.058†	0.497†	0.680†	0.044†	0.218†
50_4	**0.898**	0.044†	0.223†	0.069†	0.417†	0.536†	0.060†	0.215†
50_5	**0.869**	0.027†	0.195†	0.049†	0.380†	0.505†	0.041†	0.185†
100_4	**0.901**	0.017†	0.182†	0.088†	0.431†	0.563†	0.019†	0.183†
100_5	**0.934**	0.019†	0.174†	0.089†	0.403†	0.552†	0.039†	0.172†
100_6	**0.923**	0.030†	0.235†	0.109†	0.456†	0.570†	0.035†	0.235†
100_7	**0.964**	0.018†	0.170†	0.084†	0.396†	0.511†	0.022†	0.177†
Mean	**0.918**	0.035	0.210	0.082	0.462	0.636	0.050	0.208



(a) 10_2　**(b)** 20_3　**(c)** 40_3

(d) 40_4　**(e)** 50_5　**(f)** 100_4

(g) 100_5　**(h)** 100_6　**(i)** 100_7

图 5　所有算法在不同实例上的帕累托前沿

Figure 5　Pareto fronts achieved by all algorithms on different instances

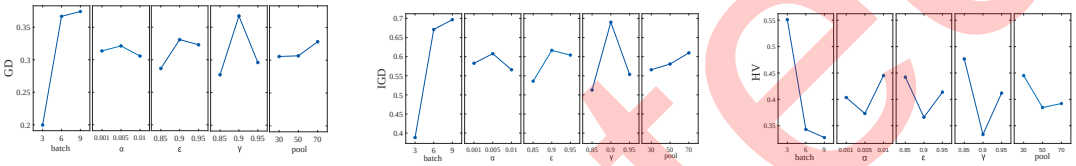
的探索提供了稳定的结构支持.

DQN 状态输入向量涵盖调度中关键的决策信息，包括工序、机器、工厂及 AGV 分配策略，同时集成了目标值与帕累托局部解集包含的独特行为特征 (NT, NI). 该输入构成使 DQN 能全面捕捉调度状态的复杂性，并使 Q 值函

表 6　弗里德曼检验结果，用于分析算法之间的显著性差异 (置信水平 $\alpha = 0.05$)

Table 6　Results of Friedman's test for significant differences among algorithms (confidence level $\alpha = 0.05$)

	GD		IGD		HV	
Algorithm	rank	p-value	rank	p-value	rank	p-value
DQN-QD	**1.00**		**1.00**		**1.00**	
NSGA-II	7.67		8.00		8.00	
MOEA/D	4.50		4.43		4.37	
MOME	6.00		6.00		6.00	
EDA-VNS	3.00	2.96E-19	3.00	1.72E-19	3.00	1.67E-19
IGSA	2.00		2.00		2.00	
SPAMA	7.33		7.00		7.00	
DQCE	4.50		4.57		4.63	



(a)　**(b)**　**(c)**

图 6　不同参数值在 GD、IGD 和 HV 上的变化

Figure 6　Trends of GD, IGD, and HV under different parameter settings

数能更准确地判断当前解在不同特征区域的优化潜力，在强化学习框架中更好地估计不同策略的价值，提供高效反馈，从而强化搜索方向的学习与泛化能力.

此外，DQN-QD 进一步通过 DQN 网络输出动态决策搜索算子的使用. 在迭代过程中，算法基于当前状态从经验池中学习策略，通过强化学习机制选择合适的搜索算子，从而避免了传统方法中因算子使用固定或随机导致的搜索效率低下问题. 同时，领域知识辅助设计的协作启发式搜索策略可与 DQN 选择机制协同运行，实现对工序顺序、机器选择与工厂分配等多资源维度的有效联动.

4.4　参数调优

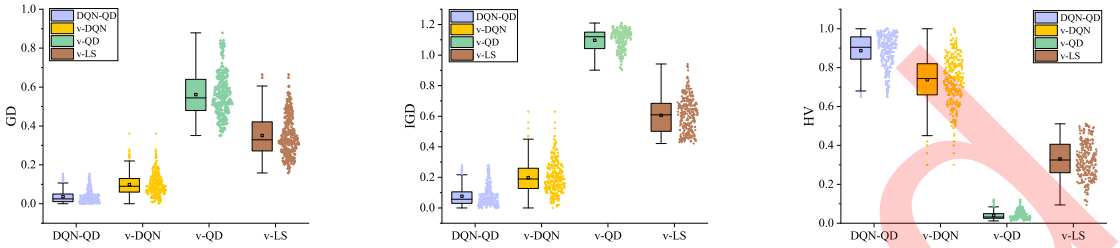
本研究采用 Taguchi 方法[43] 对以下五个参数进行调优: 批量大小 $batch$, 学习率 α, 折扣因子 γ, 贪心因子 ϵ, 以及经验池大小 $pool$. 每个参数均设定三个水平, 具体取值为: $batch \in \{3, 6, 9\}$, $\alpha \in \{0.001, 0.005, 0.01\}$, $\gamma \in \{0.85, 0.9, 0.95\}$, $\epsilon \in \{0.85, 0.9, 0.95\}$, 和 $pool \in \{30, 50, 70\}$. 采用正交试验 $L_{12}(3^5)$ 设计实验, 形成 12 组不同的参数组合 $(batch, \alpha, \gamma, \epsilon, pool)$. 此外, 所有算法的人群规模设定为 $ps = 100$. 在所有规模的实例上进行测试, 以评估不同参数组合对算法性能的影响. 图 6 展示了五个参数的因子水平趋势图, 直观地反映了不同取值对算法性能的影响. 其中, 批量大小 $batch$ 对算法性能的影响最为显著, 表明在 $batch$ 轮迭代中选择多个解可能会减缓整体进化过程, 从而降低解的探索效率. 较小的 $batch$ 值减少了评估次数, 使算法能够在更广泛的解空间中探索. 综合各评估指标的分析, 确定 DQN-QD 算法的最优参数组合为 $batch = 3, \alpha = 0.01, \gamma = 0.85, \epsilon = 0.85$, 和 $pool = 30$.

表 7　各变体在 GD、IGD 和 HV 指标上的结果

Table 7　Results of all variants on GD, IGD, and HV metrics

Instance	GD				IGD				HV			
J_F	DQN-QD	v-DQN	v-QD	v-LS	DQN-QD	v-DQN	v-QD	v-LS	DQN-QD	v-DQN	v-QD	v-LS
10_2	**0.0182**	0.0921†	0.4819†	0.2402†	**0.0465**	0.1950†	0.9526†	0.5082†	**0.9122**	0.7305†	0.0940†	0.3952†
20_2	**0.0347**	0.0853†	0.5879†	0.2982†	**0.0730**	0.1834†	1.0826†	0.5249†	**0.8742**	0.7320†	0.0420†	0.3787†
20_3	**0.0242**	0.0926†	0.4920†	0.2999†	**0.0432**	0.1766†	1.0206†	0.4856†	**0.9496**	0.7829†	0.0669†	0.4573†
30_2	**0.0255**	0.0853†	0.5617†	0.2546†	**0.0694**	0.1833†	1.0818†	0.5484†	**0.8727**	0.7325†	0.0331†	0.3552†
30_3	**0.0190**	0.0788†	0.5408†	0.3413†	**0.0466**	0.1630†	1.0253†	0.6729†	**0.9391**	0.7816†	0.0600†	0.2732†
40_2	**0.0209**	0.0904†	0.6082†	0.3110†	**0.0655**	0.1988†	1.1491†	0.5260†	**0.8810**	0.7146†	0.0243†	0.3879†
40_3	**0.0190**	0.0693†	0.5299†	0.2708†	**0.0571**	0.1580†	1.0093†	0.5180†	**0.8960**	0.7601†	0.0542†	0.3878†
40_4	**0.0973**	0.1268	0.5804†	0.4063†	**0.1909**	0.2495	1.1213†	0.7029†	**0.7463**	0.7026	0.0340†	0.2602†
50_3	**0.0434**	0.1270†	0.5832†	0.3779†	**0.0991**	0.2514†	1.1242†	0.6341†	**0.8642**	0.6797†	0.0267†	0.3107†
50_4	**0.0575**	0.1272†	0.6218†	0.4691†	**0.1106**	0.2385†	1.1690†	0.7728†	**0.8569**	0.6969†	0.0246†	0.2069†
50_5	**0.0622**	0.0786	0.5986†	0.3941†	**0.1182**	0.1434	1.1427†	0.6404†	**0.8429**	0.8183	0.0275†	0.3054†
100_4	**0.0417**	0.1162†	0.5477†	0.4277†	**0.0775**	0.2240†	1.1216†	0.7179†	**0.8882**	0.7117†	0.0259†	0.2386†
100_5	**0.0306**	0.1221†	0.5738†	0.4612†	**0.0577**	0.2282†	1.1535†	0.6976†	**0.9233**	0.7171†	0.0296†	0.2606†
100_6	**0.0330**	0.0966†	0.5997†	0.3874†	**0.0682**	0.1972†	1.1751†	0.6552†	**0.9000**	0.7355†	0.0190†	0.2888†
100_7	**0.0155**	0.0899†	0.5081†	0.3213†	**0.0291**	0.1847†	1.1276†	0.4884†	**0.9589**	0.7630†	0.0308†	0.4535†
Mean	**0.0362**	0.0985	0.5610	0.3507	**0.0768**	0.1983	1.0971	0.6062	**0.8870**	0.7373	0.0395	0.3307

https://www.sciengine.com/doi/10.1360/SSI-2025-0113



(a)　(b)　(c)

图 7　所有变体算法在不同实例上的 GD、IGD 和 HV 值

Figure 7　The GD, IGD, and HV values of all variants on different instances

4.5　消融实验

为验证所提策略的有效性, 本文对 DQN-QD 算法及其三个变体进行对比实验: 1) v-DQN, 移除 DQN 选择机制, 采用随机选择策略; 2) v-QD, 移除 QD 框架, 直接在种群中存储解; 3) v-LS, 移除启发式局部搜索算子, 仅采用随机修改工序顺序的搜索策略. 表 7 给出了 DQN-QD 算法及其变体在各个实例上的平均 GD, IGD, HV 结果. 其中, 列 'J_F' 代表作业数与工厂数之和, 末行 'Mean' 表示算法在所有实例上的整体平均性能. 每个实例的最优值以加粗字体突出标示. 此外, 本文进一步采用显著性水平为 0.05 的 Wilcoxon 检验分析 DQN-QD 与变体间的差异, 其中符号 † 表示存在显著性差异.

表 8　不同变体间显著性差异的 Friedman 检验结果 (置信水平 $\alpha = 0.05$)

Table 8　Results of Friedman's test for significant differences among variants (confidence level $\alpha = 0.05$)

	GD		IGD		HV	
Algorithm	rank	p-value	rank	p-value	rank	p-value
DQN-QD	**1.00**		**1.00**		**1.00**	
v-DQN	2.00		2.00		2.00	
v-QD	4.00	9.25E-10	4.00	9.25E-10	4.00	9.25E-10
v-LS	3.00		3.00		3.00	

由表 7可知, 在 GD, IGD 和 HV 三个指标上, DQN-QD 始终优于变体 v-DQN, v-QD 和 v-LS. 这一优势在均值结果上进一步得到验证: DQN-QD 在所有变体中表现最佳, 其中, GD 为 0.0362, IGD 为 0.0768, HV 为 0.887. 除 40_4 和 50_5 实例外, DQN-QD 在所有其余实例上均显著优于其他变体, 相较 v-DQN 亦表现出明显的优势. 此外, 表 8 给出了 Friedman 检验结果, 该检验在显著性水平 0.05 下分析 GD, IGD 和 HV 指标的显著性差异, 并报告 $rank$ 和 p-value. 其中, $rank$ 值越低, 说明算法整体性能越优; 若 p-value 小于 0.05, 则表明 DQN-QD 相较于变体存在显著性差异. 结果表明, DQN-QD 在各项指标上均优于对比算法, 具有显著的性能提升.

为进一步分析不同变体在 GD, IGD 和 HV 指标上的解分布特性及变化趋势, 本文绘制了箱型图 (见图7), 直观展示各算法在 300 次独立运行中的结果分布. 结果表明, DQN-QD 在所有评估指标上均显著优于其他变体, 验证了 DQN, QD 和 LS 三种策略对算法性能的提升作用, 其中 QD 框架贡献最为显著.

DQN-QD 在不同变体算法中的优越性可以归因于多个关键因素, 其主要包括:

- QD 框架的有效性: DQN-QD 与 v-QD 对比表明, 在 DFJSP-AGVs 中，工序调度与 AGV 运输存在高度耦合，直接影响了整体性能. 所设计的 QD 框架不仅能在结构化特征空间中存储帕累托解集，还有效利用了机器空闲和 AGV 运输的多样化行为特征来引导搜索，增强了对复杂耦合调度问题的适应性和解析能力.

- 局部搜索能力的提升: DQN-QD 与 v-LS 对比表明, 基于作业、机器、工厂和 AGV 的协作启发式搜索算子显著增强了局部搜索能力. 在传统 QD 框架缺乏细粒度局部开发能力的情形下，该策略显著强化了解集的收敛性，特别是在高维搜索空间中可有效挖掘潜在解的邻域结构. 这种具备先验知识的搜索机制对于协调多资源间调度约束具有显著优势.

- DQN 训练策略的优势: DQN-QD 与 v-DQN 对比表明, 针对调度状态的动态变化与高维组合特性, DQN 提供了策略学习上的补充. 本文通过深度神经网络对复杂调度状态进行建模，使决策不再依赖手工规则或随机机制，从而

https://www.sciengine.com/doi/10.1360/SSI-2025-0113

规避了短视性决策和行为不稳定问题. DQN 引入的策略优化机制具有动态自适应性，能够根据调度过程中不断变化的反馈信息实时调整搜索方向，从而提升解的质量与稳定性. 此外，该方法无需大量人工经验输入，在多目标、高复杂性环境中展现出更强的泛化能力和调度效率.

4.6　结论和未来工作

针对 DJFPS-AGVs, 本文提出了一种基于深度 Q 网络增强的质量-多样性算法以优化最大完工时间和能耗. 通过集成基于知识辅助的协作启发式搜索策略和 DQN 选择机制, QD 算法能够自适应选择局部搜索算子, 进而优化解的质量. 在测试实例上的实验表明, DQN-QD 优于现有最先进方法, 取得了 4% 至 15% 的性能提升. 尽管算法展现出良好性能，但仍存在一些值得研究的地方，如本文未研究机器设定、作业到达、机器故障等条件下 DQN-QD 的适用性. 这在未来研究中可以进一步探讨, 并设计具有针对性的搜索策略.

补充材料　补充材料. 本文的补充材料见网络版 infocn.scichina.com. 补充材料为作者提供的原始数据, 作者对其学术质量和内容负责.

4.7　问题描述

基于文献 [53]，为便于描述算法，本节列出了相关符号和约束条件.

表 9　符号定义
Table 9　Notations

符号	定义	符号	定义
f	工厂索引	i, i'	作业索引
j, j'	工序索引	k, k'	机器索引
a	AGV 索引	l	工厂总数
n	作业总数	w_i	作业 i 的工序数量
n_{max}	所有作业的总工序数	m	每个工厂的机器总数
v	每个工厂的 AGV 总数	F	工厂集合, $F = \{1, ..., f, ..., l\}$
I	作业集合, $I = \{1, ..., i, ..., n\}$	A	AGV 集合, $A = \{1, ..., a, ..., v\}$
J_i	作业 i 的工序集合, $J_i = \{1, ..., j, ..., w_i\}$	M	机器集合, $M = \{1, ..., k, ..., m\}$
$O_{i,j}$	作业 i 的第 j 道工序	$M_{i,j}$	工序 $O_{i,j}$ 可选的机器集合
$T_{i,j,f,k}$	工序 $O_{i,j}$ 在工厂 f 的机器 k 上的加工时间	$TR_{k,k'}$	从机器 k 运输到机器 k' 的运输时间
$TR_{0,k'}$	从起始点到机器 k' 的运输时间	PP_k	机器 k 在加工状态下单位时间的能耗
PI_k	机器 k 在空闲状态下单位时间的能耗	PT_a	AGV a 在运输状态下单位时间的能耗
H	一个足够大的整数		

决策变量:

$z_{i,f}$: 二进制变量, 若工件 i 被分配到工厂 f 加工，则取值为 1，否则为 0.

$x_{i,j,f,k}$: 二进制变量, 若工序 $O_{i,j}$ 被分配到工厂 f 的机器 k 加工，则取值为 1，否则为 0.

$y_{i,j,i',j',f,k}$: 二进制变量, 若工序 $O_{i,j}$ 和 $O_{i',j'}$ 在工厂 f 的机器 k 上加工，且 $O_{i,j}$ 先于 $O_{i',j'}$ 加工，则取值为 1，否则为 0.

$u_{i,j,f,a}$: 二进制变量, 若 AGV a 运输作业 i 到工厂 f 的机器进行工序 $O_{i,j}$ 的加工，则取值为 1，否则为 0. 若工序 $O_{i,j}$ 与前一道工序 $O_{i,j-1}$ 在同一机器上加工，无需运输，则 $u_{i,j,f,0} = 1$.

$w_{i,j,i',j',f,a}$: 二进制变量, 若 AGV a 运输作业 i 到工厂 f 进行工序 $O_{i,j}$ 加工后, 立即运输作业 i' 进行工序 $O_{i',j'}$ 加工，则取值为 1，否则为 0. 引入虚拟作业 0, 若 $w_{0,1,i',j',f,a} = 1$，则表示 AGV a 的首个运输任务是运输作业 i' 进行工序 $O_{i',j'}$ 加工. 若 $w_{i,j,0,1,f,a} = 1$，则表示 AGV a 的最后一个运输任务是运输作业 i 进行工序 $O_{i,j}$ 加工.

$C_{i,j}$: 工序 $O_{i,j}$ 的完成时间.

$S_{i,j}$: 作业 i 到达加工工序 $O_{i,j}$ 机器的时间.

$TO_{i,j,a}$: AGV a 运输作业 i 到工厂 f 加工工序 $O_{i,j}$ 所需的工作时间.

$TP_{f,k}$: 工厂 f 的机器 k 的总加工时间.

$TI_{f,k}$: 工厂 f 的机器 k 的总空闲时间.

C_{max}: 完工时间 (Makespan).

TPE: 总加工能耗.

TIE: 机器空闲状态的总能耗.

TTE: AGV 运输的总能耗.

约束条件:

$$\sum_{f \in F} z_{i,f} = 1, \forall i \in I, \tag{11}$$

$$\sum_{k\in M_{i,j}} x_{i,j,f,k} = z_{i,f}, \forall i \in I, \forall j \in J_i, \forall f \in F, \tag{12}$$

$$y_{i,j,i',j',f,k} + y_{i',j',i,j,f,k} \leqslant x_{i,j,f,k}, \quad \forall i,i' \in I, \forall j \in J_i, \forall j' \in J_{i'}, i < i' \vee (i = i' \wedge j < j'), \forall f \in F, \forall k \in M_{i,j} \cap M_{i',j'}, \tag{13}$$

$$y_{i,j,i',j',f,k} + y_{i',j',i,j,f,k} \leqslant x_{i',j',f,k}, \quad \forall i,i' \in I, \forall j \in J_i, \forall j' \in J_{i'}, i < i' \vee (i = i' \wedge j < j'), \forall f \in F, \forall k \in M_{i,j} \cap M_{i',j'}, \tag{14}$$

$$\begin{aligned} &y_{i,j,i',j',f,k} + y_{i',j',i,j,f,k} \geqslant x_{i,j,f,k} + x_{i',j',f,k} - 1, \\ &\forall i,i' \in I, \forall j \in J_i, \forall j' \in J_{i'}, i < i' \vee (i = i' \wedge j < j'), \forall f \in F, \forall k \in M_{i,j} \cap M_{i',j'}, \end{aligned} \tag{15}$$

$$C_{i,j} \geqslant C_{i,j-1} + \sum_{f\in F} \sum_{k\in M_{i,j}} \left(x_{i,j,f,k} \cdot T_{i,j,f,k}\right), \quad \forall i \in I, \forall j \in J_i \setminus \{1\}, \tag{16}$$

$$\begin{aligned} &C_{i',j'} \geqslant C_{i,j} + T_{i',j',f,k} + (y_{i,j,i',j',f,k} - 1) \cdot H, \\ &\forall i,i' \in I, \forall j \in J_i, \forall j' \in J_{i'}, i \neq i' \vee (i = i' \wedge j \neq j'), \forall f \in F, \forall k \in M_{i,j} \cap M_{i',j'}, \end{aligned} \tag{17}$$

$$\sum_{\forall a \in A \cup \{0\}} u_{i,j,f,a} = \sum_{k\in M_{i,j}} x_{i,j,f,k}, \quad \forall i \in I, \forall j \in J_i, \forall f \in F, \tag{18}$$

$$u_{i,j,f,0} = \sum_{k\in M_{i,j-1}\cap M_{i,j}} y_{i,j-1,i,j,f,k}, \quad \forall i \in I, \forall j \in J_i \setminus \{1\}, \forall f \in F, \tag{19}$$

$$u_{i,1,f,0} = 0, \forall i \in I, \forall f \in F, \tag{20}$$

$$\sum_{i\in I\cup\{0\}, i\neq i'} \sum_{j\in J_i} w_{i,j,i',j',f,a} + \sum_{j\in J_{i'}, j<j'} w_{i',j,i',j',f,a} = u_{i',j',f,a}, \quad \forall i' \in I, \forall j' \in J_{i'}, \forall f \in F, \forall a \in A, \tag{21}$$

$$\sum_{i'\in I\cup\{0\}, i'\neq i} \sum_{j'\in J_{i'}} w_{i,j,i',j',f,a} + \sum_{j'\in J_i, j'>j} w_{i,j,i,j',f,a} = u_{i,j,f,a}, \quad \forall i \in I, \forall j \in J_i, \forall f \in F, \forall a \in A, \tag{22}$$

$$\sum_{i'\in I\cup\{0\}} \sum_{j'\in J_{i'}} w_{0,1,i',j',f,a} = 1, \tag{23}$$

$$\sum_{i\in I} \sum_{j\in J_i} w_{i,j,0,1,f,a} = 1, \quad \forall f \in F, \forall a \in A, \tag{24}$$

$$C_{i,j} \geqslant S_{i,j} + \sum_{f\in F} \sum_{k\in M_{i,j}} \left(x_{i,j,f,k} \cdot T_{i,j,f,k}\right), \quad \forall i \in I, \forall j \in J_i, \forall f \in F, \forall k \in M_{i,j}, \tag{25}$$

$$\begin{aligned} &S_{i,j} \geqslant C_{i,j-1} + TR_{k',k} + (x_{i,j,f,k} + x_{i,j-1,f,k'} - u_{i,j,f,0} - 2) \cdot H, \\ &\forall i \in I, \forall j \in J_i \setminus \{1\}, \forall f \in F, \forall k \in M_{i,j}, \forall k' \in M_{i,j-1}, k \neq k', \end{aligned} \tag{26}$$

$$\begin{aligned} &TO_{i',j',a} \geqslant TR_{k,k''} + TR_{k'',k'} + (x_{i,j,f,k} + x_{i',j',f,k'} + x_{i',j'-1,f,k''} + w_{i,j,i',j',f,a} - 4) \cdot H, \\ &\forall i \in I, \forall i' \in I, \forall j \in J_i, \forall j' \in J_{i'} \setminus \{1\}, i \neq i' \vee (i = i' \wedge j < j'), \\ &\forall f \in F, \forall k \in M_{i,j}, \forall k' \in M_{i',j'}, \forall k'' \in K_{i',j'-1}, k' \neq k'', \forall a \in A, \end{aligned} \tag{27}$$

$$\begin{aligned} &TO_{i',1,a} \geqslant TR_{k,0} + TR_{0,k'} + (x_{i,j,f,k} + x_{i',1,f,k'} + w_{i,j,i',1,f,a} - 3) \cdot H, \\ &\forall i \in I, \forall i' \in I, \forall j \in J_i, i \neq i', \forall f \in F, \forall k \in M_{i,j}, \forall k' \in M_{i',1}, \forall a \in A, \end{aligned} \tag{28}$$

$$TO_{i',j',a} \geqslant TR_{0,k''} + TR_{k'',k'} + (x_{i',j',f,k'} + x_{i',j'-1,f,k''} + w_{0,1,i',j',f,a} - 3) \cdot H, \\ \forall i' \in I, \forall j' \in J_{i'} \setminus \{1\}, \forall f \in F, \forall k' \in M_{i',j'}, \forall k'' \in M_{i',j'-1}, k' \neq k'', \forall a \in A, \tag{29}$$

$$TO_{i',1,a} \geqslant TR_{0,k'} + (x_{i',j,f,k'} + w_{0,1,i',1,f,a} - 2) \cdot H, \quad \forall i' \in I, \forall f \in F, \forall k' \in M_{i',1}, \forall a \in A, \tag{30}$$

$$S_{i',j'} \geqslant S_{i,j} + TO_{i',j',a} + (w_{i,j,i',j',f,a} - 1) \cdot H, \quad \forall i, i' \in I, \forall j \in J_i, \forall j' \in J_{i'}, i \neq i' \vee (i = i' \wedge j < j'), \forall f \in F, \forall a \in A, \tag{31}$$

$$S_{i',j'} \geqslant TO_{i',j',a} + (w_{0,1,i',j',f,a} - 1) \cdot H, \quad \forall i' \in I, \forall j' \in J_{i'}, \forall f \in F, \forall a \in A, \tag{32}$$

$$C_{\max} \geqslant C_{i,w_i}, \quad \forall i \in I, \tag{33}$$

$$TP_{f,k} = \sum_{i \in I} \sum_{j \in J_i} (x_{i,j,f,k} \cdot T_{i,j,f,k}), \quad \forall f \in F, \forall k \in M, \tag{34}$$

$$TI_{f,k} \geqslant C_{i,j} - \sum_{i \in I} \sum_{j \in J_i} (x_{i,j,f,k} \cdot T_{i,j,f,k}) + (x_{i,j,f,k} - 1) \cdot H, \quad \forall i, i' \in I, \forall j \in J_i, \forall f \in F, \forall k \in M, \tag{35}$$

约束 (11): 保证每个工件仅能被分配给一个工厂.

约束 (12): 保证每道工序仅能分配给一个工厂的某台机器加工.

约束 (13)-(15): 确保工序 $O_{i,j}$ 和 $O_{i',j'}$ 被分配到同一工厂 f 的机器，其必须有确定的加工顺序.

约束 (16): 工序 $O_{i,j}$ 的完成时间必须大于等于其前一道工序的完成时间加上加工时间.

约束 (17): 保证连续工序之间的完成时间和加工时间的关系.

约束 (18): 规定作业 i 在工厂 f 进行工序 $O_{i,j}$ 加工时，只能由一台 AGV 运输.

约束 (19): 确保若工序 $O_{i,j}$ 和 $O_{i,j-1}$ 在同一机器上加工，则无需 AGV 运输.

约束 (20): 确保每个工件的第一个工序必须由 AGV 运输.

约束 (21)-(22): 确保工厂 f 的 AGV a 在每次运输任务之间有先后关系.

约束 (23)-(24): 确保每台 AGV 有首个运输任务和最后的运输任务.

约束 (25)-(26): 规定作业 i 的完成时间需大于等于其到达机器的时间加上加工时间，并考虑 AGV 运输的时间.

约束 (27)-(30): 描述了 AGV a 在不同情况下完成运输任务所需要的时间.

约束 (31)-(32): 确保 AGV a 在不同任务序列中的运输时间约束.

约束 (33): 表示完工时间 C_{max} 的计算方式.

约束 (34): 计算 $TP_{f,k}$，即每台机器的总加工时间.

约束 (35): 表示 $TI_{f,k}$ 与作业完成时间和加工时间之间的关系.

为验证 DFJSP-AGVs 集成调度模型的有效性, 本文将其在 Pycharm 中编程, 并使用 Gurobi 12.0.2 求解, 并将数值结果与所提 DQN-QD 进行对比. 模型运行的时间上限设置为 3600 秒, DQN-QD 的运行次数与终止条件与第 4.1节设置保持一致. J、F 和 M 分别表示工件、工厂和机器的数量. Gap(%) 表示相对于理想目标值的偏离程度. '\' 表示数学模型在 3600 秒内模型未发现可行解.

由表 10可知，DFJSP-AGVs 数学模型在处理小规模问题（如 5_2_4、5_2_6）时表现优越, GD 和 IGD 值为 0, HV 值较高, 说明解质量理想. 但求解时间均达到 3600 秒上限, 效率相对 DQN-QD 较低. 随着规模扩大 (如 8_2_8、10_3_6), 模型性能明显下降，部分指标劣化严重，甚至在更大规模问题 (如 12_3_8 及以上) 中无法在限定时间内获得有效解. DQN-QD 在所有实例中均能成功返回

表 10　DQN-QD 与 DFJSP-AGVs 模型的性能对比

Table 10　Performance comparison between DQN-QD and the DFJSP-AGVs Model

	Model					DQN-QD			
J_F_M	GD	IGD	HV	Gap(%)	Time(s)	GD	IGD	HV	Time(s)
5_2_4	**0.00**	**0.00**	**0.99**	14.32%	3600	0.46	0.89	0.09	2.72
5_2_6	**0.00**	**0.26**	**0.53**	22.73%	3600	0.18	0.43	0.23	2.32
5_2_8	**0.00**	**0.17**	**0.87**	30.19%	3600	0.08	0.30	0.66	2.17
8_2_4	**0.00**	**0.00**	**0.99**	29.72%	3600	0.46	0.93	0.11	3.38
8_2_6	**0.00**	**0.22**	**0.77**	45.03%	3600	0.16	0.47	0.31	3.39
8_2_8	0.48	0.81	0.10	49.71%	3600	**0.14**	**0.27**	**0.69**	3.59
10_3_6	0.25	0.74	0.07	65.57%	3600	**0.14**	**0.36**	**0.35**	4.59
10_3_8	\	\	\	\	3600	**0.25**	**0.45**	**0.49**	4.37
10_3_10	\	\	\	\	3600	**0.14**	**0.32**	**0.59**	4.36
12_3_6	**0.00**	**0.33**	**0.35**	58.97%	3600	0.15	0.48	0.34	7.70
12_3_8	\	\	\	\	3600	**0.18**	**0.34**	**0.64**	7.50
12_3_10	\	\	\	\	3600	**0.21**	**0.41**	**0.49**	8.80
15_3_6	\	\	\	\	3600	**0.23**	**0.44**	**0.52**	8.13
15_3_8	\	\	\	\	3600	**0.26**	**0.45**	**0.56**	8.34
15_3_10	\	\	\	\	3600	**0.08**	**0.31**	**0.42**	7.90
20_4_6	\	\	\	\	3600	**0.13**	**0.41**	**0.39**	12.07
20_4_8	\	\	\	\	3600	**0.11**	**0.22**	**0.78**	12.95
20_4_10	\	\	\	\	3600	**0.21**	**0.53**	**0.33**	15.83

表 11　所有对比算法的运行时间

Table 11　The running time of all comparative algorithms

Instance	Running Time (s)							
J_F	NSGA-II	MOEA/D	MOME	EDA-VNS	IGSA	SPAMA	DQCE	DQN-QD
10_2	2.35	2.47	1.16	4.01	2.39	5.21	10.56	9.25
20_2	6.62	6.58	4.26	16.82	8.89	20.98	35.24	46.30
20_3	6.78	6.59	4.27	31.41	9.05	26.94	51.13	35.98
30_2	14.28	13.11	9.50	20.58	21.11	62.82	148.56	119.18
30_3	13.18	13.19	11.96	32.26	26.72	64.64	91.66	89.34
40_2	34.40	24.54	17.44	22.83	42.92	149.06	213.29	207.23
40_3	21.94	22.23	29.61	23.21	55.44	101.74	255.43	206.36
40_4	29.15	28.20	29.39	51.08	56.17	91.39	254.01	225.05
50_3	39.15	55.27	47.33	42.11	91.23	165.31	592.27	348.00
50_4	53.57	43.16	47.13	39.31	90.11	153.04	575.17	333.82
50_5	55.74	33.79	47.24	92.62	63.77	178.81	616.34	365.47
100_4	164.75	146.46	225.11	102.42	370.99	984.73	2476.47	1744.93
100_5	146.16	202.94	225.81	154.23	425.07	946.90	2420.98	1728.38
100_6	146.44	245.44	229.04	208.54	463.04	636.77	2040.47	1557.28
100_7	146.83	242.04	191.88	233.96	469.03	616.34	1925.33	1503.37
Mean	58.76	72.40	74.74	71.69	146.39	280.31	780.46	568.00

解, 且在 8_2_8 规模以上 (除 12_3_6 规模外), GD、IGD 和 HV 指标整体优于数学模型, 同时求解时间显著减少, 其解的质量与多样性均保持稳定，充分验证了所提 DQN-QD 在求解 DFJSP-AGVs 时的高效性与扩展能力.

由表 11 可知, DQCE 运行时间最长, 其次是 DQN-QD, 接着是 SPAMA. 其余不涉及机器学习的元启发式算法运行时间相对较短. 以往研究 [54,55] 已证实, 在线深度强化学习在优化问题中展现出卓越的求解能力, 尤其是在结合神经网络训练时, 能够显著提升解的质量. 然而, 这一改进通常伴随着更高的计算资源消耗和更长的运行时间. 本文实验结果亦验证了这一现象, 其中可能的原因如下:

首先, 本文采用在线 DQN 作为动态优化策略, 其需要在运行过程中进行在线学习和决策, 该操作额外增加了计算资源需求和时间开销. 其次, 在线 DQN 通过探索与利用的交互方式, 不断从环境中学习最优策略, 进一步增加了运行时间.

相比 DQCE, DQN-QD 运行时间较短, 其归因于优化后的存档结构, 该结构用于存储和检索解, 从而降低了每次迭代的计算量. 由于 QD 生成的解被存储于特定的单元格中, 每次更新时仅需将新解与同一单元格内的解进行比较, 而无需对整个种群进行评估. 这一优化大幅减少了运行时间, 使得在评估帕累托解集的支配关系以及更新特征空间时, 相较于遍历整个种群, 能够节省大量计算成本.

4.8　调度延迟及其能耗的构成与理论分析

本节探讨与分析机器空闲和运输在目标值计算中的影响.

性质 1：机器空闲时间与作业运输时间是计算工序 $O_{i,j}$ 完工时间 $C_{i,j}$ 及对应能耗的必要条件. 若缺失任一要素，$C_{i,j}$ 及其能耗将无法被准确确定.

分析：设前序工序 $O_{i,j-1}$ 完工时间为 $C_{i,j-1}$，工序 $O_{i,j}$ 开始时间为 $S_{i,j}$，机器 k' 可用时间为 M_t，作业 i 运输时间为 $TR_{i,j,k,k'}$，若忽略工厂 f 中机器可用时间 M_t，即未考虑前一作业 i' 对所选机器 k 的占用情况，可能导致调度逻辑错误，进而使作业 i 在机器尚未空闲时被安排加工，违反 $S_{i,j} \geqslant M_t$ 的约束. 同理，若忽略运输时间 $TR_{i,j,k,k'}$，等同于假设作业 i 在上一工序 $O_{i,j-1}$ 完成后立即被安排在下一台机器 k' 上加工，这将违背作业的物理传输过程，使得 $S_{i,j} < C_{i,j-1} + TR_{i,j,k,k'}$，违反前置约束. 因此，$M_t$ 与 $TR_{i,j,k,k'}$ 是完工时间及其能耗可行性计算的必要条件.

性质 2：在已知机器空闲时间 M_t 和作业运输时间 $TR_{i,j,k,k'}$ 的前提下，工序 $O_{i,j}$ 的完工时间 $C_{i,j}$ 及其能耗可被唯一确定. 因此，上述两个要素构成了完工时间及其能耗计算的充分条件.

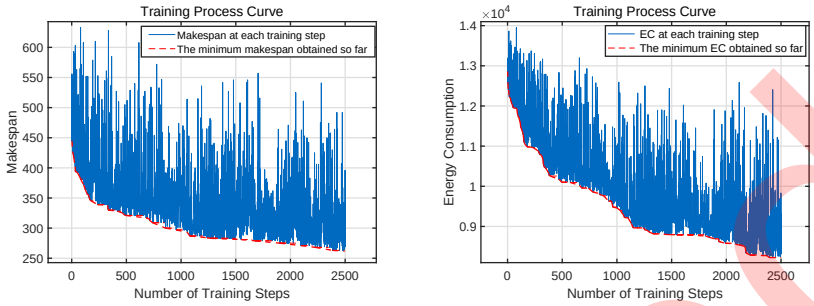
分析：设前序工序 $O_{i,j-1}$ 完工时间为 $C_{i,j-1}$，工序 $O_{i,j}$ 开始及完工时间分别为 $S_{i,j}$ 和 $C_{i,j}$，机器 k' 可用时间为 M_t，作业 i 运输时间为 $TR_{i,j,k,k'}$，加工时间为 $T_{i,j,f,k'}$. 基于主动调度假设，即除运输与资源可用性外无额外延迟因素（如抢占或设置时间），则工序 $O_{i,j-1}$ 的最早开始时间由公式 $S_{i,j} = \max\{M_t, C_{i,j-1} + TR_{i,j,k,k'}\}$ 唯一决定. 结合已知加工时间 $T_{i,j,f,k'}$，完工时间为：$C_{i,j} = S_{i,j} + T_{i,j,f,k} = \max\{M_t, C_{i,j-1} + TR_{i,j,k,k'}\} + T_{i,j,f,k'}$. 因此，$M_t$ 与 $TR_{i,j,k,k'}$ 足以唯一确定 $C_{i,j}$ 及其能耗的计算，构成其充分条件.

基于**性质 1** 及**性质 2**，本文进一步对机器空闲和作业运输进行**量化分析**：设前序工序 $O_{i,j-1}$ 完工时间为 $C_{i,j-1}$，机器 k' 可用时间为 M_t，作业 i 运输时间为 $TR_{i,j,k,k'}$，其加工时间为 $T_{i,j,f,k'}$，则工序 $O_{i,j}$ 的完工时间为：$C_{i,j} = \max\{M_t, C_{i,j-1} + TR_{i,j,k,k'}\} + T_{i,j,k'}$. 进一步引入机器空闲时间 $I_O = \max\{0, (C_{i,j-1} + TR_{i,j,k,k'}) - M_t\}$ 与作业等待时间 $W_O = \max\{0, M_t - (C_{i,j-1} + TR_{i,j,k,k'})\}$，有以下关系成立：

$$S_{i,j} = M_t + I_O = (C_{i,j-1} + TR_{i,j,k,k'}) + W_O,$$

$$C_{i,j} = S_{i,j} + T_{i,j,f,k'} = M_t + I_O + T_{i,j,f,k'} = (C_{i,j-1} + TR_{i,j,k,k'}) + W_O + T_{i,j,f,k'}.$$

https://www.sciengine.com/doi/10.1360/SSI-2025-0113



(a)　(b)

图 8　30_3 实例的 DQN-QD 训练过程曲线.(a) 每个训练步的完工时间.(b) 每个训练步的能耗.

Figure 8　The 30_3 training process curve of DQN-QD. (a) Makespan at each training step. (b) EC at each training step.

I_O 和 W_O 至少一个为零，分别对应机器空闲或运输完成后作业等待的情形. 由此可知，运输时间 $TR_{i,j,k,k'}$ 直接影响作业 i 到达时刻，进而影响机器 k' 是否空闲，以及最终的完工时间 $C_{i,j}$. 而机器 k' 空闲时间 M_t 决定作业 i 是否需要等待. 因此两者共同决定了完工时间相对于理想情况的延迟程度.

由上述分析可知，$C_{i,j}$ 随 M_t 和 $TR_{i,j,k,k'}$ 增大而非减. 若运输时间增加，可能导致作业晚到，且增加了运输能耗和机器空闲能耗；若机器晚可用，作业需等待，可能导致完工时间延长. 此外，$C_{i,j}$ 与作业运输及机器空闲存在以下关系：$C_{i,j}-(C_{i,j-1}+T_{i,j,f,k'})=\max\{M_t-C_{i,j-1},TR_{i,j,k,k'}\}$，即调度延迟来源于运输或机器空闲中的较大者. 而运输时间及机器空闲的增加，同样可能导致运输和空闲能耗的增加. 此外，上述分析明确量化了运输时间和机器空闲时间对完工时间及能耗的决定性影响，为构造特征空间中特征（描述问题行为或者特性）的选择提供了理论依据. 在加工时间 $T_{i,j,f,k'}$ 已知的前提下，选择运输次数（AGV 运输特征）与机器空闲次数（机器空闲特征）作为 QD 优化的行为特征，能够有效反应延迟情况，并基于此结合 DQN 及知识辅助的协作启发式搜索策略，指引算法朝有利于减缓延迟情况的区域探索，最终实现能耗/完工时间的协同优化.

4.9　DQN 训练过程

由于每次训练均为实时在线执行, 受篇幅限制, 无法展示所有实例的训练曲线, 因此本文随机选取 30_3, 50_4 和 100_4 实例, 并绘制训练过程中两个优化目标的变化趋势. 训练过程遵循第 4.4 节所述的实验设置. 如图 8, 9 和10 所示, (a) 和 (b) 分别展示完工时间和能耗在训练过程中的变化趋势. 从训练曲线可以看出, 无论是在小规模还是大规模实例中, 随着训练步数的增加, 完工时间和能耗均呈现下降趋势, 这一结果表明, 采用 DQN 训练局部搜索算子能够有效提升 QD 的优化性能.

4.10　真实案例验证与分析

为进一步评估所提 DQN-QD 算法的实际适用性，本文选取中国某煤机结构件制造车间[9] 作为实际应用场景开展验证. 图 11展示了该车间的平面布局. 该车间配备了 6 类主要加工设备，共配置 11 台机器，部分工艺环节具备并行加工能力. 工件在各加工设备之间的转运工作由 2 台 AGV 完成. 具体的设备分类与工艺流程安排详见表 12.

在该场景下, 本文对比评估了 DQN-QD 与其他优化算法的调度性能, 结果如表 13,14和15所示. DQN-QD 在 20 次案例测试中均表现出最小 GD 值 (平均值为 0.07), 远优于其他算法. 这表明所提

https://www.sciengine.com/doi/10.1360/SSI-2025-0113

表 12　制造车间内六道流程的可选机器

Table 12　Available machines for six processes in the manufacturing plant

类型	功能	可选机器
1	激光切割	M1, M2, M3
2	火焰切割	M4, M5
3	板材压平	M6, M7
4	弯曲	M8
5	研磨	M9, M10
6	钻孔	M11

表 13　DQN-QD 与现有算法在真实案例中的 GD 比较结果

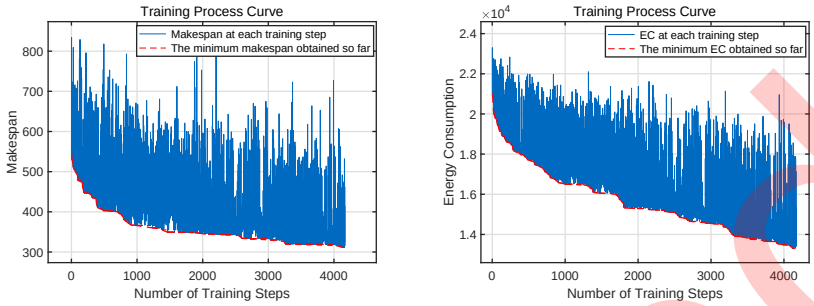
Table 13　Comparison of *GD* results between DQN-QD and existing algorithms in real cases

	GD							
Instance	DQN-QD	NSGA-II	MOEA/D	MOME	EDA-VNS	IGSA	SPAMA	DQCE
Test1	**0.20**	0.49	0.41	0.32	0.29	0.28	0.53	0.27
Test2	**0.08**	0.54	0.36	0.42	0.27	0.25	0.47	0.21
Test3	**0.12**	0.52	0.34	0.38	0.28	0.19	0.48	0.26
Test4	**0.07**	0.48	0.42	0.28	0.21	0.21	0.52	0.27
Test5	**0.09**	0.41	0.41	0.32	0.22	0.20	0.49	0.23
Test6	**0.04**	0.52	0.39	0.32	0.26	0.26	0.47	0.25
Test7	**0.08**	0.39	0.38	0.33	0.31	0.26	0.41	0.30
Test8	**0.12**	0.44	0.34	0.30	0.26	0.24	0.48	0.21
Test9	**0.10**	0.46	0.27	0.27	0.25	0.27	0.52	0.24
Test10	**0.05**	0.50	0.32	0.45	0.25	0.18	0.46	0.29
Test11	**0.02**	0.40	0.30	0.33	0.28	0.24	0.40	0.21
Test12	**0.09**	0.52	0.35	0.32	0.32	0.19	0.49	0.24
Test13	**0.06**	0.34	0.30	0.36	0.28	0.20	0.57	0.35
Test14	**0.04**	0.43	0.34	0.33	0.26	0.18	0.46	0.34
Test15	**0.00**	0.46	0.41	0.29	0.26	0.27	0.51	0.28
Test16	**0.05**	0.39	0.41	0.37	0.23	0.19	0.54	0.30
Test17	**0.02**	0.54	0.36	0.31	0.37	0.18	0.47	0.32
Test18	**0.03**	0.42	0.34	0.38	0.27	0.24	0.54	0.28
Test19	**0.01**	0.49	0.42	0.41	0.35	0.18	0.48	0.24
Test20	**0.11**	0.46	0.41	0.32	0.27	0.16	0.53	0.29
Mean	**0.07**	0.46	0.36	0.34	0.28	0.22	0.49	0.27

表 14　DQN-QD 与现有算法在真实案例中的 IGD 比较结果

Table 14　Comparison of *IGD* results between DQN-QD and existing algorithms in real cases

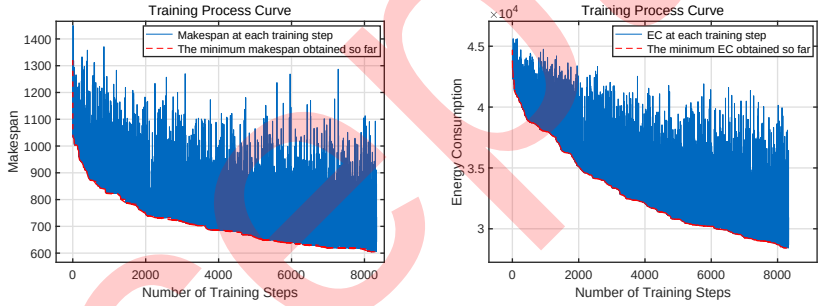
	IGD							
Instance	DQN-QD	NSGA-II	MOEA/D	MOME	EDA-VNS	IGSA	SPAMA	DQCE
Test1	**0.39**	0.83	0.71	0.84	0.61	0.42	0.86	0.63
Test2	**0.19**	0.87	0.73	0.77	0.64	0.50	0.93	0.64
Test3	**0.25**	0.89	0.74	0.76	0.62	0.44	0.95	0.67
Test4	**0.12**	0.92	0.72	0.78	0.62	0.50	0.88	0.64
Test5	**0.20**	0.91	0.71	0.76	0.62	0.36	0.96	0.64
Test6	**0.11**	0.87	0.75	0.75	0.61	0.57	0.91	0.66
Test7	**0.15**	0.93	0.65	0.78	0.58	0.52	0.89	0.58
Test8	**0.21**	0.91	0.76	0.78	0.62	0.54	0.95	0.65
Test9	**0.22**	0.92	0.78	0.77	0.65	0.39	0.90	0.63
Test10	**0.15**	0.96	0.64	0.75	0.64	0.44	0.82	0.64
Test11	**0.05**	0.94	0.70	0.79	0.62	0.53	0.96	0.65
Test12	**0.18**	0.89	0.74	0.77	0.63	0.38	0.85	0.65
Test13	**0.10**	0.94	0.70	0.79	0.62	0.43	0.96	0.62
Test14	**0.09**	0.95	0.76	0.77	0.60	0.37	0.91	0.67
Test15	**0.00**	0.91	0.71	0.80	0.64	0.43	0.84	0.59
Test16	**0.09**	0.96	0.71	0.76	0.65	0.46	0.95	0.64
Test17	**0.04**	0.87	0.73	0.79	0.63	0.44	0.92	0.62
Test18	**0.05**	0.93	0.74	0.73	0.60	0.42	0.91	0.62
Test19	**0.04**	0.92	0.72	0.76	0.59	0.47	0.86	0.65
Test20	**0.19**	0.90	0.71	0.74	0.64	0.40	0.86	0.65
Mean	**0.14**	0.91	0.72	0.77	0.62	0.45	0.90	0.64



(a)　(b)

图 9　DQN-QD 的训练过程曲线. (a) 每个训练步的完工时间. (b) 每个训练步的能耗

Figure 9　The training process curve of DQN-QD. (a) Makespan at each training step. (b) EC at each training step



(a)　(b)

图 10　100_4 实例的 DQN-QD 训练过程曲线.(a) 每个训练步的完工时间.(b) 每个训练步的能耗.

Figure 10　The 100_4 training process curve of DQN-QD. (a) Makespan at each training step. (b) EC at each training step.

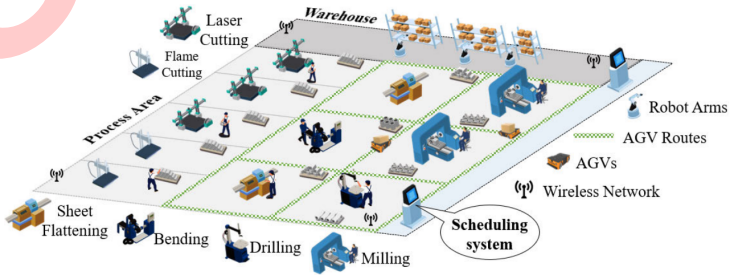


图 11　制造车间内的布局[9]

Figure 11　Layout in the manufacturing plant[9]

算法在逼近真实帕累托前沿方面具备明显优势. 同样, DQN-QD 在测试中还表现出最小的 *IGD* 值 (平均值为 0.14) 和最大的 *HV* 值 (平均值为 0.81). 这一表现得益于其质量多样性框架、基于领域知识的协作启发式搜索策略, 以及 DQN 支持的智能算子选择机制三者的有效协同. 该方法不仅提升了解的质量和分布均衡性，还增强了算法在多目标复杂调度问题中的适应性与泛化能力，特别适用于能耗与完工时间并重的工业制造场景.

https://www.sciengine.com/doi/10.1360/SSI-2025-0113

表 15　DQN-QD 与现有算法在真实案例中的 HV 比较结果

Table 15　Comparison of *HV* results between DQN-QD and existing algorithms in real cases

Instance	HV							
	DQN-QD	NSGA-II	MOEA/D	MOME	EDA-VNS	IGSA	SPAMA	DQCE
Test1	**0.52**	0.16	0.24	0.19	0.33	0.48	0.14	0.31
Test2	**0.74**	0.13	0.23	0.20	0.31	0.42	0.12	0.32
Test3	**0.68**	0.13	0.23	0.22	0.33	0.50	0.11	0.29
Test4	**0.84**	0.12	0.22	0.21	0.32	0.44	0.14	0.31
Test5	**0.74**	0.13	0.24	0.22	0.34	0.55	0.09	0.32
Test6	**0.84**	0.13	0.21	0.22	0.32	0.37	0.11	0.29
Test7	**0.79**	0.11	0.27	0.20	0.34	0.40	0.13	0.35
Test8	**0.69**	0.12	0.21	0.22	0.33	0.39	0.10	0.30
Test9	**0.70**	0.11	0.22	0.22	0.30	0.50	0.11	0.32
Test10	**0.78**	0.10	0.29	0.22	0.30	0.48	0.17	0.32
Test11	**0.93**	0.11	0.26	0.22	0.32	0.39	0.09	0.31
Test12	**0.75**	0.13	0.23	0.20	0.30	0.53	0.15	0.32
Test13	**0.86**	0.12	0.25	0.21	0.32	0.51	0.08	0.34
Test14	**0.88**	0.11	0.20	0.21	0.33	0.55	0.12	0.29
Test15	**1.00**	0.14	0.24	0.19	0.32	0.48	0.15	0.34
Test16	**0.85**	0.11	0.24	0.21	0.31	0.47	0.10	0.30
Test17	**0.96**	0.13	0.23	0.21	0.31	0.50	0.11	0.33
Test18	**0.91**	0.11	0.23	0.23	0.34	0.48	0.12	0.33
Test19	**0.97**	0.10	0.22	0.20	0.34	0.49	0.14	0.30
Test20	**0.73**	0.14	0.24	0.23	0.31	0.53	0.14	0.32
Mean	**0.81**	0.12	0.24	0.21	0.32	0.47	0.12	0.31

4.11　发现与讨论

本节对 DQN 选择机制在 QD 框架中的应用进行深入分析, 重点探讨启发式局部搜索算子在迭代过程中的作用, 旨在评估各类搜索算子对算法性能的具体影响. 为此, 本文统计每类搜索算子在所有实例中成功更新特征空间的频率, 以衡量其对解集质量与多样性的贡献. 成功更新指新生成的解在特征空间单元格内相较于已有解具有优势 (占优或互不支配). 图 12展示了四种搜索算子成功更新特征空间的比例.

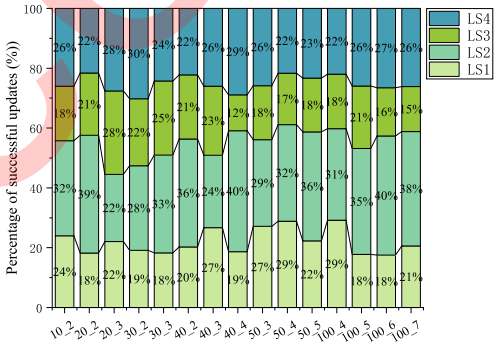


图 12　各局部搜索算子成功更新特征空间的比例

Figure 12　Percentage of successful updates for all local search operators

如图12所示, LS2 与 LS4 具有最高的成功更新率, 表明针对机器与 AGV 设计的启发式局部搜索算子对优化目标具有最显著的影响. 该结果进一步验证了机器空闲时间与 AGV 运输在 DFJSP-AGVs 中的核心作用, 也进一步印证了所设计特征空间 (以 NI 与 NT 为维度) 的合理性与区分度. 此外，本文的主要创新在于：将 DQN 与面向 DFJSP-AGVs 的 QD 优化及协作启发式搜索深度融合. 借助 DQN 对状态向量的全局建模与动态学习，算法能实现对搜索算子的自适应选择，从而提升解集收敛速度与全局质量. 通过对特征空间中帕累托解集的结构化维护，DQN-QD 构建出既具有行为多样性又具备性能优势的解集分布，凸显了质量多样性优化在复杂调度任务中的强适配性. 尽管当前方法在性能表现上优于多种对比算法，仍存在可改进方向. 如提炼更多针对机器/运输的特征及搜索策略以供算法选择制.

https://www.sciengine.com/doi/10.1360/SSI-2025-0113

参考文献

1 Mou J, Gao K, Duan P, et al. A machine learning approach for energy-efficient intelligent transportation scheduling problem in a real-world dynamic circumstances. IEEE Trans Intell Transp Syst, 2023, 24: 15527–15539

2 Zhong X Y, Han Y Y, Yao X J, et al. An evolutionary algorithm for the multi-objective flexible job shop scheduling problem with uncertain processing time. Sci Sin Inform, 2023, 53(4): 737-757 [钟小玉, 韩玉艳, 姚香娟, 等. 不确定工时下多目标柔性作业车间调度问题的进化求解方法. 中国科学：信息科学, 2023, 53(4): 737-757]

3 Pan Z, Wang L, Wang J, et al. Distributed energy-efficient flexible manufacturing with assembly and transportation: a knowledge-based bi-hierarchical optimization approach. IEEE Trans Autom Sci Eng, 2024, 21: 2161-2175.

4 Qin H X, Xiang Y, Liu F Q, et al. Enhancing quality-diversity algorithm by reinforcement learning for flexible job shop scheduling with transportation constraints. Swarm Evol Comput, 2025, 93: 101849.

5 Huang J P, Gao L, Li X Y, et al. A hierarchical multi-action deep reinforcement learning method for dynamic distributed job-shop scheduling problem with job arrivals. IEEE Trans Autom Sci Eng, 2025, 22: 2501–2513.

6 Xu G, Bao Q, Zhang H. Multi-objective green scheduling of integrated flexible job shop and automated guided vehicles. Eng Appl Artif Intell, 2023, 126: 106864.

7 Li W, Li H, Wang Y, et al. Optimizing flexible job shop scheduling with automated guided vehicles using a multi-strategy-driven genetic algorithm. Egypt Inform J, 2024, 25: 100437.

8 Chen R, Wu B, Wang H, et al. A q-learning based NSGA-II for dynamic flexible job shop scheduling with limited transportation resources. Swarm Evol Comput, 2024, 90: 101658.

9 Yao Y, Liu Q, Fu L, et al. A novel mathematical model for the flexible job-shop scheduling problem with limited automated guided vehicles. IEEE Trans Autom Sci Eng, 2024: 1–14.

10 Zhao L, Fan J, Zhang C, et al. A DRL-based reactive scheduling policy for flexible job shops with random job arrivals. IEEE Trans Autom Sci Eng, 2024, 21(3): 2912–2923.

11 Xin J, Qu Y, Zhang F, et al. Distributed model predictive contouring control for real-time multi-robot motion planning. Complex Syst Model Simul, 2022, 2(4): 273–287.

12 Zhou X, Wang F, Shen N, et al. A green flexible job-shop scheduling model for multiple AGVs considering carbon footprint. Syst, 2023, 11(8): 427.

13 Saidi-Mehrabad M, Dehnavi-Arani S, Evazabadian F, et al. An ant colony algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs. Comput Ind Eng, 2015, 86: 2–13.

14 Chaudhry I A, Rafique A F, Elbadawi I A Q, et al. Integrated scheduling of machines and automated guided vehicles (AGVs) in flexible job shop environment using genetic algorithms. Int J Ind Eng Comput, 2022, 13(3): 343–362.

15 Han X, Cheng W, Meng L, et al. A dual population collaborative genetic algorithm for solving flexible job shop scheduling problem with AGV. Swarm Evol Comput, 2024, 86: 101538.

16 Dai M, Tang D, Giret A, et al. Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints. Robot Comput Integr Manuf, 2019, 59: 143–157.

17 Pan Z, Wang L, Zheng J, et al. A learning-based multi-population evolutionary optimization for flexible job shop scheduling problem with finite transportation resources. IEEE Trans Evol Comput, 2022, (): 1–1.

18 He L, Chiong R, Li W, et al. A multiobjective evolutionary algorithm for achieving energy efficiency in production environments integrated with multiple automated guided vehicles. Knowl Based Syst, 2022, 243: 108315.

19 Meng L, Zhang C, Ren Y, et al. Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem. Comput Ind Eng, 2020, 142: 106347.

20 Meng L, Zhang C, Zhang B, et al. Mathematical modeling and optimization of energy-conscious flexible job shop scheduling problem with worker flexibility. IEEE Access, 2019, 7: 68043–68059.

21 Tutumlu B, Saraç T. A MILP model and a hybrid genetic algorithm for flexible job-shop scheduling problem with job-splitting. Comput Oper Res, 2023, 155: 106222.

22 Xie J, Li X, Gao L, et al. A hybrid genetic tabu search algorithm for distributed flexible job shop scheduling problems. J Manuf Syst, 2023, 71: 82–94.

23 An Y, Chen X, Gao K, et al. Multi-objective flexible job-shop rescheduling with new job insertion and machine preventive maintenance. IEEE Trans Cybern, 2023, 53(5): 3101–3113.

24 Song W, Chen X, Li Q, et al. Flexible job-shop scheduling via graph neural network and deep reinforcement learning. IEEE Trans Ind Inform, 2023, 19(2): 1600–1610.

25 Cao Z, Lin C, Zhou M. A knowledge-based cuckoo search algorithm to schedule a flexible job shop with sequencing flexibility. IEEE Trans Autom Sci Eng, 2021, 18(1): 56–69.

26 Li R, Gong W, Lu C, et al. A learning-based memetic algorithm for energy-efficient flexible job-shop scheduling with type-2 fuzzy processing time. IEEE Trans Evol Comput, 2023, 27(3): 610–620.

27 Zhang Q, Manier H, Manier M A. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. Comput Oper Res, 2012, 39(7): 1713–1723.

28 Karimi S, Ardalan Z, Naderi B, et al. Scheduling flexible job-shops with transportation times: mathematical models and a hybrid imperialist competitive algorithm. Appl Math Model, 2016, 41: 667–682.

29 Pal M, Mittal M L, Soni G, et al. A multi-agent system for FJSP with setup and transportation times. Expert Syst Appl, 2023, 216: 119474.

30 Berterottiere L, Dauzère-Pérès S, Yugma C. Flexible job-shop scheduling with transportation resources. Eur J Oper Res, 2024, 312(3): 890–909.

31 Li J Q, Du Y, Gao K Z, et al. A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem. IEEE Trans Autom Sci Eng, 2022, 19(3): 2153–2170.

32 Yao Y, Li X, Gao L. A DQN-based memetic algorithm for energy-efficient job shop scheduling problem with integrated limited AGVs. Swarm Evol Comput, 2024, 87: 101544.

33 Li Y, Gu W, Yuan M, et al. Real-time data-driven dynamic scheduling for flexible job shop with insufficient transportation resources using hybrid deep Q network. Robot Comput Integr Manuf, 2022, 74: 102283.

34 Du Y, Li J Q, Luo C, et al. A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations. Swarm Evol Comput, 2021, 62: 100861.

35 Zhang Z Q, Wu F C, Qian B, et al. A Q-learning-based hyper-heuristic evolutionary algorithm for the distributed flexible job-shop scheduling problem with crane transportation. Expert Syst Appl, 2023, 234: 121050.

36 Luo Q, Deng Q, Gong G, et al. An efficient memetic algorithm for distributed flexible job shop scheduling problem with transfers. Expert Syst Appl, 2020, 160: 113721.

37 Yuan M, Zheng L, Huang H, et al. Research on flexible job shop scheduling problem with AGV using double DQN. J Intell Manuf, 2023. [Accessed 2023-11-23].

38 Qin H, Han Y, Chen Q, et al. Energy-efficient iterative greedy algorithm for the distributed hybrid flow shop scheduling with blocking constraints. IEEE Trans Emerg Top Comput Intell, 2023, 7(5): 1442–1457.

39 Luo L, Zhao N, Zhu Y, et al. A* guiding DQN algorithm for automated guided vehicle pathfinding problem of robotic mobile fulfillment systems. Comput Ind Eng, 2023, 178(C).

40 Qian C, Xue K, Wang R J. Quality-diversity algorithms can provably be helpful for optimization. In: Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI), 2024. 773: 6994–7002.

41 Cully A, Clune J, Tarapore D, et al. Robots that can adapt like animals. Nature, 2015, 521(7553): 503–507.

42 Ecoffet A, Huizinga J, Lehman J, et al. First return, then explore. Nature, 2021, 590: 580–586.

43 Li R, Gong W, Wang L, Lu C, Dong C. Co-evolution with deep reinforcement learning for energy-aware distributed heterogeneous flexible job shop scheduling. IEEE Trans Syst Man Cybern Syst, 2023, pp. 1–11.

44 Mouret J B, Clune J. Illuminating search spaces by mapping elites. arXiv preprint, 2015, arXiv:1504.04909.

45 Zhang H, Qin C, Zhang W, Xu Z, Xu G, Gao Z. Energy-saving scheduling for flexible job shop problem with AGV transportation considering emergencies. Systems, 2023, 11(2).

46 Zhang F, Li R, Gong W. Deep reinforcement learning-based memetic algorithm for energy-aware flexible job shop scheduling with multi-AGV. Comput Ind Eng, 2024, 189: 109917.

47 Bilge U, Ulusoy G. A time window approach to simultaneous scheduling of machines and material handling system in an FMS. Operations Research, 1995, 43(6): 1058–1070.

48 Li M, Chen T, Yao X. How to evaluate solutions in Pareto-based search-based software engineering: A critical review and methodological guidance. IEEE Trans Softw Eng, 2022, 48(5): 1771–1799.

49 Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput, 2002, 6(2): 182–197.

50 Zhang Q, Li H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput, 2007, 11(6): 712–731.

51 Pierrot T, Richard G, Beguir K, Cully A. Multi-objective quality diversity optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference. New York, NY, USA: Association for Computing Machinery, 2022. 139–147.

52 Li R, Gong W, Wang L, Lu C, Zhuang X. Surprisingly popular-based adaptive memetic algorithm for energy-efficient distributed flexible job shop scheduling. IEEE Trans Cybern, 2023, 1–11.

53 Li W, Li H, Wang Y, Han Y. Optimizing flexible job shop scheduling with automated guided vehicles using a multi-strategy-driven genetic algorithm. Egypt Inform J, 2024, 25: 100437.

54 Luo S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. Appl Soft Comput, 2020, 91: 106208.

55 Huang J P, Gao L, Li X Y, Zhang C J. A novel priority dispatch rule generation method based on graph neural network and reinforcement learning for distributed job-shop scheduling. J Manuf Syst, 2023, 69: 119–134.

Intelligent scheduling approach for distributed flexible job-shop with deep reinforcement learning and quality-diversity optimization

Haoxiang QIN[1], Yi XIANG[1*], Yuyan HAN[2], Yuting WANG[2], Qingda CHEN[3,4] & Ping ZHOU[4]

1. *School of Software Engineering, South China University of Technology, Guangzhou 510006, China*
2. *School of Computer Science, Liaocheng University, Liaocheng 252000, China*
3. *School of Electronic Information Engineering, Inner Mongolia University, Hohhot 010021, China*
4. *The State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110004, China*

* Corresponding author. E-mail: xiangyi@scut.edu.cn

Abstract In the field of intelligent manufacturing, the integrated scheduling of automated guided vehicles (AGVs) and machines has a significant impact on makespan and energy consumption. However, existing scheduling methods struggle to efficiently coordinate AGV transportation and task execution, especially in distributed flexible job shop scheduling (DFJSP) scenarios, where this challenge is particularly pronounced. To address this issue, this paper proposes a deep reinforcement learning-enhanced Quality-Diversity (QD) optimization algorithm that effectively leverages transportation and machine behavioral features to generate high-quality and diverse Pareto-optimal solutions. First, a knowledge-assisted collaborative heuristic strategy is designed to optimize the scheduling of jobs, machines, and factories by considering AGV transportation characteristics, thereby improving the overall solution quality. Second, to address the low utilization rate of search operators, an intelligent selection mechanism based on deep reinforcement learning is introduced to overcome the limitations of random selection, enhancing both search efficiency and optimization performance. Simulation experiments demonstrate that the proposed algorithm significantly outperforms existing methods in optimizing makespan and energy consumption, validating its effectiveness.

Keywords distributed flexible job shop scheduling, automated guided vehicle transportation, quality-diversity, deep reinforcement learning, multi-objective optimization