控制与决策
Control and Decision

强化学习驱动进化的模因算法求解准时制分布式柔性作业车间调度问题

赵仕存, 周泓

引用本文:
赵仕存, 周泓. 强化学习驱动进化的模因算法求解准时制分布式柔性作业车间调度问题[J]. 控制与决策, 2026, 41(4): 905-918.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2025.0310

强化学习驱动进化的模因算法求解准时制分布式柔性作业车间调度问题

赵仕存[1,2]，周 泓[1,2†]

(1. 北京航空航天大学 经济管理学院，北京 100191；
2. 北京航空航天大学 城市运行应急保障模拟技术北京市重点实验室，北京 100191)

摘 要: 研究准时制生产条件下的分布式柔性作业车间调度问题. 企业需要根据工件的交付时间决定启用工厂的数目, 并在各工厂内部进行调度, 其目标是最小化完工时间、能量消耗和总生产成本. 鉴于此, 建立多目标混合整数线性规划模型来刻画此问题, 进而通过强化学习驱动进化的模因算法来完成求解. 首先, 通过启发式方法培育高质量的初始种群; 然后, 在进化过程中, 强化学习将交配池中的父本视为状态和动作, 并以子代的质量评估环境奖励, 目的是为每个父本推荐最合适的搭档以生成高质量的后代, 降低随机匹配的盲目性; 最后, 自适应局部搜索机制作用于进化停滞的种群, 能够进一步提升搜索质量. 通过在两类标准测试集进行仿真实验并与 5 种算法进行比较, 验证了所提出算法的有效性.

关键词: 分布式柔性作业车间调度；准时制生产；绿色调度；强化学习；模因算法；多目标优化

中图分类号: TP18 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2025.0310

引用格式: 赵仕存, 周泓. 强化学习驱动进化的模因算法求解准时制分布式柔性作业车间调度问题 [J]. 控制与决策, 2026, 41(4): 905-918.

Reinforcement learning-driven evolutionary memetic algorithm for solving just-in-time distributed flexible job shop scheduling problem

ZHAO Shi-cun[1,2]，*ZHOU Hong*[1,2†]

(1. School of Economics and Management， Beihang University， Beijing 100191， China； 2. Beijing Key Laboratory of Emergency Support Simulation Technologies for City Operations， Beihang University， Beijing 100191，China)

Abstract: This paper studies the distributed flexible job shop scheduling problem incorporating just-in-time production consideration. Enterprises need to determine the number of factories to activate based on job delivery time and schedule jobs within activated factories to minimize the makespan, energy consumption, and production cost. A multi-objective mixed-integer linear programming model is constructed and solved by a memetic algorithm with reinforcement learning-driven evolutionary strategy. A heuristic initialization strategy is adopted to breed a high-quality population. During evolution, reinforcement learning treats evolved individuals as states, and the selection of partners as actions, deriving rewards from the quality of offspring. By recommending suitable partners for each parent, the blindness inherent in random matching is reduced. Finally, an adaptive local search strategy is applied for stagnant individuals to enhance exploitation performance. The effectiveness of the proposed algorithm is verified by simulation experiments conducted on two standard test suites and comparisons with five representative algorithms.

Keywords: distributed flexible job shop scheduling； just-in-time production； green scheduling； reinforcement learning；memetic algorithm；multi-objective optimization

0 引 言

随着经济全球化的推进, 现代制造业的生产模式正逐步从集中式大规模生产转向分布式小批量生产[1-2]. 相较于传统的生产模式, 分布式制造在快速响应客户需求、提升产品质量等方面具备显著优势的同时也更具挑战性. 因为其必须同时确定分配给各

收稿日期: 2025-03-26；**录用日期:** 2025-05-06.
基金项目: 国家自然科学基金项目 (72471015).
责任编委: 王凌.
†通信作者. E-mail: h_zhou@buaa.edu.cn.

生产单元的作业以及每个单元内部的生产顺序和加工路径[3-4]. 分布式柔性作业车间调度问题 (distributed flexible job shop scheduling problem, DFJSP) 是目前最具代表性的生产场景, 近年来受到了广泛研究[5]. 由于 DFJSP 涉及到多个工厂、不同设备与资源间的协调, 以及需要考虑加工过程中各种约束和目标, 使得该问题的求解难度极高.

为求解 DFJSP, 学界开展了大量的工作. Meng 等[6] 以最小化完工时间为目标, 通过商业求解器 CPLEX 对混合整数线性规划模型和约束规划模型进行了精确求解. 尽管 CPLEX 等工具能够得到理论最优解, 但是 DFJSP 为 NP 难题, 其解空间组合爆炸, 导致精确求解工具无法作用于大规模问题[7-8]. 因此, 以遗传算法、模因算法为代表的进化算法是求解 DFJSP 的主流方式. 它们不依赖于问题特征, 逐步探索解空间中最有前途的区域, 以找到高质量调度方案. Chan 等[9] 率先研究了包含多个柔性制造单元的调度问题, 并运用遗传算法寻找在最小化完工时间目标下的最优解集; Fu 等[10] 研究了随机加工时间条件下的 DFJSP, 并提出了一种改进的进化算法来优化完工时间和总拖期; 李佳磊等[11] 提出了基于双种群的遗传算法来求解带有维修活动的 DFJSP, 目的是最小化完工时间; Xie 等[12] 以完工时间为目标, 提出了混合遗传-禁忌搜索算法, 更新了多个 DFJSP 案例的上界. 近年来, 节能调度已成为智能制造领域的核心主题[13-14]. Li 等[15] 提出了一种改进的模因算法来寻求高效且节能的调度方案, 算法采用自适应算子选择机制提升了局部搜索效率; Du 等[16] 提出了分布估计算法和变邻域搜索机制混合的优化方法, 专门用于解决涉及起重机运输的 DFJSP, 目的是最小化完工时间和能耗; 随后, Zhang 等[17] 提出了一种超启发式进化算法来求解同类问题, 以优化加权完工时间和能耗; Pan 等[18] 设计了统计学习和进化学习集成的双学习演化算法, 目的是优化完工时间和能耗; 针对带有维修活动的 DFJSP, Yan 等[19] 探寻了最小化完工时间、能耗和维修成本等目标下的最优调度方案.

随着问题复杂程度的不断提升, 单纯依赖进化算法已难以满足高效求解需求[20]. 为此, 强化学习技术 (Q学习、双Q学习、深度Q网络等) 逐渐被融入算法框架, 以提升求解性能. 一些研究利用强化学习技术自适应调整参数来提升优化效果. Li 等[21] 以完工时间和能耗为目标, 利用Q学习调整了 MOEA/D 算法中的邻域数目; Zhang 等[22] 利用Q学习调整了人工蜂群算法中的搜索概率来寻求最小化完工时间和工厂负载目标下的最优调度方案; Tang 等[23] 使用双Q学习自适应调整了遗传算法中的交叉和变异概率. 另外, 强化学习还被用于推荐合适的算子, 避免随机选择的盲目性. 唐红涛等[24] 使用Q学习为人工蜂群算法中的雇佣峰个体选取了合适的局部搜索算子; Zhao 等[25] 使用双Q学习推荐了最佳算子, 目的是在最小化完工时间和能耗的同时提升调度方案的鲁棒性; Li 等[26] 以完工时间和能耗为目标, 提出了基于深度Q网络的协同优化算法, 其中深度Q网络负责推荐合适的搜索算子. 此外, 通过强化学习选择高效的搜索机制也是提升搜索性能的有效方式. Fu 等[27] 以完工时间、能耗和工厂负载为优化目标, 设计了多种进化机制, 并在迭代过程中通过Q学习选择合适的算法进行求解.

目前, DFJSP 研究的主要优化目标为完工时间和能耗. 此外, 以上文献均未考虑工厂启动成本, 且要求所有工厂共同协作完成订单. 现实生产运作中, 每个工厂均伴随着启动成本. 因此, 企业必须要根据订单数量和交付时间合理决定启用工厂的数量. 启用数目过少会导致订单延迟 (tardiness), 而过多则会导致早交 (earliness). 延迟交货会带来罚款和信誉损失, 而提前完工则可能会引发仓库持有成本增加或产品降值. 因此, 在准时制生产条件下, 企业必须根据实际情况做出决策, 以在完工效率、能量消耗与生产成本等目标间实现权衡, 这使得问题的求解变得更加困难. 此外, 目前学界主要利用强化学习技术对演化算法进行参数自适应调整、算子智能推荐以及搜寻机制动态选择, 而关于强化学习驱动的父本匹配策略鲜有人涉及.

综上, 本文考虑准时制生产条件下的分布式柔性作业车间调度问题 (just-in-time DFJSP, JITDFJSP), 目的是优化完工时间、能量消耗和生产成本. 为求解该问题, 本文提出Q学习驱动进化的模因算法 (Q-learning-driven evolutionary memetic algorithm, QMA). 通过启发式初始化、Q学习驱动的父本匹配、自适应局部搜索等策略, 实现对 JITDFJSP 的高效求解, 获取高质量的调度解集. 该算法的主要特点如下: 1) 设计问题特定的混合编码机制来反映 JITDFJSP 的决策信息; 2) 在保证多样性的前提下, 融合 3 种启发式方法生成高质量的初始种群; 3) 提出基于Q学习的父本匹配策略, 能够提升进化效率, 降低随机匹配的盲目性; 4) 在搜索后期, 采用自适应局部搜索策略, 助力进化停滞的种群逃离局部最优区域.

1　模型建立

1.1　符号定义

模型中使用的符号和变量的含义如表 1 所示.

表1　符号说明

序号	定义
符号和参数	
f, f'	工厂索引号, $f \in F = \{1,2,\ldots,n_f\}$, F为工厂集合, n_f为工厂数目
i, i'	工件索引号, $i \in I = \{1,2,\ldots,n\}$, I为工件集合, n为工件数目
j, j'	工序索引号, $j \in J_i = \{1,2,\ldots,n_i\}$, J_i为工件i的工序集合, n_i为工件i工序数目
k	机器索引号, $k \in K_f = \{1,2,\ldots,m\}$, K_f为工厂f中的机器集合, m为工厂中的机器数目
p	机器位置索引号
G	极大正数
J'_i	工件i前n_i-1个工序的集合, $J'_i = \{1,2,\ldots,n_i-1\}$
$O_{i,j}$	工件i的第j个工序
$K_{i,j,f}$	工厂f中可以处理工序$O_{i,j}$的机器集合
$x_{f,i,j,k}$	若工厂f中的第k个机器可以处理$O_{i,j}$, 则为1; 否则为0
$p_{f,k}$	工厂f中的第k个机器上的位置数目
$P_{f,k}$	工厂f中的第k个机器上的位置集合, $P_{f,k} = \{1,2,\ldots,p_{f,k}\}$
$P'_{f,k}$	工厂f中的第k个机器上的前$p_{f,k}-1$个位置的集合, $P_{f,k} = \{1,2,\ldots,p_{f,k}-1\}$
$pt_{f,i,j,k}$	工序$O_{i,j}$在工厂f中的第k个机器上的处理时间
$C_{i,j}$	工序$O_{i,j}$的结束时间
$F_{f,k,p}$	工厂f中的第k个机器上第p个位置上工序的结束加工时间
P^O/P^I	机器的加工/空转能耗
C_f	工厂f的启动成本
D_i	工件i的交付时间
决策变量	
E_f	若启用工厂f, 则为1; 否则为0
$Z_{f,i}$	若工件i分配至工厂f中加工, 则为1; 否则为0
$X_{f,i,j,k,p}$	若工序$O_{i,j}$分配至工厂f中第k个机器上的第p个位置上, 则为1; 否则为0
$B_{i,j}$	工序$O_{i,j}$的开始时间
$S_{f,k,p}$	工厂f中的第k个机器上第p个位置上工序的开始加工时间

1.2　问题描述

一个企业经营n_f个工厂, 每个工厂f均包含由m台机器组成的同构化柔性作业车间. 有一个包含n个工件的订单需要加工, 每个工件i包含n_i道工序, 部分工序可在多台机器上加工, 但是加工时间与所选机器相关. 此外, 每个工厂f均伴随着固定启动成本C_f, 而每个工件i均有预定的交付时间, 提前或延迟交付会导致相应的罚款.

对于该问题, 考虑如下假设:

1) 工厂间是同构的, 每个工厂内部机器数目与加工能力一致;

2) 所有工件和机器在零时刻均已就位;

3) 每个机器同一时刻只允许加工一道工序;

4) 加工过程不可中断;

5) 同一工件内部工序间存在先后约束关系, 而不同工件的工序则相互独立且优先级相同.

为了更直观地理解该问题, 一个包含 5 个工件, 3 个工厂的 JITDFJSP 案例如图 1 所示. 其中: 工厂 1 和工厂 3 被启用, 由此形成了 DFJSP 问题; 然后, 工件 1、工件 3 和工件 4 被分配至工厂 1 中, 其余工件被分配至工厂 3 中; 最后, 对每个工厂内部的加工顺序和加工机器进行分配. 由此可见, 该问题包含 4 项决策内容: 选择启用的工厂、将作业分配至已启用的工厂、为各工序安排适当的机器和对所有工序进行排序.

JITDFJSP 对应的数学模型如下所示:

$$\min\ f_1 = C_{\max} = \max\{C_{i,n_i}\},\ \forall i \in I; \tag{1}$$

$$\begin{aligned}\min\ f_2 = \mathrm{TEC} = {} & P^O \cdot \sum_{f\in F}\sum_{i\in I}\sum_{j\in J_i}\sum_{k\in K_{i,j}}\sum_{t\in P_k} pt_{f,i,j,k}\cdot X_{f,i,j,k,t} + \\ & P^I \cdot \sum_{f\in F}\sum_{k\in K}\sum_{t\in P'_k}(S_{f,k,t+1} - F_{f,k,t});\end{aligned} \tag{2}$$

$$\begin{aligned}\min\ f_3 = \mathrm{Cost} = {} & E_f \cdot C_f + \\ & 2\cdot\sum_{i\in I}\max\{0, D_i - C_{i,n_i}\} + \\ & 4\cdot\sum_{i\in I}\max\{0, C_{i,n_i} - D_i\}.\end{aligned} \tag{3}$$

约束条件如下所示:

$$\sum_{f\in F} E_f \geqslant 1; \tag{4}$$

$$\sum_{f\in F} Z_{f,i} = 1,\ \forall i \in I; \tag{5}$$

$$Z_{f,i} \leqslant E_f,\ \forall f \in F,\ i \in I; \tag{6}$$

$$\begin{aligned}&\sum_{k\in K_{i,j,f}}\sum_{p\in P_{f,k}} X_{f,i,j,k,p} = Z_{f,i},\ \forall f \in F,\\ &\qquad i \in I,\ j \in J_i;\end{aligned} \tag{7}$$

$$\begin{aligned}&\sum_{i\in I}\sum_{j\in J_i} X_{f,i,j,k,p} \leqslant 1,\ \forall f \in F,\\ &\qquad k \in K_f,\ p \in P_{f,k};\end{aligned} \tag{8}$$

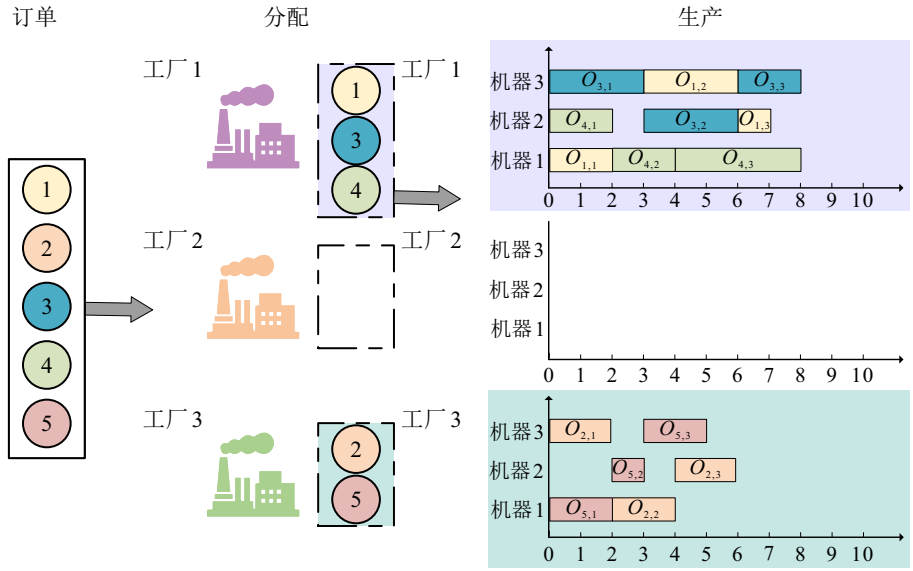


图1 JITDFJSP 示意图

$$\sum_{i'\in I}\sum_{j'\in J_{i'}} X_{f,i',j',k,p+1} \leqslant \sum_{i\in I}\sum_{j\in J_i} X_{f,i,j,k,p},$$
$$\forall f\in F,\ k\in K_f,\ p\in P_{f,k}; \tag{9}$$

$$B_{i,j}+\sum_{f\in F}\sum_{k\in K_{i,j,f}}\sum_{p\in P_{f,k}}(X_{f,i,j,k,p}\cdot pt_{f,i,j,k}) \leqslant B_{i,j+1},$$
$$\forall i\in I,\ j\in J'_i; \tag{10}$$

$$S_{f,k,p}+\sum_{i\in I}\sum_{j\in J_i}(X_{f,i,j,k,p}\cdot pt_{f,i,j,k}) \leqslant S_{f,k,p+1},$$
$$\forall f\in F,\ k\in K_f,\ p\in P'_{f,k}; \tag{11}$$

$$B_{i,j}-G\cdot(1-X_{f,i,j,k,p}) \leqslant S_{f,k,p},\ \forall f\in F,$$
$$i\in I,\ j\in J_i,\ k\in K_f,\ p\in P_{f,k}; \tag{12}$$

$$B_{i,j}+G\cdot(1-X_{f,i,j,k,p}) \geqslant S_{f,k,p},\ \forall f\in F,$$
$$i\in I,\ j\in J_i,\ k\in K_f,\ p\in P_{f,k}; \tag{13}$$

$$B_{i,j}+\sum_{f\in F}\sum_{k\in K_{i,j,f}}\sum_{p\in P_{f,k}}(X_{f,i,j,k,p}\cdot pt_{f,i,j,k}) \leqslant C_{i,j},$$
$$\forall i\in I,\ j\in J_i; \tag{14}$$

$$B_{i,j}\geqslant 0,\ \forall i\in I,\ j\in J_i; \tag{15}$$

$$S_{f,k,p}\geqslant 0,\ \forall f\in F,\ k\in K_f,\ p\in P_{f,k}. \tag{16}$$

其中: 式 (1) ~ (3) 分别为本文的 3 个目标: 完工时间、生产能耗和生产成本, 这里生产成本包含工厂启动成本和因提前/延迟交付导致的罚款. 随后, 约束 (4) ~ (16) 进一步刻画了该问题: 约束 (4) 表明至少需要启用一个工厂用于交付订单; 约束 (5) 和 (6) 限定每个工件只能分配到已经启用的工厂; 约束 (7) 表明, 一旦工件分配至对应的工厂, 其包含的所有工序均需要在该工厂中完成; 约束 (8) 要求机器上每个位置至多加工一道工序; 约束 (9) 确保工序在机器上尽可能早地加工; 约束 (10) 强调, 对于同一工件, 后续工序的开始时间不得早于前置工序的完成时间; 约束 (11) 要求, 对于同一机器, 后续位置的开始时间不早于其之前位置的完成时间; 约束 (12) 和 (13) 建立了变量$B_{i,j}$与$S_{f,k,p}$间的联系; 约束 (14) 表明工件的完成时间不早于其开始时间与加工时间的和; 约束 (15) 和 (16) 规定决策变量$B_{i,j}$和$S_{f,k,p}$的取值必须非负.

1.3 模型验证

如表 2 所示: 本节构建了包含 3 个工厂和 7 个工件, 每个工厂配备 3 台机器以及每个工件包含 4 道工序的测试案例来验证模型的正确性. 使用 CPLEX 求解器对 3 个目标进行分别求解, 设定运行时间为 3 600 s, 并配置 8 个线程进行并行计算. 最终输出的最优调度方案如图 2 所示.

由表 2 和图 2 可见, 模型能够精确地刻画 JITDFJSP 所表征的问题, 表明其正确性. 此外, 3 个目标间存在冲突, 优化其中一个目标往往会导致其他目标性能下降. 如最低能耗的调度方案在完工时间和生产成本上表现最差, 而最低生产成本的方案则对应较高的能量消耗.

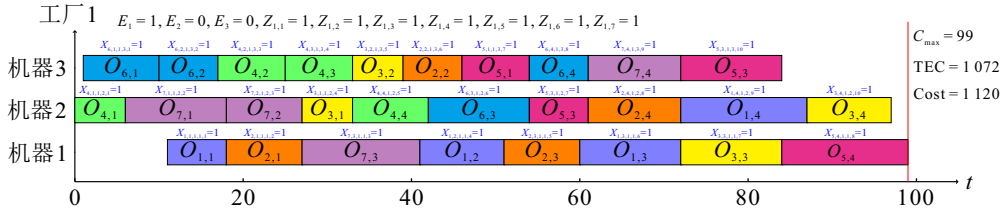
2 Q学习驱动进化的模因算法 (QMA)

2.1 编码与解码

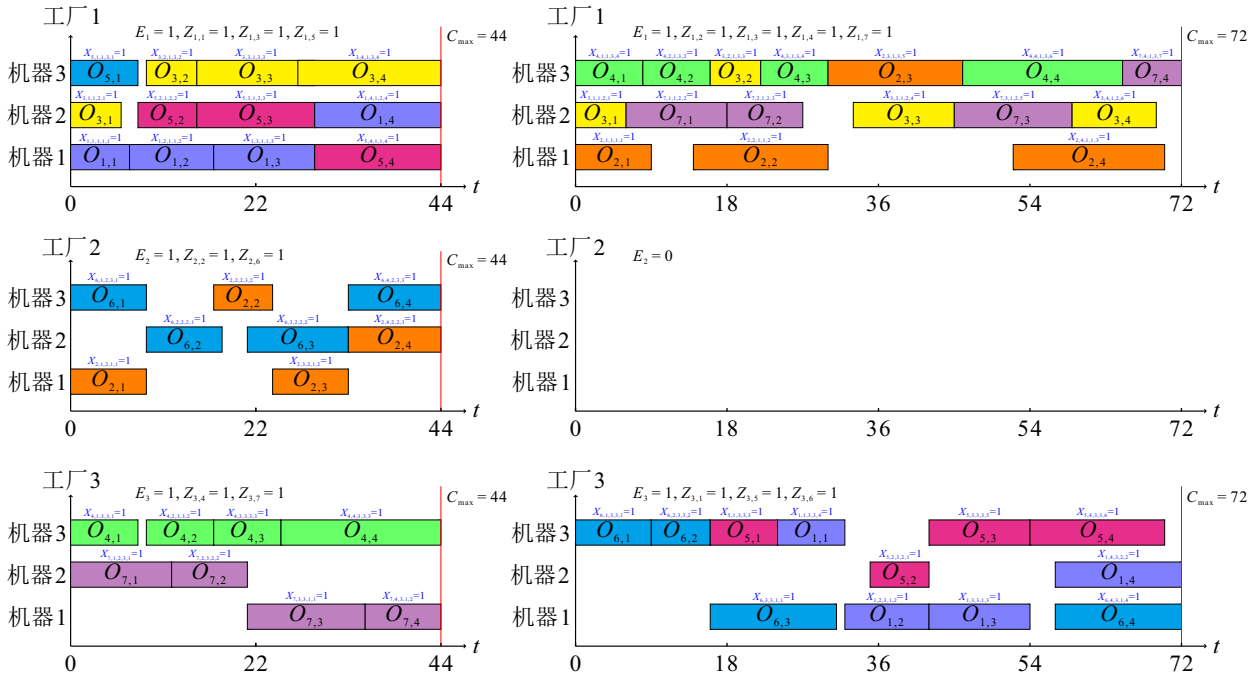
考虑到 JITDFJSP 的决策需求, 染色体包含 4 层编码信息, 即工厂启用层 (FS)、工厂分配层 (FA)、机器选择层 (MA) 和工序排序层 (OS). 其中: FS 层的长度等于工厂的数目, 其采用 0-1 编码决定是否启用对应的工厂; FA 层的长度与工件的数量相同且采用连续编码技术. 图 3 为染色体编码. 图 3 中: 启用了两个工厂 (f_1和f_3) 来加工 4 个工件, FA 层中对应取值小于 0.5 的工件分配至第 1 个启用的工厂 (f_1);

表2　案例详情

工件	工序	机器1	机器2	机器3	交付时间	工件	工序	机器1	机器2	机器3	交付时间
1	$O_{1,1}$	7	9	8	72	2	$O_{2,1}$	9	16	10	70
	$O_{1,2}$	10	12	13			$O_{2,2}$	16	16	7	
	$O_{1,3}$	12	19	13			$O_{2,3}$	9	9	16	
	$O_{1,4}$	20	15	20			$O_{2,4}$	18	11	17	
3	$O_{3,1}$	16	6	15	69	4	$O_{4,1}$	13	6	8	65
	$O_{3,2}$	11	19	6			$O_{4,2}$	16	13	8	
	$O_{3,3}$	12	12	12			$O_{4,3}$	11	15	8	
	$O_{3,4}$	17	10	17			$O_{4,4}$	17	9	19	
5	$O_{5,1}$	10	17	8	70	6	$O_{6,1}$	15	20	9	72
	$O_{5,2}$	10	7	14			$O_{6,2}$	16	9	7	
	$O_{5,3}$	16	14	12			$O_{6,3}$	15	12	12	
	$O_{5,4}$	15	15	16			$O_{6,4}$	15	17	11	
7	$O_{7,1}$	15	12	18	72	7	$O_{7,3}$	14	14	19	
	$O_{7,2}$	18	9	15			$O_{7,4}$	9	10	7	



(a)　最低能量消耗(f_2)对应的调度方案



(b)　最短完工时间(f_1)对应的调度方案

$C_{\max}=44$, TEC=1 245, Cost=1 264

(c)　最低生产成本(f_3)对应的调度方案

$C_{\max}=72$, TEC=1 310, Cost=600

图2　CPLEX 提供的最优调度方案

反之, 则分配至第 2 个启用的工厂 (f_3). MA 层和 OS 层的长度与总工序的数目相同, 前者决定了每道工序加工的机器, 而后者则确定了工序间的先后顺序. 通过这种编码方式可以清晰地反映出决策变量 $X_{f,i,j,k,t}$ 的取值. 如工序 $O_{4,1}$ 在工厂 3 中的第 1 台机器上的第 1 个位置上加工, 即 $X_{3,4,1,1,1}=1$. 同理, 可得到 $X_{3,2,1,2,1}=1$, $X_{3,4,2,2,2}=1$.

解码时, 首先根据 FS 编码层确定启用的工厂,

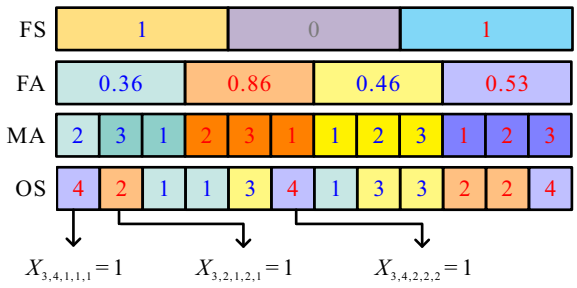


图3 染色体编码

然后按照 FA 编码层的信息将工件划分至对应的工厂. 图 3 显示, 启用工厂 1 和工厂 3 加工 4 个工件. 其中: 工件 2 和工件 4 在工厂 3 中加工, 且加工顺序为$O_{4,1}(M_1) \to O_{2,1}(M_2) \to O_{4,2}(M_2) \to O_{2,2}(M_3) \to O_{2,3}(M_1) \to O_{4,3}(M_3)$. 此外, 解码过程要兼顾工件内部约束和机器加工先后约束, 如: $O_{2,1}$和$O_{4,2}$先后在机器 2 上加工, 则按照式 (10) 和 (11), $O_{4,2}$的开始时间不早于$O_{4,1}$(工件前置工序) 与$O_{2,1}$(机器前置工序) 的完成时间. 对于机器上的第 1 个操作和工件的第 1 道工序, 由式 (15) 和 (16), 只需要满足开始时间大于等于 0 即可.

2.2 种群初始化

高质量的初始群体是决定算法搜索性能的关键. 为提升初始种群的质量并保证其多样性, 本文采取混合启发式方法[28] 对其进行初始化. 具体而言, FS、FA 和 OS 编码层随机生成对应信息; MA 编码层则结合全局选择、局部选择和随机选择确定对应工序的机器. 其中: 全局选择和局部选择分别作用于 40% 的个体, 它们通过优化机器负载来提高染色体质量, 随机选择操作则保证了解的多样性.

2.3 强化学习驱动的进化过程

QMA 算法的进化操作主要包含选择、交叉和变异 3 部分. 选择操作借助二元锦标赛机制, 确保质量高且多样性强的父本加入交配池中. 交叉操作在交配池中随机成对选取父本, 对信息进行组合以产生后代. 这种方式虽然简单, 但是, 在一定程度上增加了搜索盲目性, 降低了子代质量. 为此, 本文提出了Q学习驱动的父本匹配策略, 设计了对应的状态、动作和奖励. Q学习通过参与进化过程, 根据不同状态下父本组合的子代性能, 自适应调整匹配策略, 提升搜索效率.

状态 (state): 根据个体在 3 个目标上的适应值表现, 将所有父本划分至 8 个区域 (对应 8 种状态). 每种状态的定义方式如下所示, 其中M_{f_i}为第i个目标适应值的中位数.

状态/动作	定义
1	$f_1 \leqslant M_{f_1}, f_2 \leqslant M_{f_2}, f_3 \leqslant M_{f_3}$
2	$f_1 \leqslant M_{f_1}, f_2 \leqslant M_{f_2}, f_3 > M_{f_3}$
3	$f_1 \leqslant M_{f_1}, f_2 > M_{f_2}, f_3 \leqslant M_{f_3}$
4	$f_1 \leqslant M_{f_1}, f_2 > M_{f_2}, f_3 > M_{f_3}$
5	$f_1 > M_{f_1}, f_2 \leqslant M_{f_2}, f_3 \leqslant M_{f_3}$
6	$f_1 > M_{f_1}, f_2 \leqslant M_{f_2}, f_3 > M_{f_3}$
7	$f_1 > M_{f_1}, f_2 > M_{f_2}, f_3 \leqslant M_{f_3}$
8	$f_1 > M_{f_1}, f_2 > M_{f_2}, f_3 > M_{f_3}$

动作 (action): 在 QMA 算法中, Q学习的动作定义与状态相同, 即每个父本可与 8 个区域内的个体进行交叉操作. 每个父本 (P_1) 先确定自身状态 (s_t), 根据Q表选取最优匹配区域 (即动作a_t), 然后在该区域内随机挑选一个个体 (P_2) 进行交叉操作. 具体而言, 首先从$0 \sim 1$之间生成一个随机数r, 若其不大于Q学习中的贪心因子ε(即$r \leqslant \varepsilon$), 则选取在该状态下价值最高的动作; 否则, 任意选取一个动作.

确定父本 ($P_{1,2}$) 后, 进行交叉操作. FS 层、FA 层和 MA 层采用均匀交叉操作. 首先, 生成与 FS 层、FA 层和 MA 层长度一致的随机 0-1 数组. 根据数组取值, 将位置为 1/0 的信息从P_1/P_2传递给子代C_1; 然后, 将位置为 1/0 的信息从P_2/P_1传递给子代C_2. OS 层则采用 POX 交叉操作. 首先, 将工件序号划分为不相交的两个非空子集 ($\mathbb{J}_1, \mathbb{J}_2$); 然后, 将$P_1/P_2$中属于$\mathbb{J}_1/\mathbb{J}_2$的元素按照原位置复制到$C_1/C_2$中; 最后, 将$P_2/P_1$中属于$\mathbb{J}_2/\mathbb{J}_1$的元素按照$P_2/P_1$中的顺序填充至$C_1/C_2$中.

交叉操作执行完毕后, 对$C_{1,2}$执行变异操作, 以增强全局搜索性能. FS、FA 和 MA 编码层采用多点变异操作, OS 编码层则采取两点对换操作, 即随机选取两道不同工件的工序交换其顺序.

奖励 (reward): 父本 1 (P_1) 与父本 2 (P_2) 经过交叉、变异后, 产生两个子代 ($C_{1,2}$). QMA 根据子代的质量获取奖励 (r), 具体定义如下所示:

$$r = \begin{cases} 10, \text{ 至少有一个子代}(C_{1,2}) \\ \quad \text{支配父本}P_1\text{或}P_2; \\ 5, \text{ 至少有一个子代}(C_{1,2})\text{不被} \\ \quad \text{父本}P_1\text{或}P_2\text{支配}; \\ -5, \text{ 子代}(C_{1,2})\text{同时被}P_1\text{和}P_2\text{支配}. \end{cases}$$

随后, 对Q表进行如下更新:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \cdot (r + \gamma \cdot Q(s_{t+1}, a^*_{t+1}) - Q(s_t, a_t)). \tag{17}$$

其中: α为学习率, β为折扣因子, a^*_{t+1}为在s_{t+1}状态

下的最优行动.

2.4　自适应局部搜索

单纯依赖交叉和变异操作会限制算法的局部搜索性能. 因此, 在搜索的后期 (本文采用 CPU 时间作为终止条件, 后期定义为 CPU 运行时间超过总时间一半的阶段), 若算法处于进化停滞状态 (Pareto 前沿继承上一代的比例超过Δ), 则激活自适应局部搜索操作, 为所有个体随机作用一个局部算子 ($\mathcal{L}_{1\sim4}$) 以进一步提升搜索性能.

$\mathcal{L}_1$(针对 FA 编码层): 从因早交/拖期导致罚款最高的工厂中随机选取一个工件, 将其转移到罚款最低的工厂中加工.

$\mathcal{L}_2$(针对 MA 编码层): 从关键工厂 (完工时间最大的工厂) 的关键路径中, 随机选取一道工序, 更改其加工的机器.

$\mathcal{L}_3$(针对 OS 编码层): 从关键工厂中随机选取两道工序, 并将后序工序插入至前者之前.

$\mathcal{L}_4$(针对 OS 编码层): 搜索算子$\mathcal{L}_4$在关键工厂中采取邻域结构N_6的操作机制. 具体而言, 对于第 1 个关键路径块, 移动其内部工序至块尾; 对于最后一个关键路径块, 移动其内部工序至块首; 对于其他关键路径块, 随机移动其工序至块首或块尾.

2.5　算法步骤

QMA 算法是一种改进的模因算法, 其主要特点是结合了启发式初始化策略、Q学习驱动的父本匹配策略以及自适应局部搜索. 算法具体流程如下.

step 1: 使用启发式初始化方法生成 PS 个个体, 并计算在 3 个目标上的适应值, 设定 Pareto 前沿最优解为空集.

step 2: 通过二元锦标赛方式, 挑选出用于执行进化操作的父本并填充交配池.

step 3: 对于交配池中的每个父本 (P_1), 通过Q学习选取合适的搭档 (P_2); 然后对P_1和P_2执行交叉和变异操作, 生成子代C_1和C_2.

step 4: 根据子代 ($C_{1,2}$) 相对于父本 ($P_{1,2}$) 的质量, 确定奖励r并更新Q表.

step 5: 在搜索的后期, 若种群处于进化停滞阶段, 则启动自适应局部搜索策略.

step 6: 将遗传操作和自适应局部搜索产生的个体与原种群合并, 根据适应值剔除重复的个体.

step 7: 计算每个个体的非支配等级以及拥挤度距离[29], 筛选等级靠前的 PS 个个体进入下一代. 优先选取等级较低的个体, 若等级相同, 则选择拥挤度较大的个体以保持多样性.

step 8: 将等级为 1 的个体与 Pareto 前沿最优解合并, 对 Pareto 前沿最优解进行更新. 若满足停止条件, 则算法结束; 否则, 转至 step 2 继续搜索.

2.6　算法最优性及时间复杂度分析

QMA 算法是遗传算法与局部搜索策略相结合的混合算法, 其采用快速非支配排序和拥挤度距离筛选表现优异的个体进入下一代. 若第t代的非支配解数目少于种群规模 PS, 则非支配个体会全部保留至$t+1$代; 反之, 则每个目标上表现最好的边界解对应的拥挤度距离最大, 这些解会保留至$t+1$代. 由此可知, QMA 算法沿用的是基于最佳个体保留的进化机制. 文献 [30-31] 证明, 具备最优个体保留机制的遗传算法一定收敛至全局最优解. 此外, QMA 算法没有对解空间进行切割, 其搜索空间包含最优方案. 因此, QMA 算法能够确保在有限次迭代下收敛至全局最优解[31].

令种群规模为 PS, 算法迭代次数为 Iter. QMA 算法在进化过程中主要执行 4 部分操作: 1) 混合初始化, 对应的时间复杂度为$O(\mathrm{PS})$; 2) 交叉和变异操作, 时间复杂度为$O(\mathrm{Iter}\cdot\mathrm{PS})$; 3) 自适应局部搜索机制只在后期执行, 复杂度为$O\left(\dfrac{\mathrm{Iter}}{2}\cdot\mathrm{PS}\right)$; 4) 非支配排序的时间复杂度为$O(\mathrm{Iter}\cdot\mathrm{PS}^2)$. 由此可见, QMA 的时间复杂度主要取决于非支配排序, 即$O(\mathrm{Iter}\cdot\mathrm{PS}^2)$.

3　实验分析

3.1　实验设计

本节开展多重实验对 QMA 算法的有效性进行验证, 所有算法均采用 Matlab 语言编程, 并在 Intel i9-10900 CPU@3.4 GHz 和 16 GB RAM 的系统上运行.

实验案例包含两个 FJSP 标准数据集: 第 1 个是 Brandimarte 数据集[32], 其包含 10 个案例 ($\mathrm{MK}_1\sim\mathrm{MK}_{10}$); 第 2 个则是包含 18 个案例 ($\mathrm{DP}_1\sim\mathrm{DP}_{18}$) 的 Dauzere-Peres 数据集[33]. 工件i的交付时间计算方式如下所示:

$$D_i=\left\lceil\left(1+\frac{0.3\cdot N}{M\cdot 3}\right)\cdot\sum_{j\in J_i}\bar{pt}_{f,i,j,k}\right\rceil,\tag{18}$$

其中$\lceil a\rceil$表示对a进行向上取整操作.

企业最多可同时启用 5 个工厂, 每个工厂的启动成本为 300. 机器的空转功率设定为 1, 加工功率为 4. 此外, 本文采用 CPU 运行时间作为算法的停止标准, 每个案例的运行时间为$0.3\cdot\sum_{i\in I}n_i$ s. 所有算法独立运行 30 次, 以确保对比的有效性.

本文采用总非支配向量数目 (overall nondominated vector generation, ONVG)[34]、反世代距离 (inverted generational distance, IGD)[35] 和超体积度量 (hypervolum, HV)[36] 3 种评价指标对算法性能进行评估.

ONVG 指标的定义方式如下所示:

$$\mathrm{ONVG}(\Omega) = |\Omega|. \tag{19}$$

其中: Ω为算法搜寻的非支配解集, $|\Omega|$为集合中元素的数目.

IGD 指标的计算方式为

$$\mathrm{IGD}(\Omega, \Psi) = \frac{\sum\limits_{x\in\Psi} \min\limits_{y\in\Omega} \mathrm{dis}(x, y)}{|\Psi|}. \tag{20}$$

其中: Ψ为最优 Pareto 前沿, $\mathrm{dis}(x, y)$为解x与y适应值间的距离. 可以看出, IGD 的值越小, 非支配解越接近最优 Pareto 前沿.

HV 指标的计算方式如下所示:

$$\mathrm{HV}(\Omega, P) = \bigcup_{x\in\Omega} \nu(x, P). \tag{21}$$

其中: P为预先设定的一组参考点; HV 的值为算法搜寻的非支配解的适应值与参考点间围成的超体积, HV 值越大, 非支配解的适应值越小, 质量越高, 分布越广阔.

3.2 参数敏感性分析

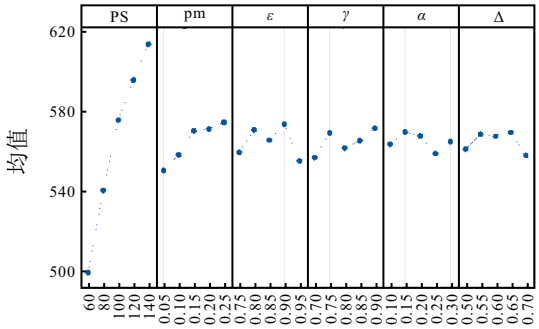
QMA 算法主要包含 6 个关键参数, 即种群大小 (PS), 变异概率 (pm), 贪心因子 (ε), 学习率 (α), 折扣因子 (γ) 和停滞阈值 (Δ). 本文通过田口试验[37] 确定参数的最佳取值, 每个参数的候选值如下所示:

$$\begin{aligned}
&\mathrm{PS} \in \{60, 80, 100, 120, 140\},\\
&\mathrm{pm} \in \{0.05, 0.10, 0.15, 0.20, 0.25\},\\
&\varepsilon \in \{0.75, 0.80, 0.85, 0.90, 0.95\},\\
&\gamma \in \{0.70, 0.75, 0.80, 0.85, 0.90\},\\
&\alpha \in \{0.10, 0.15, 0.20, 0.25, 0.30\},\\
&\Delta \in \{0.50, 0.55, 0.60, 0.65, 0.70\}.
\end{aligned}$$

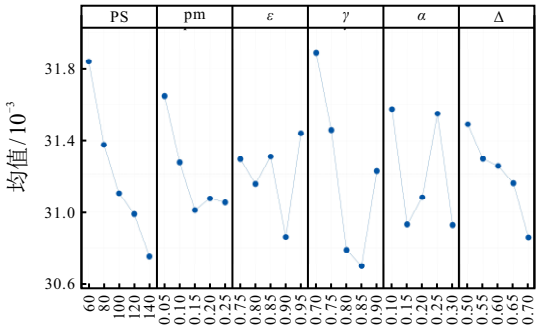
通过L25(5^6) 正交表进行参数敏感性分析, 最终 3 种评价指标下的主效应图如图 4 所示. 对于 IGD 指标, 值越小, 搜寻的非支配解集与最优 Pareto 前沿越近. 反之, ONVG 值越大, 则非支配解个数越多, HV 值越大, 非支配解集质量越高. 基于上述考虑, QMA 算法的参数设定如下: PS = 140, pm = 0.25, ε = 0.90, γ = 0.85, α = 0.15, Δ = 0.65.

3.3 对比 CPLEX

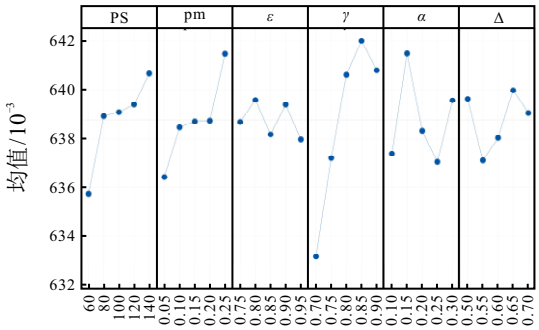
为验证所提出算法在小规模算例上的精度, 本节生成了 5 个案例, 分别利用 QMA 算法和 CPLEX 求解器来完成求解. 每个案例以FaJbMc命名, 表示



(a) ONVG 值主效应



(b) IGD 值主效应



(c) HV 值主效应

图4 不同参数在 3 种评价指标下的主效应图

表3 QMA 与 CPLEX 的对比结果

算例	CPLEX			QMA		
	$C_{\max}$	TEC	Cost	$C_{\max}$	TEC	Cost
F3J5M3	45*	768*	302	45	772	312
F3J7M3	44*	1072*	600	44	1072	558
F5J10M3	50	1468	1596	46	1448	634
F5J10M4	54	2428	2052	46	1640	642
F5J15M3	488	3108	9468	54	2248	686

有a个工厂, b个工件, 每个工厂配备c台机器. CPLEX 求解每个目标的时间为 3600 s, 并配备 8 个线程计算. QMA 算法同时优化 3 个目标, 并取非支配解在 3 个目标上的最优值与 CPLEX 进行对比. 最终结果如表 3 所示, 其中*表示对应的结果为 CPLEX 提供的最优解.

由表 3 可见, CPLEX 可获取部分算例上的最优解, 随着问题规模的提升, 其求解性能急剧下降, 只能提供可行解, 无法实现进一步的优化和提升. 在案例 F3J5M3 上, QMA 提供了$C_{\max}$目标上的全局最优解, 且在其他两个目标上与最优解的差距非常小; 在案例 F3J7M3 上, QMA 算法在$C_{\max}$和 TEC 上提供了全局最优解, 且在 Cost 上提供的解优于 CPLEX; 在其他小规模案例上, QMA 的性能均优于 CPLEX, 显示出其在小规模案例上的有效性.

3.4　消融实验

QMA 算法的核心组件包含启发式初始化、Q学习驱动的父本匹配以及自适应局部搜索. 本节开发了下述 QMA 变体以检验不同组件的有效性:

1) QMA_1采用随机方法得到初始种群;

2) QMA_2沿用传统的随机策略进行父本匹配;

3) QMA_3舍弃自适应局部搜索机制;

4) QMA_4鼓励处于相同区域的父本间进行匹配;

5) QMA_5支持来自不同区域的父本间进行匹配.

算法$\text{QMA}_1 \sim \text{QMA}_5$与 QMA 的参数设置完全一致, 表 4 为显著性水平 0.05 条件下的 Friedman 秩和检验[38] 结果.

表4　消融实验的 Friedman 秩和检验结果

算法	ONVG		IGD		HV	
	排名	p值	排名	p值	排名	p值
QMA_1	5.46	1.43e-21	5.18	1.51e-19	5.54	1.24e-20
QMA_2	2.96		3.46		3.46	
QMA_3	5.39		5.29		5.32	
QMA_4	**1.25**		3.57		2.00	
QMA_5	2.79		1.96		2.82	
QMA	3.14		**1.54**		**1.86**	

由表 4 可见, QMA_4在 ONVG 指标上表现最好, 表明其提供的非支配解数目较多, 但是其在另外两个指标上的性能低于 QMA, 表明后者提供的非支配解集质量要优于QMA_4. 此外, QMA_4与QMA_5的 HV 指标性能均优于QMA_2, 这表明父本 (P_1) 无论是单独与同一区域还是不同区域的个体 (P_2) 协作, 均可提升进化效率. 而 QMA 的算法排名略优于QMA_4与QMA_5, 这表示Q学习可在两种匹配模式下获取较好的平衡. 此外, $\text{QMA}_1 \sim \text{QMA}_5$在 IGD 与 HV 指标上的排名均低于 QMA, 这表明 QMA 搜寻的非支配解集性能更好, 更接近真实的 Pareto 前沿. 3 种指标的p值均远低于显著性水平 0.05, 证实了各算法产出的结果间存在显著差异. 由此可知, QMA 中包含的核心组件均能够有效地提升算法性能, 而当它们有机结合时, 则能够发挥出最大的协同效应.

3.5　对比实验

为进一步验证 QMA 的有效性, 本节将其与当前主流求解方法进行对比, 包括 NSGA-II[29]、MOEA/D[39], PeEA[40]、LRVMA[21] 和 SPAMA[15].

NSGA-II 和 MOEA/D 是经典的多目标优化算法, 它们分别通过 Pareto 支配和问题分解的思想驱动种群进化; PeEA 是近年来提出的算法, 其通过估计 Pareto 前沿曲率、自适应标量化函数评估并结合相似性度量等方式, 确保非支配解的最优性和多样性; LRVMA 沿用了 MOEA/D 的进化框架, 其利用Q学习自适应调整算法参数, 此外, 它还嵌入了知识驱动的变邻域搜索行为来提升解的质量; SPAMA 是专门用于求解同类型调度问题的算法, 其采用全主动解码机制降低能耗的同时设计了基于自适应的算子选择模型, 提升了局部搜索的成功率.

为保证对比的公平性, 首先, 通过田口试验为对比算法进行参数配置. 所有算法在不同的参数组合下单独运行 30 次. 最终确定每种算法的最优参数, 如表 5 所示.

表5　各种算法的最优参数

算法	参数	复杂度
NSGA-II	种群大小PS = 140, 交叉概率pc = 0.6, 变异概率pm = 0.15	$O(\text{Iter} \cdot \text{PS}^2)$
MOEA/D	种群大小PS = 140, 交叉概率pc = 0.6, 变异概率pm = 0.15, 邻域大小15	$O(\text{Iter} \cdot \text{PS} + \text{PS} \cdot T)$
PeEA	种群大小PS = 60, 交叉概率pc = 1, 变异概率pm = 0.10	$O(\text{Iter} \cdot (\text{PS}^2 \cdot \log \text{PS}))$
LRVMA	种群大小PS = 140, 贪心因子$\varepsilon = 0.95$, 折扣因子$\gamma = 0.5$, 学习率$\alpha = 0.3$	$O(\text{Iter} \cdot (\text{PS} + \text{PS} \cdot T))$
SPAMA	种群大小PS = 120, 历史记忆长度HL = 25, 奖励步长$S = 0.05$, 交叉概率pc = 0.9, 变异概率pm = 0.25.	$O(\text{Iter} \cdot \text{PS}^2)$

表 6 ～ 表 8 给出了算法在 3 种评价指标下的对比结果. 其中: Wilcoxon 秩和检验[38] 用于评估 QMA 与对比算法结果间的统计差异; 符号“+”“ = ”和“–”分别表示 QMA 相对于对比算法的表现为优于、等于、劣于; 表格最后 3 行分别统计了在所有算例中 QMA 算法与对比算法相比的胜、平、负次数. 随后, 表 9 和表 10 分别展示了 Fridman 秩和检验和 Mann-Whitney U 检验结果.

表6 算法关于 ONVG 指标的对比结果

算例	NSGA-II		MOEA/D		PeEA		LRVMA		SPAMA		QMA	
	均值	标准差	均值	标准差	均值	标准差	均值	标准差	均值	标准差	均值	标准差
MK_1	**93.37(=)**	13.74	47.3(+)	11.74	18.9(+)	6.50	30.63(+)	8.82	49.47(+)	19.62	89.1	19.16
MK_2	124.27(=)	25.69	62.03(+)	15.00	23.2(+)	11.84	46.93(+)	9.63	68.27(+)	18.61	**128.87**	17.61
MK_3	431.17(=)	74.68	223.17(+)	82.44	19.33(+)	9.54	74.7(+)	25.78	155.8(+)	40.88	**469.7**	63.20
MK_4	129.23(=)	34.22	47.5(+)	14.16	19.83(+)	8.80	33.33(+)	13.36	43.47(+)	10.43	**131.57**	35.14
MK_5	**157.67(=)**	47.65	42.17(+)	14.47	17.27(+)	6.73	23.73(+)	8.03	47.13(+)	9.24	146.43	36.32
MK_6	426.13(+)	70.39	189.97(+)	40.17	26.37(+)	27.81	131.57(+)	32.77	181.3(+)	27.32	**633.03**	45.97
MK_7	286.53(+)	60.77	140.4(+)	56.91	17.37(+)	11.42	62.07(+)	25.77	129.77(+)	19.89	**375.53**	55.19
MK_8	**186.17(=)**	84.73	59.8(+)	26.58	16.93(+)	4.09	23.6(+)	9.98	29.73(+)	7.34	162.5	46.20
MK_9	**297.17(=)**	90.77	82.57(+)	32.35	18(+)	6.02	24.33(+)	8.04	50.33(+)	15.17	260.4	51.80
MK_{10}	328.8(=)	81.26	90.33(+)	38.21	17.8(+)	5.12	29.97(+)	10.10	44.13(+)	13.01	**349.17**	49.61
DP_1	**436.27(−)**	148.46	82.77(+)	35.80	20.37(+)	5.94	45.6(+)	16.90	95.27(+)	33.65	363.87	93.74
DP_2	456.43(=)	142.36	127.17(+)	52.17	21.43(+)	6.33	50.6(+)	19.27	102.5(+)	32.37	**534.77**	190.74
DP_3	418.37(+)	159.91	115.2(+)	41.14	21.37(+)	6.49	48.1(+)	22.59	110.73(+)	31.36	**532.83**	124.28
DP_4	**618.27(=)**	214.15	108.83(+)	43.06	20.9(+)	5.86	51.17(+)	15.55	110.1(+)	44.25	577.97	111.45
DP_5	587.4(+)	202.93	142.43(+)	47.65	24.03(+)	7.14	54.27(+)	21.19	108.3(+)	38.86	**799.27**	211.08
DP_6	535.43(+)	175.01	131.57(+)	54.94	19.8(+)	6.38	57(+)	18.05	139.1(+)	31.15	**702.47**	161.18
DP_7	212.37(=)	132.27	77.3(+)	36.29	19.9(+)	4.82	32.07(+)	9.78	38.37(+)	7.80	**238.13**	91.08
DP_8	351.07(=)	143.02	94.57(+)	40.27	18.43(+)	6.62	37.37(+)	12.87	53.73(+)	14.16	**392.43**	177.32
DP_9	348.57(=)	155.47	80.8(+)	33.18	17.97(+)	4.56	37.67(+)	11.71	56.3(+)	12.59	**390.17**	171.72
DP_{10}	271.53(+)	147.86	101.5(+)	32.54	19.63(+)	4.92	34.93(+)	11.06	46.87(+)	12.44	**338.93**	115.80
DP_{11}	520.8(=)	232.33	96.07(+)	50.07	17.37(+)	4.84	35.27(+)	15.92	86.3(+)	18.02	**592.2**	181.83
DP_{12}	415.8(+)	135.56	97.37(+)	38.68	18.3(+)	6.20	41.97(+)	13.21	117.6(+)	41.75	**661.7**	132.33
DP_{13}	**246.03(=)**	99.51	70.4(+)	20.49	21.7(+)	8.98	34.17(+)	13.70	38.6(+)	10.89	205.9	72.53
DP_{14}	274.13(+)	149.25	68.03(+)	32.68	20.2(+)	4.83	33.7(+)	12.39	48.8(+)	13.82	**355.93**	124.03
DP_{15}	257.3(=)	135.36	68.1(+)	30.34	20.8(+)	6.04	38.37(+)	16.41	57.43(+)	16.19	**305.27**	162.58
DP_{16}	**323.5(−)**	158.89	89.1(+)	40.43	19.93(+)	5.77	32.97(+)	13.52	48.1(+)	14.45	237.3	88.02
DP_{17}	356.17(+)	152.10	77.07(+)	41.45	18.8(+)	5.87	33.67(+)	12.72	82.37(+)	22.15	**503.73**	152.41
DP_{18}	407.2(+)	233.31	85.23(+)	40.86	20(+)	3.70	39.03(+)	13.03	95.93(+)	34.21	**526**	142.78
胜	10		28		28		28		28			
平	16		0		0		0		0			
负	2		0		0		0		0			

表7 算法关于 IGD 指标的对比结果

算例	NSGA-II		MOEA/D		PeEA		LRVMA		SPAMA		QMA	
	均值	标准差	均值	标准差	均值	标准差	均值	标准差	均值	标准差	均值	标准差
MK_1	**0.07(−)**	*0.03*	0.12(+)	*0.02*	0.2(+)	*0.03*	0.14(+)	*0.03*	0.11(+)	*0.02*	0.08	*0.01*
MK_2	0.04(+)	*0.01*	0.14(+)	*0.03*	0.29(+)	*0.13*	0.1(+)	*0.02*	0.09(+)	*0.02*	**0.03**	*0.01*
MK_3	0.06(+)	*0.02*	0.11(+)	*0.03*	0.35(+)	*0.1*	0.1(+)	*0.02*	0.13(+)	*0.03*	**0.03**	*0.01*
MK_4	**0.06(−)**	*0.01*	0.13(+)	*0.03*	0.22(+)	*0.08*	0.15(+)	*0.04*	0.1(+)	*0.02*	0.07	*0.01*
MK_5	0.07(+)	*0.02*	0.13(+)	*0.02*	0.21(+)	*0.06*	0.12(+)	*0.02*	0.11(+)	*0.03*	**0.04**	*0.01*
MK_6	0.08(+)	*0.02*	0.2(+)	*0.04*	0.47(+)	*0.11*	0.08(+)	*0.01*	0.21(+)	*0.04*	**0.02**	*0*
MK_7	0.06(+)	*0.02*	0.13(+)	*0.03*	0.3(+)	*0.12*	0.09(+)	*0.02*	0.11(+)	*0.03*	**0.02**	*0*
MK_8	0.07(+)	*0.02*	0.09(+)	*0.02*	0.3(+)	*0.11*	0.1(+)	*0.02*	0.13(+)	*0.03*	**0.04**	*0.01*
MK_9	0.11(+)	*0.03*	0.12(+)	*0.03*	0.41(+)	*0.11*	0.07(+)	*0.01*	0.23(+)	*0.04*	**0.02**	*0*
MK_{10}	0.14(+)	*0.02*	0.14(+)	*0.04*	0.48(+)	*0.11*	0.07(+)	*0.01*	0.27(+)	*0.05*	**0.02**	*0*
DP_1	**0.03(=)**	*0*	0.05(+)	*0.01*	0.18(+)	*0.04*	0.06(+)	*0.02*	0.06(+)	*0.01*	0.03	*0*
DP_2	0.04(=)	*0.01*	0.05(+)	*0.01*	0.23(+)	*0.04*	0.07(+)	*0.02*	0.09(+)	*0.01*	**0.04**	*0.01*
DP_3	0.05(+)	*0.01*	0.06(+)	*0.02*	0.28(+)	*0.05*	0.08(+)	*0.02*	0.12(+)	*0.02*	**0.04**	*0.01*
DP_4	0.03(=)	*0.01*	0.05(+)	*0.01*	0.15(+)	*0.04*	0.06(+)	*0.01*	0.06(+)	*0.01*	**0.03**	*0*
DP_5	0.03(=)	*0.01*	0.04(+)	*0.01*	0.21(+)	*0.05*	0.06(+)	*0.02*	0.08(+)	*0.01*	**0.03**	*0*
DP_6	0.04(+)	*0.01*	0.05(+)	*0.01*	0.24(+)	*0.05*	0.07(+)	*0.01*	0.1(+)	*0.01*	**0.03**	*0*

表 7 (续)

算例	NSGA-II		MOEA/D		PeEA		LRVMA		SPAMA		QMA	
	均值	标准差	均值	标准差	均值	标准差	均值	标准差	均值	标准差	均值	标准差
DP_7	0.05(+)	*0.01*	0.07(+)	*0.02*	0.23(+)	*0.05*	0.09(+)	*0.02*	0.08(+)	*0.01*	**0.03**	*0.01*
DP_8	0.04(+)	*0.01*	0.06(+)	*0.02*	0.27(+)	*0.05*	0.1(+)	*0.03*	0.1(+)	*0.01*	**0.03**	*0.01*
DP_9	0.04(+)	*0.01*	0.06(+)	*0.02*	0.31(+)	*0.06*	0.1(+)	*0.03*	0.1(+)	*0.01*	**0.03**	*0.01*
DP_{10}	0.05(+)	*0.01*	0.06(+)	*0.02*	0.22(+)	*0.05*	0.09(+)	*0.02*	0.07(+)	*0.01*	**0.03**	*0.01*
DP_{11}	0.06(+)	*0.01*	0.07(+)	*0.02*	0.28(+)	*0.06*	0.1(+)	*0.02*	0.1(+)	*0.01*	**0.03**	*0.01*
DP_{12}	0.07(+)	*0.02*	0.08(+)	*0.02*	0.33(+)	*0.05*	0.11(+)	*0.03*	0.12(+)	*0.01*	**0.03**	*0.01*
DP_{13}	0.05(+)	*0.02*	0.08(+)	*0.01*	0.26(+)	*0.05*	0.11(+)	*0.02*	0.07(+)	*0.01*	**0.03**	*0.01*
DP_{14}	0.06(+)	*0.02*	0.09(+)	*0.03*	0.31(+)	*0.07*	0.12(+)	*0.03*	0.09(+)	*0.01*	**0.03**	*0.01*
DP_{15}	0.07(+)	*0.02*	0.09(+)	*0.03*	0.32(+)	*0.07*	0.14(+)	*0.03*	0.1(+)	*0.01*	**0.04**	*0.01*
DP_{16}	0.06(+)	*0.02*	0.08(+)	*0.02*	0.25(+)	*0.05*	0.11(+)	*0.03*	0.08(+)	*0.01*	**0.03**	*0.01*
DP_{17}	0.08(+)	*0.02*	0.1(+)	*0.03*	0.31(+)	*0.07*	0.12(+)	*0.02*	0.1(+)	*0.01*	**0.03**	*0.01*
DP_{18}	0.07(+)	*0.02*	0.08(+)	*0.02*	0.26(+)	*0.05*	0.09(+)	*0.02*	0.08(+)	*0.01*	**0.02**	*0*
胜	22		28		28		28		28			
平	4		0		0		0		0			
负	2		0		0		0		0			

表8　算法关于 HV 指标的对比结果

算例	NSGA-II		MOEA/D		PeEA		LRVMA		SPAMA		QMA	
	均值	标准差	均值	标准差	均值	标准差	均值	标准差	均值	标准差	均值	标准差
MK_1	**0.79(=)**	*0.02*	0.73(+)	*0.03*	0.61(+)	*0.05*	0.7(+)	*0.04*	0.73(+)	*0.02*	0.78	*0.01*
MK_2	0.83(=)	*0.02*	0.73(+)	*0.05*	0.52(+)	*0.12*	0.75(+)	*0.03*	0.75(+)	*0.03*	**0.84**	*0.02*
MK_3	0.86(+)	*0.03*	0.78(+)	*0.04*	0.52(+)	*0.1*	0.78(+)	*0.03*	0.75(+)	*0.03*	**0.89**	*0.02*
MK_4	**0.86(−)**	*0.03*	0.76(+)	*0.04*	0.63(+)	*0.1*	0.72(+)	*0.05*	0.78(+)	*0.03*	0.83	*0.03*
MK_5	0.84(+)	*0.03*	0.73(+)	*0.03*	0.64(+)	*0.07*	0.75(+)	*0.03*	0.76(+)	*0.03*	**0.9**	*0.02*
MK_6	0.71(+)	*0.03*	0.64(+)	*0.03*	0.35(+)	*0.08*	0.7(+)	*0.03*	0.56(+)	*0.04*	**0.77**	*0.01*
MK_7	0.87(+)	*0.02*	0.77(+)	*0.03*	0.59(+)	*0.11*	0.79(+)	*0.03*	0.8(+)	*0.03*	**0.93**	*0.01*
MK_8	0.84(+)	*0.02*	0.79(+)	*0.03*	0.57(+)	*0.1*	0.77(+)	*0.02*	0.76(+)	*0.02*	**0.9**	*0.01*
MK_9	0.77(+)	*0.04*	0.73(+)	*0.04*	0.47(+)	*0.1*	0.82(+)	*0.02*	0.64(+)	*0.04*	**0.91**	*0.01*
MK_{10}	0.76(+)	*0.03*	0.74(+)	*0.04*	0.41(+)	*0.1*	0.83(+)	*0.02*	0.62(+)	*0.04*	**0.92**	*0.01*
DP_1	0.79(+)	*0.02*	0.79(+)	*0.03*	0.55(+)	*0.06*	0.76(+)	*0.03*	0.77(+)	*0.02*	**0.81**	*0.01*
DP_2	0.83(+)	*0.02*	0.83(+)	*0.02*	0.55(+)	*0.04*	0.78(+)	*0.03*	0.77(+)	*0.02*	**0.84**	*0.02*
DP_3	0.85(+)	*0.02*	0.84(+)	*0.02*	0.56(+)	*0.07*	0.8(+)	*0.03*	0.78(+)	*0.02*	**0.87**	*0.03*
DP_4	0.8(+)	*0.02*	0.79(+)	*0.02*	0.59(+)	*0.06*	0.77(+)	*0.02*	0.78(+)	*0.01*	**0.82**	*0.01*
DP_5	0.83(+)	*0.03*	0.83(+)	*0.02*	0.57(+)	*0.06*	0.79(+)	*0.02*	0.78(+)	*0.02*	**0.86**	*0.01*
DP_6	0.81(+)	*0.03*	0.81(+)	*0.02*	0.52(+)	*0.06*	0.77(+)	*0.03*	0.73(+)	*0.01*	**0.87**	*0.01*
DP_7	0.79(+)	*0.03*	0.77(+)	*0.03*	0.52(+)	*0.06*	0.72(+)	*0.02*	0.78(+)	*0.01*	**0.83**	*0.02*
DP_8	0.81(+)	*0.03*	0.8(+)	*0.03*	0.52(+)	*0.06*	0.73(+)	*0.03*	0.78(+)	*0.01*	**0.85**	*0.02*
DP_9	0.81(+)	*0.02*	0.8(+)	*0.02*	0.47(+)	*0.05*	0.74(+)	*0.03*	0.77(+)	*0.01*	**0.85**	*0.02*
DP_{10}	0.77(+)	*0.03*	0.77(+)	*0.02*	0.5(+)	*0.06*	0.7(+)	*0.03*	0.77(+)	*0.01*	**0.82**	*0.02*
DP_{11}	0.79(+)	*0.02*	0.79(+)	*0.02*	0.49(+)	*0.08*	0.72(+)	*0.03*	0.76(+)	*0.01*	**0.85**	*0.02*
DP_{12}	0.78(+)	*0.03*	0.77(+)	*0.02*	0.47(+)	*0.06*	0.73(+)	*0.03*	0.74(+)	*0.01*	**0.88**	*0.02*
DP_{13}	0.74(+)	*0.03*	0.73(+)	*0.02*	0.49(+)	*0.06*	0.66(+)	*0.03*	0.77(+)	*0.01*	**0.8**	*0.02*
DP_{14}	0.76(+)	*0.03*	0.74(+)	*0.03*	0.45(+)	*0.08*	0.69(+)	*0.03*	0.76(+)	*0.01*	**0.83**	*0.02*
DP_{15}	0.76(+)	*0.04*	0.74(+)	*0.03*	0.47(+)	*0.08*	0.68(+)	*0.03*	0.75(+)	*0.02*	**0.82**	*0.02*
DP_{16}	0.74(+)	*0.03*	0.73(+)	*0.02*	0.5(+)	*0.06*	0.68(+)	*0.03*	0.77(+)	*0.01*	**0.8**	*0.02*
DP_{17}	0.74(+)	*0.03*	0.73(+)	*0.03*	0.48(+)	*0.07*	0.69(+)	*0.03*	0.74(+)	*0.01*	**0.83**	*0.02*
DP_{18}	0.76(+)	*0.03*	0.76(+)	*0.03*	0.54(+)	*0.06*	0.75(+)	*0.03*	0.78(+)	*0.01*	**0.88**	*0.01*
胜	25		28		28		28		28			
平	2		0		0		0		0			
负	1		0		0		0		0			

表9 对比算法间的 Friedman 秩和检验结果

算法	ONVG		IGD		HV	
	排名	p值	排名	p值	排名	p值
NSGA-II	1.71	1.21e-27	2.04	3.68e-27	2.18	2.19e-24
MOEA/D	3.32		3.46		3.68	
PeEA	6.00		6.00		6.00	
LRVMA	5.00		4.21		4.32	
SPAMA	3.67		4.18		3.75	
QMA	**1.28**		**1.11**		**1.07**	

表10 对比算法间的 Mann-Whitney U 检验结果

算法	p值(ONVG)	p值(IGD)	p值(HV)
(QMA, NSGA-II)	0.17	1.99e-06	5.32e-05
(QMA, MOEA/D)	1.29e-09	1.16e-09	1.20e-08
(QMA, PeEA)	7.02e-11	7.02e-11	7.02e-11
(QMA, LRVMA)	7.02e-11	2.03e-10	3.80e-10
(QMA, SPAMA)	9.17e-11	3.42e-10	1.05e-09

由表 6 可见: QMA 算法在 ONVG 指标上表现最好, 表明其提供的非支配解较多; NSGA-II 也在 8 个案例上表现出色; SPAMA 算法中包含全主动解码等机制, 因此占用了较多的计算资源, 导致算法在此指标上表现稍差.

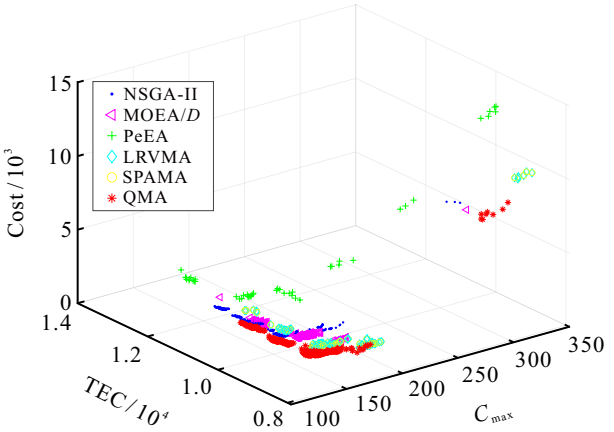
由表 7 可见: 在 IGD 指标中, QMA 在大多数算例中显著优于对比算法, 表明其提供的解不仅多样性强, 且更贴近前沿. 在MK_1、MK_4和DP_1三个案例中, NSGA-II 略优于 QMA, 但是对应的性能差异非常微小. Wilcoxon 秩和检验结果进一步表明, QMA 算法生成的结果与对比算法间存在显著的统计学差异.

表 8 中的结果显示, 就 HV 指标而言, QMA 在 26 个算例中的性能领先于对比算法, 标准差显示其性能较为稳定. 在MK_1算例中, QMA 的性能与 NSGA-II 持平, 而在MK_4算例中略逊于 NSGA-II. 在其他案例中, QMA 均取得了最佳结果. 这证实了 QMA 提供的解在适应值空间中分布广泛, 具备较高的多样性, 从而使得其在该指标上展现出更强的竞争力.

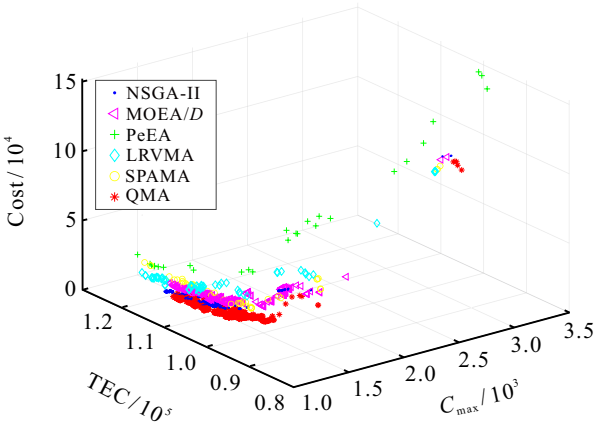
表 9 结果显示, QMA 的排名在 3 种指标下均位于第 1, 表明其性能最强, 提供的解集质量最高. 此外, 3 种指标对应的p值均小于显著性水平 0.05, 表明不同算法提供的结果间具备显著的差异.

表 10 结果显示: 在 ONVG 指标上, QMA 算法与 NSGA-II 算法的性能间无显著差异, 而与其他算法间存在显著差异; 在 IGD 与 HV 指标上, 所有的p值均小于 0.05, 算法的性能间具备显著的差异. 结合表 6 ～ 表 10 的相关内容可以推断, QMA 算法在求解 JITDFJSP 时表现出较优的性能, 能够在兼顾多样性与收敛性的同时, 提供更多调度方案.

图 5 为不同算法在MK_{10}和DP_{18}两个算例中的非支配解集分布. 由图 5 可见, QMA 算法提供的解集质量明显优于其他对比算法. 此外, 非支配解集的分布也表明 3 个目标间存在显著的冲突, 这进一步证实了采用多目标优化方法求解该模型的必要性.



(a) MK_{10}算例下对比算法的非支配解集分布



(b) DP_{18}算例下对比算法的非支配解集分布

图5 对比算法搜寻的非支配解集分布

4 结 论

研究准时制生产条件下的分布式柔性作业车间调度问题 (JITDFJSP) 具有非常重要的理论和现实意义. 本文以最小化完工时间、能量消耗和生产成本为目标, 对 JITDFJSP 进行了建模和求解. 首先, 在对其包含的 4 个子问题分别编码后, 运用强化学习驱动进化的模因算法 (QMA) 实现了求解. QMA 利用启发式初始化方法获取高质量的搜索群体, 并借助Q学习为父本推荐了最合适的搭档, 以取代传统的随机匹配, 降低了进化盲目性. 然后, 自适应局部搜索机制作用于进化停滞的种群, 进行了小步幅地局部探索, 不断地提升了搜索质量. 最后, 通过与 5 种算法对比, 验证了 QMA 可以提供收敛性强、分布性好

的非支配解集. 此外, 算法中配备的 3 种改进策略均可以显著提升搜索性能, 并共同做出最大贡献.

后续将在本研究的基础上, 进一步考虑动态因素, 如机器故障和订单插入, 以贴合实际车间状况. 此外, 分布式异构情形下的车间调度问题具备很强的现实应用背景, 在此背景下的分布式生产和订单分销的协同优化也是值得研究的方向.

参考文献 (References)

[1] 王凌, 邓瑾, 王圣尧. 分布式车间调度优化算法研究综述[J]. 控制与决策, 2016, 31(1): 1-11.
(Wang L, Deng J, Wang S Y. Survey on optimization algorithms for distributed shop scheduling[J]. Control and Decision, 2016, 31(1): 1-11.)

[2] 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, DOI: 10.13195/j.kzyjc.2025.0275.
(Guo H W, Sang H Y, Pan Q K. A learning-driven iterated local search algorithm for solving distributed flowshop robust scheduling problems[J]. Control and Decision, DOI: 10.13195/j.kzyjc.2025.0275.)

[3] 王凌, 王晶晶. 考虑运输时间的分布式绿色柔性作业车间调度协同群智能优化[J]. 中国科学: 技术科学, 2023, 53(2): 243-257.
(Wang L, Wang J J. A cooperative memetic algorithm for the distributed green flexible job shop with transportation time[J]. Science China Technological Sciences, 2023, 53(2): 243-257.)

[4] 方子丞, 李新宇, 高亮. 带负载均衡的混合算法求解分布式异构作业车间调度问题[J]. 控制理论与应用, 2024, 41(6): 977-989.
(Fang Z C, Li X Y, Gao L. Hybrid algorithm considering workload balance for solving the distributed heterogeneous job shop scheduling problem[J]. Control Theory & Applications, 2024, 41(6): 977-989.)

[5] 靳思远, 彭程, 王薇, 等. 基于向量映射代理模型的分布式柔性作业车间调度算法[J]. 控制与决策, 2025, 40(5): 1561-1570.
(Jin S Y, Peng C, Wang W, et al. Distributed flexible job shop scheduling algorithm based on avector mapping surrogate model[J]. Control and Decision, 2025, 40(5): 1561-1570.)

[6] 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.

[7] 魏光艳, 叶春明. 混合分布估计算法求解分布式柔性作业车间调度问题[J]. 运筹与管理, 2024, 33(8): 51-57.
(Wei G Y, Ye C M. Hybrid estimation of distribution algorithm for distributed flexible job shop scheduling[J]. Operations Research and Management Science, 2024, 33(8): 51-57.)

[8] Zhang X W, Liu S X, Zhao Z Y, et al. A decomposition-based evolutionary algorithm with clustering and hierarchical estimation for multi-objective fuzzy flexible jobshop scheduling[J]. IEEE Transactions on Evolutionary Computation, 2024, PP(99): 1.

[9] Chan F T S, Chung S H, Chan P L Y. Application of genetic algorithms with dominant genes in a distributed scheduling problem in flexible manufacturing systems[J]. International Journal of Production Research, 2006, 44(3): 523-543.

[10] Fu Y P, Gao K Z, Wang L, et al. Scheduling stochastic distributed flexible job shops using an multi-objective evolutionary algorithm with simulation evaluation[J]. International Journal of Production Research, 2025, 63(1): 86-103.

[11] 李佳磊, 顾幸生. 双种群混合遗传算法求解具有预防性维护的分布式柔性作业车间调度问题[J]. 控制与决策, 2023, 38(2): 475-482.
(Li J L, Gu X S. Two-population hybrid genetic algorithm for distributed flexible job-shop scheduling problem with preventive maintenance[J]. Control and Decision, 2023, 38(2): 475-482.)

[12] Xie J, Li X Y, Gao L, et al. A hybrid genetic tabu search algorithm for distributed flexible job shop scheduling problems[J]. Journal of Manufacturing Systems, 2023, 71: 82-94.

[13] 王凌, 王晶晶, 吴楚格. 绿色车间调度优化研究进展[J]. 控制与决策, 2018, 33(3): 385-391.
(Wang L, Wang J J, Wu C G. Advances in green shop scheduling and optimization[J]. Control and Decision, 2018, 33(3): 385-391.)

[14] 罗聪, 龚文引. 混合分解多目标进化算法求解绿色置换流水车间调度问题[J]. 控制与决策, 2024, 39(8): 2737-2745.
(Luo C, Gong W Y. A hybrid multi-objective evolutionary algorithm based on decomposition for green permutation flow-shop scheduling problem[J]. Control and Decision, 2024, 39(8): 2737-2745.)

[15] Li R, Gong W Y, Wang L, et al. Surprisingly popular-based adaptive memetic algorithm for energy-efficient distributed flexible job shop scheduling[J]. IEEE Transactions on Cybernetics, 2023, 53(12): 8013-8023.

[16] Du Y, Li J Q, Luo C, et al. A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations[J]. Swarm and Evolutionary Computation, 2021, 62: 100861.

[17] 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[J]. Expert Systems with Applications, 2023, 234: 121050.

[18] Pan Z X, Wang L, Wang J J, et al. A bi-learning evolutionary algorithm for transportation-constrained and distributed energy-efficient flexible scheduling[J]. IEEE Transactions on Evolutionary Computation, 2025, 29(1): 232-246.

[19] Yan Q, Wang H F, Yang S X. A learning-assisted bi-

population evolutionary algorithm for distributed flexible job-shop scheduling with maintenance decisions[J]. IEEE Transactions on Evolutionary Computation, 2024, PP(99): 1.

[20] 王艳红, 付威通, 张俊, 等. 基于改进近端策略优化算法的柔性作业车间调度[J]. 控制与决策, 2025, 40(6): 1883-1891.
(Wang Y H, Fu W T, Zhang J, et al. Flexible job-shop scheduling based on improved proximal policy optimization algorithm[J]. Control and Decision, 2025, 40(6): 1883-1891.)

[21] Li R, Gong W Y, Lu C, et al. A learning-based memetic algorithm for energy-efficient flexible job-shop scheduling with type-2 fuzzy processing time[J]. IEEE Transactions on Evolutionary Computation, 2023, 27(3): 610-620.

[22] Zhang Z P, Fu Y P, Gao K Z, et al. A learning-driven multi-objective cooperative artificial bee colony algorithm for distributed flexible job shop scheduling problems with preventive maintenance and transportation operations[J]. Computers & Industrial Engineering, 2024, 196: 110484.

[23] Tang H T, Xiao Y, Zhang W, et al. A DQL-NSGA-III algorithm for solving the flexible job shop dynamic scheduling problem[J]. Expert Systems with Applications, 2024, 237: 121723.

[24] 唐红涛, 刘歆, 张伟, 等. 基于Q-learning的改进人工蜂群算法求解分布式装配柔性作业车间绿色调度问题[J]. 工业工程与管理, 2024, 29(6): 166-179.
(Tang H T, Liu X, Zhang W, et al. Improved artificial bee colony algorithm based on Q-learning for solving green distributed assembly flexible job shop scheduling problem[J]. Industrial Engineering and Management, 2024, 29(6): 166-179.)

[25] Zhao S C, Zhou H, Zhao Y J, et al. DQL-assisted competitive evolutionary algorithm for energy-aware robust flexible job shop scheduling under unexpected disruptions[J]. Swarm and Evolutionary Computation, 2024, 91: 101750.

[26] Li R, Gong W Y, Wang L, et al. Co-evolution with deep reinforcement learning for energy-aware distributed heterogeneous flexible job shop scheduling[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2024, 54(1): 201-211.

[27] Fu Y P, Zhang Z P, Huang M, et al. Multi-objective integrated energy-efficient scheduling of distributed flexible job shop and vehicle routing by knowledge-and-learning-based hyper-heuristics[J]. IEEE Transactions on Emerging Topics in Computational Intelligence, 2025, PP(99): 1-14.

[28] Zhang G H, Gao L, Shi Y. An effective genetic algorithm for the flexible job-shop scheduling problem[J]. Expert Systems with Applications, 2011, 38(4): 3563-3573.

[29] Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197.

[30] Rudolph G. Convergence analysis of canonical genetic algorithms[J]. IEEE Transactions on Neural Networks, 1994, 5(1): 96-101.

[31] 徐宗本, 聂赞坎, 张文修. 遗传算法的几乎必然强收敛性 —— 鞅方法[J]. 计算机学报, 2002, 25(8): 785-793.
(Xu Z B, Nie Z K, Zhang W X. Almost sure convergence of genetic algorithms: A martingale approach[J]. Chinese Journal of Computers, 2002, 25(8): 785-793.)

[32] Brandimarte P. Routing and scheduling in a flexible job shop by tabu search[J]. Annals of Operations Research, 1993, 41(3): 157-183.

[33] Dauzère-Pérès S, Roux W, Lasserre J B. Multi-resource shop scheduling with resource flexibility[J]. European Journal of Operational Research, 1998, 107(2): 289-305.

[34] van Veldhuizen D A, Lamont G B. On measuring multiobjective evolutionary algorithm performance[C]. Proceedings of the Congress on Evolutionary Computation. La Jolla, 2000: 204-211.

[35] Coello C A C, Cortés N C. Solving multiobjective optimization problems using an artificial immune system[J]. Genetic Programming and Evolvable Machines, 2005, 6(2): 163-190.

[36] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach[J]. IEEE Transactions on Evolutionary Computation, 1999, 3(4): 257-271.

[37] Roy R K. Design of experiments using the Taguchi approach: 16 steps to product and process improvement[M]. John Wiley & Sons, 2001.

[38] Hollander M, Wolfe D A, Chicken E. Nonparametric statistical methods[M]. John Wiley & Sons, 2013.

[39] Zhang Q, Li H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition[J]. IEEE Transactions on Evolutionary Computation, 2007, 11(6): 712-731.

[40] Li L, Yen G G, Sahoo A, et al. On the estimation of Pareto front and dimensional similarity in many-objective evolutionary algorithm[J]. Information Sciences, 2021, 563: 375-400.

作者简介

赵仕存 (1996−), 男, 博士生, 主要研究方向为智能优化算法与应用, E-mail: zhaoshicun@buaa.edu.cn;

周泓 (1965−), 男, 教授, 博士, 博士生导师, 主要研究方向为生产与物流系统中的优化调度理论及方法、智能启发式优化算法、供应链管理, E-mail: h_zhou@buaa.edu.cn.