

文章编号: 1006-3080(2025)05-0623-10

DOI: 10.14135/j.cnki.1006-3080.20250117002

基于强化学习的正弦优化算法求解能耗分布式流水车间 节能调度问题

郎 峻¹, 殷晓明², 顾幸生¹

(1. 华东理工大学能源化工过程智能制造教育部重点实验室, 上海 200237;

2. 中烟机械技术中心, 上海 201206)

摘要: 针对分布式流水车间节能调度中最大完工时间(makespan)与总能耗(TEC)的多目标优化难题, 本文提出一种基于强化学习的正弦优化算法(RLSOA)。算法通过双重 Q-learning 策略协同优化加工序列与速度调整: 底层 Q-learning 优先加速关键路径任务以缩短 makespan, 顶层 Q-learning 降低非关键任务速度以减少 TEC。结合自适应参数与 4 种速度调整算子, 设计基于精英解导向的局部搜索策略, 平衡全局探索与局部开发。基于 480 组不同规模算例的实验表明, 相较于 KCA、INSGA 等对比算法, RLSOA 在覆盖率(C-metric)和反世代距离(IGD)指标上平均提升 23.6% 和降低 41.8%。消融实验验证双重 Q-learning 与局部搜索分别贡献 65.3% 和 28.7% 的解质量提升。统计检验($p < 0.05$)证实本文算法优越性, 为分布式制造系统提供了高效的节能调度工具。

关键词: 节能; 分布式流水车间调度; 强化学习; 元启发式算法; 多目标优化

中图分类号: Q232

文献标志码: A

在气候变化和全球变暖的压力下, 开发有效的节能技术, 采用先进的调度方案, 成为当前生产调度领域的研究热点。生产规模和企业协作的增强要求制造企业在分布于多个地理位置的工厂之间有效分配生产资源, 因此基于多工厂生产调度的分布式车间调度问题应运而生, 其目的是在不同生产能力和劳动力的限制下, 合理配置资源, 节约成本, 提高效率。能量有效的分布式流水车间置换调度问题是指在分布式流水车间环境中考虑能量消耗的作业调度过程。近年来, 随着智能优化算法的快速发展, 一些启发式规则和元启发式算法^[1]被应用于求解能耗分布式流水车间节能调度问题(EEDPFSP)。然而, EEDPFSP 的计算具有挑战性, 特别是随着产品和机器数量的增加, 高效的调度算法的开发还明显不足。

EEDPFSP 需要整合多个目标, 包括最小化能耗、减少完工时间和提高机器利用率等。由于这些目标经常相互冲突, 因此开发能够平衡这些冲突目标的算法, 并提供一组 Pareto 解决方案是现今面临的挑战。为了解决这些问题, 本文在强化学习框架下设计了一种新的智能优化算法以求解多目标 EEDPFSP。

节能调度的研究工作可以追溯到 2010 年前后, 但直到近几年才有系统的研究。Dai 等^[2]提出了一种基于关机+开机能量消耗与空闲运行能量消耗之比权衡最大完工时间和能耗的节能模型, 并为不能完全关机的机器设计了速度缩放框架。Elisabet 等^[3]建立了一个批调度模型, 同时考虑了产品流、原材料和设备消耗以满足环境要求。对于传统的流水车间调度问题, Fang 等^[4]用 10 个不同的阈值进行峰值功

收稿日期: 2025-01-17

基金项目: 国家自然科学基金(61973120)

作者简介: 郎 峻(1998—), 女, 安徽芜湖人, 硕士生, 主要研究方向为生产计划与调度, 智能优化方法。

通信联系人: 顾幸生, E-mail: xsgu@ecust.edu.cn

引用本文: 郎 峻, 殷晓明, 顾幸生. 基于强化学习的正弦优化算法求解能耗分布式流水车间节能调度问题[J]. 华东理工大学学报(自然科学版), 2025, 51(5): 623-632.

Citation: LANG Jun, YIN Xiaoming, GU Xingsheng. Sine Optimization Algorithm Based on Reinforcement Learning for Energy-Efficient Distributed Flow Shop Scheduling Problem[J]. Journal of East China University of Science and Technology, 2025, 51(5): 623-632.

耗约束,提出了一种以尽可能小的速度尽早处理作业的节能规则;Yüksel 等^[5]提出了多目标离散人工蜂群算法、多目标遗传算法和多目标局部搜索遗传算法,并比较了这些算法的优越性;Ztop 等^[6]假设一台机器具有 3 个处理速度(快速、正常和慢速)级别,提出了 3 种具有速度缩放结构的节能算法。Jiang 等^[7]设计了一个根据能耗和完工时间匹配解的动态策略。Li 等^[8]设计了两种考虑最小能耗的解码启发式算法,并提出了一种右移启发式算法,以减少机器空闲时间和能耗。Naderi 等^[9]提出了分布式置换流水车间调度问题(Distributed Permutation Flow Shop Scheduling Problem, DPFSP)。由于实际制造业中生产过程的差异和生产环境的限制,DPFSP 扩展了许多不同的模型,包括分布式两机流水车间调度问题^[10]、分布式混合流水车间调度问题^[11]、分布式无空闲流水车间调度问题^[12]、分布式阻塞流水车间调度问题^[13]、具有多处理器任务的分布式混合流水车间^[14]和分布式无等待流水车间调度问题^[15]等。然而,对分布式环境下的节能流水车间调度研究较少。Wang 等^[16]提出了一种基于知识的协同算法,包括协同初始化、协同搜索和基于知识的局部强化搜索。Li 等^[17]设计了一种改进的 NSGA II 算法,并在人工蜂群算法的启发下构造了一种新的种群生成方法。

目前,应用于生产调度的强化学习算法一般采用基于值的强化学习算法,通过选择具有最大状态-动作值的作用构造最优策略。此类强化学习方法具有较高的样本利用率,但容易过拟合。代表性算法包括 SARSA、Q-learning 和 DQN^[18]。为了解决柔性作业车间调度问题,Bouazza 等^[19]应用 Q-learning 方法选择最合适的机器选择规则实现作业分配。Said 等^[20]设计了两个深度 Q-learning 网络处理实时调度,并根据最优完工时间与解之间的误差设计了 Q-learning 的奖励公式,该策略通过给予更好的解、更高的奖励引导解的进化。针对动态车间调度问题中的状态空间问题,Wang^[18]提出了一种聚类方法降低状态的维数,并定义了一种加权 Q-learning 方法降低聚类误差。为了解决置换流水车间调度问题,Li 等^[21]使用 Q-learning 选择人工蜂群算法的最优邻域结构。尽管强化学习方法已经成功地应用于车间调度问题,但鲜有文献探讨分布式环境下的强化学习方法。

本文针对复杂的最小化能耗分布式流水车间调度问题,设计了基于强化学习的正弦优化算法,以最小化总能耗和最大完工时间为目标函数;提出了 4 种根据能耗计算特性调整的加速方法,并给出双层 Q-learning 策略学习作业序列调整规则和加工速度调整

规则;将 SOA 的位置更新公式扩展为一个自适应参数,这种改进可以平衡对解空间的探索;设计了一种基于速度调整的局部搜索策略,以提高 RLSOA 的搜索强度,当双层 Q-learning 策略无法搜索到更好的解时,该策略可以进一步减少最大完工时间或总能耗(TEC)。

1 能耗分布式置换流水车间调度模型

能耗分布式置换流水车间调度问题在规模为 $n \times m \times F$ (n 为工件数, m 为机器数, F 为工厂数)的 DPFSP 模型上扩展了加工速度设置,每台设备的加工速度定义成 s 种不同的等级 $v = 1, 2, \dots, s$,在工件加工过程中,设备速度不可以改变。工件 i 在设备 j 上的加工工序为 $O_{i,j}$,其标准加工时间 $t_{i,j} > 0$ 。若以速度 $v_{i,j}$ 执行工序,则工序 $O_{i,j}$ 的实际加工时间为 $p_{i,j} = t_{i,j}/v_{i,j}$,设备单位能耗为 $PP_{f,j,v}$ 。设备加工速度越大,设备能耗越高,同时工件加工时长就越短。当分配给工厂的所有任务完成,设备才可关闭;当没有工件在设备 j 上加工时,设备的单位能耗为 $SP_{f,j}$ 。本文以最小化最大完工时间 $C_{\max}$ 和总能耗 TEC 为优化目标, $C_{\pi(k),j,f}$ 表示加工序列中处于位置 k 的工件在设备 j 上的完工时间, C_f 为工厂 f 的完工时间, $PEC_{f,j}$ 和 $SEC_{f,j}$ 分别表示工厂 f 的设备 j 运行在加工状态和就绪状态的能耗。该问题的求解分为 3 个子问题:工厂的分配、每个工厂中工件的加工顺序以及每道工序的加工速度。

EEDPFSP 的混合整数线性规划模型定义如下^[22]:

$$\min\{C_{\max}, \text{TEC}\} \quad (1)$$

s.t.

$$\sum_{k=1}^{n_f} \sum_{f=1}^F X_{i,\pi(k),f} = 1 \quad \forall i \quad (2)$$

$$\sum_{i=1}^n \sum_{f=1}^F X_{i,\pi(k),f} = 1 \quad \forall k \quad (3)$$

$$\sum_{v=1}^s Z_{i,j,v} = 1 \quad \forall i, j \quad (4)$$

$$p_{i,j} = t_{i,j} \times \sum_{v=1}^s \frac{Z_{i,j,v}}{v_{i,j}} \quad \forall i, j \quad (5)$$

$$C_{\pi(k),j,f} = \sum_{i=1}^n X_{i,\pi(k),f} \times p_{i,j} \quad k = 1, j = 1, \forall f \quad (6)$$

$$C_{\pi(k),j,f} \geq C_{\pi(k),j-1,f} + \sum_{i=1}^n X_{i,\pi(k),f} \times p_{i,j} \quad \forall k, j, f \quad (7)$$

$$C_{\pi(k),j,f} \geq C_{\pi(k-1),j,f} + \sum_{i=1}^n X_{i,\pi(k),f} \times p_{i,j} \quad \forall k > 1, j, f \quad (8)$$

$$C_f \geq C_{\pi(k),j,f} \quad \forall k, j \quad (9)$$

$$C_{\max} = \max_f C_f \quad \forall f \quad (10)$$

$$\text{PEC}_{f,j} = \sum_{i=1}^n \sum_{k=1}^{n_f} (X_{i,\pi(k),f} \times p_{i,j} \times \sum_{v=1}^s Z_{i,j,v} \times \text{PP}_{f,j,v}) \quad \forall f, j \quad (11)$$

$$\text{SEC}_{f,j} = (C_f - \sum_{i=1}^n p_{i,j} \times \sum_{k=1}^{n_f} X_{i,\pi(k),f}) \times \text{SP}_{f,j} \quad \forall f, j \quad (12)$$

$$\text{TEC} = \sum_{f=1}^F \sum_{j=1}^m (\text{PEC}_{f,j} + \text{SEC}_{f,j}) \quad (13)$$

$$X_{i,\pi(k),f} \in \{0, 1\} \quad \forall i, k, f \quad (14)$$

$$Z_{i,j,v} \in \{0, 1\} \quad \forall i, j, v \quad (15)$$

$$C_{\pi(k),j,f} \geq 0 \quad \forall k, j, f \quad (16)$$

其中, $C_{\pi(k),0,f} = 0$ 。式(1)表示模型的目标函数;式(2)、(3)所示确保加工序列和工件的位置相互匹配;式(4)约束每个加工任务 $O_{i,j}$ 只能以不变的速度加工;式(5)计算 $O_{i,j}$ 的实际加工时间;式(6)计算每个工厂首个工件在首台设备上的完工时间;式(7)所示确保工件 i 完成前一项工序的加工后才能传送到后一台设备进行后续加工任务;式(8)确保每台设备在当前的工件加工完成前不能处理其他加工任务;式(9)定义了每个工厂的最大完工时间;式(10)定义了 makespan;式(11)~(13)计算加工状态和空闲状态下的能耗计算公式以及总能耗;式(14)~(16)定义了各变量的范围。

2 基于强化学习的正弦优化算法

2.1 编码和解码

为了完整描述一个可行解,编码应该包括3个部分:工件的工厂分配信息、每个工厂的加工顺序和每个加工操作的加工速度。本文采用 Wang 等^[16]提出的编码策略,工件的加工顺序由 F 维向量 $\boldsymbol{\pi} = \{\{\pi_1^1, \pi_2^1, \dots, \pi_{n_1}^1\}, \dots, \{\pi_1^F, \pi_2^F, \dots, \pi_{n_F}^F\}\}$ 表示,其中每个向量表示工厂中工件的加工顺序。处理速度的编码由 $n \times m$ 维的速度矩阵 $\mathbf{V} = (v_{i,j})_{n \times m}$ 表示,其中 $v_{i,j}$ 表示 $O_{i,j}$ 的加工速度。因此,可行解表示为 $\text{sol} = (\boldsymbol{\pi}, \mathbf{V})$ 。

初始化阶段,随机生成工件的加工序列并确保每个工厂都含有待加工的工件,使用最小允许速度

(Slowest Allowable Speed, SAS)规则对每道工序分配合适的加工速度,在加工速度尽量小的条件下,使得加工流程尽量连续。

2.2 速度调整算子

与普通的 DPFSP 问题相比,具有能耗约束的 DPFSP 问题需要额外考虑处理速度。加工速度越快,机器的能耗越高,同时,机器的空载也会产生一定能耗^[23]。如果处理速度很慢,则会延长完工时间。因此,需要调整合适的加工速度平衡完成时间和机器能耗,减少机器空闲时间。为了提高调速效率,减少计算时间,从开始时刻到结束时刻,两次操作之间没有空闲时间的连续路径。对于关键路径上的工厂,可以通过调整其操作减少完工时间^[24]。根据以上分析,设计速度调整算子如下:

(1) Speedup(f): 对于关键路径上的所有关键任务 $O_{i,j}$,若执行前有一段空闲时间,则在不增加工厂 f 的能耗的条件下,尽可能地提高操作 $O_{i,j-1}$ 的执行速度,并将操作整体左移,如图 1(a) 所示。

(2) Speeddown(f): 对于非关键路径上的所有加工任务,在不增加总体 makespan 的条件下,将加工速度降低到最小,如图 1(b) 所示。

(3) LocalSpeedup(f): 对于关键路径上的所有关键任务 $O_{i,j}$,若执行前有空闲时间,则增大 $O_{i,j-1}$ 的加工速度直到该工厂的完工时间无法继续降低或加工速度已达到最大。

(4) LocalSpeeddown(f): 对于非关键路径上的所有加工任务,降低工件的加工速度直到该工厂的能耗无法继续降低或加工速度已达到最小。

2.3 加工序列调整算子

对于加工序列 $\boldsymbol{\pi}$,针对每项优化目标 obj_l 设计 4 种工件序列调整算子(Job-sequence Adjustment Operators, JAO),其中,针对不同的优化目标将完工时间或能耗最大的工厂作为关键工厂,以 $\text{obj}_{\text{makespan}}$ 为例, π_d 为关键工厂中的连续 d 个工件序列:

(1) 基于关键工厂间的插入操作:随机选择 π_d 从关键工厂 f_{obj} 移除,并插入到关键工厂除外的任意工厂 f 所有可能的位置,执行 Speedup(f) 和 Speedup(f_{obj})。

(2) 基于关键工厂内的插入操作:在关键工厂 f_{obj} 中随机选择 π_d ,剩余工件为 π_r ,将 π_d 插入到 π_r 所有可能的位置,执行 Speedup(f_{obj})。

(3) 基于关键工厂间的交换操作:在关键工厂 f_{obj} 中随机选择 π_d ,并将其和任意工厂 f 中所有连续的 d 个工件进行交换,执行 Speedup(f) 和 Speedup(f_{obj})。

(4) 基于关键工厂内的交换操作:在关键工厂

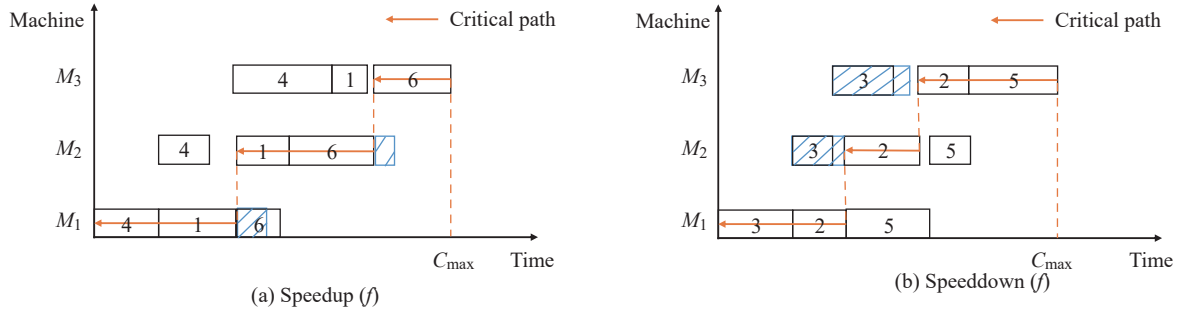


图 1 速度调整策略甘特图

Fig. 1 Gantt chart of speed adjustment strategy

f_{obj} 中随机选择 π_d , 剩余工件为 π_r , $\pi_r \geq \pi_d$, 将 π_d 和 π_r 中所有连续的 d 个工件进 f 行交换, 执行 Speedup(f_{obj})。

以上方法若旧解被新解支配, 返回新解; 否则保留 $obj_{makespan}$ 最小的解。

式(17)表示对函数值取整, 并取绝对值。参数 d 采用 SOA(Sine Optimization Algorithm)中的正弦公式计算:

$$d = \lfloor r_1 \times \sin r_2 \rfloor \quad (17)$$

$$r_1 = \beta \times n_f \times \left(1 - \frac{T_{dur}}{T_{max}}\right) \quad (18)$$

其中, n_f 为工厂 f 的工件数量, T_{max} 为最大 CPU 运行时间, T_{dur} 为程序已运行时间。 β 为控制参数, 由于 $d \leq 0.5 \times n_f$, 所以 β 的取值范围为 (0,0.5), r_2 为 $[0, 2\pi]$ 之间的随机数。

2.4 双重 Q-learning 策略

本节设计了一种双重 Q-learning 策略。底层 Q-learning 的任务是搜索具有更优 makespan 的个体, 通过调整工件加工顺序和提高关键任务的加工速度, 提高设备利用率, 减少加工时间。顶层 Q-learning 的任务是搜索 TEC 更优的个体解, 通过调整工件加工顺序和降低非关键任务的加工速度, 减少空闲时间和设备能耗。

此策略将个体抽象为一个智能体, 智能体状态的变化给 Q-learning 提供反馈, 以判断此次执行的动作是否可以提高解的质量, 智能体通常选择 Q 值最大的动作获得最高的奖励。为了准确执行反馈, 使用解的目标函数表现智能体的状态, 本文将 8 种加工工序调整算子作为顶层和底层的动作集。

在多目标问题中, 解的质量主要体现在非支配关系以及距离参考点的位置远近上。将历史中优化目标最小/最大的值定义为最小/最大参考点, 对于解 X 及其执行完动作后得到的新解 X' , 奖励函数定义如下:

$$R_t = \frac{\text{dis}(X', X_{\max})}{\text{dis}(X', X_{\max}) + \text{dis}(X', X_{\min})} + \begin{cases} 1, X' < X \\ 0, \text{otherwise} \\ -1, X < X' \end{cases} \quad (19)$$

其中, $\text{dis}(X', X_{\min})$ 为归一化后新解和最小参考点的欧氏距离。 $X' < X$ 表示解 X' 支配解 X 。根据新解和旧解的支配关系给予奖励, 同时考虑新解自身在种群进化中的优势, 距离最远参考点越远, 或距离最近参考点越近, 则奖励越大。智能体每轮迭代执行 Q 值最大的动作, 为了避免出现执行单一动作的情况, 设置控制参数 randp, 当生成的 (0,1) 间随机数小于 randp 时, 随机执行任一动作。伪代码如下:

Algorithm 1: Pseudocode of Dual Q-learning

Initialize Q-table matrices $Q_{\text{bottom}}, Q_{\text{top}}$;

Choose agent X and update d ;

If $\text{rand}(0, 1) < \text{randp}$

JAO($X, \text{rand_makespan}$); //randomly perform JAO on X

Else

JAO($X, \max Q_{\text{bottom}}$); //assign JAO with the highest reward value to X

End if

If $\text{rand}(0, 1) < \text{randp}$

JAO($X, \text{rand_TEC}$);

Else

JAO($X, \max Q_{\text{top}}$);

End if

Update $Q_{\text{bottom}}, Q_{\text{top}}$;

2.5 基于速度调整的局部搜索策略

为了加强算法在精英解附近的搜索深度, 提出一种局部搜索策略。针对种群中所有的非支配解, 随机选择一项优化目标, 在忽略另一项优化目标的前提下, 对非支配解附近的搜索区域进行纵向深度搜索。具体策略为: 从关键工厂中随机选择任意工件移除, 再将工件重新插入到关键工厂除外的所有工厂任何可行位置, 最终保留该优化目标最小的解, 在执行局部速度调整后更新外部档案集。策略的伪

代码如下:

Algorithm 2: Pseudocode of Local Search

Begin

For non-dominated solution X do

If $\text{rand} < 0.5$

obj = makespan ; $F_{\text{critical}} = F_{\text{makespan}}$;

Else

obj = TEC ; $F_{\text{critical}} = F_{\text{TEC}}$;

End if

job = $\pi_{F_{\text{critical}}}(\text{random})$; //Select a random job from

F_{critical}

Remove job;

Reinsert job into all positons in all factories except

F_{critical} , keeping the minimum obj;

If obj == makespan

LocalSpeedup(f), $f = 1, \dots, F$;

Else if obj == TEC

LocalSpeeddown(f), $f = 1, \dots, F$;

End if

If not $X < X'$

Update(PeratoArchive, X') ; //Update external

archive

End if

End for

End

2.6 RLSOA 算法流程图

RLSOA 算法包括种群初始化、基于 Q-learning 的全局搜索、选择策略以及局部搜索 4 个阶段。具体算法流程如图 2 所示。

3 仿真实验与统计分析

3.1 实验设置

为检验所提出的 RLSOA 求解 EEDFSP 的优越性能, 本文采用文献 [17] 提供的实验算例进行仿真求解, 其中, 工件规模 $n = \{20, 40, 60, 80\}$, 设备规模 $m = \{4, 8, 16\}$, 工厂规模 $F = \{2, 3, 4, 5\}$, 每种规模组合各有 10 个不同的算例, 共记 480 个算例。每个算例独立运行 10 次, 将 10 次计算获得的非支配解集作为该算例最终得到的 Pareto 前沿。工件加工时间设置为 $[5, 50]$ 内的随机整数, 设备加工速度设置为 5 个等级 $\{1, 1.3, 1.55, 1.75, 2.10\}$ 。算法基于 Visual Studio 2022 C++ 实现, 使用主频为 2.4 GHz 的 Intel (R) Xeon (R) CPU E5-2640 v4 处理器和 64 GB 内存的 PC 计算。运行的终止条件为最大 CPU 占用时间 $T_{\text{max}} =$

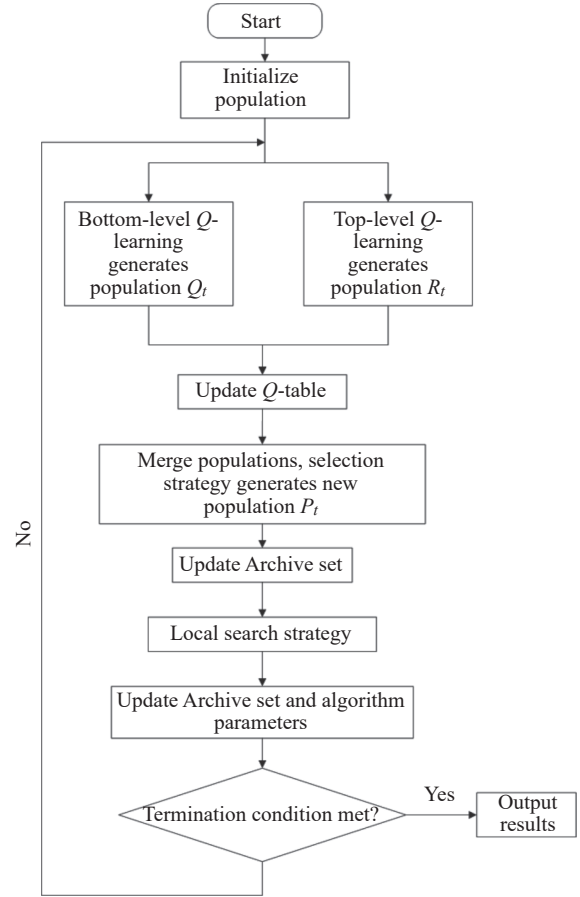


图 2 RLSOA 算法流程图

Fig. 2 Flowchart of the RLSOA

$0.5 \times n(s)$ 。

3.2 参数设置

RLSOA 有 3 个主要参数: 种群规模 popsize, 正弦算法的控制参数 beta 与 Q-learning 策略执行概率 randp, 如表 1 所示。采用 ANOVA 对参数的各组合进行分析。为防止结果过拟合, 随机生成校正算例: $n = \{20, 40, 60, 80\}$, $m = \{4, 8, 16\}$, $F = \{2, 3, 4\}$, 共 36 种不同组合, 标准加工时间为 $[5, 50]$ 间的随机整数。采用贡献度 CON 作为不同参数组合所求 Pareto 解集质量的衡量指标。

表 1 RLSOA 的 ANOVA 因子水平表

Table 1 Factor levels for ANOVA of RLSOA

Level	Value		
	popsize	beta	randp
1	10	0	0.1
2	20	0.1	0.2
3	30	0.2	0.3
4	40	0.3	0.4

$$\text{CON}(i) = \frac{|F_i|}{|F_{\text{ref}}|} \quad (20)$$

其中,用 $F_i(i=1,2,\dots,36)$ 表示每个参数组合求得的 Pareto 解集, F_{ref} 为 F_1-F_{36} 综合获得的 Pareto 解集, $F'_i = F_i \cap F_{\text{ref}}$, $|F'_i|$ 表示 F'_i 中非支配解的个数, $|F_{\text{ref}}|$ 表示 F_{ref} 中非支配解的个数, CON 值越大,说明解的质量越好。ANOVA 实验结果和各参数在不同规模下对 RLSOA 的影响趋势分别如表 2 和图 3 所示。

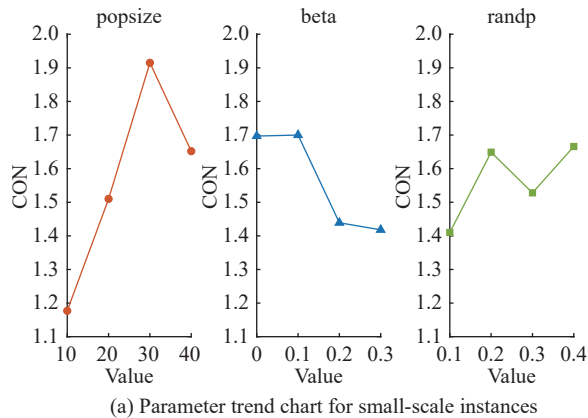
表 2 ANOVA 实验结果

Table 2 Experimental results of ANOVA

Factor	DF	F	p
popsze	3	1.406	0.001
beta	3	1.096	0.024
randp	3	0.870	0.874

DF—Degree of freedom

由表 2 可以看出, popsize 和 beta 的 p 值均小于



0.05, 说明该参数对算法性能有显著性影响。由于算例的规模跨度较大, 因此将问题规模划分为小规模(即工件数小于等于 40)和大规模(即工件数大于 40)两种情况分别对参数选择进行讨论。种群大小 popsize 决定了采样点的多少, 种群过小无法覆盖整个搜索空间; 而随着问题规模增大, 求解的复杂度提高, 种群过大会花费大量的计算时间, 降低收敛速度。由图 3 可以看出, 在小规模问题上, popsize 取值为 30 时 CON 值最大, 解的质量最好; 在大规模问题上, popsize 的最优值为 10(CON 值最大)。beta 控制了个体更新时变化量的波动大小。图 3 显示在不同规模的问题上, beta 最优的取值均为 0.1。randp 对算法没有显著性影响, 综合在两种规模问题上的表现, randp 取值为 0.2。综上所述, 各参数的取值为: beta=0.1, randp=0.2, 小规模问题 popsize=30, 大规模问题 popsize=10。

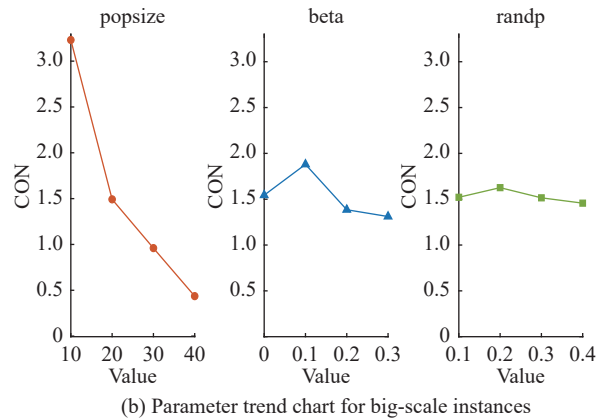


图 3 各参数在不同规模下对 RLSOA 的影响趋势图

Fig. 3 Trend chart of the influence of each parameter on the RLSOA at different scales

3.3 RLSOA 消融实验

为了验证各个改进策略对 RLSOA 性能的影响程度, 本节对比 RLSOA 和两种算法变体的各项性能指标。两种算法变体包括去除局部搜索策略(RLSOA1)和去除双层 Q-learning 策略(RLSOA2)。采用反世代距离^[25](Inverted Generational Distance, IGD)和覆盖率^[26](Coverage metric, C-metric)评价 Pareto 解集的质量。

表 3 所示为 RLSOA1 和 RLSOA 的 C-metric 指标统计。当工厂数为 2 时, RLSOA 受 RLSOA1 支配的解均少于 RLSOA1 受 RLSOA 支配的解。当工厂个数为 3、4 时, 仅有两个规模算例的平均覆盖率 $C(\text{RLSOA1}, \text{RLSOA})$ 略小于 $C(\text{RLSOA}, \text{RLSOA1})$ 。当工厂规模为 5 时, RLSOA 在大部分规模的算例上好于去掉局部搜索策略。局部搜索策略在一定程度上可以扩展算法搜索的深度, 在精英解附近进一步开

发探索, 但由于局部搜索复杂度较高, 因此受问题规模大小和计算时间的限制较大。

表 4 所示为 RLSOA2 和 RLSOA 的 C-metric 指标统计结果。结果表明在各个不同规模的算例上, RLSOA 获得的 Pareto 解集的质量都要优于 RLSOA2。说明 Q-learning 策略的引入很大程度上改善了算法的求解性能, 通过奖励机制减少了无效搜索策略的使用, 提高精英解集更新的概率。同时, Q-learning 策略对算法的优化效果没有受到算例规模大小的显著影响, 表明引入的 Q-learning 策略有一定的稳定性和鲁棒性。

表 5、表 6 所示为 RLSOA 与 RLSOA1、RLSOA2 的 IGD 值对比。统计结果表明当工厂个数小于 5 时, RLSOA 在所有规模的工件和机器中产生更高质量的解集, 当工厂数为 5 时, RLSOA1 在个别规模的算例上其平均 IGD 值略小于 RLSOA。说明当工

表 3 RLSOA 与 RLSOA1 的 C-metric 值结果
Table 3 Results of C-metric values for RLSOA and RLSOA1

(n,m)	C(RLSOA, RLSOA1)				C(RLSOA1, RLSOA)			
	$F=2$	$F=3$	$F=4$	$F=5$	$F=2$	$F=3$	$F=4$	$F=5$
(20,4)	0.265	0.255	0.291	0.355	0.637	0.645	0.569	0.438
(20,8)	0.261	0.290	0.420	0.505	0.626	0.591	0.464	0.236
(20,16)	0.230	0.345	0.469	0.485	0.665	0.564	0.441	0.401
(40,4)	0.049	0.328	0.416	0.299	0.936	0.572	0.521	0.650
(40,8)	0.260	0.308	0.287	0.364	0.601	0.626	0.666	0.597
(40,16)	0.231	0.207	0.288	0.375	0.614	0.736	0.664	0.596
(60,4)	0.234	0.468	0.396	0.339	0.529	0.278	0.360	0.487
(60,8)	0.281	0.331	0.304	0.438	0.469	0.479	0.512	0.325
(60,16)	0.162	0.160	0.300	0.304	0.643	0.709	0.477	0.488
(80,4)	0.264	0.422	0.332	0.466	0.601	0.342	0.489	0.289
(80,8)	0.234	0.211	0.300	0.339	0.588	0.584	0.469	0.464
(80,16)	0.219	0.304	0.293	0.348	0.679	0.583	0.561	0.376

表 4 RLSOA 与 RLSOA2 的 C-metric 值结果
Table 4 Results of C-metric values for RLSOA and RLSOA2

(n,m)	C(RLSOA, RLSOA2)				C(RLSOA2, RLSOA)			
	$F=2$	$F=3$	$F=4$	$F=5$	$F=2$	$F=3$	$F=4$	$F=5$
(20,4)	0.174	0.150	0.291	0.335	0.786	0.748	0.608	0.544
(20,8)	0.212	0.216	0.381	0.467	0.728	0.699	0.544	0.467
(20,16)	0.430	0.366	0.327	0.246	0.452	0.546	0.624	0.696
(40,4)	0.056	0.204	0.248	0.262	0.889	0.718	0.724	0.697
(40,8)	0.193	0.230	0.164	0.204	0.726	0.673	0.807	0.702
(40,16)	0.371	0.202	0.269	0.275	0.528	0.739	0.660	0.666
(60,4)	0.174	0.377	0.414	0.431	0.775	0.581	0.543	0.517
(60,8)	0.449	0.241	0.197	0.370	0.461	0.687	0.734	0.601
(60,16)	0.384	0.255	0.354	0.341	0.534	0.660	0.586	0.544
(80,4)	0.230	0.251	0.261	0.436	0.710	0.648	0.684	0.527
(80,8)	0.333	0.166	0.288	0.306	0.618	0.773	0.588	0.684
(80,16)	0.248	0.354	0.287	0.447	0.649	0.590	0.633	0.514

厂数量越多时,局部搜索策略对算法性能的提升逐渐减弱。为了弥补局部搜索的缺点,设计了一个自适应参数 d 。根据每个工厂内工件数量以及算法的迭代轮数自适应地改变个体解位置变化量的大小。结合 ANOVA 结果,可以看到自适应参数对解集质量有显著提高。

3.4 RLSOA 对比实验及统计分析

为了检验 RLSOA 求解 EEDPFSP 问题的效果,

表 5 RLSOA、RLSOA1 与 RLSOA2 的 IGD 值计算结果($F=2$, $F=3$)

Table 5 Results of IGD values for RLSOA, RLSOA1 and RLSOA2($F=2$, $F=3$)

(n,m)	$F=2$			$F=3$		
	RLSOA	RLSOA1	RLSOA2	RLSOA	RLSOA1	RLSOA2
(20,4)	0.003	0.011	0.009	0.003	0.014	0.009
(20,8)	0.004	0.013	0.01	0.004	0.014	0.009
(20,16)	0.005	0.015	0.006	0.005	0.011	0.006
(40,4)	0.001	0.023	0.015	0.003	0.013	0.009
(40,8)	0.004	0.013	0.01	0.005	0.015	0.01
(40,16)	0.006	0.017	0.008	0.003	0.016	0.01
(60,4)	0.004	0.023	0.014	0.008	0.013	0.011
(60,8)	0.009	0.019	0.01	0.006	0.018	0.012
(60,16)	0.005	0.033	0.011	0.004	0.022	0.01
(80,4)	0.005	0.024	0.017	0.007	0.011	0.015
(80,8)	0.006	0.029	0.011	0.004	0.021	0.016
(80,16)	0.004	0.032	0.012	0.007	0.026	0.009

表 6 RLSOA、RLSOA1 与 RLSOA2 的 IGD 值计算结果($F=4$, $F=5$)

Table 6 Results of IGD values for RLSOA, RLSOA1 and RLSOA2($F=4$, $F=5$)

(n,m)	$F=4$			$F=5$		
	RLSOA	RLSOA1	RLSOA2	RLSOA	RLSOA1	RLSOA2
(20,4)	0.004	0.01	0.009	0.006	0.008	0.01
(20,8)	0.006	0.01	0.008	0.009	0.005	0.009
(20,16)	0.006	0.009	0.008	0.006	0.007	0.01
(40,4)	0.005	0.012	0.01	0.004	0.016	0.01
(40,8)	0.005	0.016	0.013	0.005	0.018	0.01
(40,16)	0.004	0.018	0.008	0.005	0.015	0.01
(60,4)	0.008	0.01	0.01	0.007	0.012	0.009
(60,8)	0.005	0.014	0.014	0.009	0.008	0.012
(60,16)	0.006	0.017	0.009	0.005	0.011	0.008
(80,4)	0.005	0.017	0.011	0.008	0.006	0.009
(80,8)	0.005	0.014	0.009	0.007	0.011	0.012
(80,16)	0.007	0.023	0.011	0.007	0.013	0.009

将其与两种较新的与能耗相关的多目标优化算法 KCA(Knowledge-Based Cooperative Algorithm) 和 INSGA(Improved Nondominated Sorting Genetic Algorithm)进行比较,CPU 占用时间为 $T_{\max} = 0.5 \times n(s)$,

KCA 和 INSGA 按照文献 [15-16] 进行了复现。

表 7、表 8 分别展示了 RLSOA 与 KCA 和 INSGA 的 C-metric 指标对比结果。在不同规模下, RLSOA 产生的 pareto 解数量明显更多。表 9、表 10 所示均为 3 种算法的 IGD 指标对比, 可以看出, RLSOA 在各个规模上的结果都要小于对比算法近一个数量级。随着问题规模增大, RLSOA 仍保持着平稳的优势。RLSOA 包括两种不同的速度调整策略, 一类是

表 7 RLSOA 与 KCA 的 C-metric 值结果

Table 7 Results of C-metric values for RLSOA and KCA

(n, m)	C(SOA, KCA)				C(KCA, SOA)			
	$F=2$	$F=3$	$F=4$	$F=5$	$F=2$	$F=3$	$F=4$	$F=5$
(20,4)	0.000	0.001	0.006	0.032	1.000	0.996	0.976	0.882
(20,8)	0.001	0.000	0.000	0.000	0.994	1.000	1.000	1.000
(20,16)	0.003	0.000	0.000	0.000	0.993	1.000	1.000	1.000
(40,4)	0.008	0.002	0.003	0.010	0.936	0.981	0.976	0.959
(40,8)	0.095	0.003	0.000	0.000	0.720	0.980	1.000	1.000
(40,16)	0.079	0.012	0.001	0.000	0.813	0.957	0.998	1.000
(60,4)	0.013	0.053	0.105	0.147	0.956	0.786	0.686	0.524
(60,8)	0.195	0.099	0.130	0.165	0.476	0.744	0.734	0.667
(60,16)	0.157	0.107	0.028	0.044	0.688	0.826	0.952	0.945
(80,4)	0.115	0.047	0.124	0.182	0.710	0.825	0.669	0.517
(80,8)	0.280	0.229	0.197	0.154	0.413	0.573	0.585	0.591
(80,16)	0.277	0.173	0.110	0.114	0.545	0.710	0.867	0.875

表 8 RLSOA 与 INSGA 的 C-metric 值结果

Table 8 Results of C-metric values for RLSOA and INSGA

(n, m)	C(SOA, INSGA)				C(INSOA, SOA)			
	$F=2$	$F=3$	$F=4$	$F=5$	$F=2$	$F=3$	$F=4$	$F=5$
(20,4)	0.076	0.046	0.047	0.190	0.873	0.921	0.886	0.714
(20,8)	0.104	0.099	0.090	0.123	0.808	0.838	0.843	0.819
(20,16)	0.218	0.195	0.166	0.125	0.725	0.736	0.746	0.821
(40,4)	0.136	0.147	0.065	0.074	0.776	0.795	0.895	0.898
(40,8)	0.171	0.115	0.137	0.105	0.738	0.808	0.780	0.801
(40,16)	0.231	0.148	0.192	0.196	0.765	0.771	0.748	0.704
(60,4)	0.195	0.124	0.088	0.154	0.602	0.805	0.833	0.822
(60,8)	0.154	0.135	0.109	0.143	0.657	0.699	0.743	0.746
(60,16)	0.198	0.178	0.194	0.204	0.654	0.597	0.583	0.633
(80,4)	0.261	0.189	0.136	0.114	0.548	0.660	0.744	0.824
(80,8)	0.170	0.098	0.102	0.108	0.608	0.684	0.693	0.714
(80,16)	0.194	0.163	0.140	0.188	0.609	0.640	0.584	0.583

在保证其中一项优化目标不变的情况下, 尽可能地最小化另一项优化目标; 另一类则是只考虑其中一项优化目标尽可能地最小化。两种策略分别应用在种群更新和局部搜索中, 可以有效地扩大搜索范围, 在全局搜索的同时保证在精英解附近的搜索深度。RLSOA 相较于对比算法引入了强化学习中的 Q-表 9 RLSOA、KCA 与 INSGA 的 IGD 值计算结果($F=2, F=3$)

(n, m)	$F=2$			$F=3$		
	RLSOA	KCA	INSOA	RLSOA	KCA	INSOA
(20,4)	0.001	0.064	0.037	0.001	0.083	0.045
(20,8)	0.002	0.064	0.047	0.002	0.089	0.052
(20,16)	0.004	0.068	0.048	0.004	0.072	0.054
(40,4)	0.003	0.055	0.045	0.002	0.058	0.055
(40,8)	0.006	0.044	0.052	0.002	0.066	0.067
(40,16)	0.007	0.056	0.058	0.002	0.058	0.053
(60,4)	0.005	0.06	0.033	0.003	0.047	0.047
(60,8)	0.007	0.045	0.056	0.003	0.035	0.042
(60,16)	0.007	0.044	0.058	0.005	0.038	0.042
(80,4)	0.01	0.055	0.028	0.004	0.043	0.033
(80,8)	0.009	0.042	0.05	0.004	0.036	0.046
(80,16)	0.011	0.045	0.061	0.005	0.036	0.044

表 10 RLSOA、KCA 与 INSGA 的 IGD 值计算结果($F=4, F=5$)

Table 10 Results of IGD values for RLSOA, KCA and INSGA($F=4, F=5$)

(n, m)	$F=4$			$F=5$		
	RLSOA	KCA	INSOA	RLSOA	KCA	INSOA
(20,4)	0.001	0.077	0.133	0.004	0.082	0.025
(20,8)	0.002	0.096	0.052	0.003	0.093	0.045
(20,16)	0.002	0.073	0.054	0.002	0.076	0.056
(40,4)	0.001	0.067	0.053	0.001	0.067	0.048
(40,8)	0.003	0.066	0.058	0.001	0.063	0.049
(40,16)	0.003	0.06	0.056	0.002	0.061	0.047
(60,4)	0.003	0.045	0.051	0.005	0.046	0.051
(60,8)	0.003	0.038	0.043	0.004	0.036	0.047
(60,16)	0.004	0.041	0.036	0.004	0.038	0.032
(80,4)	0.004	0.038	0.045	0.006	0.038	0.052
(80,8)	0.003	0.023	0.034	0.003	0.027	0.036
(80,16)	0.003	0.03	0.029	0.004	0.033	0.029

learning 方法,进行搜索策略的筛选,增大有效搜索的概率,从整体上提高了算法性能。

分别采用 Friedman 和 Wilcoxon 检验法分析 RLSOA 与 KCA、INSGA 两种算法之间的性能差异。表 11 所示的 Friedman 检验结果表明在 95% 的置信区间上,几种算法间存在显著性差异,其中 RLSOA 的秩均值为 1.00,优势最明显。表 12 所示为在 95% 置信区间上采用 Wilcoxon 法验证时,算法两两间的差异,由于两个算法对比组都为正秩且 p 值均小于 0.05,因此,可以认为 RLSOA 对 EEDPFSP 的求解效果均优于 KCA 和 INSGA 两个对比算法。

表 11 Friedman 检验结果
Table 11 Friedman test result

Algorithm	Average of rank	p
RLSOA	1.00	
KCA	2.57	
INSGA	2.43	1.486×10^{-16}

表 12 Wilcoxon 检验结果
Table 12 Friedman test result

Constrastive algorithm	Positive or negative of rank	p
RLSOA-KCA	+	1.622×10^{-9}
RLSOA-INSGA	+	1.618×10^{-9}

4 结束语

本文设计了一种基于 Q-learning 的多目标正弦优化算法,以最小化 EEDPFSP 的总能耗和最大完工时间。算法提出了一种双重 Q 学习策略,通过学习不同的启发式方法综合优化两个目标。在工序调整策略中使用正弦公式调整自适应参数,平衡对解空间的搜索。同时本文还探讨了一种基于速度调整的局部搜索策略,进一步探索非支配解附近的区域。在一系列仿真实验中,所提出的 RLSOA 均取得了最佳结果和卓越性能。在未来的研究中,我们将考虑实际制造系统中时间、奖励和需求的不确定性,以及分布式系统中的配送中心和运输路线,建立更符合实际生产过程的分布式车间调度模型,并设计有效的元启发式或超启发式方法进行求解。

参考文献:

[1] 张振,张丫丫,孙美玲,等. 基于多目标生物地理学优化算法的模糊节能分布式流水车间调度[J]. 华东理工大学学

报(自然科学版), 2024, 50(6): 869-877.

- [2] DAI M, TANG D, GIRET A, *et al.* Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm[J]. *Robotics and Computer-Integrated Manufacturing*, 2013, 5: 418-429.
- [3] CAPÓN-GARCÍA E, BOJARSKI A D, ESPUÑA A, *et al.* Multiobjective optimization of multiproduct batch plants scheduling under environmental and economic concerns[J]. *AIChE Journal*, 2011, 57(10): 2766-2782.
- [4] FANG K, UHAN N A, ZHAO F, *et al.* Flow shop scheduling with peak power consumption constraints[J]. *Springer US*, 2013, 206(1): 115-145.
- [5] Yüksel D, TAGETIREN M F, KANDILLER L, *et al.* An energy-efficient bi-objective no-wait permutation flowshop scheduling problem to minimize total tardiness and total energy consumption[J]. *Computers & Industrial Engineering*, 2020, 145: 106431.
- [6] ZTOP H, TASGETIREN M F, ELIHYI D T, *et al.* An energy-efficient permutation flowshop scheduling problem[J]. *Expert Systems with Applications*, 2020, 150: 113279.
- [7] JIANG E D, WANG L. An improved multi-objective evolutionary algorithm based on decomposition for energy-efficient permutation flow shop scheduling problem with sequence-dependent setup time[J]. *International Journal of Production Research*, 2018, 120: 1-16.
- [8] LI J Q, SANG H Y, HAN Y Y, *et al.* Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions[J]. *Journal of Cleaner Production*, 2018, 181(3): 584-598.
- [9] NADERI B, RUIZ R. The distributed permutation flowshop scheduling problem[J]. *Computers & Operations Research*, 2010, 37(4): 754-768.
- [10] DENG J, WANG L, SHEN J, *et al.* An improved harmony search algorithm for the distributed two machine flow-shop scheduling problem[J]. *Springer Berlin Heidelberg*, 2016, 1: 22-27.
- [11] HAO J H, LI J Q, DU Y, *et al.* Solving distributed hybrid flowshop scheduling problems by a hybrid brain storm optimization algorithm[J]. *IEEE Access*, 2019, 7: 66879-66894.
- [12] YING K C, LIN S W, CHENG C Y, *et al.* Iterated reference greedy algorithm for solving distributed no-idle permutation flowshop scheduling problems[J]. *Computers & Industrial Engineering*, 2017, 110: 413-423.
- [13] COMPANYS R, RIBAS I. Efficient constructive procedures for the distributed blocking flow shop scheduling problem[C]//2015 International Conference on Industrial Engineering and Systems Management (IESM). Seville, Spain: IEEE Xplore, 2016, 1: 92-98.
- [14] YI K C, LIN S W. Minimizing makespan for the distributed hybrid flowshop scheduling problem with multiprocessor tasks[J]. *Expert Systems with Application*, 2018, 3: 1-12.

- [15] AVCI M, AVCI M G, HAMZADAYI A. A branch-and-cut approach for the distributed no-wait flowshop scheduling problem[J]. *Computers & Operations Research*, 2022, 3: 106009.
- [16] WANG J J, WANG L. A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop[J]. *IEEE Transactions on Systems, Man and Cybernetics: Systems*. Piscataway, NJ, USA, 2019, 12: 1-15.
- [17] LI Y Z, PAN Q K, GAO K Z, *et al*. A green scheduling algorithm for the distributed flowshop problem[J]. *Applied Soft Computing*, 2021, 109: 107526.
- [18] WANG Y F. Adaptive job shop scheduling strategy based on weighted Q-learning algorithm[J]. *Journal of Intelligent Manufacturing*, 2020, 31(2): 417-432.
- [19] BOUAZZA W, SALLEZ Y, BELDJILALI B. A distributed approach solving partially flexible job-shop scheduling problem with a Q-learning effect[J]. *Ifac Papersonline*, 2017, 50(1): 15890-15895.
- [20] SAID N, SAMAHA Y, AZAB E, *et al*. An online reinforcement learning approach for solving the dynamic flexible job-shop scheduling problem for multiple products and constraints[C]//International Conference on Computational Science and Computational Intelligence (CSCI 2021). USA: IEEE, 2021.
- [21] LI H R, LI X Y, GAO L. A discrete artificial bee colony algorithm for the distributed heterogeneous no-wait flowshop scheduling problem[J]. *Applied Soft Computing Journal*, 2021, 100: 1-11.
- [22] ZHAO F, DI S, WANG L. A hyperheuristic with q-learning for the multiobjective energy-efficient distributed blocking flow shop scheduling problem[J]. *IEEE Transactions on Cybernetics*, 2022, 10: 3192112.
- [23] 张梓琪, 钱斌, 胡蓉, 等. 基于多维 EDA 算法的低碳分布式装配流水线车间调度 [J]. *控制与决策*, 2022, 37(5): 1367-1377.
- [24] LU C, GAO L, YI J, *et al*. Energy-efficient scheduling of distributed flow shop with heterogeneous factories: A real-world case from automobile industry in China[J]. *IEEE Transactions on Industrial Informatics*, 2020, 17(10): 6687-6696.
- [25] HAN W, DENG Q, GONG G, *et al*. Multi-objective evolutionary algorithms with heuristic decoding for hybrid flow shop scheduling problem with worker constraint[J]. *Expert Systems with Applications*, 2021, 168: 114282.
- [26] 姜一啸, 吉卫喜, 何鑫, 苏璇. 基于改进非支配排序遗传算法的多目标柔性作业车间低碳调度 [J]. *中国机械工程*, 2022, 33(21): 2564-2577.

Sine Optimization Algorithm Based on Reinforcement Learning for Energy-Efficient Distributed Flow Shop Scheduling Problem

LANG Jun¹, YIN Xiaoming², GU Xingsheng¹

(1. Key Laboratory of Smart Manufacturing in Energy Chemical Process, Ministry of Education, East China University of Science and Technology, Shanghai 200237, China; 2. China Tobacco Machinery Technology Center, Shanghai 201206, China)

Abstract: To address the multi-objective optimization problem of minimizing makespan and total energy consumption (TEC) in energy-efficient distributed permutation flow shop scheduling (EEDPFSP), this paper proposes a reinforcement learning-based sine optimization algorithm (RLSOA). The algorithm integrates a dual Q-learning strategy: The bottom-level Q-learning prioritizes accelerating critical tasks to reduce makespan, while the top-level Q-learning adjusts processing speeds of non-critical tasks to lower TEC. An adaptive parameter derived from the sine formula dynamically controls the search scope, and a local search strategy guided by elite solutions balances global exploration and local exploitation. Experiments on 480 instances with varying scales demonstrate that RLSEA outperforms state-of-the-art algorithms (KCA and INSGA) with 23.6% higher C-metric and 41.8% lower IGD on average. Ablation studies reveal that the dual Q-learning and local search contribute 65.3% and 28.7% to solution quality improvement, respectively. Statistical tests (Friedman and Wilcoxon, $p < 0.05$) confirm the superiority of RLSEA. The proposed algorithm provides an efficient tool for energy-efficient scheduling in distributed manufacturing systems.

Key words: energy-efficient; distributed flow shop scheduling; reinforcement learning; meta-heuristic; multi-objective optimization

(责任编辑: 王晓丽)