计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2024,33(8):240−249 [doi: 10.15888/j.cnki.csa.009629]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

改进蒲公英优化算法的有限缓冲区流水车间调度①

李伟铭[1], 杨敬辉[1,2]

[1](上海第二工业大学 计算机与信息工程学院, 上海 201209)
[2](上海第二工业大学 智能制造与控制工程学院, 上海 201209)
通信作者: 杨敬辉, E-mail: jhyang@sspu.edu.cn

摘　要: 针对考虑缓冲区容量限制和机器加工档位的流水车间调度问题 (FSSP_LBMPG), 建立了有限缓冲区绿色流水车间数学规划模型, 模型以最小化最大完工时间和最小化加工能量消耗为目标函数, 将缓冲区容量纳入约束, 通过合理选择机器的加工档位达到协调加工速度和加工能耗的效果. 针对问题模型特点, 提出了一种改进蒲公英优化算法 (IDOA), 算法首先根据调度问题的特点设计了双层实数编码机制表示问题的解, 通过引入一种初始化机制, 提高初始解质量和求解效率. 算法迭代过程中, 设计了实数交叉策略和变邻域搜索策略, 弥补了原始蒲公英算法局部搜索能力较差的缺点, 提高了改进算法的开发能力. 最后通过设计案例上的对比实验, 表明所提改进措施能有效增强算法性能, 也验证了算法的有效性和鲁棒性.

关键词: 流水车间; 缓冲区; 蒲公英优化算法; 加工档位

引用格式: 李伟铭,杨敬辉.改进蒲公英优化算法的有限缓冲区流水车间调度.计算机系统应用,2024,33(8):240–249. http://www.c-s-a.org.cn/1003-3254/9629.html

Limited Buffer Flow Workshop Scheduling Based on Improved Dandelion Optimization Algorithm

LI Wei-Ming[1], YANG Jing-Hui[1,2]

[1](School of Computer and Information Engineering, Shanghai Polytechnic University, Shanghai 201209, China)
[2](School of Intelligent Manufacturing and Control Engineering, Shanghai Polytechnic University, Shanghai 201209, China)

Abstract: To solve the flow shop scheduling problem with limited buffers and machine processing gears (FSSP_LBMPG), this research establishes a mathematical programming model for green flow shops with limited buffers. The model has two objective functions: the minimized values of maximum completion time and processing energy consumption. With buffer capacity as a constraint, the processing speed and energy consumption are coordinated by reasonably selecting machine processing gears. Based on the characteristics of the problem model, an improved dandelion optimization algorithm (IDOA) is proposed. The algorithm first designs a DOA double-layer real-valued encoding mechanism to represent the solution to the problem according to the characteristics of the scheduling problem. By introducing an initialization mechanism, the quality and efficiency of the initial solution are improved. During algorithm iteration, a real-valued crossover strategy and a variable neighborhood search strategy are designed to compensate for the poor local search ability of the original dandelion algorithm and enhance the development capabilities of the improved algorithm. Comparative experiments on designed cases show that the proposed improved algorithm effectively enhances the performance of the original algorithm, thereby verifying the effectiveness and robustness of the improved algorithm.

Key words: flow workshop; buffer; dandelion optimization algorithm; processing gear

① 基金项目: 上海市教委产学研项目 (A30DB1621008B)
收稿时间: 2024-02-21; 修改时间: 2024-03-19; 采用时间: 2024-05-06; csa 在线出版时间: 2024-07-03
CNKI 网络首发时间: 2024-07-08

复杂生产流程车间的优化调度研究对于实现智能制造[1–3]具有重要推动作用. 有限缓冲区流水车间调度问题 (flow shop scheduling problem with limited-buffer, FSSP_LB) 是对传统的置换流车间调度问题 (PFSP) 的一种扩展. 近年来, 已经引起了广泛的关注[4,5].

缓冲区约束通常存在于各种生产行业中, 由于机器中间缓冲区或生产温度的限制, 使得工件加工完成后被阻塞在当前机器上. 在阻塞情况下, 前一台机器会保持当前已完成的作业, 直到下一台机器或缓冲区可用[6,7]. 产品属性的影响也会导致阻塞现象. 例如, 在钢铁制造中, 有两个关键的阶段: 加热和轧制钢板. 铸块在煤坑中加热到很高的温度后, 被送到轧制钢板磨坊轧制. 由于加热设备的停止和开启需要很长时间, 为了提高效率和节省成本, 在设备运行时, 希望加工过程是连续的, 此时的铸块加工过程可以被考虑为缓冲区容量为 0 的流水车间调度问题.

为了解决阻塞约束问题, 许多工业生产工厂已经开始采用优化作业排序策略以提高生产效率. 高效的作业调度算法对于企业提高生产效率有重要的推动作用[3], 针对有限缓冲区流水车间调度问题, Liang 等[8]提出了自适应差分进化算法来解决以最大时间和最大工作延迟为目标的双目标优化问题问题. Bai 等[9]研究了一个具有有限缓冲区的非排列双代理流水车间调度问题, 任务随着时间的推移而释放, 并提出了一种混合粒子群优化 (HPSO) 算法来寻求高质量的解决方案. Kazemi Esfeh 等[10]提出了一种新的数学模型来解决存在中间缓冲区、预算和人力资源等约束的柔性流水车间问题. Janeš 等[11]探讨了有限缓冲条件下混合流水车间中产品批次的订单和批次大小的确定问题, 并开发了一种基于改进的稳态遗传算法的调度方法对问题进行求解并取得了良好的效果. 徐震浩等[12]针对缓冲区空间和时间同时受限的流水车间调度问题, 以最小化完工时间为优化目标, 提出一种改进的离散帝国竞争算法求解.

近年来, 随着能源价格的持续走高, 考虑节能减排的绿色制造浪潮正在全球范围内兴起, Lu 等[13]以最小化完工时间和总能耗 (TEC) 为目标, 针对具有有限缓冲的分布式置换流水车间节能调度问题进行了研究, 并提出了一种基于 Pareto 的协同多目标优化算法进行求解. Jiang 等[14]建立了针对有限缓冲区的混合整数线性规划模型, 该模型考虑最小化总加权延迟和总加工能耗两个目标. 在基于分解的多目标进化算法框架下开发了一种高效的多目标优化算法. 取得了良好的效果. Yaurima-Basaldua 等[15]针对复杂的多阶段、多产品、多机器和批量生产环境, 考虑完成时间和能耗为优化目标, 提出了基于非支配排序遗传算法双目标算法, 实验结果表明, 与实际生产相比, 它可以节省 48% 的生产时间和 47% 的电力消耗. Han 等[16]提出了一种将基于模拟退火算法、Hopfield 神经网络算法与局部调度规则相结合的新方法对缓冲区柔性流水车间进行了研究.

蒲公英优化算法 (dandelion optimization algorithm, DOA) 由 Zhao 等[17]于 2022 年提出, 通过模拟蒲公英种子依靠风长距离飞行的过程对问题进行优化求解, 作者在 CEC2017 测试集上, 将蒲公英优化算法与 9 种先进的元启发式算法进行比较, 对算法的优化精度、稳定性、收敛性和延展性等进行了评估. 实验结果表明, 与已有算法相比, 蒲公英优化算法具有较好的优化能力和较强的鲁棒性. 本文将蒲公英优化算法的应用拓展到车间调度问题当中, 首先建立了以完工时间和完工能耗为目标函数的有限缓冲区流水车间调度模型, 设计了适应于调度问题的双层编码机制表示问题的解. 之后通过引入一种种群初始化策略, 提高初始种群的多样性, 在算法迭代过程中, 基于编码染色体设计了实数交叉策略和变邻域搜索策略, 弥补了原始蒲公英算法局部搜索能力较差的缺点, 提高了改进算法的开发能力. 最后, 通过仿真实验和算法比较, 验证了所提 IDOA 算法的有效性.

1 问题描述与模型

考虑缓冲区容量限制和机器加工档位的流水车间调度问题 (flow shop scheduling problem with limited buffers and machine processing gears, FSSP_LBMPG), 可以描述如下: 有n个工件$\{J_1, J_2, \cdots, J_n\}$, 需在总计$m$台机器$\{M_1, M_2, \cdots, M_m\}$上完成加工, 不同工件的优先级相同. 每个工件都要在 m 台机器完成加工, 即工件的加工顺序以及机器的加工顺序均相同, 表示为$O_i = \{O_{i1}, O_{i2}, \cdots, O_{im}\}$, O_{im}表示工件 i 由加工机器m负责加工的工序, 机器加工过程中, 可以选择不同的加工档位, 加工档位越高, 加工工件的速度越快, 但是相应的, 加工能耗越高. 机器加工工件的过程中不允许换挡, 相邻机器之间存在缓冲区容量限制, 当缓冲区已满时, 机器需要停下来等待, 目标是确定工件在机器上的加工顺

序以及工件在机器上的加工档位, 以减小完成所有工件加工的完工时间和能源消耗. 模型中所用到的符号及其含义如表 1 所示.

表 1 符号及其含义

符号	含义
m	加工机器数量
n	加工工件数量
O_{ij}	工件i在机器j上加工的工序
C_{ij}	工件i在机器j上的加工结束时间
C_{lj}	机器M_j上第l个工件的加工释放时间
T_{ij}	选择最低加工档位(视为1)时O_{ij}的处理时间
B_j	机器M_j和M_{j+1}之间的缓冲区大小
S	可选加工档位集合, $S=\{S_1,S_2,\cdots,S_S\}$
v_{ij}	工件i在机器j上的加工档位, $v_{ij}\in\{S_1,S_2,\cdots,S_S\}$
t_{ij}	工件i在机器j上的加工时间, $t_{ij}=T_{ij}/v_{ij}$
π	调度问题的解, $\pi=\{\pi_1,\pi_2,\cdots,\pi_n\}$
$P_{jv_{ij}}$	当机器j以档位v_{ij}加工工件i时的功率
P_j'	机器j的空闲功率
x_{ilj}	如果工件i位于π中第l个位置且在机器j上加工时为1, 否则为0
y_{ijs}	如果工序O_{ij}被安排以加工档位S_s在机器M_j上加工时为1, 否则为0
$C_{\max}$	总完工时间
E_{cost}	总加工能耗
E_{idle}	总空闲能耗
E	总完工能耗

给出本文问题模型目标函数计算公式如下:

$$\begin{cases} C_{\pi_1 1}=t_{\pi_1 1}=T_{\pi_1 1}/v_{\pi_1 1} \\ C_{\pi_i 1}=C_{\pi_{i-1}1}+T_{\pi_{i-1}1}/v_{\pi_{i-1}1},\ i=2,\cdots,B_1+1 \\ C_{\pi_1 j}=C_{\pi_1(j-1)}+T_{\pi_1 j}/v_{\pi_1 j},\ j=1,\cdots,m \\ C_{\pi_i 1}=\max\{C_{\pi_{i-1}1}+T_{\pi_i 1}/v_{\pi_i 1},C_{\pi_i-B_1-1,2}\},\ i>B_1+1 \\ C_{\pi_i j}=\max\{C_{\pi_{i-1}j},C_{\pi_i(j-1)}\}+T_{\pi_i j}/v_{\pi_i j}, \\ \quad i=2,\cdots,B_j+1,\ j=2,\cdots,m-1 \\ C_{\pi_i j}=\max\{\max\{C_{\pi_{i-1}j},C_{\pi_i(j-1)}\}+T_{\pi_i j}/v_{\pi_i j},C_{\pi_i-B_j-1,j+1}\}, \\ \quad i=2,\cdots B_j+1,\ j=2,\cdots,m-1 \\ C_{\pi_i m}=\max\{C_{\pi_{i-1}m},C_{\pi_i(m-1)}\}+T_{\pi_i m}/v_{\pi_i m},\ i=2,\cdots,n \end{cases} \tag{1}$$

则最大完工时间计算公式为:

$$C_{\max}=C_{\pi_n m} \tag{2}$$

总能耗的计算公式为:

$$\begin{aligned} E=&E_{\text{cost}}+E_{\text{idle}}=\sum_{i=1}^{n}\sum_{j=1}^{m}P_{jv_{ij}}\times T_{ij}/v_{ij} \\ &+\sum_{i=2}^{n}\sum_{j=1}^{m}P_j'\times(C_{\pi_i j}-C_{\pi_{i-1}j}-T_{\pi_i j}/v_{ij}) \end{aligned} \tag{3}$$

s.t.

$$\sum_{j=1}^{m}x_{ilj}=1,\forall i,l \tag{4}$$

$$\sum_{i=1}^{n}x_{ilj}=1,\forall l,j \tag{5}$$

$$\sum_{s=1}^{S}y_{ijs}=1,\forall i,j \tag{6}$$

$$t_{ij}=T_{ij}\times\sum_{s=1}^{S}(y_{ijs}/S_s),\forall i,j \tag{7}$$

$$C_{ij+1}-C_{ij}\geqslant t_{ij},\ j=1,2,\cdots,m-1,\forall i \tag{8}$$

$$C_{lj}-C_{l-1j}\geqslant\sum_{i=1}^{n}(x_{ilj}\times t_{ij}),\ l\geqslant 2,\forall i,j \tag{9}$$

$$x_{ilj}\in\{0,1\},\forall i,l,j \tag{10}$$

$$y_{ijs}\in\{0,1\},\forall i,j,s \tag{11}$$

其中, 式 (1)、式 (2) 为完工时间的计算公式, 式 (3) 为总能耗的计算公式. 式 (4) 阐述了不同机器之间不得协同加工工件的规定. 式 (5) 说明了不同工件不能在同一台机器上同时进行加工的限制. 式 (6) 表明在工件的工序加工过程中, 机器的加工档位是固定的. 式 (7) 表示工件在机器上的实际加工时间的计算方法. 式 (8) 规定了工件的加工工艺路线是固定的. 式 (9) 明确了在同一台机器上, 在同一时刻只能有一个工件进行加工. 式 (10) 和式 (11) 为 0-1 变量约束.

2 改进蒲公英优化算法

2.1 蒲公英算法基本流程

蒲公英算法由 Zhao 等[17]在 2022 年提出, 算法主要模拟蒲公英种子依靠风进行长距离飞行的过程, 成熟之后的蒲公英种子要进行 3 个阶段: 在上升阶段蒲公英种子在风速的作用下上升, 到达一定高度后, 进入下降阶段, 直到最终随机落地, 生长出新的蒲公英. DOA 算法模拟了这个过程并引入了 Brownian 运动和 *Levy* 飞行描述种子的运动轨迹.

(1) 上升阶段

在晴天, 蒲公英在风速的影响下被随机吹到各个位置, 风速越大, 飞得越远, 在这种情况下, 蒲公英个体

的位置更新公式为:

$$x_i^{t+1} = x_i^t + \alpha \times v_x \times v_y \times \ln Y \times (x_{rand}^t - x_i^t) \tag{12}$$

其中, x_i^t为当前种群下第i个个体在第t次迭代中所在的位置, x_{rand}^t为搜索空间内产生的随机数, $\ln Y$表示服从$\mu = 0, \sigma^2 = 1$的对数正态分布, 用于模拟风速的影响, 表达式为:

$$\ln Y = \begin{cases} \dfrac{1}{y\sqrt{2\pi}} \mathrm{e}^{\left[-\frac{1}{2\sigma^2}(\ln y)^2\right]}, y \geqslant 0 \\ 0, \qquad\qquad\qquad y < 0 \end{cases} \tag{13}$$

其中, y表示标准正态分布$N(0,1)$, 式 (12) 中的参数α为自适应参数, 用于调整搜索步长, 计算公式为:

$$\alpha = rand \times \left(\frac{1}{T^2}t^2 - \frac{2}{T}t + 1\right) \tag{14}$$

其中, $rand$为 (0, 1) 之间的随机数, T为算法最大迭代次数, 式 (12) 中的参数v_x, v_y表示蒲公英由于涡流分离作用而产生的升力系数, 计算公式为:

$$v_x = \frac{1}{\mathrm{e}^\theta} \times \cos\theta, v_y = \frac{1}{\mathrm{e}^\theta} \times \sin\theta \tag{15}$$

其中, θ为$[-\pi, \pi]$之间的随机数.

雨天, 蒲公英种子无法上升, 而是在当前位置周围盘旋, 此时的位置更新公式为:

$$\begin{cases} x_i^{t+1} = x_i^t \times k \\ k = 1 - rand \times q \\ q = \dfrac{1}{T^2 - 2T + 1}t^2 - \dfrac{2}{T^2 - 2T + 1}t + 1 + \dfrac{1}{T^2 - 2T + 1} \end{cases} \tag{16}$$

其中, k用于调节蒲公英种子的局部搜索区域. 蒲公英种子在上升阶段的综合位置更新公式为:

$$x_i^{t+1} = \begin{cases} x_i^t + \alpha \times v_x \times v_y \times \ln Y \times (x_{rand}^t - x_i^t), rand < 1.5 \\ x_i^t \times k, \qquad\qquad\qquad\qquad \text{else} \end{cases} \tag{17}$$

其中, $rand$是服从标准正态分布的随机数. 为了使算法全局搜索能力更强, 临界值设为 1.5, 使得蒲公英种群在上升阶段尽可能地遍历整个搜索空间, 提高算法全局搜索能力

(2) 下降阶段

通过模拟蒲公英种子的下降阶段增强算法的勘探能力, 此阶段算法通过引入 Brownian 运动和采用上升阶段后的平均位置信息, 使得种群朝着更有潜力的区域发展, 此阶段个体的位置更新公式为:

$$x'^{t+1}_i = x_i^{t+1} - \alpha \times \beta(x_{\text{mean}}^{t+1} - \alpha \times \beta \times x_i^{t+1}) \tag{18}$$

其中, β表示服从正态分布的 Brownian 运动随机数, x_i^{t+1}为第$t+1$次迭代时, 经过上升阶段的第i个个体的位置, x'^{t+1}_i表示经过下降阶段后个体的新位置, x_{mean}^{t+1}为第$t+1$次迭代时, 种群上升阶段后的平均位置.

(3) 落地阶段

在前两个阶段的基础上, 蒲公英种子随机选择降落地点, 随着迭代的逐步进行, 算法有望收敛到全局最优解. 落地阶段蒲公英个体利用当前精英个体的位置, 在其局部邻域中搜索, 位置更新公式为:

$$x'^{t+1}_i = x_{\text{elite}}^t + Levy(\lambda) \times \alpha \left(x_{\text{elite}}^t - x'^{t+1}_i \times \frac{2t}{T}\right) \tag{19}$$

其中, x_{elite}^t为第t次迭代最优蒲公英种子的位置, $Levy(\lambda)$为$Levy$飞行分布函数, 计算公式如下:

$$Levy(\lambda) = s \times \frac{w \times \sigma}{|r_1|^{\frac{1}{\gamma}}} \tag{20}$$

其中, γ为[0, 2]之间的随机数, $s = 0.01$为固定常数, w, r_1为[0, 1]之间的随机数, σ的计算公式为:

$$\sigma = \left(\frac{\Gamma(1+\gamma) \times \sin\left(\frac{\pi\gamma}{2}\right)}{\Gamma\left(\frac{1+\gamma}{2}\right) \times \gamma \times 2^{\left(\frac{\gamma-1}{2}\right)}}\right)^{\frac{1}{\gamma}} \tag{21}$$

2.2 改进蒲公英优化算法

蒲公英优化算法通过上升、下降、落地 3 大策略, 大大提高了算法的搜索能力, 能够搜索到更多的存在潜在最优解的区域, 具有良好的探索性能. 付出的代价是算法的开发能力较差, 由于算法全局探索和局部开发能力不平衡, 导致算法收敛时间长. 为了提高蒲公英优化算法求解流水车间调度问题时的综合性能, 本文提出一种改进蒲公英优化算法 (improved dandelion optimization algorithm, IDOA), 改进算法首先建立一种双层编码机制, 分别用来求解工件排序子问题和档位选择子问题. 采用对数函数计算方法以统一完工时间及加工能耗之间的量纲. 引入初始化机制, 提高初始种群的多样性, 提高算法的全局搜索能力. 种群迭代过程中, 在原有蒲公英算法上升、下降、落地 3 大搜索策略的基础上, 设计了基于机器档位选择编码向量的实数交叉策略与结合交换、插入、逆序 3 种邻域结构的变邻域搜索策略, 弥补了原始蒲公英算法局部搜索能力较差的缺点, 提高了改进算法的开发能力.

2.2.1 编码与解码

本文建立的考虑缓冲区容量限制的流水车间调度问题需要确定工件在机器上的加工顺序与机器选择的加工档位 2 个子问题. 为此, 本文建立一种双层编码机制, 每个解决方案都由 2 个部分组成, 分别是工件排序编码向量和机器档位选择编码向量. 这 2 个向量都是由随机生成的 0–1 之间的数字组成, 工件排序向量的长度与车间需加工的总工件数相等, 机器档位选择向量长度为$n\times m$.

在对编码向量进行解码计算目标函数值时, 首先通过 ROV 转换将工件排序编码向量从实数域转换到整数域, 机器档位选择编码向量通过计算向量值乘以机器可选档位数再向上取整, 计算得到的值即为对应档位中选择的加工档位, 需要注意的是, 本文设计的机器档位编码按照工件顺序进行编码而与工件排序编码无关. 如图 1 所示为一个拥有 5 个工件, 5 台加工机器, 每台加工机器具备 3 个加工档位[1, 1.2, 1.4]的案例示意图, ⌐表示向上取整. 通过计算得到工件 5 在机器 1–5 上选择的加工档位分别为 1, 1.4, 1.2, 1.4, 1.

图 1 编码解码示意图

如图 2 所示为在不同缓冲区约束条件下, 使用主动调度的解码方式, 对图 1 中的编码向量进行解码之后的甘特图示意图, 图 2(a) 为缓冲区容量限制为 0 时, 解码得到调度方案的完工时间值为 27.5, 图 2(b) 为缓冲区容量限制为无限大时, 解码得到调度方案的完工时间值为 25.25, 缓冲区容量越小, 机器在加工工件时为避免违反缓冲区容量约束, 工件在机器上的加工完成时间可能越晚, 从而影响最终的加工完成时间.

2.2.2 适应度函数

为比较不同解决方案之间的优劣关系, 采用对数函数统一能耗和完工时间之间的量纲, 并通过归一化方法将多目标调度问题转换为单目标调度问题, 适应度函数的计算公式为:

$$f(X)=\lambda_1\times\lg C+\lambda_2\times\lg E \tag{22}$$

其中, λ_1,λ_2 为适应度函数中时间和能耗的占比且 $\lambda_1+\lambda_2=1$.

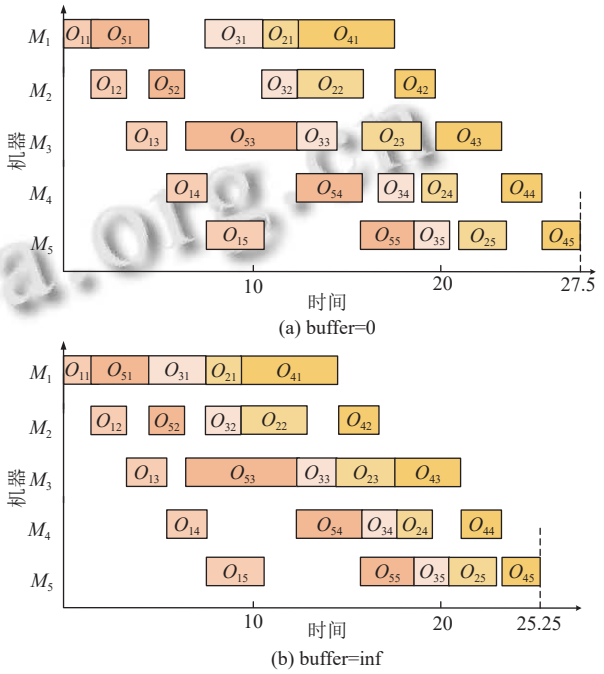


图 2 不同缓冲区解码示意图

2.2.3 种群初始化

为了提高初始种群的多样性从而增强算法的全局探索能力, 本文采用随机初始化的方法产生工件排序向量编码, 对于机器档位编码, 考虑降低能耗与降低完工时间的影响, 机器档位编码种群中 1/10 的个体选择最高的加工档位进行加工以降低解决方案的完工时间, 最大化机器利用率, 属于用能耗换加工效率. 1/10 的初始种群个体选择最低的加工档位以降低能耗的影响, 通过延长加工时间, 降低生产能耗. 其余种群中的初始个体采用随机初始化的方法产生机器档位选择编码种群. 通过以上方式, 既达到了提高初始解质量的目的, 在对机器加工档位选择编码向量进行实数交叉变异策略之时, 能够进行更加充分地搜索, 增强算法搜索能力.

2.2.4 实数交叉策略

本文设计的编码方案包括工序排序编码向量和机器加工档位选择编码向量两部分, 针对工序排序编码向量, 原蒲公英优化算法利用上升、下降、落地 3 大搜索策略, 对工序排序编码向量进行充分的搜索, 然而

在这个阶段机器加工档位选择编码向量并没有参与到搜索过程当中, 因此本文针对机器档位选择编码向量, 设计了一种基于实数编码的交叉策略, 选择不同的机器加工档位可能会提升解的质量, 从而在蒲公英种群迭代过程中, 使算法对解空间进行更高效搜索, 提高算法的探索能力, 搜索到更多的可能存在最优解的区域.

本文基于实数对机器档位选择编码向量进行交叉产生实数子代, 从而在算法迭代过程中, 产生的子代向量可以继续参与到原蒲公英优化算法的进化策略当中, 而不需要再将 ROV 值转换为蒲公英优化算法进化过程中需要的实数值, 基于实数的交叉策略的具体步骤为: 随机生成一个与机器档位选择编码向量长度相等的只包含整数 0, 1 值的数组R, 之后交换父代机器档位选择编码M_1, M_2中数组R中值为 1 的位置上的基因值. 产生子代机器选择编码, 如图 3 所示.

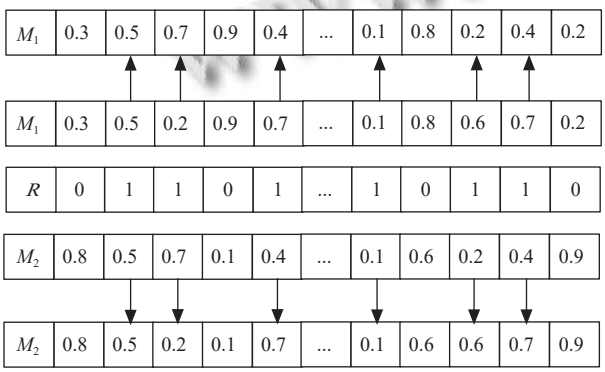


图 3 实数交叉示意图

2.2.5 变邻域搜索策略

为进一步提高算法的开发能力, 本文在基于机器档位选择编码向量的实数交叉策略的基础上, 针对工件排序编码向量, 进一步提出一种基于实数编码的变邻域搜索策略以提高解的质量, 增强改进算法的局部搜索能力. 邻域策略包括: (1) 交换邻域, 随机产生工件排序编码向量长度范围内的两个不相等的正整数, 交换对应整数位置上的值. (2) 插入邻域, 随机产生工件排序编码向量长度范围内的两个不相等的正整数, 将第 1 个整数位置上的值插入到第 2 个整数位置后方. (3) 逆序邻域, 将两个整数中间位置上的值逆序. 通过变邻域搜索操作, 算法在潜在最优解区域周围能够进行更加充分的搜索, 从而大大提升了算法的开发能力. 邻域结构示意图如图 4 所示.

2.2.6 算法基本流程

改进蒲公英优化算法的基本流程如图 5 所示.

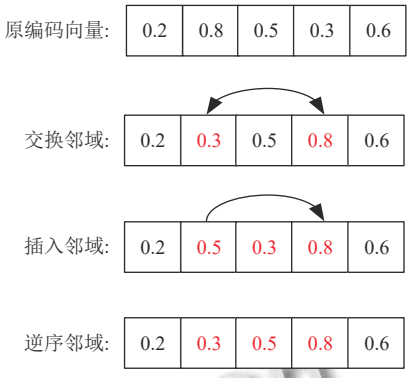


图 4 变邻域搜索示意图

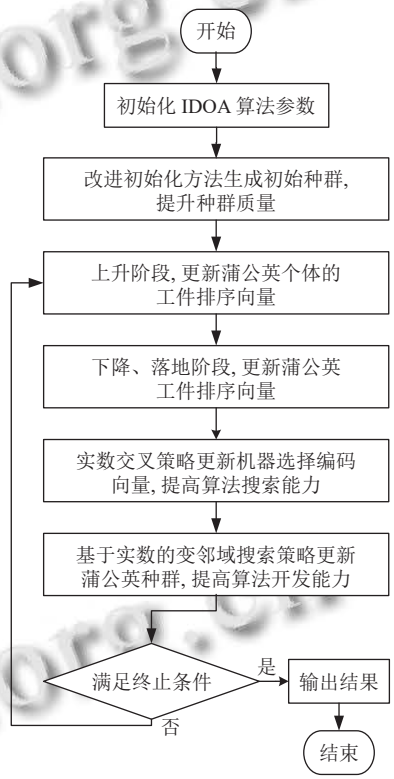


图 5 改进蒲公英优化算法流程图

3 仿真实验

为了验证算法性能, 本文基于典型的 10 个流水车间调度问题设计测试算例, 即 Reeves 系列测试集 rec01, rec03, ⋯, rec19. 参考文献[18]中有关机器加工档位的设定, 本文中的机器加工档位值设置为 $S=\{S_1, S_2, S_3\}=\{1, 1.2, 1.4\}$, 机器加工功率$P_{jv_{ij}}=4v_{ij}^2$, 机器空闲功率设为固定值 1[19], 不同机器间的缓冲区容量设为相同的固定值. 算法使用 Matlab R2021b 编程, 在 i7 处理器 3.3 GHz, 16 GB 内存的环境下运行.

3.1 改进蒲公英算法 (IDOA) 与 DOA 对比

为了验证本文所提改进策略对于算法效果的提升,

采用缓冲区等于 1 的测试集对 IDOA 和 DOA 进行对比, 同时为了降低随机性的影响, 算法在每个算例上均运行 10 次, 使用 Best 表示算法在不同算例下 10 次求解结果中, 适应度函数的最优值. Avg 表示算法在不同算例下 10 次求解结果的平均值. 算法种群规模设为 100, IDOA 实数交叉概率设为 0.8, 如表 2 所示为算法在不同算例下分别运行 20 代、50 代、100 代后适应度函数的指标值. 从表 2 可以看出, 在所有 4 类规模的 10 个测试算例上, 本文的改进蒲公英优化算法在最优值指标和平均值指标上均取得了较原始蒲公英优化算法更好的解, 这是因为 DOA 算法虽然具有良好的勘探能力, 但当 DOA 算法寻找到潜在最优解周围区域时, 由于缺少良好的局部开发能力, 导致算法在潜在最优解周围收敛速度很慢, 而本文提出的 IDOA 算法, 当寻找到潜在最优解周围时, 通过改进的实数交叉策略和变邻域搜索策略, 改进算法可以进行更加充分且有效的有针对性的搜索, 大大提高了算法的局部搜索能力, 增强了算法的性能.

从算法收敛性能看, 如图 6 所示为 DOA 算法和 IDOA 算法在 4 个算例上运行的最优值收敛曲线图, 从图 6 中可以看出在刚开始迭代时, IDOA 最优解的质量明显优于 DOA 算法, 证明了本文提出的初始化机制对于提高初始种群质量的有效性, 随着算法的迭代运行, DOA 算法虽然比 IDOA 算法收敛速度更快, 但是寻找到的最优解的质量要明显劣于 IDOA 算法, 侧面印证了原始 DOA 算法开发能力较弱的问题, 从而导致算法过早收敛于局部最优解.

表 2 IDOA 与 DOA 的对比结果

问题	规模	迭代次数											
		20				50				100			
		IDOA		DOA		IDOA		DOA		IDOA		DOA	
		Best	Avg	Best	Avg	Best	Avg	Best	Avg	Best	Avg	Best	Avg
rec01	20×5	**3.7195**	**3.7250**	3.7305	3.7350	**3.7178**	**3.7196**	3.7267	3.7331	**3.7178**	**3.7195**	3.7267	3.7325
rec03	20×5	**3.6642**	**3.6671**	3.6742	3.6807	**3.6560**	**3.6602**	3.6718	3.6767	**3.6558**	**3.6595**	3.6718	3.6761
rec05	20×5	**3.7080**	**3.7088**	3.7166	3.7198	**3.7020**	**3.7045**	3.7121	3.7161	**3.7020**	**3.7045**	3.7112	3.7157
rec07	20×10	**3.9113**	**3.9144**	3.9165	3.9237	**3.9046**	**3.9079**	3.9159	3.9222	**3.9043**	**3.9072**	3.9159	3.9218
rec09	20×10	**3.9022**	**3.9039**	3.9091	3.9157	**3.8936**	**3.8963**	3.9050	3.9117	**3.8917**	**3.8960**	3.9050	3.9111
rec11	20×10	**3.8740**	**3.8827**	3.8940	3.8966	**3.8716**	**3.8781**	3.8887	3.8933	**3.8716**	**3.8780**	3.8887	3.8917
rec13	20×15	**4.0567**	**4.0581**	4.0689	4.0718	**4.0525**	**4.0544**	4.0608	4.0652	**4.0524**	**4.0542**	4.0602	4.0647
rec15	20×15	**4.0505**	**4.0552**	4.0595	4.0666	**4.0464**	**4.0513**	4.0595	4.0635	**4.0464**	**4.0511**	4.0595	4.0632
rec17	20×15	**4.0475**	**4.0514**	4.0611	4.0649	**4.0441**	**4.0463**	4.0558	4.0610	**4.0438**	**4.0462**	4.0558	4.0598
rec19	30×10	**4.0755**	**4.0792**	4.0851	4.0919	**4.0680**	**4.0712**	4.0795	4.0860	**4.0665**	**4.0694**	4.0765	4.0840

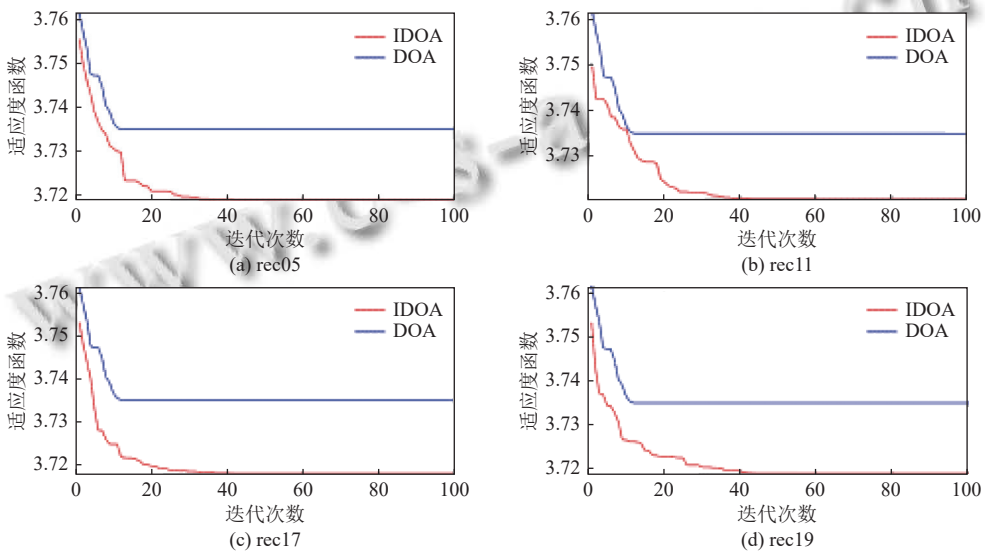


图 6 IDOA 与 DOA 收敛曲线图

3.2 IDOA 与其他优化算法的对比

为验证算法性能, 将 IDOA 算法与广泛应用于车间调度问题的遗传算法 (GA)、模拟退火算法 (SA) 以及双层变异迭代贪婪算法 (IGDLM)[20]进行比较. GA、SA、IDOA 算法的种群规模设为 100, 迭代次数设为 100, 遗传算法的交叉概率为 0.8, 变异概率设为 0.1. 模

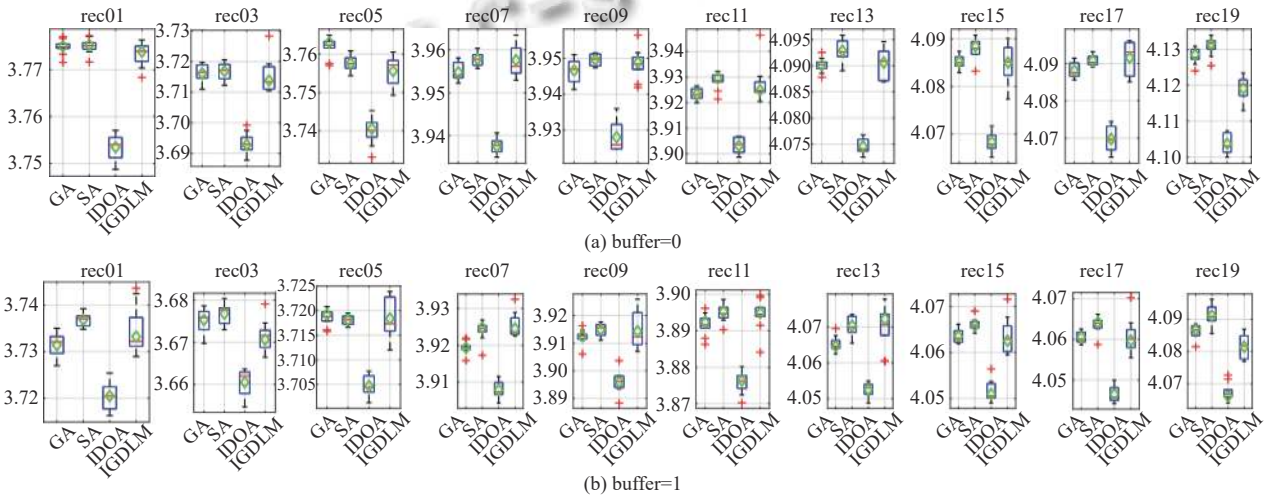
拟退火算法初始温度设为 50, 最低温度设为 1E–6, 降温速率设为 0.99. IGDLM 算法参考文献[20]进行参数取值. 同样进行 10 次重复实验以降低随机性的影响, 算法对比结果如表 3 所示.

表 3 IDOA、GA、SA、IGDLM 在不同缓冲区容量下的对比结果

缓冲区大小	算法	指标	算例									
			rec01	rec03	rec05	rec07	rec09	rec11	rec13	rec15	rec17	rec19
buffer=0	IDOA	Best	3.7488	3.6878	3.7331	3.9350	3.9247	3.8987	4.0725	4.0652	4.0650	4.0999
	GA		3.7715	3.7109	3.7570	3.9522	3.9413	3.9200	4.0878	4.0829	4.0856	4.1241
	SA		3.7716	3.7122	3.7544	3.9557	3.9474	3.9214	4.0890	4.0832	4.0891	4.1255
	IGDLM		3.7683	3.7104	3.7493	3.9529	3.9419	3.9204	4.0869	4.0773	4.0851	4.1130
	IDOA	Avg	3.7535	3.6935	3.7398	3.9375	3.9280	3.9031	4.0746	4.0683	4.0698	4.1036
	GA		3.7748	3.7161	3.7619	3.9548	3.9466	3.9234	4.0901	4.0852	4.0881	4.1281
	SA		3.7750	3.7169	3.7576	3.9579	3.9496	3.9282	4.0929	4.0879	4.0908	4.1310
	IGDLM		3.7733	3.7152	3.7556	3.9576	3.9483	3.9270	4.0906	4.0850	4.0915	4.1192
buffer=1	IDOA	Best	3.7163	3.6547	3.7012	3.9043	3.8883	3.8702	4.0488	4.0490	4.0441	4.0640
	GA		3.7270	3.6698	3.7158	3.9159	3.9061	3.8863	4.0625	4.0619	4.0586	4.0815
	SA		3.7348	3.6730	3.7166	3.9173	3.9111	3.8903	4.0655	4.0642	4.0587	4.0856
	IGDLM		3.7289	3.6665	3.7120	3.9225	3.9070	3.8841	4.0605	4.0595	4.0555	4.0769
	IDOA	Avg	3.7205	3.6605	3.7047	3.9079	3.8960	3.8757	4.0526	4.0515	4.0465	4.0674
	GA		3.7314	3.6753	3.7186	3.9195	3.9124	3.8917	4.0652	4.0635	4.0605	4.0863
	SA		3.7369	3.6768	3.7181	3.9240	3.9149	3.8948	4.0704	4.0661	4.0636	4.0911
	IGDLM		3.7343	3.6715	3.7184	3.9254	3.9143	3.8944	4.0699	4.0635	4.0608	4.0815
buffer=2	IDOA	Best	3.7136	3.6481	3.6937	3.9012	3.8849	3.8680	4.0491	4.0470	4.0393	4.0642
	GA		3.7249	3.6606	3.7057	3.9140	3.9025	3.8843	4.0612	4.0581	4.0545	4.0790
	SA		3.7295	3.6659	3.7085	3.9203	3.9066	3.8882	4.0672	4.0622	4.0596	4.0807
	IGDLM		3.7240	3.6588	3.7079	3.9175	3.9042	3.8865	4.0629	4.0561	4.0541	4.0716
	IDOA	Avg	3.7173	3.6535	3.6972	3.9058	3.8894	3.8746	4.0516	4.0504	4.0437	4.0669
	GA		3.7271	3.6643	3.7081	3.9172	3.9074	3.8873	4.0639	4.0613	4.0561	4.0810
	SA		3.7334	3.6684	3.7111	3.9231	3.9115	3.8922	4.0691	4.0641	4.0616	4.0838
	IGDLM		3.7298	3.6655	3.7152	3.9243	3.9097	3.8903	4.0674	4.0622	4.0599	4.0771

从表 3 可以看出, 本文所提的 IDOA 在对比结果中的优化效果明显优于遗传算法、模拟退火算法和双层变异迭代贪婪算法, 在所有的测试算例上, 不论是求解的最优值, 还是 10 次求解结果的平均值, IDOA 均取得了比对比算法更优的结果. 双层变异迭代贪婪算法虽然在求解本文的考虑机器档位和缓冲区容量的流水车间调度问题时效果要略优于模拟退火算法和遗传算法, 但是由于缺乏良好的勘探策略, 影响了算法整体的求解效果.

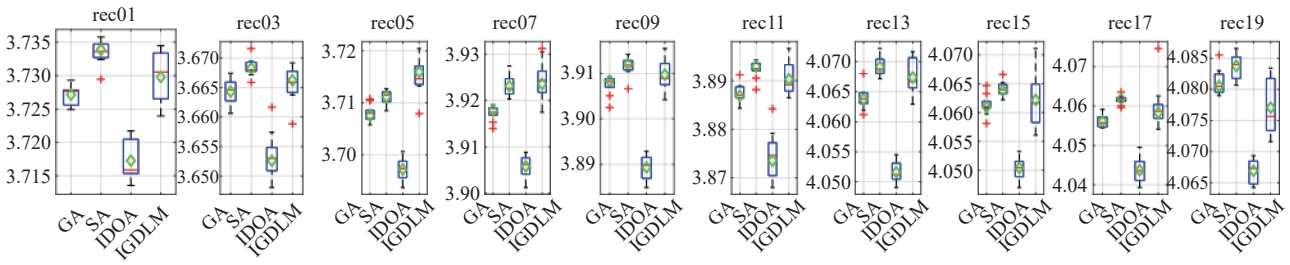
算法收敛性对比如图 7–图 9. 图 7 为不同缓冲区容量下的算法对比箱线图, 绿色菱形表示 10 次求解结果的平均值.



(a) buffer=0

(b) buffer=1

图 7 不同缓冲区下算法对比箱线图



(c) buffer=2

图 7　不同缓冲区下算法对比箱线图 (续)

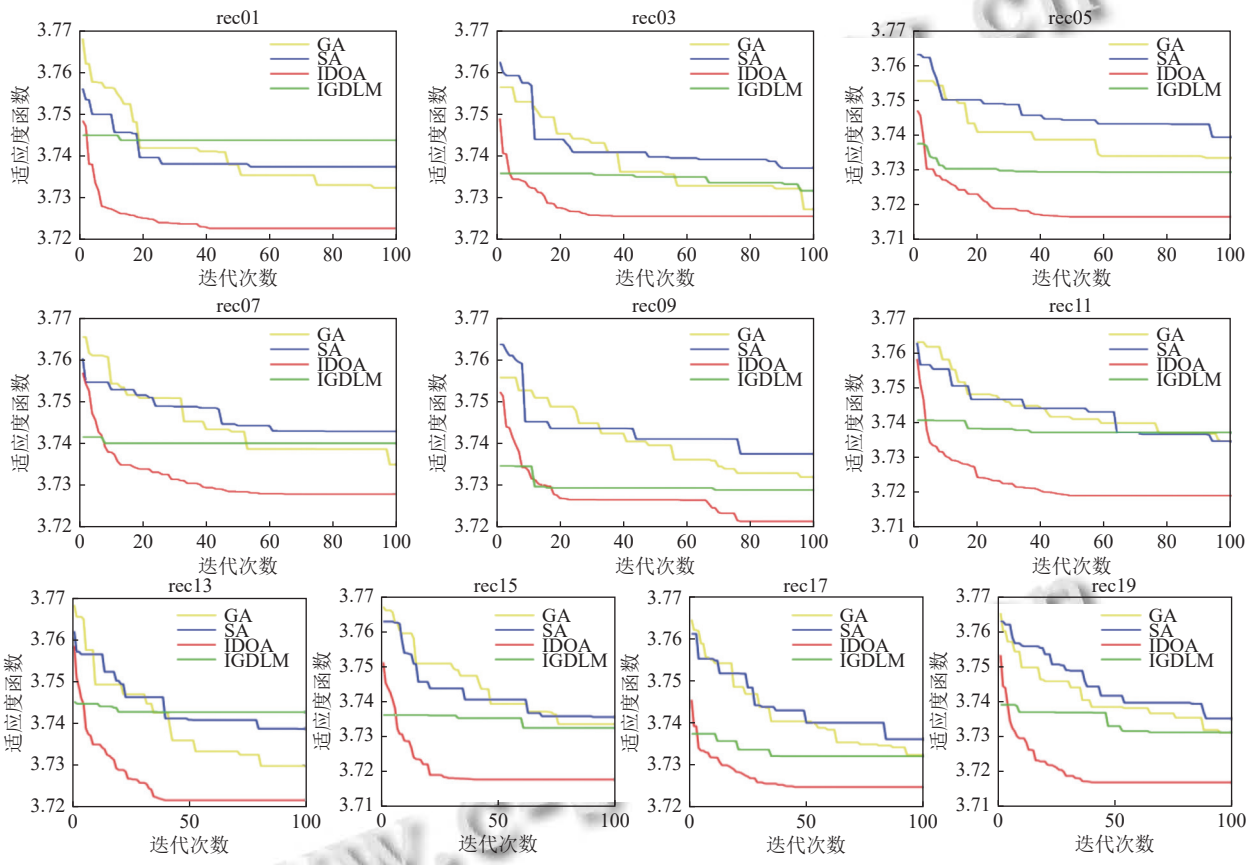


图 8　收敛曲线对比

图 8 所示为算法在缓冲区容量为 1 时运行的最优值收敛曲线图, 从图 8 中可以看出, 在求解稳定性上, IDOA 算法性能要差于对比算法, 但 IDOA 算法在求解质量上要远优于对比的遗传算法、模拟退火算法与双层变异迭代贪婪算法. 验证了 IDOA 算法在求解本文提出的 FSSP_LBMPG 问题时的有效性. 如图 9 所示为 IDOA 算法在 rec11 算例下求得的最优结果的甘特图, 算法求得的工序排序结果为[14 16 9 18 19 1 12 10 7 20 4 13 2 8 3 5 11 15 17 6], 甘特图中矩阵上的数字表示工件选择的机器加工档位.

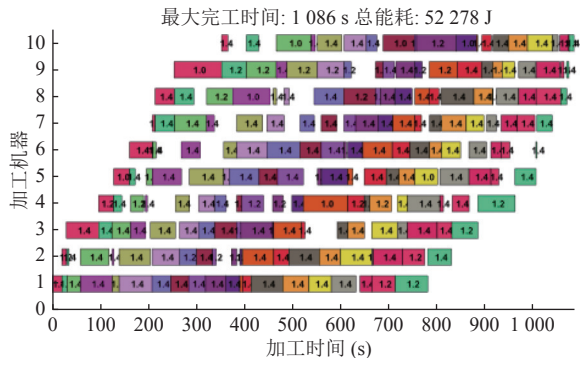


图 9　甘特图

4 结语

通过考虑机器加工过程中加工档位和加工时间、加工能耗之间的关系, 本文建立了以最小化完工时间和最小化能量消耗为目标函数的考虑缓冲区容量限制和机器加工档位的流水车间调度模型, 并提出一种改进蒲公英优化算法对模型进行求解, 改进算法在原有蒲公英优化算法上升、下降、落地 3 大进化策略的基础上, 进一步提出了一种种群初始化机制以提高初始种群的质量, 加快搜索速度. 算法迭代过程中, 提出了针对机器档位编码的实数交叉策略以及变邻域搜索策略, 弥补了原始 DOA 算法局部搜索能力差, 开发性能疲弱的缺点, 大大提高了算法的性能. 最后通过在测试算例上将 IDOA 与 DOA 算法, IDOA 与 GA、SA、IGDLM 算法进行对比分析验证了 IDOA 算法在求解本文所提出的问题时的有效性和鲁棒性.

参考文献

1 Qin M, Wang RS, Shi ZS, *et al*. A genetic programming-based scheduling approach for hybrid flow shop with a batch processor and waiting time constraint. IEEE Transactions on Automation Science and Engineering, 2021, 18(1): 94–105. [doi: 10.1109/TASE.2019.2947398]

2 Meng T, Pan QK, Wang L. A distributed permutation flowshop scheduling problem with the customer order constraint. Knowledge-based Systems, 2019, 184: 104894. [doi: 10.1016/j.knosys.2019.104894]

3 Qin HX, Han YY, Wang YT, *et al*. Intelligent optimization under blocking constraints: A novel iterated greedy algorithm for the hybrid flow shop group scheduling problem. Knowledge-based Systems, 2022, 258: 109962. [doi: 10.1016/j.knosys.2022.109962]

4 轩华, 郑倩倩, 李冰. 带不相关并行机的阻塞 FFP 的混合遗传算法. 计算机工程与设计, 2021, 42(4): 949–956. [doi: 10.16208/j.issn1000-7024.2021.04.008]

5 温廷新, 关婷誉. 考虑能耗和运输的有限缓冲区混合流水车间调度. 系统仿真学报, 2024, 36(6): 1344–1358. [doi: 10.16182/j.issn1004731x.joss.23-0343]

6 Han X, Han YY, Zhang B, *et al*. An effective iterative greedy algorithm for distributed blocking flowshop scheduling problem with balanced energy costs criterion. Applied Soft Computing, 2022, 129: 109502. [doi: 10.1016/j.asoc.2022.109502]

7 Aqil S, Allali K. Two efficient nature inspired meta-heuristics solving blocking hybrid flow shop manufacturing problem. Engineering Applications of Artificial Intelligence, 2021, 100: 104196. [doi: 10.1016/j.engappai.2021.104196]

8 Liang J, Wang P, Guo L, *et al*. Multi-objective flow shop scheduling with limited buffers using hybrid self-adaptive differential evolution. Memetic Computing, 2019, 11(4): 407–422. [doi: 10.1007/s12293-019-00290-5]

9 Bai DY, Liu TY, Zhang YC, *et al*. Scheduling non-permutation flowshop with finite buffers and two competitive agents. Computers & Industrial Engineering, 2023, 177: 108939.

10 Kazemi Esfeh M, Shojaie AA, Javanshir H, *et al*. Flexible flow shop scheduling problem with reliable transporters and intermediate limited buffers via considering learning effects and budget constraint. Complexity, 2022, 2022: 1253336.

11 Janeš G, Ištoković D, Jurković Z, *et al*. Application of modified steady-state genetic algorithm for batch sizing and scheduling problem with limited buffers. Applied Sciences, 2022, 12(22): 11512. [doi: 10.3390/app122211512]

12 徐震浩, 王程, 顾幸生. 基于 DICA 的存储受限流水车间调度. 华东理工大学学报 (自然科学版), 2018, 44(4): 563–572. [doi: 10.14135/j.cnki.1006-3080.20170627003]

13 Lu C, Huang YX, Meng LL, *et al*. A Pareto-based collaborative multi-objective optimization algorithm for energy-efficient scheduling of distributed permutation flow-shop with limited buffers. Robotics and Computer-integrated Manufacturing, 2022, 74: 102277. [doi: 10.1016/j.rcim.2021.102277]

14 Jiang SL, Zhang L. Energy-oriented scheduling for hybrid flow shop with limited buffers through efficient multi-objective optimization. IEEE Access, 2019, 7: 34477–34487. [doi: 10.1109/ACCESS.2019.2904848]

15 Yaurima-Basaldua VH, Tchernykh A, Villalobos-Rodríguez F, *et al*. Hybrid flow shop with unrelated machines, setup time, and work in progress buffers for bi-objective optimization of tortilla manufacturing. Algorithms, 2018, 11(5): 68. [doi: 10.3390/a11050068]

16 Han ZH, Han C, Lin S, *et al*. Flexible flow shop scheduling method with public buffer. Processes, 2019, 7(10): 681. [doi: 10.3390/pr7100681]

17 Zhao SJ, Zhang TR, Ma SL, *et al*. Dandelion optimizer: A nature-inspired metaheuristic algorithm for engineering applications. Engineering Applications of Artificial Intelligence, 2022, 114: 105075. [doi: 10.1016/j.engappai.2022.105075]

18 胡蓉, 董钰明, 钱斌. 基于探路者算法的绿色有限缓冲区流水线调度. 系统仿真学报, 2021, 33(6): 1384–1396. [doi: 10.16182/j.issn1004731x.joss.20-0077]

19 Lei DM, Zheng YL, Guo XP. A shuffled frog-leaping algorithm for flexible job shop scheduling with the consideration of energy consumption. International Journal of Production Research, 2017, 55(11): 3126–3140. [doi: 10.1080/00207543.2016.1262082]

20 秦浩翔, 韩玉艳, 陈庆达, 等. 求解阻塞混合流水车间调度的双层变异迭代贪婪算法. 控制与决策, 2022, 37(9): 2323–2332. [doi: 10.13195/j.kzyjc.2021.0607]

(校对责编: 张重毅)