考虑预保养的多时间约束柔性作业车间绿色调度

王玉芳[1,2,3], 郑明波[1], 张　艺[1], 华晓麟[1†], 章殿清[1]

(1. 南京信息工程大学 自动化学院, 江苏 南京 210044; 2. 南京信息工程大学 大气环境与装备技术协同创新中心, 江苏 南京 210044;
3. 南京信息工程大学 气象能源利用与控制工程技术研究中心, 江苏 南京 210044)

摘要: 考虑到绿色生产中模具切换、工件运输以及加工负载对机器能耗的影响, 将模具设置时间、工件运输时间、可变机器能耗以及预保养机制纳入调度问题模型. 建立以最小化完工时间、能耗和总噪声为目标的柔性作业车间绿色调度模型, 并提出基于目标平衡的Jaya算法进行求解. 考虑多目标优化初始种群质量对进化效率的影响, 设计一种混合初始化策略, 加强初始种群目标间的平衡, 加快算法的收敛速度. 为所有非最优个体设计一种多样化的Jaya优化算子, 增加种群多样性, 提高算法的全局搜索能力. 为了避免算法陷入局部最优, 提高解的分布性, 设计了3种基于目标平衡的局部搜索策略. 通过消融实验验证所提策略的有效性, 与其他算法的实验对比结果验证了所提算法的优越性.

关键词: 柔性作业车间绿色调度; 多时间约束; 能耗可变; 预保养; 多目标优化

引用格式: 王玉芳, 郑明波, 张艺, 等. 考虑预保养的多时间约束柔性作业车间绿色调度. 控制理论与应用, 2025, 42(12): 2598 – 2612

DOI: 10.7641/CTA.2025.50183

Flexible job-shop green scheduling with multiple time constraints considering pre-maintenance

WANG Yu-fang[1,2,3], ZHENG Ming-bo[1], ZHANG Yi[1], HUA Xiao-lin[1†], ZHANG Dian-qing[1]

(1. School of Automation, Nanjing University of Information Science & Technology, Nanjing Jiangsu 210044, China;
2. Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET),
Nanjing University of Information Science & Technology, Nanjing Jiangsu 210044, China;
3. Engineering Research Center on Meteorological Energy Using and Control (C-MEIC),
Nanjing University of Information Science & Technology, Nanjing Jiangsu 210044, China)

Abstract: Considering the requirements for mold switching, job transportation and the impact of processing load on machine energy consumption in green production, the setting time, transportation time, variable machine energy consumption and the pre-maintenance mechanism are incorporated into the scheduling problem model. A flexible job-shop green scheduling model is established with objectives of minimizing the completion time, energy consumption and total noise. The Jaya algorithm based on objective balance is proposed to solve it. Considering the influence of the initial population quality on the evolutionary efficiency, a hybrid initialization strategy is designed, which enhances the objective balance of the initial population and accelerates the convergence speed of the algorithm. For all non-optimal individuals, a diversified Jaya optimization operator is put forward to increase population diversity and global search capability. To prevent the algorithm from getting stuck in a local optimum and enhance the distribution of the solutions, three local search strategies based on objective balance are designed. The effectiveness of the proposed strategies is verified through ablation experiments. The experimental comparison results with four algorithms have demonstrated the superiority of the proposed algorithm.

Key words: flexible job-shop green scheduling; multiple time constraints; variable energy consumption; premaintenance; multiobjective optimization

Citation: WANG Yufang, ZHENG Mingbo, ZHANG Yi, et al. Flexible job-shop green scheduling with multiple time constraints considering pre-maintenance. *Control Theory & Applications*, 2025, 42(12): 2598 – 2612

收稿日期: 2025−04−25; 录用日期: 2025−09−19.

†通信作者. E-mail: 1922163398@qq.com; Tel.: +86 18136262608.

本文责任编委: 李德伟.

国家自然科学基金项目(51705260)资助.

Supported by the National Natural Science Foundation of China (51705260).

1 引言

近年来, 随着全球化的不断发展, 产品更新迭代速度加快, 生产企业之间的竞争愈发激烈. 不同于传统的生产模式, 柔性作业车间调度能满足现代化企业多品种、小批量的生产需求[1], 柔性作业车间调度问题(flexible job-shop scheduling problem, FJSP)已成为各企业亟需求解的问题. 当前, 全球能源短缺和环境问题日益突出, 节能减排和绿色生产已成为现代工业发展的必然趋势[2]. 绿色调度是柔性作业车间实现节能减排和绿色生产的关键环节[3], 生产企业通过合理的调度和优化, 能够最大程度地减少能源和资源的浪费, 降低环境污染. 虽然有许多研究者(如Xu等[4], 罗文冲等[5])采用不同的方法解决绿色调度问题, 但普遍只关注生产车间的加工阶段, 未考虑到运输和设置时间所消耗的成本与能耗.

由于企业生产规模较大, 运输时间的长短直接影响到完工时间和设备的利用率[6]. 忽略运输时间可能导致工序加工的等待时间增加, 造成设备空闲和生产瓶颈, 降低生产效率[7]. 此外,在实际生产中, 由于各工件不同的加工工艺, 机器从加工一种类型的工件转换到其他类型工件前, 需要进行加工模具更换和预热等设置操作, 这些操作不仅会占用机器的有效工作时间, 还可能引发额外的成本和生产延误[8]. 因此, 实际生产调度中的运输时间和设置时间不容忽略, 近年来受到了研究者的广泛关注. Meng等[9]考虑了序列相关的安装时间和运输时间的柔性作业车间调度问题, 建立了以能耗为目标的混合整数线性规划, 通过扩展案例验证了模型的有效性. Zhang等[10]将运输、安装、交付时间等约束与能源消耗相结合, 建立了整数规划模型. Pa1等[11]针对考虑运输和设置时间的FJSP, 提出一种多代理的分解方式求解该问题. 虽然上述研究考虑了生产实际中加工、设置和运输等时间的影响, 但都假设机器单位加工能耗不变. 而在实际生产中, 因为机器长期高负载或超负荷运行会导致一系列连锁反应, 直接影响生产效率和能耗成本, 因此必须考虑机器能耗的可变性[12]. 具体而言, 长时间的加工会造成机器疲劳损伤和塑性变形, 这会影响主轴材料的温度. 温度越高意味着机床运行所需的功率越大[13]. 此外, 机器因持续高负载会加剧机器磨损, 尤其是关键部件如轴承、齿轮的损耗, 增加了运动阻力, 从而需要更多的能量来驱动[14]. 如果不及时采取适当的保养措施,机器的磨损便会不断累积, 不仅浪费更多能耗, 还可能导致机器的严重损坏, 影响生产的连续性和稳定性. 因此, 为了减少持续加工对机器正常运作造成的影响, 有必要通过机器预保养和合理调度减轻过载情况. 综上, 本文对考虑预保养的多时间约束柔性作业车间绿色调度问题(flexible job-shop green scheduling problem with pre-maintenance and multiple temporal constraints, FJGSPPMT)开展研究.

FJSP具有高度复杂性、多目标性和NP难等特性[15], 而FJGSPPMT相比于FJSP, 增加了运输和设置时间, 模型约束更复杂, 且需要考虑机器预保养, 求解的子问题更多. 因此, 求解最优调度方案更困难, 传统的精确算法难以找到最优解. 而启发式进化算法凭借其快速逼近最优解的能力、应对多目标优化的灵活性, 以及通过模拟自然进化实现的自学习特点, 成为解决FJGSPPMT的有效手段. 其中, Jaya算法摆脱了传统启发式算法因参数选择不当而导致性能瓶颈的局限性, 它无需精细调节参数, 且通过简单的“趋优避劣”原则实现种群进化, 兼具了结构的简洁性和实施的高效性. 自Jaya算法由Rao[16]提出以来, 就展现出良好的全局搜索能力, 尤其适合解决FJGSPPMT这类复杂且目标多样的问题. 然而, Jaya算法存在着收敛速度慢且易陷入局部最优等缺陷, 许多研究者针对这些问题对Jaya算法进行了改进. Fan等[17]提出了一种结合禁忌搜索的混合Jaya算法, 通过设计3种不同的关键路径搜索方式, 帮助算法挖掘更优解, 避免陷入局部最优. Caldeirar等[18]针对特定问题提出基于邻域的局部搜索, 提高了搜索空间的利用率. 虽然上述研究针对Jaya算法易陷入局部最优的缺陷设计了局部搜索策略, 但并未对其全局搜索进行进一步优化. 因此, 根据上述研究基础, 本文从局部搜索和全局搜索两方面进行改进, 提出了一种基于目标平衡的Jaya算法(Jaya based on multi-objective balance, MBJaya)求解FJGSPPMT. 首先, 针对传统Jaya算法收敛速度慢的问题, 设计一种混合初始化规则, 提高初始种群的质量并维持其在不同目标之间的平衡, 加快算法收敛速度; 其次, 设计一种多样化Jaya优化算子, 通过多种不同的优化方式、不同的优劣个体引导种群进化, 增加种群多样性, 提高算法的全局搜索能力. 再次, 针对Jaya易陷入局部最优的问题, 设计基于目标平衡的局部搜索策略, 为算法寻求更优解.

2 考虑预保养的多时间约束柔性作业车间绿色调度模型

2.1 问题描述

FJGSPPMT具体描述如下: n个工件在m台独立的机器上进行加工, 每个工件J_i由h_i道工序组成, 每道工序有一或多台机器可以加工; 每道工序的加工时间以及加工前的设置时间因加工机器不同存在差异. 每台机器在累计加工一段时间之后单位加工能耗会增加, 如果在加工一道工序之前进行预保养, 机器的单位能耗恢复初始值. 由于加工所需要的最大完成时间、总能耗以及噪声等都会因工序选择的加工机器的不同而发生改变, 因此, 本文所研究的调度问题的优化任务为: 在满足加工约束条件的前提下, 合理地安

排各工序的加工机器、加工顺序以及加工前的预保养选择, 以使所期望的指标达到最优. FJGSPPMT需要满足假设条件: 1) 同一工件的相邻工序之间需要考虑运输时间; 2) 同一机器上加工的相邻工序需要考虑设置时间, 若为同种工件的相邻工序, 则无设置时间; 3) 在整个生产活动开始之前所有机器预保养一次, 累计负载为零, 单位加工能耗为初始值; 4) 机器设有缓冲区, 存放加工完的工件, 缓冲区容量无限大; 5) 运输资源无限, 且无需等待; 6) 工序加工不可中断; 7) 所有工件均在0时刻可以被加工; 8) 不同工件工序之间相互独立, 同一工件工序之间存在先后约束.

FJGSPPMT模型所使用的符号如表1所示.

表 1 FJGSPPMT 模型符号定义

Table 1 Symbol definition of FJGSPMT model

符号	定义
n	工件总数
m	机器总数
i	工件序号
j	工序序号
k	机器序号
J_i	第i个工件
h_i	J_i的工序数
M_k	第k台机器
O_{ij}	工件i的第j道工序
t_{ijk}^{p}	O_{ij}在M_k上的加工时间
t_{ijk}^{s}	O_{ij}在M_k上的设置时间
t_{ijk}^{tr}	O_{ij}在到达M_k之前的运输时间
t_{ijk}^{pm}	O_{ij}在M_k上的预保养时间
S_{ij}^{pm}	O_{ij}的预保养开始时间
C_{ij}^{pm}	O_{ij}的预保养结束时间
S_{ij}^{st}	O_{ij}设置开始时间
C_{ij}^{st}	O_{ij}设置结束时间
S_{ij}^{pt}	O_{ij}加工开始时间
C_{ij}^{pt}	O_{ij}加工结束时间
S_{ij}^{tr}	O_{ij}运输开始时间
C_{ij}^{tr}	O_{ij}运输结束时间
S_{kl}^{M}	M_k上的第l道工序加工开始时间
C_{kl}^{M}	M_k上的第l道工序加工结束时间
C_i	J_i完成时间
pe_{kt}	M_k在t时刻的单位加工能耗
pe_k	M_k的初始单位加工能耗
se_k	M_k的单位设置能耗
te	机器之间的单位时间运输能耗
ie_k	M_k的单位空载能耗
it_k	M_k的总空载时间
pm_k	M_k的预保养时间
sumpt_{kt}	M_k在t时刻的累计加工负载
L	一个足够大的正数
pme_k	M_k的预保养单位能耗
N_{ijk}	M_k加工O_{ij}的噪声值
$C_{\max}$	最大完工时间

为了将问题描述清楚, 表2给出了一个3 × 3的FJGSPPMT示例, 给出3个工件在3台机器上的加工时间、设置时间和各机器的预保养时间. 其中, O_{32}可以在M_1或M_3上加工, 若选择M_1, 则加工时间为8, 设置时间为3, 若加工前进行预保养, 则M_1预保养时间为4. “–”表示该机器不具备加工该道工序的能力.

表 2 3 × 3 FJGSPPMT示例

Table 2 Example of 3 × 3 FJGSPPMT

工件	工序	加工时间/设置时间		
		M_1	M_2	M_3
J_1	O_{11}	7/2	6/3	7/3
	O_{12}	5/3	—	6/2
	O_{13}	8/2	10/4	—
J_2	O_{21}	6/3	5/2	5/3
	O_{22}	7/2	5/3	5/3
	O_{23}	5/2	6/3	—
J_3	O_{31}	3/3	4/2	—
	O_{32}	8/3	—	5/2
预保养时间		4	3.5	3.8

2.2 模型建立

在实际生产中, FJGSPPMT有多个评价指标, 在不同的约束条件下, 采用优化方法对输入的工件信息、机器信息等进行合理分配, 最终输出非劣的调度方案.

2.2.1 目标函数

本文优化目标为最大完工时间最小、能耗最小、总噪声值最小, 如式(1)所示:

$$\min f = \{f_1, f_2, f_3\}. \tag{1}$$

1) 最大完工时间指所有工件中完成时间最大值, 它表示一个生产活动的周期, 是衡量企业加工经济效益的重要指标, 其表达式如下:

$$f_1 = \min(C_{\max}) = \min(\max_{1\leqslant i\leqslant n} C_i). \tag{2}$$

2) 总能耗(total energy consumption, TEC)最小, 该目标用于描述车间加工生产活动中的各种能耗之和, 是一个企业是否达到节能生产的标准, 包含加工能耗PE、空闲能耗IE、运输能耗TE、设置能耗SE和保养能耗PME, 表达式如下:

$$f_2 = \min \mathrm{TEC} = \min(\mathrm{PE} + \mathrm{IE} + \mathrm{TE} + \mathrm{SE} + \mathrm{PME}). \tag{3}$$

加工能耗PE如式(4)所示, 与机器单位加工能耗pe_{kt}息息相关. 考虑到机器长期运行导致能耗升高, 定义可变的机器单位加工能耗pe_{kt} 如式(5)所示:

$$\mathrm{PE} = \sum_{i=1}^{n}\sum_{j=1}^{h_i}\sum_{k=1}^{m} X_{ijk} \times \int_{S_{ij}^{\mathrm{pt}}}^{C_{ij}^{\mathrm{pt}}} \mathrm{pe}_{kt}\mathrm{d}t, \tag{4}$$

$$\mathrm{pe}_{kt}=\begin{cases}\mathrm{pe}_k,\ \mathrm{sumpt}_{kt}<t_{\min},\\ \mathrm{pe}_k\left(1+\mathrm{ep}\times\left(\mathrm{sumpt}_{kt}-t_{\min}\right)\right),\\ \qquad t_{\min}\leqslant\mathrm{sumpt}_{kt}\leqslant t_{\max},\\ \mathrm{pe}_k\left(1+\mathrm{ep}\times\left(t_{\max}-t_{\min}\right)\right),\\ \qquad \mathrm{sumpt}_{kt}>t_{\max},\end{cases}\tag{5}$$

式中: pe_k为机器M_k的初始单位加工能耗, $[t_{\min}, t_{\max}]$为累计加工负载sumpt_{kt}对pe_k的影响范围, ep为增长率. 其他各能耗的计算公式如下:

$$\mathrm{IE}=\sum_{k=1}^{m}\mathrm{it}_k\times\mathrm{ie}_k,\tag{6}$$

$$\mathrm{it}_k=\sum_{x=1}^{n}\sum_{y=1}^{h_x}\sum_{i=0}^{n}\sum_{j=1}^{h_i}\left(S_{xy}^{\mathrm{pm}}-C_{ij}^{\mathrm{pt}}\right)\times Z_{ijxyk},\tag{7}$$

$$\mathrm{TE}=\sum_{i=1}^{n}\sum_{j=2}^{h_i}\sum_{k=1}^{m}t_{ijk}^{\mathrm{t}}\times\mathrm{te},\tag{8}$$

$$\mathrm{SE}=(\sum_{i=1}^{n}\sum_{k=1}^{m}t_{ijk}^{\mathrm{s}}+\sum_{i=1}^{n}\sum_{j=1}^{h_i}\sum_{x=1}^{n}\sum_{y=2}^{h_x}\sum_{k=1}^{m}t_{xyk}^{\mathrm{s}}\times(1-Z_{ijxyk}))\times\mathrm{se}_k,\tag{9}$$

$$\mathrm{PME}=\sum_{i=1}^{n}\sum_{j=1}^{h_i}\sum_{k=1}^{m}\mathrm{pm}_k\times Q_{ijk}\times\mathrm{pme}_k.\tag{10}$$

3) 噪声污染最小, 该目标用于描述车间机器在加工过程中由于机械设备本身的振动、部件之间的碰撞以及所造成的气流扰动等而产生的噪声值大小[19]. 由于人一直接触噪声和断续接触噪声所受影响程度不同, 本文采用平均噪声衡量车间的总噪声, 车间生产总噪声计算公式如式(11)所示:

$$f_3=\min\{10\times\lg\frac{\sum_{i=1}^{n}\sum_{j=1}^{h_i}\sum_{k=1}^{m}10^{0.1\times N_{ijk}}\times t_{ijk}^{\mathrm{p}}\times X_{ijk}}{\sum_{k=1}^{m}t_{ijk}^{\mathrm{p}}\times X_{ijk}}\}.\tag{11}$$

总能耗最小化目标涵盖加工、空闲、运输、设置及预保养等环节的能耗, 直接反映生产过程的资源利用效率与节能水平; 总噪声值最小化目标则致力于降低车间机器运行产生的噪声污染, 二者共同构成绿色调度在节能减排和环境友好方面的核心优化方向.

2.2.2　决策变量

在建立FJGSPPMT中, 需引入以下决策变量.

$$X_{ijk}=\begin{cases}1,\ 若X_{ijk}在M_k上加工,\\ 0,\ 其他,\end{cases}\tag{12}$$

$$Y_{ijk}=\begin{cases}1,\ 若O_{i(j-1)}不由M_k加工且O_{ij}由M_k加工,\\ 0,\ 其他,\end{cases}\tag{13}$$

$$Z_{ijxyk}=\begin{cases}1,\ 若O_{ij}先于紧邻工序O_{xy}在M_k上加工,\\ 0,\ 其他,\end{cases}\tag{14}$$

$$Q_{ijk}=\begin{cases}1,\ 若M_k在加工O_{ij}前预保养,\\ 0,\ 其他.\end{cases}\tag{15}$$

2.2.3　约束条件

求解多目标FJGSPPMT时应满足以下约束条件:

$$S_{ij}^{\mathrm{pt}}+X_{ijk}\times t_{ijk}^{\mathrm{p}}\leqslant C_{ij}^{\mathrm{pt}},\tag{16}$$

$$C_{i(j-1)}^{\mathrm{pt}}+t_{ijk}^{\mathrm{t}}\leqslant S_{ij}^{\mathrm{pt}}+L\left(1-Y_{ijxyk}\right),\tag{17}$$

$$\mathrm{CM}_{k(l-1)}+\mathrm{pm}_k\times Q_{ijk}+\mathrm{st}_{ijk}\times Y_{ijk}\leqslant\mathrm{SM}_{kl},\tag{18}$$

$$S_{ij}^{\mathrm{pm}}+\mathrm{pm}_k\times Q_{ijk}\leqslant C_{ij}^{\mathrm{pm}},\tag{19}$$

$$C_{ij}^{\mathrm{pt}}\leqslant C_{\max},\tag{20}$$

$$S_{ij}^{\mathrm{st}}+t_{ijk}^{\mathrm{s}}\times Y_{ijk}\leqslant S_{xy}^{\mathrm{st}}+L\left(1-Z_{ijxyk}\right),\tag{21}$$

$$\sum_{k=1}^{m}X_{ijk}=1,\tag{22}$$

$$C_{ij}^{\mathrm{st}}\geqslant S_{ij}^{\mathrm{st}}\geqslant 0,\ C_{ij}^{\mathrm{tr}}\geqslant S_{ij}^{\mathrm{tr}}\geqslant 0,\ C_{ij}^{\mathrm{pt}}\geqslant S_{ij}^{\mathrm{pt}}\geqslant 0,\ C_{ij}^{\mathrm{pm}}\geqslant S_{ij}^{\mathrm{pm}}\geqslant 0,\ C_i\geqslant 0,\tag{23}$$

其中: 式(12)–(15)表示决策变量; 式(16)–(17)表示同一个工件的工序具有加工顺序约束; 式(18)表示同一台机器上的相邻工序之间有设置时间约束; 式(19)表示机器上的保养时间约束, 若选择在加工O_{ij}前进行预保养, 则需要先花费pm_k时间对M_k预保养, 才能开始对工件的设置或加工; 式(20)表示工件的完工时间约束, 即每一个工件的完工时间不可能超过总的完工时间; 式(21)表示任意时刻一台机器只能加工一道工序; 式(22)表示在某一时刻一道工序只能由一台机器加工; 式(23)表示各个参数均非负.

3　基于目标平衡的Jaya优化算法

Jaya算法是一种具有高效性、全局搜索能力和鲁棒性的优化算法, 该算法求解效率高, 易于实现, 且参数设置简单, 只需预定义种群大小和最大迭代次数, 适用于解决调度问题. Jaya算法的主要思想是, 给定问题的解决方案应该朝着最佳解决方案移动, 并且远离最坏的解决方案, 核心迭代进化公式如式(24)所示.

$$X'_{jk(i+1)}=X_{jki}+r_{1ji}\left(X_{\mathrm{jbest}i}-|X_{jki}|\right)-r_{2ji}\left(X_{\mathrm{jworst}i}-|X_{jki}|\right),\tag{24}$$

式中: i代表迭代次数, k表示种群中的个体, j表示个体中的第j维变量, X_{jki}是第i代个体k中变量j的值, $X'_{jk(i+1)}$是其更新后的值, $X_{\mathrm{jbest}i}$是第i代最优个体中变量j的值, $X_{\mathrm{jworst}i}$是第i代最劣个体中变量j的值, r_{1ji}和r_{2ji}为2个在$[0,1]$范围内的随机数. 式中$r_{1ji}(X_{\mathrm{jbest}i}-|X_{jki}|)$表示个体趋向于最优个体, 而$-r_{2ji}(X_{\mathrm{jworst}i}-|X_{jki}|)$则表示个体有远离最劣个体的趋势. 若$X'_{jk(i+1)}$优于$X_{jki}$, 则替换$X_{jki}$, 否则不变.

虽然Jaya算法简易高效, 但存在收敛速度较慢且

易陷入局部最优的缺陷, 针对这些问题, 本文设计了一种MBJaya以求解多目标FJGSPPMT. MBJaya主要思路如下: 1) 采用多规则初始化种群, 为算法提供高质量起点, 加快收敛速度, 并维持种群多样性, 减少冗余运算; 2) 为加快算法收敛并提高种群多样性, 提出多样性Jaya优化算子, 遍历种群中所有非最优解, 使其快速进化, 提高算法全局搜索能力; 3) 为了提升算法的局部搜索能力和非支配解的分布性, 设计了多种基于目标平衡的局部搜索策略, 深度挖掘非支配解周围的更优解, 提高种群的进化效率.

3.1 编码和解码

FJGSPPMT分为机器选择(machines selection, MS)、工序排序(operations sequence, OS)和预保养选择(pre-maintenance selection, PS) 3个子问题, 因此编码采用整数编码生成MS, OS和PS 3层编码串. 以表2描述的示例为例, 编码过程如图1所示.

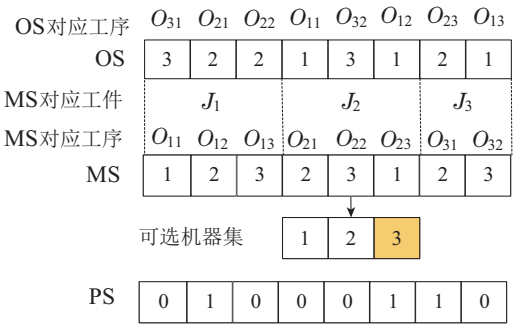


图 1 编码示意图

Fig. 1 Encoding diagram

在图1中, 编码长度为工序总数, OS串元素值为工件号, 工件号出现的次数对应工件的工序号; MS串元素值为工序选择的机器序号, 对应的工件工序按照$J_1-J_2-J_3$顺序排列; PS串元素值表示工序加工前是否选择进行预保养, 1表示进行预保养, 0表示不进行, 工序对应顺序与MS串相同. 由图1可知, O_{22}的可选机器为M_1, M_2和M_3, O_{22}选择M_3加工, 且O_{22}在加工前不需要进行预保养.

编码由MS, OS和PS串组成, 因此需要分别对其进行解码. 对MS串进行解码时, 从左到右依次读取每道工序选择的机器序号, 生成机器分配方案. 对PS串进行解码时, 从左到右依次读取每道工序加工前是否选择预保养, 生成预保养选择方案. 对OS串进行解码时从左到右依次读取, 根据OS串中给出的工序序列信息, 与MS串中机器分配方案和PS串中的预保养选择方案相结合, 对工序进行“插空”排列, 在机器上选择合适的位置进行解码, 以此生成调度方案. 综合考虑运输和设置时间约束, 设计一种适用于FJGSPPMT的改进贪婪解码方法(improvd greed decode method, I-GDM), 伪代码如算法1所示(见表3).

表 3 算法1: IGDM

Table 3 Algorithm 1: IGDM

输入: 个体的OS、MS、PS所有工序数allpnum, 各机器的加工信息

输出: 工序开始预保养时间$S_{ij}^{\rm pm}$、工序结束预保养时间$C_{ij}^{\rm pm}$、工序开始设置时间$S_{ij}^{\rm st}$、设置完成时间$C_{ij}^{\rm st}$; 工序开始运输时间$S_{ij}^{\rm tr}$、运输结束时间$C_{ij}^{\rm tr}$; 工序开始加工时间$S_{ij}^{\rm pt}$、加工完成时间$C_{ij}^{\rm pt}$; 机器加工时间$[{\rm ST}_k, {\rm ET}_k]$

for $t=1$ to allpnum **do**

  $O_{ij}={\rm OS}(t)$; $M_k={\rm MS}(O_{ij})$; $Q_{ijk}={\rm PS}(O_{ij})$; success$=0$; % success用于判断是否插入空闲时间

  获取O_{ij}加工所需设置时间$t_{ijk}^{\rm st}$、加工时间$t_{ijk}^{\rm pt}$、预保养时间$t_{ijk}^{\rm pm}$和运输时间$t_{ijk}^{\rm tr}$;

  获取机器M_k空闲时间$[{\rm AST}_k, {\rm AET}_k]$;

  if $j==1$ **then** % O_{ij}是工件首工序

    if 机器未进行加工 **then**

      $S_{ij}^{\rm pm}=0$; $C_{ij}^{\rm pm}=S_{ij}^{\rm pm}+t_{ijk}^{\rm pm}$; $S_{ij}^{\rm st}=C_{ij}^{\rm pm}$; $C_{ij}^{\rm st}=S_{ij}^{\rm st}+t_{ijk}^{\rm st}$; $S_{ij}^{\rm pt}=C_{ij}^{\rm st}$; $C_{ij}^{\rm pt}=S_{ij}^{\rm pt}+t_{ijk}^{\rm pt}$; $S_{ij}^{\rm tr}=0$; $C_{ij}^{\rm tr}=0$; success $=1$;

    else % 机器已进行加工

      if $[{\rm AST}_k, {\rm AET}_k]\approx\varnothing$ && ${\rm AST}_k+t_{ijk}^{\rm pm}+t_{ijk}^{\rm st}+t_{ijk}^{\rm pt}\leqslant{\rm AET}_k$ **then** % 机器有空闲且满足插入

        $S_{ij}^{\rm pm}={\rm AST}_k$; $C_{ij}^{\rm pm}=S_{ij}^{\rm pm}+t_{ijk}^{\rm pm}$; $S_{ij}^{\rm st}=C_{ij}^{\rm pm}$; $C_{ij}^{\rm st}=S_{ij}^{\rm st}+t_{ijk}^{\rm st}$; $S_{ij}^{\rm pt}=C_{ij}^{\rm st}$; $C_{ij}^{\rm pt}=S_{ij}^{\rm pt}+t_{ijk}^{\rm pt}$; $S_{ij}^{\rm tr}=0$; $C_{ij}^{\rm tr}=0$; success $=1$;

      end if

    end if

  else % 不是工件首工序

    if $[{\rm AST}_k, {\rm AET}_k]\approx\varnothing$ **then** % 机器有空闲

      if 插入工件号与机器上一道加工工件号不一样 && $\max(C_{i(j-1)}+{\rm tt}_{ijk}, {\rm AST}_k+t_{ijk}^{\rm pm}+t_{ijk}^{\rm st})+t_{ijk}^{\rm pt}\leqslant{\rm AET}_k$ **then**

        $S_{ij}^{\rm pt}=\max({\rm AST}_k+t_{ijk}^{\rm pm}+t_{ijk}^{\rm st}, C_{i(j-1)}+{\rm tt}_{ijk})$; $C_{ij}^{\rm pt}=S_{ij}^{\rm pt}+t_{ijk}^{\rm pt}$; $C_{ij}^{\rm st}=S_{ij}^{\rm pt}$; $S_{ij}^{\rm st}=C_{ij}^{\rm st}-t_{ijk}^{\rm st}$; $C_{ij}^{\rm pm}=S_{ijk}^{\rm pt}$; $S_{ij}^{\rm pm}=C_{ij}^{\rm pm}-t_{ijk}^{\rm pm}$; $C_{ij}^{\rm tr}=S_{ij}^{\rm pt}$; $S_{ij}^{\rm tr}=C_{ij}^{\rm tr}-t_{ijk}^{\rm tr}$; success $=1$;

      else if %插入工件号与机器上一道加工工件号一样且为紧邻工序 **then**

        if ${\rm AST}_k+t_{ijk}^{\rm pm}+t_{ijk}^{\rm pt}\leqslant{\rm AET}_k$ **then**

          $S_{ij}^{\rm pm}={\rm AST}_k$; $C_{ij}^{\rm pm}=S_{ij}^{\rm pm}+t_{ijk}^{\rm pm}$; $S_{ij}^{\rm st}=C_{ij}^{\rm pm}$; $C_{ij}^{\rm st}=S_{ij}^{\rm st}$; $S_{ij}^{\rm tr}={\rm AST}_k$; $C_{ij}^{\rm tr}=S_{ij}^{\rm tr}$; $S_{ij}^{\rm pt}=C_{ij}^{\rm st}$; $C_{ij}^{\rm pt}=S_{ij}^{\rm pt}+t_{ijk}^{\rm pt}$; success $=1$;

        end if

      end if

    end if

  end if

  if success $==0$ **then**

    $S_{ij}^{\rm pt}=\max(C_{i(j-1)}^{\rm pt}+t_{ijk}^{\rm tr}, {\rm ET}_k+t_{ijk}^{\rm pm}+t_{ijk}^{\rm st})$; $C_{ij}^{\rm pt}=S_{ij}^{\rm pt}+t_{ijk}^{\rm pt}$; $C_{ij}^{\rm st}=S_{ij}^{\rm st}$; $S_{ij}^{\rm st}=C_{ij}^{\rm st}-t_{ijk}^{\rm st}$; $C_{ij}^{\rm pm}=S_{ijk}^{\rm pt}$; $S_{ij}^{\rm pm}=C_{ij}^{\rm pm}-t_{ijk}^{\rm pm}$; $C_{ij}^{\rm tr}=S_{ij}^{\rm pt}$; $S_{ij}^{\rm tr}=C_{ij}^{\rm tr}-t_{ijk}^{\rm tr}$; success $=1$;

(转下页)

(接上页)

28　**end if**
29　更新工序O_{ij}的最早完工时间C_{ij}, 机器M_k加工时间$[\mathrm{ST}_k, \mathrm{ET}_k]$;
30 **end for**
31 输　出: S_{ij}^{pm}, C_{ij}^{pm}, S_{ij}^{st}, C_{ij}^{st}, S_{ij}^{tr}, C_{ij}^{tr}, S_{ij}^{pt}, C_{ij}^{pt}和$[\mathrm{ST}_k, \mathrm{ET}_k]$

3.2　种群初始化

构建一个优质的初始种群能提升算法的进化起点, 加速收敛过程并减少冗余计算. 传统的初始化方式仅考虑机器负载平衡进行启发式初始化, 忽略了其他目标在初始阶段的重要性, 使得算法在前期获得的解不能在多个目标上获得较优质量. 因此, 本文针对多目标问题采用一种融合多种策略的混合初始化方法, 具体为MS串采用基于目标平衡GLR[20]规则, 不同于传统的仅将机器累计加工时间作为衡量机器选择标准的初始化方式, 基于目标平衡的混合初始化方式通过计算各机器累计时间负载(加工时间和设置时间), 能耗负载以平衡机器的完工时间和总能耗, 旨在优化初始解在完工时间和能耗两个目标中的质量. PS串根据MS各机器上累计加工时间判断是否需要预保养. OS串采用随机初始化.

MS串采用全局选择(global selection, GS)、局部选择(local selection, LS)和随机选择(random selection, RS) 3种初始化规则, 全局选择从所有工件工序的角度记录每台机器的累计用时(加工时间与设置时间)和加工能耗, 每次工序的加工机器都选择总用时和总能耗最小的机器.

PS串的具体操作如下:

步骤 1　初始化各机器的累计加工负载为0.

步骤 2　遍历当前个体已生成的MS编码串, 读取当前基因位的加工机器序号以及加工时间.

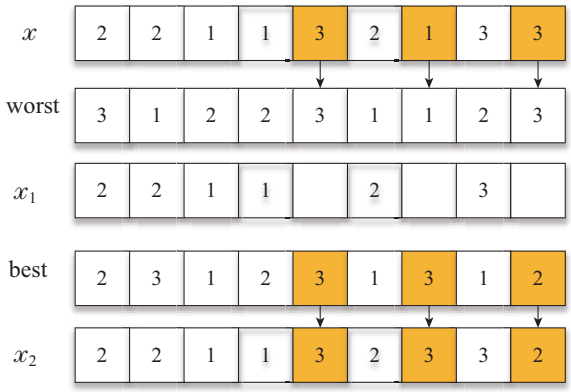
步骤 3　根据累计加工负载对机器单位能耗的影响范围$[t_{\min}, t_{\max}]$得到当前工序预保养的概率. 若选择预保养, 则当前工序选择加工的机器累计加工负载清零. 反之, 则将当前工序加工时间累计到总加工负载中.

步骤 4　重复执行步骤2–3, 直至确定所有工序前的预保养选择.

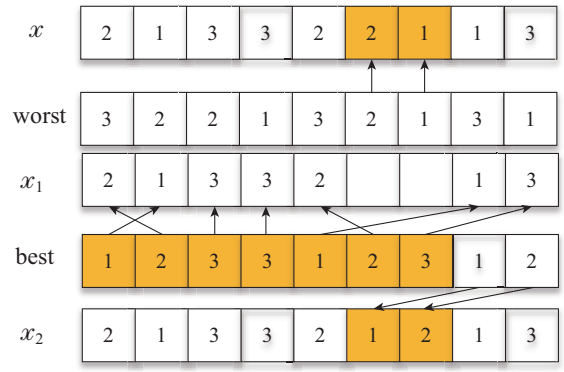
3.3　多样性Jaya 优化算子

传统Jaya算法依赖单一最优和最劣个体引导进化, 易导致算法过早收敛. 针对此弊端, 本文设计了多样性Jaya优化算子, 引导普通个体更新的最优与最劣个体并非传统单一固定的, 而是从优质、劣质个体等级中随机选择, 这样不仅能让普通个体快速进化, 还能保证种群的多样性, 有效避免算法陷入局部最优. 具体操作为: 对所有个体进行非支配排序, 从非支配等级为1的个体中随机选出一个作为最优个体, 从非支配等级最大的个体中随机选择一个作为最劣个体, 对所有等级不为1的个体, 采用进化机制L_1, L_2和L_3进行机器选择和工序排序的进化操作, 采用L_4进行预保养选择的进化操作, 4种进化机制共同构成多样性Jaya优化算子, 以提高种群多样性, 避免算法过早收敛.

进化机制L_1使当前解更加接近最优解, 远离最差解. 执行过程如图2所示.



(a) MS串优化



(b) OS串优化

图 2　进化机制L_1

Fig. 2　Evolutionary mechanism L_1

执行过程的具体步骤如下:

1) MS串优化.

步骤 1　遍历当前个体x的MS串基因位, 找出与最劣个体worst编码值相同的基因并标记, 将未标记的编码复制到x_1.

步骤 2　对于x_1编码为空的基因位, 将最优个体best相同基因位上的编码复制到x_1的空基因位上, 得到x_2即为进化后的个体.

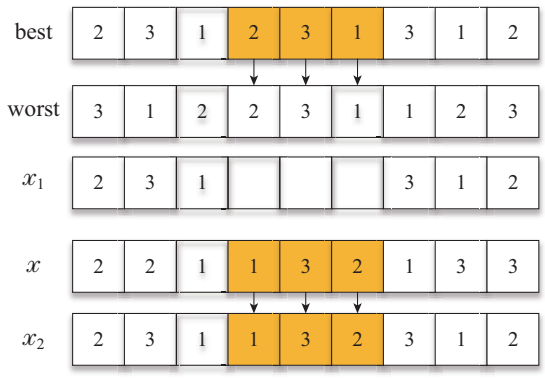
2) OS串优化.

步骤 1　遍历当前个体x的OS串, 标记与最劣个体worst编码值相同的基因位, 并将x中未标记的编码复制到x_1中.

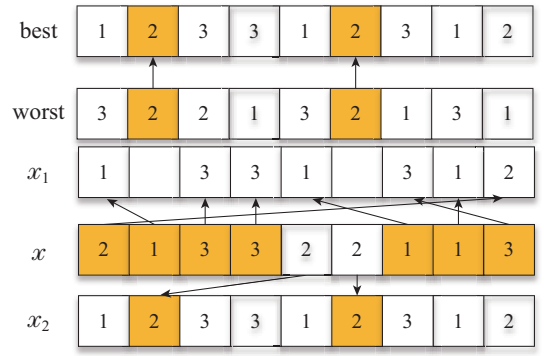
步骤 2 从最优个体best工序排序编码中从头开始找与x_1中编码相同的基因位, 进行标记.

步骤 3 将x_1的编码复制到x_2中, 将best中未标记的编码依次填入x_2中空基因位处, 得到x_2即为进化后的个体.

进化机制L_2与L_1相似, 区别在于在进化的过程中当前个体x与最优个体best的位置互换, 通过不同的更新方式引导非最优个体, 维持种群多样性, 防止算法陷入局部最优. 执行过程如图3所示.



(a) MS串优化



(b) OS串优化

图 3 进化机制L_2

Fig. 3 Evolutionary mechanism L_2

若仅采用L_1和L_2引导个体, 可能会生成大量重复的新个体, 因此引入进化机制L_3, 将当前个体x与最优个体best进行基于工件优先顺序的交叉(precedence preserving order-based crossover, POX) 和多点交叉(multiple point crossover, MPX)[21], 从交叉后的子代中随机选择一个不受x支配的个体作为新个体, 若子代都被x支配, 则不更新.

前3种进化机制都是引导个体优化OS串和MS串, 针对PS串的更新, 本文设计了进化机制L_4, 执行过程如图4所示, 具体步骤如下:

步骤 1 找出当前个体x与最优个体best中选择预保养的工序数, 计算两者之差x_num.

步骤 2 若$x_\text{num} < 0$,则遍历x, 找出与最劣个体worst中PS都为0的基因位并标记; 若$x_\text{num} > 0$, 则找出x与worst中PS都为1的基因位并标记.

步骤 3 若$x_\text{num} < 0$, 在标记的基因位中随机选择$|x_\text{num}|$个基因复制到x对应的基因位; 若$x_\text{num} > 0$, 则在标记的基因位中随机选择x_num个基因复制到x对应的基因位.

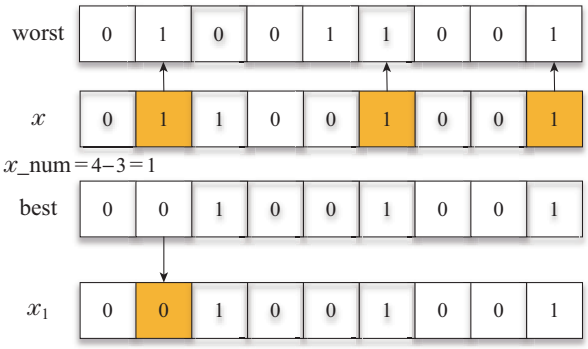


图 4 进化机制L_4

Fig. 4 Evolutionary mechanism L_4

3.4 基于目标平衡的局部搜索

求解多目标调度问题需要同时考虑多个目标的协同优化, 而许多研究者往往忽略了对多个目标的均衡优化, 致使某单个目标大幅提升的同时劣化了其他目标, 导致该个体非支配等级不高. 因此在局部搜索的过程中需要兼顾个体在各个目标中的优化. 此外, 种群中普通个体通过趋优避劣引导进化, 其质量受最优个体的引导, 因此, 为了提高种群的进化效率, 同时避免算法陷入局部最优, 设计多种基于目标平衡的局部搜索策略(local search based on multi-objectives balance, LSMB), 为算法挖掘更优质的解. 具体操作为: 对各目标值中最优的部分个体执行基于目标值的负载平衡(即将目标值当作负载计算)算子; 对所有个体中非支配等级为1的个体进行基于关键路径的邻域搜索; 对所有个体中非支配等级为1的个体进行针对预保养选择的贪婪搜索.

3.4.1 基于目标值的负载平衡算子

由于本文求解的调度问题模型有3个调度目标, 因此, 分别设计基于不同目标值的负载平衡算子B_1, B_2和B_3, 3种算子并行更新种群, 具体操作过程如下:

1) 基于完工时间的负载平衡算子B_1.

步骤 1 将所有个体按照完工时间进行排序, 取前prop比例的个体, 对每一个个体进行以下操作.

步骤 2 分别提取当前个体的每台机器的完工时间.

步骤 3 随机选择具有最大完工时间的机器上的一道工序, 从其可选机器集中选择具有最小完工时间的机器加工该工序.

步骤 4　计算更新后的个体的目标值, 若新个体不受旧个体支配则更新, 反之不更新.

步骤 5　重复步骤2–4, 直至完成prop比例的种群个体更新.

2) 基于总能耗的负载平衡算子B_2.

操作步骤与B_1类似, 区别在于所有个体按照总能耗进行排序, 且更换加工机器的工序从具有最大能耗的机器上随机找, 并更换到其可选机器集中具有最小能耗的机器上.

3) 基于噪声值的负载平衡算子B_3.

操作步骤与B_1类似, 区别在于所有个体按照总噪声值进行排序, 且更换加工机器的工序从具有最大总噪声值的机器上随机找, 并更换到其可选机器集中具有最小总噪声值的机器上.

3.4.2　基于关键路径的邻域搜索

为了提高算法的开发挖掘能力, 减少无效搜索, 设计基于关键路径的邻域搜索, 关键路径已被证实为能搜索解决方案的有效邻域, 是指解决方案中从开始到结束的最长路径[22], 它决定了整个任务的完成时间, 基于关键路径的块结构邻域会减少搜索空间. 传统的关键路径搜索一般都是通过对关键路径上的工序进行小的扰动来搜索邻域, 如将关键路径上的关键工序转移到非关键路径的机器上, 这样有可能缩短当前解的最大完工时间. 而在FJGSPPMT中, 同一工件相邻工序在同一台机器上加工无设置时间, 因此调整关键路径上每台机器加工工序的顺序, 可以减少总设置时间, 从而减少最大完工时间和能耗. 另外, 通过取消关键路径上工序加工前的预保养, 省去预保养的时间, 可以减小完工时间. 因此, 本节设计了一种基于关键路径的邻域搜索策略, 分别对MS, OS和PS更新, MS更新方式为移动关键路径上随机一道工序到加工时间最小, 运输时间最小, 能耗最小的机器上, OS更新方式为调整关键路径上工序加工的顺序, 以减少总设置时间, PS更新方式为取消关键路径上随机某道工序加工前的预保养, 以减少总完工时间. 详细步骤见算法2(见表4).

3.4.3　贪婪搜索

为了在保持能耗目标质量的同时优化完工时间, 针对预保养操作, 设计一种贪婪搜索算子, 以增加能耗为代价减少完工时间, 或在不牺牲完工时间的前提下减少能耗, 平衡完工时间与能耗两个目标值. 若取消工序加工前的预保养可以使其插入到机器加工空闲中, 则取消预保养, 以优化非支配个体的完工时间, 若当前机器没有满足上述条件的工序, 则找出原先无预保养但增加预保养只占用机器空闲时段的某道工序, 在不增加完工时间的情况下优化总能耗. 详细步骤见算法3(见表5).

表 4 算法2: 基于关键路径的邻域搜索

Table 4 Algorithm 2: Neighborhood search based on critical path

输入: 非支配等级为1的个体X集合
输出: 更新后的个体X集合
for $i = 1 : \text{length}(X)$ **do** % 对每一个非支配等级为1的个体进行操作
　$x = X(i)$, 找出x的关键路径, 共l条
　for $j = 1 : l$ **do**
　　随机选择该关键路径上的一道工序
　　将它更换到可选加工机器中加工时间最小的机器, 更新个体$x \to x_1$
　　将它更换到可选加工机器中运输时间最小的机器, 更新个体$x \to x_2$
　　将它更换到可选加工机器中加工能耗最小的机器, 更新个体$x \to x_3$
　　若该工序加工前有预保养操作, 取消预保养, 更新个体$x \to x_4$
　　找出该关键路径上在每台机器上的关键块$(\text{CBS}_k)(1 \leqslant k \leqslant m)$
　　for $z = 1 : m$ **do**
　　if CBS_k中有同工件的但不为紧邻加工顺位的相邻工序O_{xy}, $O_{x(y-1)}(1 \leqslant x \leqslant n, 2 \leqslant y \leqslant h_x)$ **then**
　　将O_{xy}插入到$O_{x(y-1)}$之后, 更新个体$x \to x_5$
　　else 计算连续工序之间的设置时间, 并找出CBS_k上具有最大设置时间的工序O_{bst};
　　if O_{bst}是CBS_k上的第一道工序 **then**
　　将O_{bst}随机插入到一个中间的工序之后, 更新个体$x \to x_5$
　　else if O_{bst}是CBS_k上的中间的工序 **then**
　　将O_{bst}插入到第一道工序之前或者插入到最后一道工序之后, 更新个体$x \to x_5$
　　else if O_{bst}是CBS_k上的最后一道工序 **then**
　　将O_{bst}随机插入到一个中间的工序之前, 更新个体$x \to x_5$
　　end if
　　end if
　　end for
　　计算新解$x_1 - x_5$的目标值, 找出最优非支配解, 如果该解支配旧解x则替换, 否则不替换
　end for
end for

3.5　算法复杂度分析

MBJaya的时间复杂度取决于算法自身的参数与算子, 主要有种群大小popsize、解维度dim、解码算子、全局搜索算子以及局部搜索算子. MBJaya求解FJGSPPMT时, 可知解码算子的时间复杂度为O(popsize×dim), 全局搜索算子的时间复杂度为O(popsi-

ze×dim), 局部搜索算子的时间复杂度为O(prop×popsize × dim), 时间复杂度最高为 O (popsize × dim). MBJaya的空间复杂度指其运行时占用的临时存储空间, MBJaya本身采用3层序列编码, 且搜索算子均属于对原有个体进行部分替换的操作, 所以算法运行时没有额外占用临时存储空间, 即MBJaya的空间复杂度为O(popsize×dim).

表 5 算法3: 贪婪搜索算子

Table 5 Algorithm 3: Greedy search operator

输入: 非支配等级为1的个体X集合
输出: 更新后的个体X集合
for $i = 1$: length(X) **do** % 对每一个非支配等级为1的个体进行操作
  找出当前个体x完工时间最大的机器M_k(若有多个随机选择一个)
  $\mathrm{S} = 0$; % 判断是否插入
  获取M_k上各工序加工所需设置时间、加工时间、预保养时间、运输时间和M_k空闲时间$[\mathrm{AST}_k, \mathrm{AET}_k]$(共$l$段);
  for $j = 1 : l$ **do** % 对空闲段依次操作
    依次读取第j段空闲后加工前有预保养的工序O_{ij}所需设置时间t^{st}_{ijk}、加工时间t^{pt}_{ijk}和运输时间t^{tr}_{ijk}
    if $\max(C_{i(j-1)} + t^{\mathrm{tr}}_{ijk}, \mathrm{AST}_k + t^{\mathrm{st}}_{ijk}) + t^{\mathrm{pt}}_{ijk} \leqslant \mathrm{AET}_k$ **then**
      $\mathrm{PS}(O_{ij}) = 0$; $\mathrm{S} = 1$;
      更新个体$x \to x_1$
      break
    end if
    if $\mathrm{S} == 0$ && 第j段空闲后的工序O_{xy}无预保养操作 **then**
      if $\mathrm{AST}_k + t^{\mathrm{pm}}_{ij} \leqslant \mathrm{AET}_k$ **then**
        $\mathrm{PS}(O_{xy}) = 1$; 更新个体$x \to x_1$
        break
      end if
    end if
  end for
  计算x_1的目标值, 如果x_1不受旧解支配则替换x, 否则替换
end for

4 实验分析

本文所有算法均采用MATLAB2018b软件编程, 在具有2.9 GHz主频, 16 G内存的Intel i7十代CPU的计算机上运行.

4.1 算例生成及评价指标

由于已有研究中尚未有FJGSPPMT的测试算例, 本文对文献[23]中的通用性算例(MK-data和Da-data)进行拓展, 由于文章篇幅有限, 生成10组MK测试算例MTMK01–MTMK10和3组不同规模的Da测试算例MTDa01, MTDa10, MTDa18. 模型特有的数据在拓展信息中, 拓展的具体信息为: $\mathrm{pe}_k \in [11, 15]$, $\mathrm{se}_k \in [3, 5]$, $\mathrm{ie}_k \in [1.5, 2.5]$, $\mathrm{te} = 5$ kw, $N_{ijk} \in [61, 80]$, $t_{\min} \in [15, 25]$, $t_{\max} \in [30, 50]$, $p = 0.5$. 本文采用文献[24]中的反世代距离(inverted generational distance, IGD)、超体积(hypervolume, HV)和贡献率(contribution rate, CR)作为评价指标, 其含义和公式如下:

1) IGD用于衡量算法得到的Pareto前沿集合与真实Pareto前沿集合的距离, IGD值越小, 代表算法得到的解越接近真实解. IGD计算如式(25)所示:

$$\mathrm{IGD} = \frac{\sum\limits_{\bar{r} \in \mathrm{PF}^*} d'\bar{r}}{N}, \tag{25}$$

其中: PF^*表示本文所有算法求得的Pareto前沿, $d'\bar{r} = \min\limits_{\bar{q} \in \mathrm{PF}} |\bar{r} - \bar{q}|$表示$\mathrm{PF}^*$上的点$\bar{r}$到当前算法解集PF中个体$\bar{q}$的最小欧氏距离, N为非支配解的个数.

2) HV用于度量算法在目标空间中所能达到的解的多样性和覆盖面积. 较大的HV值表示算法的综合性能较好. HV计算如式(26)所示:

$$\mathrm{HV} = \mathrm{volume}(\bigcup_{i=1}^{\mathrm{PF}} u_i), \tag{26}$$

其中: u_i是在得到的PF中的一个解和参考点之间形成的超立方体.

3) CR用于评价算法生成的解是否均匀地分布在Pareto前沿上. CR越大, 说明算法的性能越好. CR计算如式(27)所示:

$$\mathrm{CR} = \frac{|\{X \mid X \in \mathrm{PF} \& X \in \mathrm{PF}^*\}|}{|\mathrm{PF}^*|}, \tag{27}$$

其中, PF是对比算法生成的非支配解集, PF^*表示本文所有算法求得的Pareto前沿.

4.2 模型准确性验证

为验证FJGSPPMT的准确性, 采用求解器Cplex和Gurobi对模型进行求解, 在MTMK算例中提取3种小规模测试算例(Ce03表示$m = 3$, $n = 3$, 最大工序数为3)进行实验, MBJaya与求解器均优化$\mathrm{C}_{\max}$. 结果如表6所示, 其中"–"表示算法无法得到最优解.

表 6 MBJaya与求解器的对比结果

Table 6 Comparison results between MBJaya and solver

算例	Cplex		Gurobi		MBJaya	
	$\mathrm{C}_{\max}$	CPU/s	$\mathrm{C}_{\max}$	CPU/s	$\mathrm{C}_{\max}$	CPU/s
Ce03	10	5.30	10	3.43	11	2.73
Ce05	46	318.96	46	283.54	52	13.02
Ce06	—	—	35	20 153	40	15.73

由表6可知, 在Ce03和Ce05中, MBJaya求解得到的完工时间分别为11和52, 而Cplex和Gurobi求出最优解完工时间都为10和46, 虽然求解器能求出较优解,

但运行时间较长, 且Gurobi较Cplex求解速度快. 在Ce06中, Cplex未能求出最优解, Gurobi在很长时间内求解出较优解35, MBJaya求解得到的完工时间为40. 通过与求解器对比, 验证了模型的准确性和MBJaya的有效性.

4.3　参数设置

MBJaya 算法中包含种群规模popsize和局部搜索种群比例prop共2个参数, 为了合理选择参数值, 采用田口实验[25]来确定最恰当的参数组合为popsize=200, prop = 0.1. 为了保证优化算法的收敛性和结果的可靠性, 确保算法有足够的探索空间, 同时避免因过度迭代导致的计算资源浪费, 经过反复实验确定评价次数采用100 000次作为算法迭代的终止条件, 以保证优化质量的同时兼顾计算效率.

4.4　消融实验

为了证明MBJaya算法中改进策略的有效性, 在MBJaya的框架下, 依次对策略组合选择, 得到如表7所示的8种算法, 其中: I_1表示本文设计的混合初始化策略INIT, I_2表示文献[26]中初始化方式; J_1表示本文设计的多样性Jaya优化算子, J_2表示文献[18]中的Jaya优化方式; L_1表示本文设计的基于目标平衡的局部搜索策略LSMB, L_2表示文献[16]中的局部搜索方式.

表 7　选择不同策略的8种算法

Table 7　8 algorithms with different strategies

策略	算法		
	I_1/I_2	J_1/J_2	L_1/L_2
MBJaya0	I_2	J_2	L_2
MBJaya1	I_1	J_2	L_2
MBJaya2	I_2	J_1	L_2
MBJaya3	I_2	J_2	L_1
MBJaya4	I_1	J_1	L_2
MBJaya5	I_1	J_2	L_1
MBJava6	I_2	J_1	L_1
MBJaya	I_1	J_1	L_1

将本实验8种算法与第3.5节的4种对比算法共12种算法分别独立运行20次, 得到最优解集形成的并集, 对此并集进行非支配排序, 将等级为1的解视为Pareto前沿解. 计算各种算法随评价次数eval变化的IGD平均值, 绘制收敛曲线如图5所示.

从图5中MBJaya0和MBJaya1曲线对比可以看出, MBJaya1的IGD值始终在MBJaya0的下方, 说明本文设计的多目标平衡混合初始化策略相比较文献[26]的初始化策略, 为算法提供了更高的起点, 加快了算法的收敛速度. 同时可以看出采用I_1的MBJaya4, MBJaya5和MBJaya算法均具有较优的进化起点, 进一步验证了混合初始化策略的作用. 比较MBJaya0和MBJaya2可得, MBJaya2的IGD值在整个评价过程中均低于MBJaya0, 表明本文设计的多样性Jaya优化算子对种群整体进化具有更大的推动作用, 能够提供更多个体优化的路径. 由MBJaya0和MBJaya3曲线对比可知, MBJaya3的IGD在评价初期与MBJaya0不分上下, 而在评价过程中后期两者差距逐渐增大,前者远远小于后者, 这是因为本文设计的LSMB策略包含多种兼顾多个目标优化的局部搜索策略, 在评价中后期可以通过搜索优质解的邻域挖掘更优解.

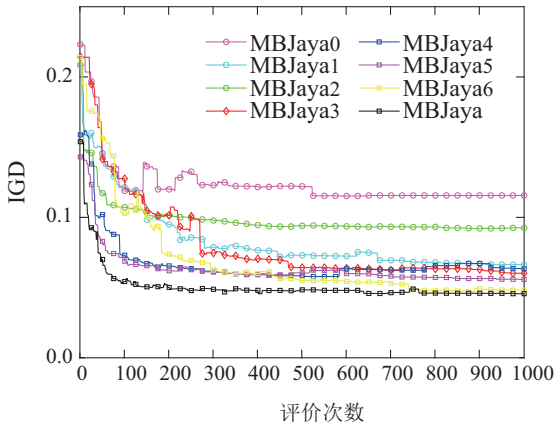


图 5　8种算法IGD收敛图

Fig. 5　IGD convergence graphs for 8 algorithms

融合了I_1和J_1两种策略的MBJaya4的IGD一直都小于MBJaya1和MBJaya2, 这是由于多样性Jaya优化算子增强了算法的全局搜索能力, 并利用混合初始化提供的高质量初始种群, 提高了种群的进化效率. 在多样性Jaya优化算子基础上加入LSMB的MBJaya6在评价初期就比MBJaya2的IGD值低, 在整个评价过程中都优于MBJaya2, 这是因为LSMB相比L_2更能提升种群优质个体的质量, 以提高Jaya优化的效率. 相应地, 从MBJaya6与MBJaya3两条曲线比较可得, 虽然MBJaya6与MBJaya3在评价后期IGD差距不大, 但在前半段评价过程中低于后者甚多, 这是因为多样性Jaya优化算子加快种群进化的效率, 提高了种群质量, 使得LSMB能更早地在质量高的个体邻域挖掘更优质的解. 综合3种改进策略的MBJaya算法具有最优的性能, 在进化过程中MBJaya的IGD几乎都是最小的, 说明3种改进策略互相促进, 使MBJaya具有较高的收敛速度以及较好的寻优能力.

为了评估算法中的3种主要改进策略的影响, 将8种算法在MTMK05得到的各算法最终的非支配解集, 计算HV和IGD值, 采用多元方差分析, HV和IGD分别作为响应变量, F值和P值用于分析实验结果. F值用于衡量该因素对结果的影响, P值用于测试每个因素的不同水平之间是否存在显著差异. MBJaya的分析结果如表8所示, F的值越大, 则该策略对算法的影响越大, 当P值小于0.05时, 表明该策略的影响程度与其他策略有显著差异.

表 8 MBJaya的方差分析结果

Table 8 MBJaya's analysis of variance results

策略	响应变量HV		响应变量IGD	
	F	P	F	P
INIT	1.697	0.026	2.978	0.016
JAYA	1.392	0.303	1.453	0.294
LSMB	1.749	0.025	5.377	0.008

从表8中数据可以看出,无论是以HV还是IGD为响应变量, LSMB的F值都是最大的, 且P值都小于0.05, 说明与其他两种策略相比, LSMB对算法有显著影响. 此外, INIT和JAYA对算法的影响都差不多, 相比之下INIT更有优势. 表9给出了8种算法在13组算例(机器数$m\times$工件数n表示算例规模)中求得的IGD, HV和CR, 加粗的数据为各性能指标的最优值.

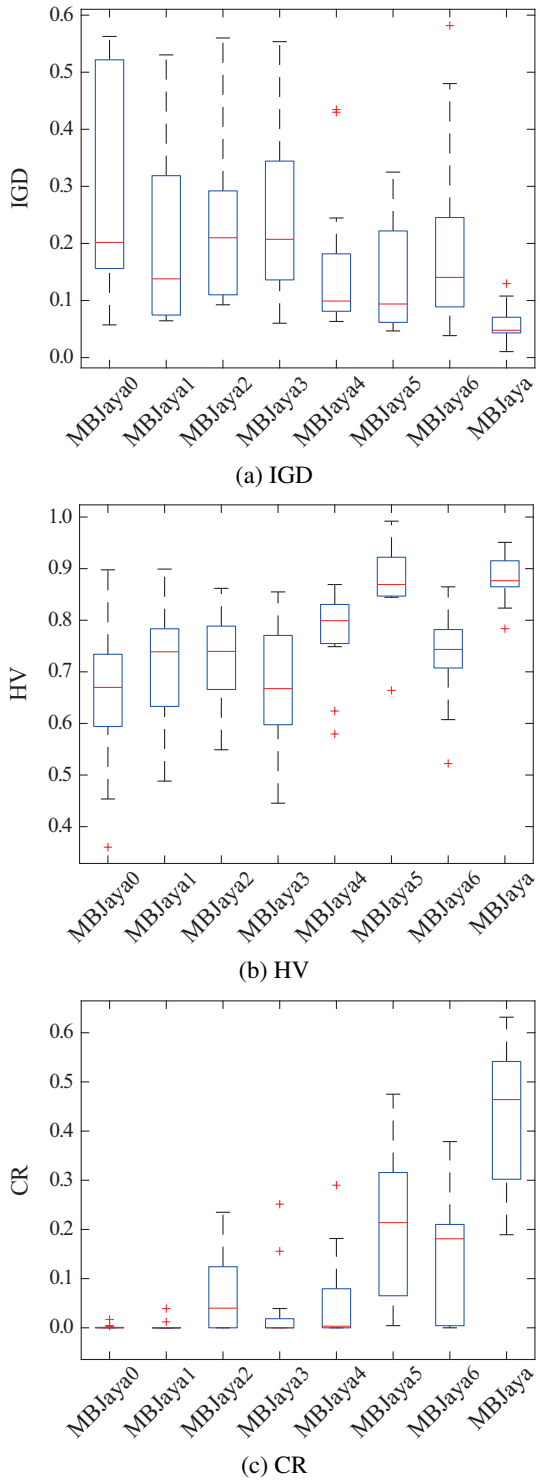
从表9可以看出MBJaya在绝大多数的算例中取得了最优的IGD, HV和CR, 这是因为MBJaya融合了多样性Jaya优化算子和基于目标平衡的局部搜索策略, 提高全局搜索能力的同时, 在非支配解邻域挖掘更优质的解, 使种群快速进化, 逼近并占据真实前沿解. MBJaya在7个算例中得到了最优HV, 这是因为MBJaya采用的混合初始化以及基于目标平衡的局部搜索都考虑了个体在各目标中的优化, 提高了进化过程中的分布性.

表 9 8种算法IGD, HV, CR对比结果

Table 9 Comparison results of 8 algorithms IGD, HV, CR

算例	规模	指标	MBJaya0	MBJaya1	MBJaya2	MBJaya3	MBJaya4	MBJaya5	MBJaya6	MBJaya
		IGD	0.1256	0.1499	0.1101	0.1576	0.1035	0.0529	**0.0385**	0.0389
MTMK01	10×6	HV	0.7282	0.7766	0.7808	0.8366	0.7991	0.8445	0.8466	**0.8751**
		CR	0.0044	0.0000	0.0400	0.1556	0.0000	0.0044	0.0044	**0.5911**
		IGD	0.1569	0.1206	0.2100	0.2072	0.0991	**0.0718**	0.1533	0.0977
MTMK02	10×6	HV	0.7532	0.7389	0.6688	0.6801	0.8246	0.8539	0.7350	**0.8732**
		CR	0.0000	0.0000	0.2287	0.0116	0.0736	0.1240	0.0039	**0.4070**
		IGD	0.1917	0.0720	0.2337	0.2163	0.0684	0.1491	0.1691	**0.0249**
MTMK03	15×8	HV	0.6792	0.7640	0.7032	0.6484	0.8350	0.8468	0.7425	**0.9058**
		CR	0.0000	0.0000	0.0000	0.0000	0.0033	**0.4641**	0.2059	0.2222
		IGD	0.0573	0.0645	0.1131	0.0829	0.1609	**0.0467**	0.1394	0.0616
MTMK04	15×8	HV	0.8977	0.8991	0.8119	0.8389	0.7488	**0.9139**	0.7941	0.9135
		CR	0.0625	0.0729	0.1927	0.0208	0.0000	0.0469	0.1302	**0.1979**
		IGD	0.1535	0.0663	0.0925	0.0602	0.0633	0.0558	0.0482	**0.0456**
MTMK05	15×4	HV	0.6698	0.8487	0.8427	0.8549	**0.8692**	0.8692	0.8647	0.8674
		CR	0.0168	0.0391	0.0559	0.2514	0.0782	0.0447	0.2961	**0.3194**
		IGD	0.2018	0.1209	0.2524	0.1917	0.1289	0.0765	0.2359	**0.0105**
MTMK06	10×10	HV	0.6858	0.7428	0.6568	0.6675	0.7757	0.8468	0.6861	**0.9203**
		CR	0.0000	0.0000	0.1527	0.0000	0.1816	0.0720	0.0605	**0.5072**
		IGD	0.2982	0.0754	0.0956	0.1008	0.0920	0.3171	0.0822	**0.0515**
MTMK07	20×5	HV	0.7515	0.8038	0.7504	0.7485	0.7568	**0.8539**	0.7779	0.8235
		CR	0.0000	0.0120	0.2349	0.0392	0.0030	0.2139	0.2229	**0.2500**
		IGD	0.1834	0.2809	0.2512	0.1479	0.0829	0.3250	0.0911	**0.0541**
MTMK08	20×10	HV	0.6649	0.6691	0.8618	0.7178	0.8291	**0.9515**	0.7738	0.7837
		CR	0.0000	0.0000	0.0415	0.0000	0.0830	0.1774	0.1811	**0.5170**
		IGD	0.2911	0.1835	0.1272	0.2538	0.0836	0.2808	0.1405	**0.0455**
MTMK09	20×10	HV	0.5803	0.7072	0.7396	0.6044	0.8353	**0.9216**	0.7145	0.8766
		CR	0.0000	0.0000	0.1147	0.0000	0.0505	0.2982	0.1881	**0.3486**
		IGD	0.5267	0.1380	0.1098	0.2984	**0.0760**	0.1311	0.1377	0.1297
MTMK10	20×15	HV	0.3602	0.6436	0.7396	0.5572	0.8183	0.6640	0.7463	**0.8570**
		CR	0.0000	0.0000	0.0000	0.0000	**0.2899**	0.0237	0.3787	0.1893
		IGD	0.5210	0.5304	0.4112	0.4818	0.2446	0.0939	**0.2739**	0.0479
MTDA01	10×5	HV	0.5986	0.6011	0.6854	0.6164	0.7651	0.9233	0.7434	**0.9509**
		CR	0.0000	0.0000	0.0000	0.0000	0.0000	0.3684	0.0000	**0.6316**
		IGD	0.5239	0.5169	0.4295	0.5263	0.4299	0.2023	0.4801	**0.1078**
MTDA10	15×8	HV	0.6097	0.5890	0.6260	0.5767	0.6240	0.9920	0.6074	**0.9082**
		CR	0.0000	0.0000	0.0000	0.0000	0.0000	0.2222	0.1389	**0.6111**
		IGD	0.5627	0.4324	0.5600	0.5536	0.4346	0.0637	0.5817	**0.0447**
MTDA18	20×10	HV	0.4536	0.4882	0.5490	0.4454	0.5795	0.9138	0.5223	**0.9441**
		CR	0.0000	0.0000	0.0000	0.0000	0.0000	0.4750	0.0000	**0.5250**

为了更清晰地观察各算法的性能表现, 将表9中的数据绘制8种算法的IGD, HV和CR的箱线图, 如图6所示.



(a) IGD

(b) HV

(c) CR

图 6　8种算法的IGD, HV和CR的箱线图

Fig. 6　Box plots of IGD, HV and CR for 8 algorithms

从图6中IGD、HV和CR的箱线图可以进一步得出结论, 采用不同策略算法的IGD、HV和CR都比未采用该策略的算法好, 说明本文所提的3种策略均能提升算法性能, 获得更好的结果. IGD和HV箱线图中出现的异常点是由于在某个算例上, 该算法求解的非支配解集不同于其他算例远优于/劣于其他算法, 因此这些异常点不影响算法的性能比较.

4.5　算法性能验证

为了验证MBJaya算法性能, 选择Sun等[23]提出的同样求解多时间约束问题的混合多目标进化算法(hybrid many-objective evolutionary algorithm, HEMA), 钟小玉等[27]提出的经典变邻域非支配改进遗传算法VNS-NSGA-II, Wang等[28]提出启发式规则与人工蜂群算法的组合方法(hybrid combination of heuristic rules and the ABC algorithm, HMABC)和张洪亮等[26]提出的同类设计框架Jaya算法进行比较. 所有算法使用的种群规模与最大评价次数与MBJaya相同, 其他参数采用各自论文中设定参数. 各算法的IGD, HV和CR指标结果如表10所示, 加粗表示性能指标最优.

从表10可以看出, MBJaya算法的IGD、HV和CR指标在绝大多数算例上均为最优. IGD指标值越小, 代表算法得到的解越接近真实解, 在所有算例上, MBJaya的IGD值几乎都为最小, 对比结果表明, MBJaya算法的Pareto前沿解具有更好的分布性和更强的收敛性. 在所有算例上, MBJaya的HV几乎都是最大的, 由此可知, MBJaya算法的搜索范围比其他算法更广, 说明多样性Jaya优化算子策略提高了算法全局寻优能力. CR值代表各算法得到的非支配解在最优非支配解集中的占比, 可以看出, MBJaya算法的CR值在几乎所有算例中都是最大的, 说明基于目标平衡的局部搜索策略充分挖掘了解空间, 找到了更优非支配解, 为最终的非支配前沿做出了较大的贡献.

为了更直观地验证MBJaya算法的优势, 绘制表10中5种算法的IGD, HV和CR的箱线图, 如图7所示.

可以看出, MBJaya得到的IGD的中位线最小, HV和CR的中位线最大, 且MBJaya的IGD和HV箱体范围最小, 说明MBJaya相较于其他4种算法具有更好的稳定性. 需要说明的是, MBJaya在IGD, HV箱线图中均有异常点, 这个异常点产生的原因是在某些算例中, MBJaya 的 IGD 或 HV 并非最优, 但相比较得到最优IGD或HV的算例差距较小, 因此该异常点的出现不影响算法的对比.

为了观察算法性能在不同规模算例上的变化, 将表10中5种算法在MTMK01–MTMK10上的IGD和HV值绘制折线图, 如图8所示.

从图8中可以看出, 无论是在IGD的折线图还是在HV的折线图上, MBJaya算法的变化趋势都不大, 且在绝大部分算例上都保持着最优的性能, 说明MBJaya算法的性能不随算例的规模增大而衰弱.

为了清晰地展示各算法在不同规模算例中求解的非支配解集的分布情况, 绘制了3种不同规模算例的

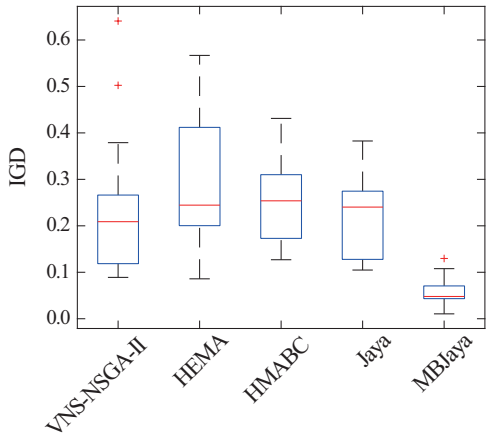
三维目标分布图, 如图9所示, 每种对比算法采用不同的形状表示.

表 10 5种算法IGD, HV, CR对比结果

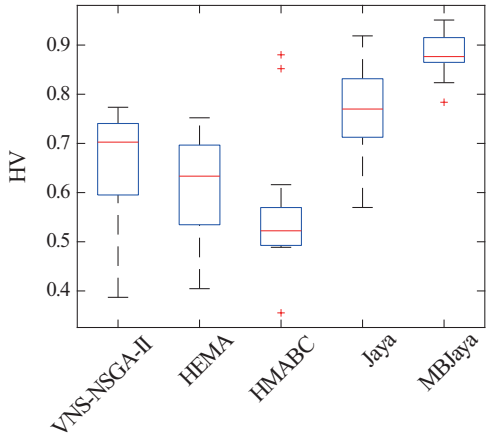
Table 10 Comparison results of 5 algorithms IGD, HV, CR

算例	指标	VNS	HEMA	HMABC	Jaya	MBJaya
	IGD	0.089 0	0.085 9	0.213 7	0.122 9	**0.038 9**
MTMK01	HV	0.769 9	0.752 1	0.498 5	0.802 1	**0.875 1**
	CR	0.017 8	0.000 0	0.182 2	0.000 0	**0.591 1**
	IGD	0.156 3	0.217 4	0.219 8	0.163 4	**0.097 7**
MTMK02	HV	0.730 7	0.648 8	0.522 3	0.830 5	**0.873 2**
	CR	0.007 8	0.000 0	0.089 1	0.054 3	**0.407 0**
	IGD	0.119 1	0.155 7	0.146 4	0.128 2	**0.024 9**
MTMK03	HV	0.713 7	0.633 4	0.554 1	0.834 7	**0.905 8**
	CR	0.009 8	0.000 0	0.045 8	0.049 0	**0.222 2**
	IGD	0.211 4	0.275 5	0.261 0	0.126 4	**0.061 6**
MTMK04	HV	0.702 6	0.690 8	0.525 8	0.910 1	**0.913 5**
	CR	0.005 2	0.000 0	0.182 3	0.088 5	**0.197 9**
	IGD	0.089 5	0.159 7	0.181 8	0.104 8	**0.045 6**
MTMK05	HV	0.773 5	0.723 1	0.493 5	**0.918 7**	0.867 4
	CR	0.000 0	0.044 7	0.117 3	0.027 9	**0.319 4**
	IGD	0.153 4	0.234 3	0.289 2	0.261 8	**0.010 5**
MTMK06	HV	0.697 4	0.640 6	0.535 9	0.609 7	**0.920 3**
	CR	0.000 0	0.000 0	0.025 9	0.000 0	**0.507 2**
	IGD	0.208 9	0.213 9	0.325 9	0.259 1	**0.051 5**
MTMK07	HV	0.773 0	0.713 9	0.503 1	0.761 6	**0.823 5**
	CR	0.006 0	0.000 0	0.006 0	0.012 0	**0.250 0**
	IGD	0.228 6	0.392 0	0.431 2	0.266 6	**0.054 1**
MTMK08	HV	0.722 7	0.623 1	0.355 4	0.769 9	**0.783 7**
	CR	0.000 0	0.000 0	0.000 0	0.000 0	**0.517 0**
	IGD	0.213 6	0.345 1	0.308 4	0.240 3	**0.045 5**
MTMK09	HV	0.690 9	0.606 2	0.489 7	0.789 6	**0.876 6**
	CR	0.000 0	0.000 0	0.000 0	0.000 0	**0.348 6**
	IGD	**0.117 0**	0.244 5	0.253 8	0.161 4	0.129 7
MTMK10	HV	0.602 7	0.486 3	0.488 7	0.768 0	**0.857 0**
	CR	0.005 9	0.000 0	0.106 5	0.005 9	**0.189 3**
	IGD	0.502 4	0.566 8	0.127 0	0.298 5	**0.047 9**
MTDA01	HV	0.572 5	0.547 0	0.880 2	0.718 6	**0.950 9**
	CR	0.000 0	0.000 0	0.000 0	0.000 0	**0.631 6**
	IGD	0.640 8	0.557 4	0.138 4	0.339 8	**0.107 8**
MTDA10	HV	0.409 0	0.497 5	0.852 0	0.694 2	**0.908 2**
	CR	0.000 0	0.000 0	0.027 8	0.000 0	**0.611 1**
	IGD	0.379 0	0.471 5	0.315 1	0.382 6	**0.044 7**
MTDA18	HV	0.387 0	0.404 6	0.616 1	0.569 7	**0.944 1**
	CR	0.000 0	0.000 0	0.000 0	0.000 0	**0.525 0**

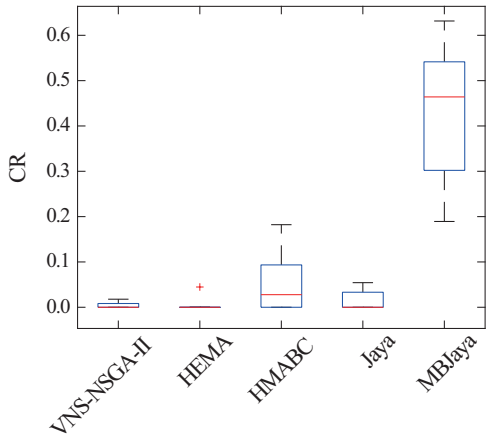
从图 9 中可以看出, 在 MTMK05 和 MTMK10 中, MBJaya算法求得的非支配解的个数最多, 且在3种不同规模算例中MBJaya的解集最靠近原点, 占据最多的前沿面上的解, 说明MBJaya相较于其他4种对比算法, 寻优能力更强, 能挖掘更优更多的解.



(a) IGD



(b) HV



(c) CR

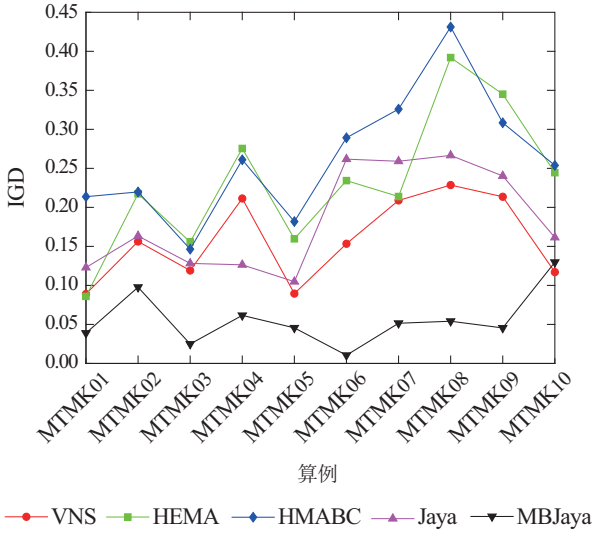
图 7 5种算法的IG, HV和CR的箱线图

Fig. 7 Box plots of IGD, HV and CR for 5 algorithms

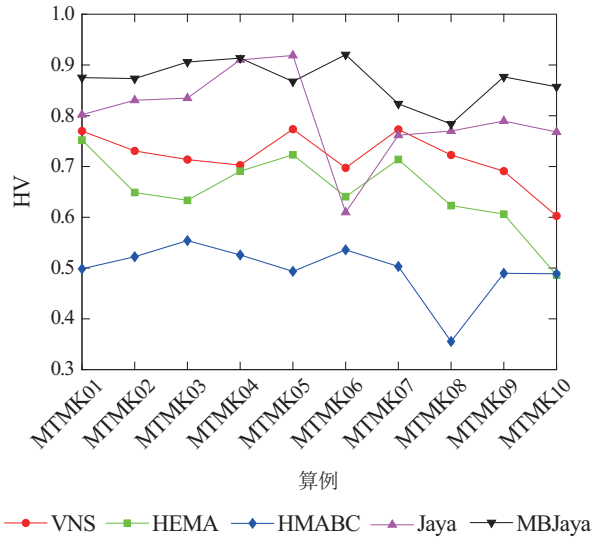
5 结论

本文考虑了实际生产中的运输时间和设置时间, 同时还考虑了机器累计加工时长对机器单位能耗的影响, 将预保养机制纳入调度方案, 建立了考虑预保养的多时间约束柔性作业车间绿色调度模型, 提出了一种基于目标平衡的Jaya算法来解决多目标FJGS-PPMT. 为弥补传统初始化方式仅针对时间负载进行

机器选择易造成初始种群单一目标质量高但分布性不强的不足, 提出了考虑机器累计时间和累计能耗平衡的混合初始化策略, 提高初始种群的质量, 加快了算法的收敛速度; 为了解决传统Jaya算子单一的进化方向造成种群多样性低的问题, 提出了多样性Jaya优化算子, 通过4种进化机制引导非最优个体, 促使种群快速进化且保持种群多样性, 提高算法的全局搜索能力; 为避免Jaya算法陷入局部最优, 针对模型特征, 提出基于目标平衡的局部搜索策略, 充分挖掘优质个体的更优解, 提高算法的寻优能力, 对多个目标同时优化, 提高种群整体质量, 发挥优质个体对种群的引导作用. 通过消融实验对本文设计的策略与其他算法策略的比较, 验证本文改进策略有效性, 利用本文提出的MBJaya算法和已有研究中的4种算法求解13组算例并进行比较, 结果表明MBJaya算法具有更好的分布性和收敛性, 求解FJGSPPMT具有显著优势.



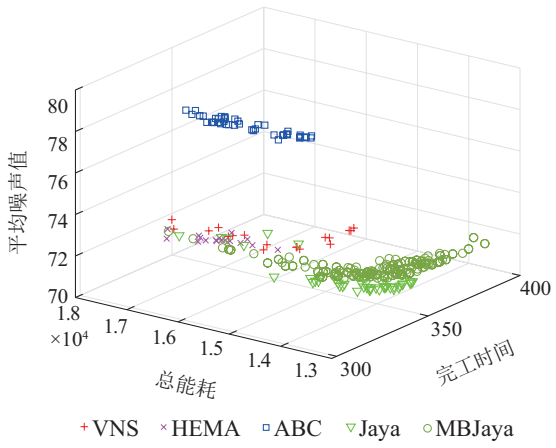
(a) IGD



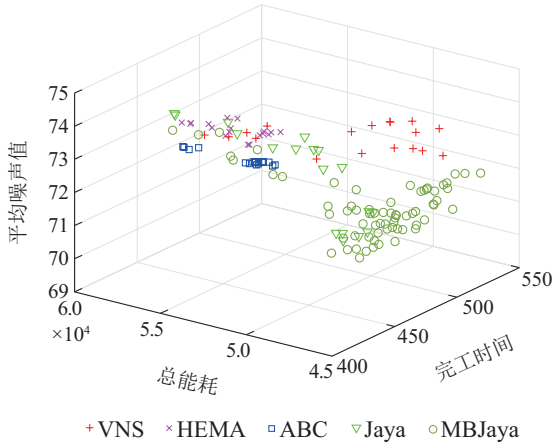
(b) HV

图 8　5种算法的IGD和HV折线图

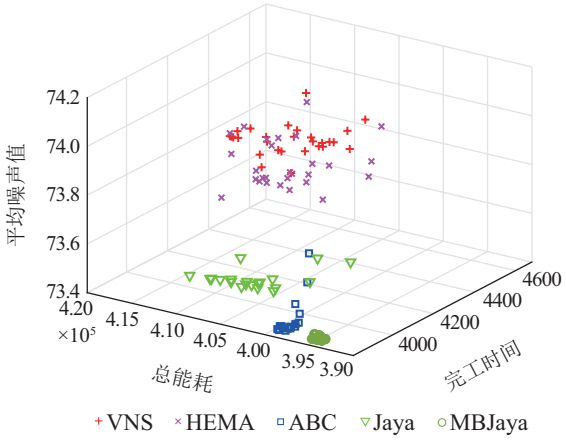
Fig. 8　Line charts of IGD and HV for 5 algorithms



(a) MTMK05



(b) MTMK10



(c) MTDA10

图 9　5种算法的解集分布图

Fig. 9　Distribution map of solution sets for 5 algorithms

本文对于MBJaya算法求解FJGSPPMT的研究还存在一些不足. 企业实际生产的需求规模较大, 工件种类以及数量较多, 因此, 需要对批量调度进行研究. 此外, 自动导向车批量运输工件的数量对其运输速度和运输消耗的能耗也会有影响, 值得进一步研究.

参考文献:

[1] JIN Siyuan, PENG Cheng, WANG Wei, et al. Distributed flexible job-shop scheduling algorithm based on a vector mapping surrogate

model. *Control and Decision*, 2025, 40(5): 1561 – 1570.
(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. 控制与决策, 2025, 40(5): 1561 – 1570.)

[2] GONG Q S, LI J L, JIANG Z G, et al. A hierarchical integration scheduling method for flexible job shop with green lot splitting. *Engineering Applications of Artificial Intelligence*, 2024, 129: 107595.

[3] LI Yibing, HUANG Weixing, WU Rui. Research on multi-objective green flexible job-shop scheduling based on improved ABC algorithm. *China Mechanical Engineering*, 2020, 31(11): 1344 – 1350, 1385.
(李益兵, 黄炜星, 吴锐. 基于改进人工蜂群算法的多目标绿色柔性作业车间调度研究. 中国机械工程, 2020, 31(11): 1344 – 1350, 1385.)

[4] XU G, BAO Q, ZHANG H. Multi-objective green scheduling of integrated flexible job shop and automated guided vehicles. *Engineering Applications of Artificial Intelligence*, 2023, 126: 106864.

[5] LUO Wenchong, QIAN Bin, HU Rong, et al. Hyper-heuristic cross-entropy algorithm for distributed assembly flexible job-shop scheduling problem. *Control Theory & Applications*, 2021, 38(10): 1551 – 1568.
(罗文冲, 钱斌, 胡蓉, 等. 超启发式交叉熵算法求解分布式装配柔性作业车间调度问题. 控制理论与应用, 2021, 38(10): 1551 – 1568.)

[6] CHEN X L, LI J Q, SANG H Y. Improved artificial immune algorithm for the flexible job shop problem with transportation time. *Measurement & Control*, 2020, 53(9/10): 2111 – 2128.

[7] ZIAEE M, MORTAZAVI S M J, AMRA M. Flexible job shop scheduling problem considering machine and order acceptance, transportation costs, and setup times. *Soft Computing*, 2022, 26(7):3527 – 3543.

[8] YUNUSOGLU P, YILDIZ S T. Solving the flexible job shop scheduling and lot streaming problem with setup and transport resource constraints. *International Journal of Systems Science-Operations & Logistics*, 2023, 10(1): 2221072.

[9] MENG L L, ZHANG B, GAO K Z, et al. An MILP model for energy-conscious flexible job shop problem with transportation and sequence-dependent setup times. *Sustainability*, 2022, 15(1): 776.

[10] ZHANG G H, YAN S F, SONG X H, et al. Evolutionary algorithm incorporating reinforcement learning for energy-conscious flexible job-shop scheduling problem with transportation and setup times. *Engineering Applications of Artificial Intelligence*, 2024, 133: 107974.

[11] PAL M, MITTAL M, SONI G, et al. A multi-agent system for FJSP with setup and transportation times. *Expert Systems with Applications*, 2023, 216: 119474.

[12] WANG Jianhua, PAN Yujie, SUN Rui. Flexible job-shop green scheduling algorithm considering machine tool depreciation. *Journal of Computer Applications*, 2020, 40(1): 43 – 49.
(王建华, 潘宇杰, 孙瑞. 考虑机床折旧的柔性作业车间绿色调度算法. 计算机应用, 2020, 40(1): 43 – 49.)

[13] YANG B, LIAW P K, MORRISON M, et al. Temperature evolution during fatigue damage. *Intermetallics*, 2005, 13(3): 419 – 428.

[14] YAN J, HUA D. Energy consumption modeling for machine tools after preventive maintenance. *IEEE International Conference on Industrial Engineering and Engineering Management*. Macao, China: IEEE, 2010: 2201 – 2205.

[15] BAYKASOGLU M, MADENOGLU F S. Greedy randomized adaptive search procedure for simultaneous scheduling of production and preventive maintenance activities in dynamic flexible job shops. *Soft Computing*, 2021, 25(23): 14893 – 14932.

[16] RAO R. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *International Journal of Industrial Engineering Computations*, 2016, 7(1): 19 – 34.

[17] FAN J X, SHEN W M, GAO L, et al. A hybrid Jaya algorithm for solving flexible job shop scheduling problem considering multiple critical paths. *Systems*. 2021, 60: 298 – 311.

[18] CALDEIRAR H, GNANAVELBABU A. A Pareto based discrete Jaya algorithm for multi-objective flexible job shop scheduling problem. *Expert Systems with Applications*. 2021, 170: 114567.

[19] NIE Wenqian, WANG Ting. Flexible job shop scheduling study considering carbon emissions and noise pollution. *Logistics Engineering and Management*, 2023, 45(8): 57 – 62, 20.
(聂文倩, 王婷. 考虑碳排放量和噪声污染的柔性作业车间调度研究. 物流工程与管理, 2023, 45(8): 57 – 62, 20.)

[20] ZHANG G H, GAO L, SHI Y. An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications*. 2011, 38(4): 3563 – 3573.

[21] LUAN F, ZHAO H X, LIU S Q, et al. Enhanced NSGA-II for multi-objective energy-saving flexible job shop scheduling. *Sustainable Computing-Informatics & Systems*, 2023, 39: 100901.

[22] ZHANG H L, XU G J, PAN R L, et al. A novel heuristic method for the energy-efficient flexible job-shop scheduling problem with sequence-dependent set-up and transportation time. *Engineering Optimization*, 2021, 54(10): 1646 – 1667.

[23] SUN J H, ZHANG G H, LU J, et al. A hybrid many-objective evolutionary algorithm for flexible job-shop scheduling problem with transportation and setup times. *Computers & Operations Research*, 2021, 132: 105263.

[24] TIAN Y, CHENG R, ZHANG X Y, et al. PlatEMO: A MATLAB platform for evolutionary multi-objective optimization. *IEEE Computational Intelligence Magazine*, 2017, 12(4): 73 – 87.

[25] ANSARI T, CHANDRA G, GUPTA P K, et al. Synthesis of pine needle cyanoethyl cellulose using Taguchi L25 orthogonal array. *Industrial Crops & Products*, 2023, 191: 115973.

[26] ZHANG Hongliang, XU Gongjie, BAO Qiang, et al. Flexible job-shop green scheduling considering transportation time and machine preventive maintenance. *Computer Integrated Manufacturing Systems*, 2024, 30(9): 3111 – 3124.
(张洪亮, 徐公杰, 鲍蔷, 等. 考虑运输时间和机器预维护的柔性作业车间绿色调度. 计算机集成制造系统, 2024, 30(9): 3111 – 3124.)

[27] ZHONG Xiaoyu, HAN Yuyan, YAO Xiangjuan, et al. An evolutionary algorithm for the multi-objective flexible job shop scheduling problem with uncertain processing time. *Scientia Sinica (Informationis)*, 2023, 53(4): 737 – 757.
(钟小玉, 韩玉艳, 姚香娟, 等. 不确定工时下多目标柔性作业车间调度问题的进化求解方法. 中国科学: 信息科学, 2023, 53(4): 737 – 757.)

[28] WANG Y F, GE J R, MIAO S, et al. Application of hybrid artificial bee colony algorithm based on load balancing in aerospace composite material manufacturing. *Expert Systems with Applications*, 2023: 119375.

作者简介:

王玉芳 博士, 副教授, 研究生导师, 目前研究方向为智能优化算法及生产车间调度, E-mail: wangyufang@nuist.edu.cn;

郑明波 硕士研究生, 目前研究方向为智能优化算法及生产车间调度, E-mail: 2441686421@qq.com;

张 艺 硕士研究生, 目前研究方向为智能优化算法及生产车间调度, E-mail: 820107163@qq.com;

华晓麟 硕士研究生, 目前研究方向为智能优化算法及生产车间调度, E-mail: 1922163398@qq.com;

章殿清 硕士研究生, 目前研究方向为智能优化算法及生产车间调度, E-mail: 842101808@qq.com.