Operations Research and Fuzziology 运筹与模糊学, 2025, 15(2), 738-749
Published Online April 2025 in Hans. https://www.hanspub.org/journal/orf
https://doi.org/10.12677/orf.2025.152121

Hans汉斯

基于分解多目标进化算法的柔性作业车间调度问题研究

师　辉

五邑大学电子与信息工程学院，广东　江门

收稿日期：2025年2月28日；录用日期：2025年4月14日；发布日期：2025年4月21日

摘　要

本文聚焦于多目标柔性作业车间调度问题(MOFJSP)，为更贴近实际生产环境的复杂性，提出了一种考虑机器设置时间和工件时滞约束的调度模型，旨在最小化最大完工时间和总能耗。本文基于分解多目标进化算法(MOEA/D)框架，结合启发式方法，对种群初始化过程进行了优化改进。通过在15个标准柔性作业车间调度实例上的对比实验，改进后的MOEA/D算法与5种现有的多目标优化算法进行了性能评估。实验结果表明，改进后的MOEA/D算法能够生成更高质量的初始解集，显著提升了算法的收敛速度，并在帕累托最优解的分散性方面表现出色。这一改进不仅增强了算法在多目标优化问题中的整体性能，还为解决复杂调度问题提供了一种高效且实用的解决方案。

关键词

柔性作业车间调度，能源消耗，MOEA/D，启发式方法

Research on Flexible Job Shop Scheduling Problem Based on Decomposition Multi Objective Evolutionary Algorithm

Hui Shi

School of Electronic and Information Engineering, Wuyi University, Jiangmen Guangdong

Received: Feb. 28th, 2025; accepted: Apr. 14th, 2025; published: Apr. 21st, 2025

Abstract

This paper focuses on the Multi-objective Flexible Job Shop Scheduling Problem (MOFJSP). In order

文章引用: 师辉. 基于分解多目标进化算法的柔性作业车间调度问题研究[J]. 运筹与模糊学, 2025, 15(2): 738-749.
DOI: 10.12677/orf.2025.152121

to be closer to the complexity of the actual production environment, a scheduling model considering machine setup time and job delay constraints is proposed to minimize the maximum completion time and total energy consumption. Based on the framework of Decomposed Multi-objective Evolutionary Algorithm (MOEA/D), combined with heuristic methods, the population initialization process is optimized and improved. Through comparative experiments on 15 standard flexible job shop scheduling instances, the performance of the improved MOEA/D algorithm is evaluated with five existing multi-objective optimization algorithms. Experimental results show that the improved MOEA/D algorithm can generate higher quality initial solution sets, significantly improve the convergence speed of the algorithm, and perform well in the dispersion of Pareto optimal solutions. This improvement not only enhances the overall performance of the algorithm in multi-objective optimization problems, but also provides an efficient and practical solution for solving complex scheduling problems.

Keywords

Flexible Job Shop Scheduling, Energy Consumption, MOEA/D, Heuristic Method

1. 引言

柔性作业车间调度问题(Flexible Job Shop Scheduling Problem, FJSP)是一种具有高度复杂性的大规模组合优化问题，广泛应用于多品种小批量生产任务的排产场景。在现代制造企业中，当接到客户订单后，企业需根据订单任务要求、车间设备配置以及人力资源状况制定相应的调度方案。订单任务通常涉及多个不同类别的产品，而车间设备则包含多台功能各异的机器。每个产品的加工过程需经过多道工序，且每道工序均可在一组指定机器中的任意一台上进行加工。图 1 展示了一个简单的 FJSP 示例。

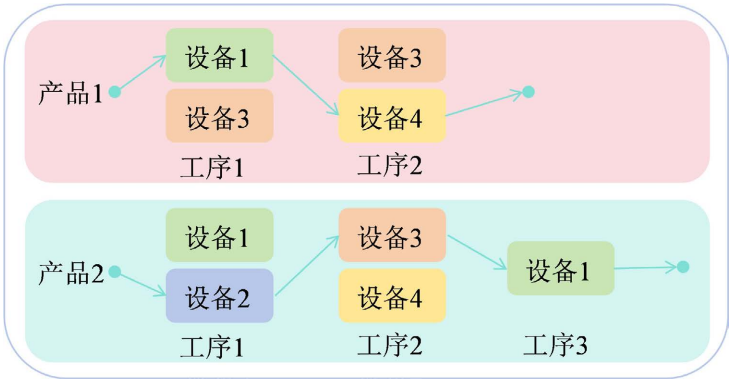

Figure 1. Simple flexible job shop scheduling example
图 1. 简单的柔性作业车间调度示例

在生产企业的排产过程中，通过优化多个目标维度以实现生产效率的最大化是关键所在。常见的优化目标包括最小化最大完工时间、实现机器负载均衡、最小化总机器负载、降低瓶颈机器的负载、减少延迟惩罚以及控制总成本等。随着绿色制造理念的兴起，为了有效控制成本并减少环境污染，最小化能

源消耗或碳排放也逐渐成为生产企业亟需考虑的重要优化指标。排产调度的核心在于通过对上述多个目标的同时优化，借助高效的优化算法寻找最优可行解，从而为企业的生产活动提供科学指导。然而，对于多目标柔性作业车间调度问题(Multi-objective Flexible Job Shop Scheduling Problem, MOFJSP)而言，其面临的挑战在于如何在提升算法收敛能力的同时，增强最优解的多样性，以确保在复杂多变的生产环境中实现高效且可持续的生产调度。

针对 MOFJSP，学术界提出了众多有效的算法。基于分解的多目标进化算法(Decomposed Multi-objective Evolutionary Algorithm, MOEA/D) [1]作为一种经典框架，已被广泛应用于解决多目标优化问题，并衍生出大量改进算法。Wang 等人提出用基于距离和角度的评估指标识别最差子问题，并通过局部搜索强化该子问题的解的质量[2]。Xiao 等人根据种群的进化程度自适应调整邻域大小，并提出一种全局更新策略，使得新解不仅替换子问题的邻域，还能替换最适合新解的子问题邻域[3]。Li 等人用 Q-learning 算法自适应调整邻域大小，并提出一种自适应方法为每个解选择合适的局部搜索算子[4]。Wu 等人在为子问题生成新解后，采用局部搜索进一步优化，并提出用聚类方法更新邻域解。此外，他们还提出根据作业相似度对染色体解码，以平衡两个优化目标之间的关系[5]。Liu 等人提出根据进化代数自适应选择交叉策略和调整变异概率的方法，并对新解采用四种邻域结构进行局部搜索[6]。Jiang 等人提出利用子问题邻域的非支配解进行合作搜索，并针对优化目标设计了一种强化的局部搜索算子[7]。Shen 等人在更新子问题邻域解时引入精英档案，并提出了一种自适应选择差分进化算子的策略，以提高算法的搜索能力和解的质量[8]。这些研究通过结合 MOEA/D 框架与不同的搜索算子和改进策略，显著提升了算法在解决 MOFJSP 时的性能。但现有研究中关于种群初始化的策略大多仅依赖于简单的启发式算法。这种初始化方法难以满足复杂优化问题的多样化需求，并可能导致算法收敛速度较慢以及最优解多样性不足的问题。因此，如何设计更有效的种群初始化策略，以提升算法的整体性能，仍然是当前研究中的一个重要挑战。

2. 问题建模

2.1. 问题描述

FJSP 可以定义为：给定一组工件 $J=\{J_1,J_2,\cdots,J_n\}$ 和一组机器 $M=\{M_1,M_2,\cdots,M_m\}$。每个工件 J_i 包含多道工序 $O_i=\{O_{i,1},O_{i,2},\cdots,O_{i,P_i}\}$，同一工件的工序需按既定工序加工。每道工序可在一组指定的可选机器 $M_{i,j}$ 上加工。该问题有以下基本假设：在初始时刻(0 时刻)，所有作业和机器均处于就绪状态；不考虑工件在机器间的运输时间；每个机器前的缓冲区容量无限，可容纳任意数量的工件；每台机器在同一时间只能加工一个工件；每道工序只能在一台机器上完成；一旦机器开始加工某工序，机器不会停机，直到该工序完成。

本研究在 MOFJSP 中引入了机器设置时间和工件时滞时间这两个关键因素。其中，机器设置时间是指在加工工件之前，为调整机器状态(如更换刀具、清洁机器等)所需的时间；而工序时滞是指某道工序完成后，工件需要经历的冷却或加热的等处理时间。这两种时间因素在相关文献中已被应用[9] [10]。对于每道工序，不同可选机器的设置时间可能不同。若同一机器上连续加工的两道工序属于同一工件的相邻工序，那么不考虑设置时间。此外，本研究要优化的目标是最小化最大完工时间和总能源消耗。

2.2. 模型构建

在本节中，针对上述问题构建了相应的数学模型。首先，定义了一些必要的符号和决策变量。模型所涉及的符号及含义如表 1 所示，模型变量如表 2 所示。

Table 1. Symbol and meanings of models
表 1. 模型的符号及含义

符号	说明
n	工件数量
m	机器数量
i	工件下标
j	工序下标
k	机器下标
l	某机器第 l 个加工位置
l_k	机器 k 上最大的 l 值
P_i	工件 J_i 的工序数
$O_{i,j}$	工件 J_i 的第 j 个工序
$P_{i,j,k}$	工序 $O_{i,j}$ 在机器 k 上的处理时间
$S_{i,j,k}$	工序 $O_{i,j}$ 在机器 k 上的设置时间
$D_{i,j}$	工序 $O_{i,j}$ 的时滞
$EP_{i,j,k}$	工序 $O_{i,j}$ 在机器 k 上单位时间的处理能源消耗
$ES_{i,j,k}$	工序 $O_{i,j}$ 在机器 k 上单位时间的设置能源消耗
EI_k	机器 k 上单位时间的空闲能源消耗
C_i	工件 i 的最大完成时间

Table 2. Decision variable and their implications of models
表 2. 模型的决策变量及含义

符号	说明
$x_{i,j,k,l}$	如果工序 $O_{i,j}$ 在机器 k 的第 l 个位置加工，则 $x_{i,j,k,l}=1$，否则 $x_{i,j,k,l}=0$
$s_{k,l}$	机器 k 的第 l 个位置的设置开始时间
$e_{k,l}$	机器 k 的第 l 个位置的处理结束时间

然后，对该问题构建的数学模型如下。等式 1 和等式 2 分别用于最小化最大完工时间和总能耗。等式 3 至等式 5 分别给出了加工能源消耗(PEC)，设置能源消耗(SEC)，空闲能源消耗(IEC)的计算公式。等式 6 确保每道工序仅在一个机器加工一次。等式 7 描述了在机器 k 上位置 l 的设置开始时间与加工结束时间的关系。约束 8 规定，工件的第二道工序及之后工序的加工开始时间取决于两部分：一是前一道工序的加工结束时间加上时滞，二是当前工序所在机器的加工结束时间加上设置时间。约束 9 规定，在工序分配到机器时，应优先选择前面的空闲位置。约束 10 确保在同一机器上，后一位置的设置开始时间不得早于前一位置的加工结束时间。约束 11 要求任意机器位置的设置开始时间必须为正值。

$$\min F_1 = \max C_i, 1 \le i \le n \tag{1}$$

$$\min F_2 = PEC + SEC + IEC \tag{2}$$

$$PEC = \sum_{i=1}^{n}\sum_{j=1}^{P_i}\sum_{k=1}^{m}\sum_{l=1}^{l_k} P_{i,j,k} * x_{i,j,k,l} * EP_{i,j,k} \tag{3}$$

$$SEC = \sum_{i=1}^{n}\sum_{j=1}^{P_i}\sum_{k=1}^{m}\sum_{l=1}^{l_k} S_{i,j,k} * x_{i,j,k,l} * ES_{i,j,k} \tag{4}$$

$$IEC = \sum_{k=1}^{m}\left[e_{k,l_k} - \sum_{l=1}^{l_k}(e_{k,l} - s_{k,l}) \right] * EI_k \tag{5}$$

$$\sum_{k=1}^{m}\sum_{l=1}^{l_k} x_{i,j,k,l} = 1, \forall 1 \le i \le n, 1 \le j \le P_i \tag{6}$$

$$e_{k,l} = s_{k,l} + \sum_{i=1}^{n}\sum_{j=1}^{P_i}\left(S_{i,j,k} + P_{i,j,k}\right) * x_{i,j,k,l}, \forall 1 \le k \le m, 1 \le l \le l_k \tag{7}$$

$$\begin{aligned} &\max\left[\sum_{k=1}^{m}\sum_{l=1}^{l_k}\left(x_{i,j,k,l} * e_{k,l}\right) + D_{i,j}, \sum_{k=1}^{m}\sum_{l=1}^{l_k} x_{i,j+1,k,l} * \left(e_{k,l-1} + S_{i,j+1,k}\right)\right] \\ &\le \sum_{k=1}^{m}\sum_{l=1}^{l_k} x_{i,j+1,k,l} * \left(s_{k,l} + S_{i,j+1,k}\right), \forall 1 \le i \le n, 1 \le j \le P_j - 1 \end{aligned} \tag{8}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{P_i} x_{i,j,k,l-1} \ge \sum_{i=1}^{n}\sum_{j=1}^{P_i} x_{i,j,k,l}, 1 \le k \le m, 2 \le l \le l_k \tag{9}$$

$$s_{k,l} \ge e_{k,l-1}, \forall 1 \le k \le m, 2 \le l \le l_k \tag{10}$$

$$s_{k,l} \ge 0, \forall 1 \le k \le m, 1 \le l \le l_k \tag{11}$$

3. 改进的 MOEA/D

3.1. MOEA/D

MOEA/D 的核心思想在于通过将多目标优化问题分解为多个相互关联的单目标子问题来简化问题的求解过程。每个子问题通过其邻域子问题生成新解，并将新解反馈至邻域子问题中，从而实现信息的共享与传递。这种机制使得后续子问题能够充分利用前序子问题产生的优质解，进而加速算法的收敛并提升解的质量。MOEA/D 框架具有高度的灵活性，能够集成多种搜索算子，分解方法和解的更新方法。这里给出了契比雪夫聚合函数的公式，如公式 12 所示。

$$g^{te}\left(x^i \mid \lambda^i, z^*\right) = \max_{1 \le j \le m}\left\{\lambda_j^i \left| f_j\left(x^i\right) - z_j^* \right|\right\} \tag{12}$$

其中，$g^{te}\left(x^i \mid \lambda^i, z^*\right)$ 表示子问题 i 的目标函数值，λ^i 表示子问题 i 的权重向量，z^* 是参考点，m 表示目标维度，λ_j^i 表示 λ^i 第 j 维的值，$f_j\left(x^i\right)$ 表示 x^i 的第 j 维目标值，z_j^* 表示 z^* 第 j 维的值。

3.2. 初始种群策略

FJSP 可分解为两个关键子问题：一是确定所有工序的加工顺序，二是为每道工序分配合适的机器。为此，染色体编码需要采用双层编码结构，分别表示工序处理序列和机器分配序列。现有文献[3] [4]中提及的工序序列初始化方法包括：将处理时间最短的工序排在前面、将剩余处理时间最长的工件排在前面以及随机排列工序顺序。对于机器序列的初始化方法，则包括选择当前负载最小的机器、选择能耗最低的机器、选择处理时间最短的机器以及随机选择机器。然而，这些初始化方法在提升初始种群多样性方面存在局限性。

例如，在初始化包含 100 个个体的种群时，若每个个体的工序处理序列均采用随机排列方式生成，同时，机器分配序列的生成策略为：50%的个体选择处理完成时间最早的机器，而另外 50%的个体选择

能源消耗最少的机器。这种初始化策略在处理小规模实例(如Mk01)时，尚能在一定程度上满足优化需求，但对于大规模实例(如DP01)，其初始种群的多样性则明显不足。如图2所示，红色点表示选择处理完成时间最早的机器生成的解，蓝色点表示选择能源消耗最少的机器生成的解。由此可见，这种初始化方法在处理大规模实例时，难以有效覆盖目标空间，限制了算法的全局搜索能力和解的多样性。

为克服传统初始化方法在多样性方面的不足，本研究提出了一种改进的染色体机器分配序列编码策略。具体而言，在对单个染色体的机器分配序列进行编码时，同时融合了两种启发式规则：选择处理完成时间最早的机器和选择能源消耗最少的机器。这一策略使得每条染色体的机器分配序列能够综合考虑加工效率与能耗优化。

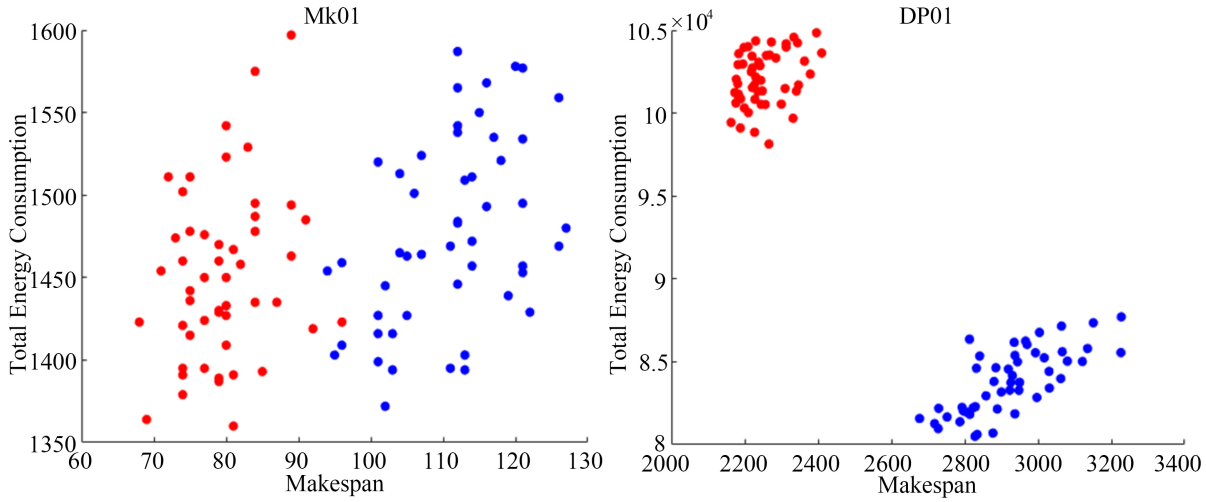

Figure 2. Initial population distribution without improvement
图2. 未改进前的初始种群分布

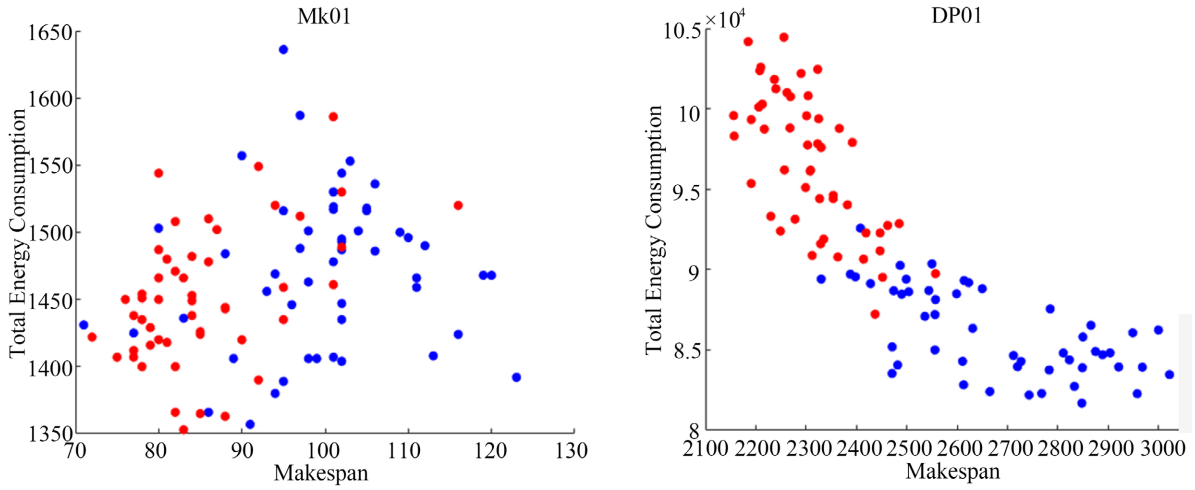

Figure 3. Initial population distribution after the improvement
图3. 改进后的初始种群分布

进一步地，在为每道工序分配机器时，引入概率机制：以概率 p 选择处理完成时间最早的机器，以概率 1 − p 选择能源消耗最少的机器。通过这种方式，机器分配序列的生成将根据概率 p 的取值，灵活偏向于其中一种启发式规则。这种基于概率的初始化方法不仅增强了种群的多样性，还提升了算法在不同

规模实例上的适应性。例如，对于实例 Mk01 和 DP01，采用该初始化方法后，其初始种群的多样性表现如图 3 所示。结果显示，该种群初始化方法能够有效改善种群的多样性，为后续优化过程提供更丰富的目标空间覆盖。该种群初始化方法的具体步骤如下：

步骤 0：设定概率值 p，表示为工序分配机器时，以概率 p 选择处理完成时间最早的机器，以概率 1 − p 选择能源消耗最少的机器。概率值 p 的取值范围是{0, 0.1, 0.2, ···, 1}，共 11 种可能，因此存在 11 种不同的初始化策略。

步骤 1：对于每一种初始策略，生成 10 个个体。工序处理序列采用随机排列方式生成，机器分配序列则根据概率 p 和 1 − p 选择处理完成时间最早的机器和能源消耗最少的机器。通过这种方式，11 种初始策略共生成 110 个初始个体。

步骤 2：运用非支配排序算法和拥挤度距离计算方法[11]从 110 个个体筛选最好的 N = 100 个个体构成初始种群。

3.3. IMOEA/D 算法

Table 3. The algorithm flow of IMOEA/D
表 3. IMOEA/D 的算法流程

算法 1：IMOEA/D 的算法流程

输入：种群大小 N，评估次数 FE = 0，最大评估次数 maxFE = 20000，参数设置

输出：精英档案 EP

步骤 1：初始化

1.1 初始化均匀分布的权重向量 $\lambda^1,\lambda^2,\cdots\lambda^N$，为每个子问题找到其邻域 $B(i)$。EP = ∅

1.2 使用 3.2 节的初始化策略初始种群 $x^1,x^2,\cdots,x^N$，计算参考点 $z^*=\left(z_1^*,\cdots,z_m^*\right)$，设置 $z_j^*=\ min\left\{f_j\left(x^1\right),\cdots,f_j\left(x^N\right)\right\}$

1.3 初始化每个子问题的效益值 Pi = 1

while FE < maxFE

for k = 1:5

步骤 2：选择子问题：设置 I = ∅，选择 m 个子问题的下标存入 I 中，这些子问题的目标是 f_j。使用 10 元锦标赛从种群中选择 N/5-m 个子问题的下标存入 I 中。设置 OffSprings = ∅

for i = I

步骤 3：进化

3.1 选择交配范围 P：生成一个(0,1)的随机数 rand，设置 $\delta = 0.9$

$$\mathrm{P}=\begin{cases}B(i)\cdots, rand\le\delta\\ \{1,2,\cdots N\}, rand\ge\delta\end{cases}$$

3.2 繁殖：从 P 中随机选择两个值 r_1 和 r_2，并通过交叉和变异算子生成一个新解 x_{new}，添加 x_{new} 到 OffSprings，FE = FE + 1

3.3 更新参考点 z^*：对于所有的 j，如果 $f_j\left(x_{new}\right)<z_j^*$，更新 $z_j^*\ =\ f_j\left(x_{new}\right)$

3.4 更新解：设置 c = 0，然后执行下面步骤：

(1) 随机从 P 中选择一个值 k，根据等式 12 计算 x_{new} 和 x^k 的函数值，若 $g^{te}(x_{new}\mid\lambda^k,z^*)\le g^{te}(x^k\mid\lambda^k,z^*)$，则将 x^k 替换为 x_{new}。c = c + 1

(2) 将 k 从 P 中删除，若 c = nr，返回到步骤 4；否则返回到(1)

end

end

续表

步骤 4：更新子问题的效益值

4.1 若 EP 是 1000 的倍数，计算每个子问题 i 目标的相对减少量：

$$\Delta^i = \frac{\text{old function value} - \text{new function value}}{\text{old function value}}$$

4.2 更新每个子问题的效益值：

$$\pi^i = \begin{cases} 1 & ,\Delta^i > 0.001 \\ \left(0.95 + 0.05 * \frac{\Delta^i}{0.001}\right)\pi^i & ,\Delta^i \le 0.001 \end{cases}$$

步骤 5：更新精英档案 EP：将 Offspring 与 EP 合并，并将非支配解存入 EP，同时去除重复解。

步骤 6：权重向量调整：若 FE > 0.8* maxFE 且 FE % 1000 == 0，调整权重向量如下：

6.1 从 EP 和种群中选择与权重向量最匹配的解，构成外部种群。

6.2 从外部种群中删除拥挤度大的解和对应的权重向量

6.3 从 EP 中添加新的解和生成新的权重向量，构成新的种群和权重向量。

end

本研究针对 FJSP，提出了一种改进的基于分解的多目标进化算法(Improved Multi-objective Evolutionary Algorithm Based on Decomposition, IMOEA/D)，IMOEA/D 的算法流程如表 3 所示。在步骤 2 中，算法根据子问题的效益值选择待更新子问题，效益值反映了子问题在优化过程中的难易程度或改进潜力，帮助算法动态调整资源分配，以提高整体优化效率。在步骤 3 中，进化操作采用交叉和变异算子，在交叉操作中，工序处理序列采用优先工序交叉，机器分配序列采用多点交叉；在变异操作中，工序处理序列采用交换变异，机器分配序列采用多点变异[4]。在步骤 6 中，通过权重向量调整策略，算法能够有效获得更均匀的非支配解。该权重向量调整策略的具体细节可参考文献[12]。

4. 仿真实验

4.1. 实验数据

本节通过不同规模的柔性作业车间调度实例[13]，对本文提出的 IMOEA/D 算法的有效性进行了验证。其中，Mk01 至 Mk10 实例为中等规模实例，DP01 至 DP05 为大规模实例。这些实例的相关数据如表 4 所示。

Table 4. The job quantity, machine quantity and maximum operation quantity for jobs of the instances
表 4. 实例的工件数、机器数和工件的最大工序数

实例	工件数	机器数	工件的最大工序数
Mk01	10	6	6
Mk02	10	6	6
Mk03	15	8	10
Mk04	15	8	9
Mk05	15	4	9
Mk06	10	10	15
Mk07	20	5	5
Mk08	20	10	14
Mk09	20	10	14

续表

Mk10	20	15	14
DP01	30	10	10
DP02	30	10	10
DP03	30	10	10
DP04	30	10	10
DP05	30	10	10

4.2. 实验说明

本研究选取五种多目标优化算法与 IMOEA/D 进行性能对比，这些算法分别是 IMOEAD_HS [2]，MOEAD_AWA [12]，MOEAD_DRA [14]，MOEA/D，RVEA [15]。其中，IMOEAD_HS 是针对 MOFJSP 设计的算法，RVEA 是参考向量引导的进化算法，MOEAD_AWA 采用自适应权重向量调整机制，MOEAD_DRA 则对 MOEA/D 的父本选择方式进行了改进。

为了量化多目标优化算法性能，本研究采用超体积(HV) [16]和分散性(Spread) [17]作为性能评价指标。HV 指标衡量的是算法所得到的帕累托(Pareto)前沿与预设的最大目标值参考点之间形成的封闭空间的体积，该指标综合反映了算法的收敛性和多样性。具体而言，HV 值越大，表明算法所得到的 Pareto 前沿与参考点围成的区域体积越大，算法性能越优。在本研究中，为便于计算 HV 值，对目标值进行了归一化处理，选取的参考点为(1, 1)。分散性衡量算法生成的 Pareto 最优解集在真实 Pareto 前沿上的分散程度。由于实际工程优化问题的真实 Pareto 前沿未知，故用所有对比算法求得的 Pareto 最优解集并集中的最优解来代替。spread 值越小表示 Pareto 最优解的分散程度越高。

本研究选取的 15 个实例在每个算法上单独运行 30 次，对 30 次结果取均值作为最终实验结果。此外，为了比较 IMOEA/D 和其他对比算法之间是否有显著差异，通常采用显著水平为 0.05 的 Wilcoxon 符号秩检验。如果检验结果小于 0.05，则说明算法之间存在显著差异。本文中将 IMOEA/D 与其他算法从性能及差异显著性两个方面进行比较，显著优于用“+”表示，无显著差别用“≈”表示，显著劣于用“−”表示。IMOEA/D 及其他算法的相同参数包括：算法终止条件是最大目标函数的评估次数为 20000、种群大小为 100、邻域大小为 10、交叉概率为 1、变异概率为 1/D，D 为机器分配序列染色体的长度。其他没有共用的参数参考原论文中的参数设置。

4.3. 实验结果与分析

Table 5. Results of HV values for each algorithm
表 5. 各算法的 HV 值结果

实例	IMOEAD_HS	RVEA	MOEAD_AWA	MOEA/D	MOEAD_DRA	IMOEA/D
Mk01	0.6319 (−)	0.7579 (−)	0.7795 (≈)	0.7770 (≈)	0.7692 (−)	**0.7889**
Mk02	0.2902 (−)	0.6422 (−)	0.6706 (−)	0.6615 (−)	0.6437 (−)	**0.7610**
Mk03	0.3853 (−)	0.8504 (−)	**0.9050** (≈)	0.8625 (−)	0.8789 (−)	0.8965
Mk04	0.4520 (−)	0.7106 (−)	0.7145 (−)	0.6746 (−)	0.7152 (≈)	**0.7439**
Mk05	0.3749 (−)	0.7410 (−)	0.7561 (≈)	0.7265 (−)	0.7395 (−)	**0.7695**
Mk06	0.1394 (−)	0.6112 (−)	0.6003 (−)	0.5223 (−)	0.6024 (−)	**0.8047**
Mk07	0.2648 (−)	0.7074 (≈)	0.7091 (≈)	0.6384 (−)	0.6817 (−)	**0.7355**

续表

Mk08	0.4134 (−)	0.7523 (−)	0.7961 (≈)	0.6702 (−)	0.7481 (−)	**0.8064**
Mk09	0.1503 (−)	0.6521 (−)	0.6704 (−)	0.5331 (−)	0.6618 (−)	**0.7031**
Mk10	0.1069 (−)	0.5875 (−)	0.6621 (−)	0.4747 (−)	0.6410 (−)	**0.8186**
DP01	0.1597 (−)	0.5057 (−)	0.5386 (−)	0.3940 (−)	0.5234 (−)	**0.7988**
DP02	0.1823 (−)	0.5470 (−)	0.6049 (−)	0.4602 (−)	0.6001 (−)	**0.8084**
DP03	0.1744 (−)	0.4769 (−)	0.5131 (−)	0.3989 (−)	0.4994 (−)	**0.7641**
DP04	0.1325 (−)	0.4950 (−)	0.5250 (−)	0.3880 (−)	0.5114 (−)	**0.7865**
DP05	0.1648 (−)	0.5108 (−)	0.5392 (−)	0.4158 (−)	0.5201 (−)	**0.7873**
−/≈/+	15/0/0	14/1/0	10/5/0	14/1/0	14/1/0	

Table 6. Results of Spread values for each algorithm
表 6. 各算法的 Spread 值结果

实例	IMOEAD_HS	RVEA	MOEAD_AWA	MOEA/D	MOEAD_DRA	IMOEA/D
Mk01	0.9221 (−)	0.9538 (≈)	**0.7920** (≈)	1.0001 (−)	0.8576 (≈)	0.8508
Mk02	0.9344 (−)	0.8660 (−)	0.9335 (−)	1 (−)	0.9533 (−)	**0.8322**
Mk03	1 (−)	0.9738 (−)	0.9965 (−)	1.0003 (−)	0.9965 (−)	**0.9481**
Mk04	0.9258 (≈)	0.9313 (≈)	0.9441 (≈)	1.0001 (−)	0.9222 (≈)	**0.8674**
Mk05	0.9814 (−)	0.9221 (−)	0.8783 (≈)	1.0002 (−)	0.8923 (≈)	**0.8624**
Mk06	1.0175 (≈)	1.0514 (≈)	1.0995 (≈)	**1.0005** (≈)	1.0567 (≈)	1.0674
Mk07	**0.8790** (≈)	0.9052 (≈)	0.929 (≈)	1.0005 (−)	0.9376 (≈)	0.8931
Mk08	0.9577 (−)	0.9707 (−)	0.9096 (≈)	1.0002 (−)	0.9423 (−)	**0.8877**
Mk09	0.9492 (≈)	0.9725 (≈)	1.1169 (−)	1.0019 (−)	1.0173 (≈)	**0.9489**
Mk10	1.1175 (≈)	1.0613 (≈)	1.2327 (≈)	**1.0002** (≈)	1.0948 (≈)	1.1656
DP01	1.0831 (≈)	1.1293 (−)	1.0638 (≈)	1.0064 (≈)	1.0967 (≈)	**1.0014**
DP02	1.1333 (−)	1.0963 (−)	1.1333 (≈)	**1.0154** (≈)	1.1077 (≈)	1.0277
DP03	1.0916 (≈)	**1.0257** (≈)	1.1461 (≈)	1.1155 (≈)	1.0986 (≈)	1.0748
DP04	1.1768 (≈)	1.0467 (≈)	1.1121 (≈)	1.1610 (−)	1.0824 (≈)	**1.0444**
DP05	1.0737 (−)	0.9925 (≈)	1.0953 (−)	1.1468 (−)	1.0465 (≈)	**0.9595**
−/≈/+	7/8/0	6/9/0	4/11/0	10/5/0	3/12/0	

表 5，表 6 分别展示了 IMOEA/D 与五种对比算法在所有实例上的 HV 和 Spread 指标结果。“−/≈/+”符号用于统计 IMOEA/D 与对比算法之间的显著性差异，表格中对表现较好数据进行了加粗标注。从表 5 可以看出，IMOEA/D 在超过 60%的测试实例上，其 HV 指标显著优于五种比对算法。从表 6 可以看出，虽然 IMOEA/D 在少数实例的 Spread 指标上优于对比算法，但在大多数实例上与对比算法无显著差异，这表明在解的分布均匀性方面，IMOEA/D 与其他算法相当。综上所述，IMOEA/D 的有效性得到了验证。

为了进一步验证 IMOEA/D 和五种对比算法的收敛性能，表 7 列出了每个算法在各目标上的最优值，其中 f_1 表示最大完工时间，f_2 表示总能耗，加粗字体表示较优值。每个算法对每个实例独立运行 30 次，取其中最好的目标值作为该算法在该实例上的最优目标值。从表 7 可以看出，在 9 个实例的所有目标值

上，IMOEA/D 均优于五种对比算法。尽管在某些实例上 IMOEA/D 的目标值未能达到最好，但与最优值的差距非常小。这进一步证明了 IMOEA/D 在 MOFJSP 中的有效性。

Table 7. Results of convergence for each algorithm
表 7. 各算法的收敛性结果

实例	IMOEAD_HS		RVEA		MOEAD_AWA		MOEA/D		MOEAD_DRA		IMOEA/D	
	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
Mk01	68	**1191**	68	**1191**	68	**1191**	68	1206	68	1200	**65**	**1191**
Mk02	52	1031	51	**1000**	50	1017	49	1023	50	1010	**47**	1004
Mk03	**313**	7380	**313**	7246	**313**	7242	**313**	7498	322	7233	**313**	**7211**
Mk04	106	2675	109	2626	109	**2617**	108	2631	107	2632	**104**	2620
Mk05	275	4597	276	4579	274	4583	276	4653	271	4570	**270**	**4558**
Mk06	140	3374	144	3031	147	3052	136	3071	140	3085	**131**	**2959**
Mk07	227	4378	228	**4275**	229	4325	223	4336	**221**	4285	**221**	4279
Mk08	**779**	20,258	**779**	**20,083**	**779**	20,281	780	20,365	790	20,310	782	20,099
Mk09	504	16,898	513	16,574	515	16,667	510	17,057	511	16,512	**501**	**16,413**
Mk10	401	14,803	403	13,845	403	14,043	**373**	14,208	**373**	14,025	377	**13,485**
DP01	2227	83,938	2265	80,484	2284	80,618	2148	78,045	2210	80,532	**2126**	**74,906**
DP02	2434	88,786	2466	85,741	2460	85,945	2341	85,152	2445	83,298	**2335**	**82,024**
DP03	2199	80,543	2225	76,646	2240	76,653	2103	75,278	2156	75,241	**2094**	**72,159**
DP04	2261	82,882	2284	78,867	2293	79,245	**2177**	76,063	2214	75,613	**2177**	**74,437**
DP05	2275	84,250	2304	81,114	2328	80,481	**2169**	80,343	2177	79,825	2181	**76,165**
较优	2	1	2	4	2	2	4	0	2	0	12	11

5. 结论

本研究聚焦于 MOFJSP，综合考虑了机器设置时间和工件时滞时间因素，并构建了以总能耗和最大完工时间为目标的优化模型。为有效解决该问题，本文提出一种有效的种群初始化方法，该方法适用于不同规模的柔性作业车间调度实例，并能显著提升初始种群的质量。通过在 15 个不同规模实例上的对比实验，IMOEA/D 在 HV 指标上显著优于五种对比算法，充分证明了其在解的收敛性和多样性方面的优越性。本研究可以推广到更广泛的调度领域，为解决复杂调度问题提供了新的思路和方法。

参考文献

[1] Zhang, Q. and Li, H. (2007) MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, **11**, 712-731. https://doi.org/10.1109/tevc.2007.892759

[2] Wang, Z., He, M., Wu, J., Chen, H. and Cao, Y. (2024) An Improved MOEA/D for Low-Carbon Many-Objective Flexible Job Shop Scheduling Problem. *Computers & Industrial Engineering*, **188**, Article 109926. https://doi.org/10.1016/j.cie.2024.109926

[3] Xiao, B., Zhao, Z., Wu, Y., Zhu, X., Peng, S. and Su, H. (2024) An Improved MOEA/D for Multi-Objective Flexible Job Shop Scheduling by Considering Efficiency and Cost. *Computers & Operations Research*, **167**, Article 106674. https://doi.org/10.1016/j.cor.2024.106674

[4] Li, R., Gong, W. and Lu, C. (2022) A Reinforcement Learning Based RMOEA/D for Bi-Objective Fuzzy Flexible Job

Shop Scheduling. *Expert Systems with Applications*, **203**, Article 117380. https://doi.org/10.1016/j.eswa.2022.117380

[5] Wu, X. and Xie, Z. (2023) Improved MOEA/D with Local Search for Solving Multi-Stage Distributed Reentrant Hybrid Flow Shop Scheduling Problem. *Expert Systems with Applications*, **232**, Article 120893. https://doi.org/10.1016/j.eswa.2023.120893

[6] Liu, Z., Liang, X., Hou, L., Yang, D. and Tong, Q. (2023) Multi-Strategy Dynamic Evolution-Based Improved MOEA/D Algorithm for Solving Multi-Objective Fuzzy Flexible Job Shop Scheduling Problem. *IEEE Access*, **11**, 54596-54606. https://doi.org/10.1109/access.2023.3281364

[7] Jiang, E. and Wang, L. (2018) A Modified MOEA/D for Energy-Efficient Flexible Job Shop Scheduling Problem. 2018 *IEEE International Conference on Industrial Engineering and Engineering Management* (*IEEM*), Bangkok, 16-19 December 2018, 1476-1480. https://doi.org/10.1109/ieem.2018.8607582

[8] Shen, X., Sun, Y. and Zhang, M. (2016) An Improved MOEA/D for Multi-Objective Flexible Job Shop Scheduling with Release Time Uncertainties. 2016 *IEEE Congress on Evolutionary Computation* (*CEC*), Vancouver, 24-29 July 2016, 2950-2957. https://doi.org/10.1109/cec.2016.7744162

[9] Defersha, F.M. and Rooyani, D. (2020) An Efficient Two-Stage Genetic Algorithm for a Flexible Job-Shop Scheduling Problem with Sequence Dependent Attached/Detached Setup, Machine Release Date and Lag-Time. *Computers & Industrial Engineering*, **147**, Article 106605. https://doi.org/10.1016/j.cie.2020.106605

[10] Li, M. and Lei, D. (2021) An Imperialist Competitive Algorithm with Feedback for Energy-Efficient Flexible Job Shop Scheduling with Transportation and Sequence-Dependent Setup Times. *Engineering Applications of Artificial Intelligence*, **103**, Article 104307. https://doi.org/10.1016/j.engappai.2021.104307

[11] Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002) A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, **6**, 182-197. https://doi.org/10.1109/4235.996017

[12] Qi, Y., Ma, X., Liu, F., Jiao, L., Sun, J. and Wu, J. (2014) MOEA/D with Adaptive Weight Adjustment. *Evolutionary Computation*, **22**, 231-264. https://doi.org/10.1162/evco_a_00109

[13] Zhang, G., Yan, S., Song, X., Zhang, D. and Guo, S. (2024) Evolutionary Algorithm Incorporating Reinforcement Learning for Energy-Conscious Flexible Job-Shop Scheduling Problem with Transportation and Setup Times. *Engineering Applications of Artificial Intelligence*, **133**, Article 107974. https://doi.org/10.1016/j.engappai.2024.107974

[14] Zhang, Q., Liu, W. and Li, H. (2009) The Performance of a New Version of MOEA/D on CEC09 Unconstrained MOP Test Instances. 2009 *IEEE Congress on Evolutionary Computation*, Trondheim, 18-21 May 2009, 203-208. https://doi.org/10.1109/cec.2009.4982949

[15] Cheng, R., Jin, Y., Olhofer, M. and Sendhoff, B. (2016) A Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation*, **20**, 773-791. https://doi.org/10.1109/tevc.2016.2519378

[16] Zitzler, E. and Thiele, L. (1999) Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, **3**, 257-271. https://doi.org/10.1109/4235.797969

[17] Wang, Y., Wu, L. and Yuan, X. (2010) Multi-Objective Self-Adaptive Differential Evolution with Elitist Archive and Crowding Entropy-Based Diversity Measure. *Soft Computing*, **14**, 193-209. https://doi.org/10.1007/s00500-008-0394-9