文章编号:1671-6833(2026)01-0049-09

带强制工期约束的混合柔性流水线调度

轩 华[1], 李坤博[1], 曹 颖[2]

(1. 郑州大学 管理学院,河南 郑州 450001;2. 河南科技大学 土木建筑学院,河南 洛阳 471000)

摘 要: 针对每阶段包含不相关并行机的混合柔性流水线问题,考虑强制工期和运输时间,以最小化总加权完成时间为目标建立整数规划模型,结合改进遗传算法和邻域搜索策略,提出一种人工蜂群算法和鲸鱼优化算法的混合算法以获取近优解。算法采用基于工件号编码以及 NEH 启发式法生成初始工件序列集,雇佣蜂阶段引入改进遗传算法产生更优质的工件序列,跟随蜂阶段利用 5 种邻域搜索策略以得到更好的邻域序列,在侦察蜂阶段设计基于最差解的鲸鱼优化算法提高算法搜索能力。仿真实验测试了混合人工蜂群和鲸鱼优化算法内改进项的有效性以及不同规模的算例。实验结果表明:所提出的混合算法具有较好的求解性能。

关键词:混合柔性流水线; 强制工期; ABC-WOA 混合算法; NEH 启发式法

中图分类号:TB49;N945 **文献标志码**:A **doi**:10.13705/j.issn.1671-6833.2025.04.018

混合流水线调度(hybrid flowline scheduling, HFS)在制造业中较为常见,对其进行科学的决策有助于提高企业的生产效率和竞争力。经典 HFS 假定各工件按同一方向访问所有加工阶段[1],但在实际生产中,出于工艺要求等原因,一些产品的加工可能会忽略或越过某些阶段,如某些钢种需要经过不同的工位,即产生了混合柔性流水线调度(hybrid flexible flowline scheduling,HFFS)。这种跳过阶段的特点提升了调度模型适应实际环境的能力,在包装制造、钢铁生产、半导体制造等行业均可找到其应用[2-4]。

HFFS 得到了很多学者的持续关注。针对炼钢-连铸过程,Long 等[3]考虑可调处理时间,提出结合质量改进、新精英策略和重启策略的改进遗传算法(genetic algorithm, GA)以最小化 makespan、总等待时间、处理时间与其基础处理时间的偏差。Oujana 等[5]考虑准备时间和不相关并行机约束,提出了基于线性规划技术的求解方法以最小化总拖期。轩华等[6]考虑恶化效应,提出了一种结合头脑风暴优化算法与候鸟优化算法的启发式算法以最小化 makespan 与能耗的加权和。Yu 等[7]结合基于属性特征的双向量表述提出了改进模因算法,目标是最小化滞留时间和提前/拖期惩罚。黄辉等[8]针对序列设置时间,以最小化 makespan 和负荷均衡指标为目标,提出了改进的非支配解排序遗传算法。李浩平等[9]针对批量流,结合粒子群算法和 GA 提出基于双层编码的混合算法最小化 makespan 和机床负荷平衡。

在 HFS 中,交货期是常考虑的一个关键因素,对其研究多围绕加权拖期、加权提前/拖期、延迟工件数等开展,旨在缩短产品交付时间与其交货期的差距。然而,当延迟交付对生产线影响巨大或者企业非常重视准时性时,实际生产过程则不允许延迟,即所有产品必须在强制工期内交付使用。上述考虑强制工期的 HFFS 有着广泛的工业背景[10-11]。例如,在易腐制造领域,由于农产品的易腐性,每个工件有一个无法改变的强制工期。

围绕强制工期约束所开展的研究在车间调度领域已有相关报道。Janaki 等[12]研究了带强制工期和均匀分布任务持续时间的单机调度,利用 GA 来最小化延迟时间。宋存利[13]研究了禁止拖期交付的无等待流水线调度,提出了基于有向无环图的精确搜索算法以及基于该算法的分段迭代搜索算法。Ying 等[14]为求解含强制工期的两代理比例流水线调度,提出了两种多项式时间优化算法,分别最小化

收稿日期:2025-01-12;**修订日期**:2025-03-23

基金项目:河南省省级科技研发计划联合基金项目(242103810046);河南省哲学社会科学规划项目(2023BJJ085);河南省自然科学基金资助项目(252300421945)

作者简介:轩华(1979—),女,河南睢县人,郑州大学教授,博士,主要从事生产计划与调度、智能优化算法等研究,E-mail:hxuan@zzu.edu.cn。

引用本文:轩华,李坤博,曹颖. 带强制工期约束的混合柔性流水线调度[J]. 郑州大学学报(工学版),2026,47(1):49-57. (XUAN H, LI K B, CAO Y. Hybrid flexible flowline scheduling with deadline constraints[J]. Journal of Zhengzhou University (Engineering Science),2026,47(1):49-57.)

最大完成时间和总完成时间。对于无拖期作业车间调度,Shi 等[15]考虑加班,提出了结合模拟退火的混合 GA 以最小化总提前库存和加班成本;史双元等[16]考虑加班,提出了多目标金鹰优化算法以最小化总加班时间和 makespan;史双元等[17]考虑外协,提出了多目标差分进化-变邻域搜索算法以最小化 makespan 和总外协成本。

综上所述,本文提出了考虑强制工期约束的 HFFS(HFFS with deadline constraints,HFFSDC),由于 HFFS 是 NP-hard 问题[5],故所研究的更复杂的 HFFSDC 也是 NP-hard 问题。本文的贡献如下。

(1)虽然从不同角度研究 HFFS 以及带强制工期的生产调度问题已有一些成果,但在 HFFS 中兼顾强制工期的研究还相对较少。强制工期的引入增加了问题的求解复杂度,若使用常规编码解码策略,可能会产生超出强制工期约束的不可行解。因此,本文结合机器空闲规则、最早完成时间规则以及基于强制工期的前移策略提出两阶段动态解码方案。

(2)在以往研究中,处理 HFFS 时多假定工件访问所有加工阶段,但将工件在跳过阶段的处理时间视为零,该方式需对工件在所有阶段的机器选择和工件排序进行安排。为了减少工件跳过阶段对其他阶段的安排产生影响且提高运行效率,本文通过设置工件实际加工阶段集的方式,确定工件实际所需访问的阶段。

(3)为有效解决 HFFSDC,引入改进 GA 和邻域搜索提高人工蜂群算法(ABC)的局部搜索能力,引入基于最差解的鲸鱼优化算法(WOA)防止算法过快收敛。因此,结合改进 GA、邻域搜索策略提出 ABC-WOA 混合算法以获取近优解。

1 问题建模

1.1 问题描述

所研究的 HFFSDC 包含 n 个工件和 o 个加工阶段,每个阶段 $j(j\in\{1,2,\cdots,o\})$ 有 $m_j(m_j\geqslant 1)$ 台不相关并行机;工件 $i(i\in\{1,2,\cdots,n\})$ 的加工过程可能跳过一些阶段,即它遍历的加工阶段数可能会小于总阶段数。令 u^i 为工件 i 所经加工阶段的集合,则工件 i 实际访问的阶段数可表示为 $|u^i|$。$u^i(g)$ 为 u^i 中排在第 g 个位置的阶段号,$g\in\{1,2,\cdots,u_i\}$。工件 i 在相邻两个未缺失阶段 $u^i(g)$ 和 $u^i(g+1)$ 之间的运输时间为 $T^i_{u^i(g),u^i(g+1)}$,st_{ji} 为工件 i 在阶段 j 的开始时间,ct_{ji} 为工件 i 在阶段 j 的完成时间,p_{jki} 为工件 i 在阶段 j 的机器 k 上的处理时间。设 X_{jki} 为二元变量,若工件 i 在阶段 j 由机器 k 加工,$X_{jki}=1$,否则 $X_{jki}=0$。每个工件有对应的强制工期 $\overline{d}_i$,这意味着各工件必须在该时间内完成加工,因此在确定工件加工序列时需要核查和调整以满足强制工期要求。当工件 i 访问阶段 j 时,由于该阶段的各台机器相互独立,所以不同的机器选择会影响工件在此阶段的处理时间,进而影响它的完成时间。调度的目的是决定工件的加工序列以及各工件的机器选择以使总加权完成时间最小。其他的假设描述包括:①所有工件在零时刻均可用于加工;②每道工序的加工不允许中断;③机器在任何时候都可用,不考虑故障和维护;④每台机器一次只能处理一个工件,每个工件一次只能由一台机器加工。

1.2 模型构建

HFFSDC 的整数规划模型的构建如下。

优化目标为最小化总加权完成时间:

$$\min z=\sum_{i=1}^{n} w_i\cdot ct_{u^i(|u^i|)i}。\tag{1}$$

式中:w_i 为工件 i 的权重。

确保每个工件在任意一个未缺失阶段只能由一台机器加工:

$$\sum_{k=1}^{m_{u^i(g)}} X_{u^i(g)ki}=1,\forall i,g。\tag{2}$$

工件在每个未缺失阶段的开始和完成时间关系为

$$ct_{u^i(g)i}=st_{u^i(g)i}+\sum_{k=1}^{m_{u^i(g)}} X_{u^i(g)ki}\cdot p_{u^i(g)ki},\forall i,g。\tag{3}$$

同一阶段分配到同一台机器加工的两个工件之间的加工优先级关系分别为

$$Z_{jkli}+Z_{jkil}=1,\forall i,l,j,k,l\neq i;\tag{4}$$

$$s_{jki}+V(3-X_{jki}-X_{jkl}-Z_{jkli})\geqslant c_{jkl},\forall i,l,j,k,l\neq i。\tag{5}$$

式中:Z_{jkli} 为二元变量,若工件 l 和 i 均分配到阶段 j 的机器 k 且工件 l 是工件 i 的紧前工件,$Z_{jkli}=1$,否则 $Z_{jkli}=0$;s_{jki} 和 c_{jki} 分别为工件 i 在阶段 j 的机器 k 上的开始时间和完成时间;V 为一个很大的数。

工件在相邻两个未缺失阶段的工序优先级关系为

$$st_{u^i(g)i}+\sum_{k=1}^{m_{u^i(g)}} X_{u^i(g)ki}\cdot p_{u^i(g)ki}+T^i_{u^i(g),u^i(g+1)}\leqslant st_{u^i(g+1)i},\forall i,g\in\{1,\cdots,|u^i|-1\}。\tag{6}$$

为确保虚拟工件 0 是安排在每台机器上第一个工件的紧前工件有

$$\sum_{i=1}^{n} Z_{jk0i}\leqslant 1,\forall j,k。\tag{7}$$

为保证工件 i 在最后阶段的完成时间不能超过强制工期有

$$ct_{u^i(|u^i|)i} \leqslant \overline{d}_i, \forall i。\tag{8}$$

变量的取值为

$$Z_{jkli}, X_{jki} \in \{0,1\}, \forall j,k,i,l,l \neq i;\tag{9}$$

$$c_{jki}, s_{jki}, st_{ji}, ct_{ji} \geqslant 0, \forall j,k,i。\tag{10}$$

2　算法设计

2.1　总体流程

ABC-WOA 混合算法的具体流程概括如下。

步骤 1　参数设置。设置迭代标号 q、总迭代数 $iter$、序列集规模 $Popsize$、总累计迭代数 lj、累计未改进次数 $\varphi=0$、迭代次数 $ik=1$。

步骤 2　初始化。采用 NEH 启发式法产生 $Popsize$ 个工件序列构成初始工件序列集 τ_q。

步骤 3　若达到总迭代数,输出最佳解,否则记录当前最佳解,继续步骤 4。

步骤 4　雇佣蜂阶段。根据轮盘赌规则从工件序列集 τ_q 选择 $Popsize$ 个工件序列执行交叉变异算子,将产生的子代与序列集 τ_q 合并。计算所形成的规模为 $2Popsize$ 的新序列集内每个工件序列的适应度值,按其降序排列工件序列,选取适应度值大的前 $Popsize$ 个工件序列作为新序列集 τ_q。

步骤 5　跟随蜂阶段。针对新序列集 τ_q,应用轮盘赌规则选择 $Popsize$ 个工件序列,对其执行 5 种邻域搜索策略。若新工件序列的适应度值较大,则用新工件序列取代序列集 τ_q 内对应的工件序列。

步骤 6　计算更新后的序列集 τ_q 内各工件序列的目标值,选择具有最小总加权完成时间的解作为最佳解,若它优于历史最佳解,令 $ik=ik+1$,$\varphi=0$,返回步骤 3;否则,令 $\varphi=\varphi+1$,若 $\varphi<lj$,令 $ik=ik+1$,转至步骤 3,若 $\varphi \geqslant lj$,继续步骤 7。

步骤 7　侦察蜂阶段。将工件序列集 τ_q 按照适应度值升序排列,选择前 $Popsize/2$ 个工件序列,随机利用鲸鱼优化算法的线性包围策略和螺旋包围策略。无论是否产生适应度值更大的新工件序列,都使用新工件序列取代原序列。令 $ik=ik+1$,返回步骤 3。

2.2　初始化方法

2.2.1　*编码和解码*

采用基于工件排序的整数编码方式,取所有工件的工件号组成一个工件序列,其长度等于总工件数 n。工件号所处位置即为该工件的加工顺序。根据机器空闲规则、最早完成时间规则以及基于强制工期的前移策略进行两阶段动态解码以确定工件的加工安排。

步骤 1　对于虚拟工件 0,令 $ct_{j0}=0(j \in \{1,2,\cdots,o\})$;

步骤 2　按照给定工件序列依次计算工件 i 在每个未缺失阶段的机器上的开始时间 $s_{u^i(g)ki}$、完成时间 $c_{u^i(g)ki}$ 以及记录每台机器的负载。计算工件开始时间时,比较该工件在前一个未缺失阶段的完成时间 $c_{u^i(g-1)ki}$ 与运输时间 $T^i_{u^i(g-1),u^i(g)}$ 之和(记为 CT)和所选机器上工件 i 的紧前工件的完成时间(记为 QT),若 $CT \geqslant QT$,则 $s_{u^i(g)ki}=CT$;若 $CT<QT$,如果该机器在 CT 和工件 i 的紧前工件的开始时间(记为 BT)之间存在长度为 $p_{u^i(g)ki}$ 的空闲时间段,则将该工件安排至该时间段,记录最早空闲时间段的起点作为 $s_{u^i(g)ki}$,否则 $s_{u^i(g)ki}=QT$。进而,得到 $c_{u^i(g)ki}$,更新该机器在各时刻的负载状态,将工件 i 记入已安排工件子序列。

步骤 3　在决定工件 i 的加工机器时,利用基于最早完成时间规则的机器选择策略,即计算工件 i 在当前阶段的所有并行机上的完成时间,选取完成时间最早的一台机器作为工件 i 在该阶段的加工机器。若最早完成时间相同,则选择机器号最小的机器。

步骤 4　若工件 i 的完成时间超过强制工期,则将该工件的加工位置前移,即从{1, 2, 3}产生随机数 rd,将工件 i 前移 rd 个位置,根据前述规则确定前移工件 i 以及后移的 rd 个工件的开始/完成时间和处理这 $rd+1$ 个工件的机器;若移动后工件 i 的完成时间满足强制工期约束,则继续判断已安排工件子序列内下一工件,否则重复该前移过程,直至已安排工件子序列内所有工件满足强制工期约束。返回步骤 2,直至完成所有工件。

2.2.2　*初始工件序列集*

引入 NEH 启发式法产生 $Popsize$ 个工件序列以构成初始工件序列集。计算每个工件在所有机器的总处理时间,按照降序排列各工件;选取总处理时间最长的前两个工件作为部分工件序列;随机选取一个未加工工件,插入到部分工件序列的所有可能位置,计算此时该序列的总加权完成时间,选择总加权完成时间最小的序列作为这些工件的当前序列。重复上述步骤直至完成所有工件的插入,形成完整的工件序列。

2.3　雇佣蜂阶段

2.3.1　选择

采用轮盘赌规则从工件序列集中选择 $Popsize$ 个工件序列,被选中的概率由适应度值决定。由于本文的目标是最小化总加权完成时间,所以第 h 个工件序列的适应度值和选择概率分别为

$$F_h = \frac{1}{\sum_{i=1}^{n} w_i \cdot ct_{u^i(|u^i|)i}}; \tag{11}$$

$$SP_h = \frac{F_h}{\sum_{h=1}^{Popsize} F_h}。 \tag{12}$$

2.3.2 多点交叉

采取两种基于工件号的多点交叉算子,如图 1 所示。随机选择两个父代工件序列 r_1 和 r_2,产生 1 个[0,1]的随机数 $rand$,若 $rand<pc$(交叉概率),则从两种交叉算子中随机选择一种进行交叉。具体过程如下。

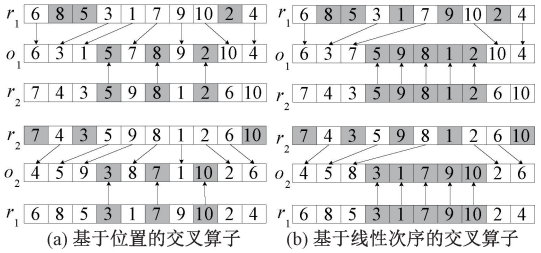

图 1 多点交叉算子

Figure 1 Multi-point crossover operator

(1)基于位置的交叉算子。随机产生 $\varepsilon_1(\varepsilon_1<n)$个位置,交换被选中位置上的工件号,删除未选中位置上的重复工件号,依次在未选中位置上放入对应父代工件序列内除交换工件号之外的剩余工件号,生成子代工件序列 o_1 和 o_2。

(2)基于线性次序的交叉算子。随机选择两个位置 π_1 和 $\pi_2(1\leqslant\pi_1<\pi_2\leqslant n)$,交换这两个位置之间的工件片段,依次删除未选中位置上的重复工件号,填入对应父代工件序列的除所选工件片段之外的剩余工件号,生成子代工件序列 o_1 和 o_2。

2.3.3 反转逆序变异

当产生的随机数 $rand<pm$(变异概率)时,利用翻转逆序变异扩大搜索范围,即随机产生两个位置 π_1 和 π_2 $(1\leqslant\pi_1<\pi_2\leqslant n)$,将两个位置之间的工件片段进行翻转。

2.4 跟随蜂阶段

根据 HFFSDC 的特点,设计了 5 种产生邻域解的邻域搜索策略,如图 2 所示。

对序列集内的工件序列分别进行上述 5 种邻域搜索,若产生的新序列中最大适应度值超过原序列,则将具有最大适应度值的新序列取代当前序列。

2.5 侦察蜂阶段

WOA 具有操作简单、控制参数少及跳出局部最优能力强等特点,因此引入基于最差解的 WOA 改善解的质量。

(1)线性包围猎物策略 Q_1。首先,找到当前工件序列集内的最佳工件序列 P_{best},若待更新工件序列 P_t 和 P_{best} 内同一位置上的工件号相同,则保留该位置以及对应的工件号。然后对 P_{best} 内其余位置的工件号进行循环移位。

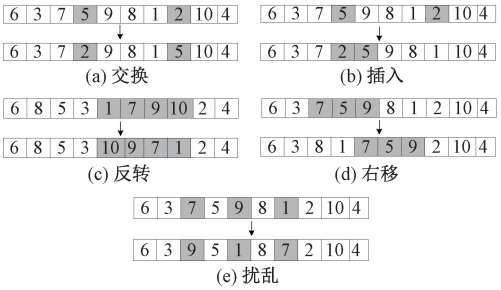

图 2 邻域搜索策略

Figure 2 Neighborhood search strategies

(2)螺旋包围猎物策略 Q_2。螺旋包围猎物是在线性包围猎物的基础上,对通过线性包围生成的位置整体进行循环移位来实现的。具体实现过程如图 3 所示。

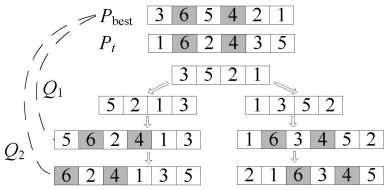

图 3 包围猎物策略

Figure 3 Encirclement strategy

3 数值实验与分析

3.1 实验设计

为了验证所提出的 ABC-WOA 混合算法求解 HFFSDC 的有效性,将其与结合 GA 和邻域搜索的人工蜂群算法(ABC&GA),混合变邻域搜索的改进 GA[18](IGA&VNS),以及基于邻域搜索的改进离散候鸟优化算法[19](IDMBO)进行对比测试。利用 MATLAB R2022b 在 PC Intel(R) Core i5-13600K/CPU3.5 GHz/RAM32 G 的计算机上实现上述算法。实验数据设定如下:$n=\{20,40,60,80,100\}$,$o=\{5,10,15\}$,跳过阶段概率[20] $p'=\{0.2,0.4,0.6\}$,$m_j\sim U[3,5]$,$p_{jki}\sim U[1,20]$,$T^i_{u^i(g),u^i(g+1)}\sim U[1,10]$,$w_i\sim U[1,10]$。定义各工件的总平均处理时间 $\bar{p}_i$ 及总运输时间 Ts_i:

$$\bar{p}_i = \sum_{j=1}^{o}\left(\sum_{k=1}^{m_j} p_{jki}/m_j\right); \tag{13}$$

$$Ts_i = \sum_{g=1}^{|u^i|-1} T^i_{u^i(g),u^i(g+1)}。 \tag{14}$$

据此产生各工件的强制工期[21]:

$$\bar{d}_i = 2(\bar{p}_i + Ts_i)(1+\eta)。 \tag{15}$$

式中:$\eta \in U[0,1]$。

为公平比较上述算法,以 ABC-WOA 混合算法运行 100 次迭代所需时间为所有算法的最大运行时间。根据正交试验对该算法涉及的主要参数进行最佳设置。$pm=\{0.2,0.3\}$,$lj=\{4,5,6\}$,共得到 24 种参数组合。选择小规模 $n=40$,$o=5$,$p'=0.2$ 以及大规模 $n=80$,$o=10$,$p'=0.4$ 作为两组实验算例,对这 24 种参数组合进行测试,每种组合随机运行 10 次,取平均值作为测试结果,并据此绘制 ABC-WOA 混合算法的平均目标值主效应图如图 4 所示。

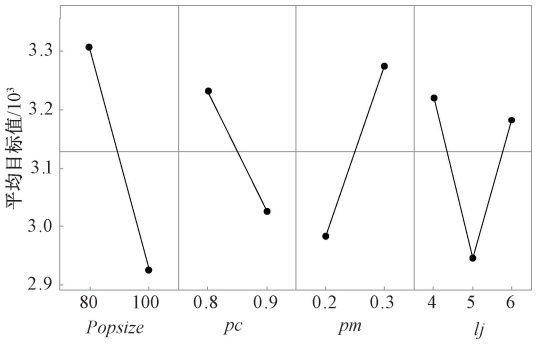

图 4　平均目标值的主效应图

Figure 4　Main effect plot of average objective value

各对比算法的关键参数也依据正交试验进行不同的取值测试,得到最佳参数设置,如表 1 所示。

表 1　各算法的参数设置

Table 1　Parameter settings for each algorithm

算法	参数
ABC&GA	$Popsize=100$,$pc=0.8$,$pm=0.2$
IGA&VNS	$Popsize=100$,$pc=0.9$,$pm=0.2$,变邻域搜索循环次数 $P=100$
IDMBO	$Popsize=101$,最大巡回次数 $G_{\max}=5$,邻域解集个数 $\beta=3$
ABC-WOA	$Popsize=100$,$pc=0.9$,$pm=0.2$,$lj=5$

为了说明所提 ABC-WOA 混合算法中各改进部分的影响,开展以下数值实验。

(1)NEH 启发式法初始化对比实验。对比使用 NEH 启发式法和不使用 NEH 启发式法产生的初始工件序列集的解的质量,为客观评价初始工件序列集解对算法的影响提供依据。

(2)鲸鱼优化算法对比实验。比较 ABC-WOA 混合算法引入鲸鱼优化算法前后的性能,为人工蜂群算法与鲸鱼优化算法的融合提供依据。

(3)不同规模算例对比实验。对比 4 种算法求解不同问题规模的优化结果,为评估所提算法的优化效果提供数据支持。

3.2　NEH 启发式法初始化对比实验

为验证 NEH 启发式法对改善初始工件序列集的影响,进行初始化对比实验。取 $n=\{20,40,60,80,100\}$,$o=\{5,10\}$,$p'=\{0.2,0.4\}$,产生 20 种规模问题,每种规模问题随机运行 10 次,分别测试由 NEH 启发式法和随机法产生的初始工件序列集的解的质量。表 2 列出了 NEH 启发式法和随机法得到的初始工件序列集的最佳目标值和平均目标值。由表 2 可知,对几乎所有规模问题,NEH 启发式法产生的初始解的性能均优于随机法。

表 2　NEH 启发式法初始化对比结果

Table 2　Comparison results of NEH heuristic initialization

$n\times o\times p'$	最佳目标值		平均目标值	
	NEH 启发式法	随机法	NEH 启发式法	随机法
20×5×0.2	4 368.7	**4 355.7**	**4 507.1**	4 731.5
20×5×0.4	**4 078.6**	4 181.9	**4 181.2**	4 456.5
20×10×0.2	**9 575.2**	9 789.5	**9 820.8**	10 240.3
20×10×0.4	**8 938.7**	9 081.5	**9 070.6**	9 402.5
40×5×0.2	**12 592.2**	14 509.7	**14 206.7**	15 592.9
40×5×0.4	**8 029.6**	8 626.1	**8 471.8**	9 503.5
40×10×0.2	**19 113.5**	20 540.7	**19 878.8**	22 054.4
40×10×0.4	**19 730.6**	20 808.4	**20 450.5**	21 854.7
60×5×0.2	**17 839.5**	20 109.4	**19 072.2**	22 312.1
60×5×0.4	**11 587.2**	12 989.7	**12 477.4**	14 259.4
60×10×0.2	**36 153.7**	37 018.6	**37 483.5**	39 480.1
60×10×0.4	**28 463.2**	29 013.2	**29 763.8**	30 897.2
80×5×0.2	**37 198.6**	38 181.3	**42 442.7**	47 691.4
80×5×0.4	**29 887.4**	30 646.2	**33 095.1**	35 305.3
80×10×0.2	**68 205.8**	73 891.2	**71 527.9**	78 105.5
80×10×0.4	**43 074.8**	45 387.0	**44 638.6**	48 257.6
100×5×0.2	**45 081.6**	50 706.3	**48 706.7**	57 261.3
100×5×0.4	**34 991.3**	38 074.9	**36 504.4**	40 990.7
100×10×0.2	**76 063.6**	85 354.4	**81 390.2**	91 368.0
100×10×0.4	**61 414.5**	66 844.4	**63 975.7**	69 484.4

3.3　鲸鱼优化算法对比实验

由于人工蜂群算法与 GA 和邻域搜索的融合较为常见,而鲜有与鲸鱼优化算法的混合,由此本实验提出了引入鲸鱼优化算法前后的对比实验以说明鲸鱼优化算法的融合效果。

以 3.2 节 20 种规模问题为例,每个规模问题随机运行 10 次,以 100 次迭代作为停止条件分别测试引入和不引入鲸鱼优化算法的 ABC-WOA 混合算法,得到平均目标值以及平均 CPU 时间,结果见表 3。由表 3 可知,在求解不同的规模问题时,引入鲸鱼优化算法的 ABC-WOA 混合算法得到的平均目标值均优于引入前。

3.4　不同规模算例对比实验

$\{n,o,p'\}$的不同组合共产生 45 种规模问题,每种规模问题随机进行 10 次,取其平均值作为测试

表 3 有无鲸鱼优化算法的 ABC-WOA 混合算法对比结果

Table 3 Comparison results of ABC-WOA hybrid algorithm with and without whale optimization algorithm

$n\times o\times p'$	目标值		CPU 时间/s	
	有优化	无优化	有优化	无优化
20×5×0.2	**4 051.67**	4 131.82	37.39	**36.43**
20×5×0.4	**3 640.46**	3 643.06	28.49	**28.14**
20×10×0.2	**8 837.01**	9 082.64	71.35	**69.05**
20×10×0.4	**8 099.86**	8 332.42	65.16	**63.60**
40×5×0.2	**12 256.33**	12 672.64	82.37	**79.89**
40×5×0.4	**9 725.67**	10 456.84	67.04	**66.52**
40×10×0.2	**23 568.71**	25 642.34	171.24	**167.17**
40×10×0.4	**14 785.63**	15 311.53	140.36	**137.80**
60×5×0.2	**18 367.54**	19 752.42	174.28	**172.04**
60×5×0.4	**14 743.73**	17 311.31	142.95	**137.08**
60×10×0.2	**34 678.38**	37 157.44	351.54	**341.97**
60×10×0.4	**25 832.64**	26 664.34	382.35	**371.87**
80×5×0.2	**35 467.35**	37 432.95	332.48	**327.05**
80×5×0.4	**25 693.56**	28 933.47	273.16	**267.38**
80×10×0.2	**55 764.76**	59 674.64	672.17	**665.06**
80×10×0.4	**35 713.53**	39 261.63	538.27	**531.94**
100×5×0.2	**43 678.86**	47 114.56	565.48	**556.46**
100×5×0.4	**35 735.84**	36 457.33	431.94	**430.38**
100×10×0.2	**75 822.66**	82 456.45	932.29	**921.60**
100×10×0.4	**61 467.84**	65 356.74	891.37	**878.48**
平均	**27 396.60**	29 537.06	317.58	**312.50**

结果。定义相对偏差百分比 δ_c 为

$$\delta_c = \frac{Z_c - Z_{\text{ABC-WOA}}}{Z_{\text{ABC-WOA}}} \times 100\%。 \tag{16}$$

式中:$Z_{\text{ABC-WOA}}$ 和 Z_c 分别为由 ABC-WOA 混合算法和当前算法 c($c \in \{\text{ABC\&GA}, \text{IGA\&VNS}, \text{IDMBO}\}$)得到的平均目标值。

所有规模问题的测试结果见表 4 和表 5,可知对于中小规模问题,在平均的 CPU 时间 165.66 s 内,由 ABC-WOA 混合算法得到的平均目标值在 ABC&GA、IGA&VNS 和 IDMBO 的基础上分别改进了 16.85%,3.77%和 10.85%;对于大规模问题,在平均的 CPU 时间 695.40 s 内,由 ABC-WOA 混合算法得到的平均目标值在 ABC&GA、IGA&VNS 和 IDMBO 的基础上分别改进了 19.01%,4.53%和 13.46%。在相同的 CPU 时间内,由 ABC-WOA 混合算法得到的解的质量几乎均优于其他 3 种算法。

表 6 列出了不同 p'取值下 4 种算法求解所有规模问题的平均测试结果。当 $p' = 0.2$ 时,在平均 CPU 时间 436.60 s 内,由 ABC-WOA 混合算法得到的平均目标值在 ABC&GA、IGA&VNS 和 IDMBO 的

表 4 中小规模问题的测试结果

Table 4 Test results of small and medium-sized problems

$n\times o\times p'$	目标值				$\delta_{\text{ABC\&GA}}$/%	$\delta_{\text{IGA\&VNS}}$/%	δ_{IDMBO}/%	CPU 时间/s
	ABC&GA	IGA&VNS	IDMBO	ABC-WOA				
20×5×0.2	4 783.32	4 117.32	4 423.65	**3 984.99**	20.03	3.32	11.01	35.43
20×10×0.2	10 613.65	9 489.99	10 015.32	**9 260.65**	14.61	2.48	8.15	69.86
20×15×0.2	17 434.65	15 595.32	16 563.99	**15 348.32**	13.59	1.61	7.92	100.68
40×5×0.2	12 573.65	11 639.99	12 167.32	**11 334.65**	10.93	2.69	7.35	83.10
40×10×0.2	25 265.65	22 690.65	24 735.32	**21 519.99**	17.41	5.44	14.94	172.93
40×15×0.2	28 833.99	26 425.65	27 346.32	**25 747.32**	11.99	2.63	6.21	261.80
60×5×0.2	22 575.99	19 422.65	21 692.99	**18 785.99**	20.17	3.39	15.47	177.71
60×10×0.2	40 966.99	38 218.99	39 316.99	**36 940.65**	10.90	3.46	6.43	349.66
60×15×0.2	54 852.32	48 609.99	51 832.99	**46 918.65**	16.91	3.60	10.47	539.54
20×5×0.4	4 075.99	3 435.32	3 726.32	**3 425.65**	18.98	0.28	8.78	27.05
20×10×0.4	9 212.99	8 324.32	8 706.99	**8 264.99**	11.47	0.72	5.35	55.31
20×15×0.4	12 392.99	11 374.65	11 932.32	**11 203.65**	10.62	1.53	6.50	88.94
40×5×0.4	11 425.99	9 101.65	10 578.99	**8 899.32**	28.39	2.27	18.87	74.50
40×10×0.4	18 127.65	15 543.65	16 793.99	**14 664.32**	23.62	6.00	14.52	147.78
40×15×0.4	33 262.32	29 155.99	31 320.65	**27 763.99**	19.80	5.01	12.81	224.89
60×5×0.4	17 937.65	16 012.32	16 842.65	**14 542.99**	23.34	10.10	15.81	144.06
60×10×0.4	33 369.65	30 477.99	31 647.99	**29 287.32**	13.94	4.07	8.06	297.53
60×15×0.4	52 330.65	46 720.32	49 855.32	**44 692.32**	17.09	4.54	11.55	459.25
20×5×0.6	3 616.99	**3 263.65**	3 402.32	3 263.99	10.81	−0.01	4.24	25.37
20×10×0.6	8 325.99	7 225.32	7 980.65	**7 178.65**	15.98	0.65	11.17	47.64
20×15×0.6	12 839.99	11 400.32	12 338.99	**11 123.99**	15.43	2.48	10.92	69.82
40×5×0.6	8 911.65	7 946.99	8 463.65	**7 781.99**	14.52	2.12	8.76	54.43
40×10×0.6	18 088.32	16 134.32	17 174.99	**15 029.65**	20.35	7.35	14.27	111.84
40×15×0.6	28 294.99	24 603.32	26 575.99	**23 344.65**	21.21	5.39	13.84	167.31
60×5×0.6	14 988.65	13 586.32	14 320.32	**12 691.32**	18.10	7.05	12.84	116.46
60×10×0.6	25 312.65	22 748.65	24 284.65	**21 009.32**	20.48	8.28	15.59	230.32
60×15×0.6	42 204.65	38 877.65	41 044.65	**36 942.32**	14.24	5.24	11.10	339.59
平均	21 208.15	18 968.27	20 188.38	**18 183.39**	16.85	3.77	10.85	165.66

表 5　大规模问题的测试结果

Table 5　Test results of large-scale problems

$n\times o\times p'$	目标值				$\delta_{ABC\&GA}$/%	$\delta_{IGA\&VNS}$/%	δ_{IDMBO}/%	CPU 时间/s
	ABC&GA	IGA&VNS	IDMBO	ABC-WOA				
80×5×0.2	39 783.24	33 119.24	36 978.91	**32 005.57**	24.30	3.48	15.54	329.44
80×10×0.2	64 206.57	57 809.91	61 380.24	**55 397.91**	15.90	4.35	10.80	671.85
80×15×0.2	92 648.57	83 423.57	89 698.91	**80 921.57**	14.49	3.09	10.85	1 041.71
100×5×0.2	50 117.24	44 471.91	48 892.24	**41 858.91**	19.73	6.24	16.80	581.08
100×10×0.2	90 332.24	81 432.57	85 387.57	**77 658.57**	16.32	4.86	9.95	934.23
100×15×0.2	109 643.24	96 895.24	101 827.24	**93 004.24**	17.89	4.18	9.49	1 200.00
80×5×0.4	25 661.24	22 169.24	23 903.24	**20 725.24**	23.82	6.97	15.33	273.34
80×10×0.4	43 712.22	40 005.88	41 302.89	**37 860.85**	15.45	5.67	9.09	553.90
80×15×0.4	69 827.78	63 048.89	66 530.98	**60 283.78**	15.83	4.59	10.36	875.80
100×5×0.4	39 341.01	34 805.78	38 008.89	**34 075.99**	15.45	2.14	11.54	453.75
100×10×0.4	71 719.87	64 323.02	70 016.77	**61 052.89**	17.47	5.36	14.68	921.49
100×15×0.4	117 723.46	97 570.23	111 002.89	**94 915.10**	24.03	2.80	16.95	1 200.00
80×5×0.6	20 724.21	16 790.96	20 265.22	**16 744.21**	23.77	0.28	21.03	221.25
80×10×0.6	37 379.26	31 559.30	32 979.20	**31 174.05**	19.91	1.24	5.79	435.08
80×15×0.6	53 134.94	47 236.65	51 125.76	**44 903.77**	18.33	5.20	13.86	659.10
100×5×0.6	31 156.05	27 175.25	30 245.19	**25 325.19**	23.02	7.31	19.43	346.07
100×10×0.6	53 135.06	47 088.95	51 135.07	**44 308.07**	19.92	6.28	15.41	738.37
100×15×0.6	74 958.99	69 153.89	74 259.64	**64 353.05**	16.48	7.46	15.39	1 080.56
平均	60 289.18	53 226.69	57 496.71	**50 920.50**	19.01	4.53	13.46	695.40

表 6　不同 p' 取值时各算法的测试结果

Table 6　Test results of various algorithms with different values of p'

p'	目标值				$\delta_{ABC\&GA}$/%	$\delta_{IGA\&VNS}$/%	δ_{IDMBO}/%	CPU 时间/s
	ABC&GA	IGA&VNS	IDMBO	ABC-WOA				
0.2	44 308.75	39 557.53	42 150.67	38 045.87	16.46	3.97	10.79	436.60
0.4	37 341.43	32 804.62	35 478.06	31 443.89	18.76	4.33	12.83	386.51
0.6	28 871.49	25 652.77	27 706.42	24 344.95	18.59	5.37	13.81	309.55

基础上分别改进了 16.46%,3.97% 和 10.79%;当 $p'=0.4$ 时,在平均 CPU 时间 386.51 s 内,由 ABC-WOA 混合算法得到的平均目标值在其他 3 种算法的基础上分别改进了 18.76%,4.33% 和 12.83%;当 $p'=0.6$ 时,在平均 CPU 时间 309.55 s 内,由 ABC-WOA 混合算法得到的平均目标值在其他 3 种算法的基础上分别改进了 18.59%,5.37% 和 13.81%。随着 p' 的增加,工件访问的加工阶段有所减少,各算法所消耗的 CPU 时间随之缩短,ABC&GA、IGA&VNS 和 IDMBO 的相对偏差百分比呈上升趋势,而所提 ABC-WOA 混合算法在解的质量方面的表现越有优势。

为了说明实验结果具有统计学意义,在 95% 的置信水平下,得出各算法在求解不同规模问题的目标值区间图如图 5 所示,可以看出,ABC-WOA 混合算法对比 3 种对比算法具有显著性优势。

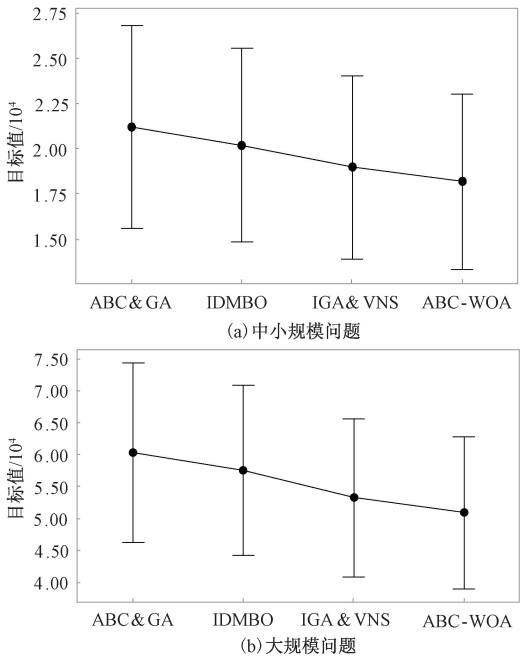

图 5　目标值区间图

Figure 5　Interval graph of target values

4 结论

本文研究了考虑不相关并行机、运输时间和强制工期约束的HFFS,目标是最小化总加权完成时间。考虑到工件跳过阶段对于调度的影响,通过设置工件实际加工阶段集的方式,确定工件在所访问阶段的排序以及机器选择问题。为了使所有工件都满足强制工期约束,提出基于强制工期前移策略的两阶段动态解码方案。为了提高算法初始解的质量、局部搜索能力且防止过快收敛,在ABC算法的基础上,融入NEH启发式法、改进GA和邻域搜索策略提出了ABC-WOA混合算法。不同算法的对比实验验证了引入NEH启发式法和鲸鱼优化算法的有效性;通过不同规模问题的仿真实验,比较所提混合算法与现有算法,测试结果显示了ABC-WOA混合算法求解HFFSDC的有效性和优越性。

本文研究还存在一些局限:对于更复杂的实际生产,模型和算法的适应性需进一步增强;绿色调度是现代制造业发展的一大趋势,未来可继续深入探讨节能和经济影响在车间调度领域的应用。

参考文献:

[1] 轩华, 耿祝新, 李冰. 组合缓冲约束下的多目标混合流水线节能调度[J]. 郑州大学学报(工学版), 2025, 46(1): 17-25.
XUAN H, GENG Z X, LI B. Multi-objective hybrid flowline energy-saving scheduling with combined buffer constraints[J]. Journal of Zhengzhou University (Engineering Science), 2025, 46(1): 17-25.

[2] OUJANA S, AMODEO L, YALAOUI F, et al. Mixed-integer linear programming, constraint programming and a novel dedicated heuristic for production scheduling in a packaging plant[J]. Applied Sciences, 2023, 13(10): 6003.

[3] LONG J Y, ZHENG Z, GAO X Q, et al. Scheduling a realistic hybrid flow shop with stage skipping and adjustable processing time in steel plants[J]. Applied Soft Computing, 2018, 64: 536-549.

[4] SCHUMACHER C, BUCHHOLZ P, FIEDLER K, et al. Local search and tabu search algorithms for machine scheduling of a hybrid flow shop under uncertainty[C]// 2020 Winter Simulation Conference (WSC). Piscataway: IEEE, 2020: 1456-1467.

[5] OUJANA S, YALAOUI F, AMODEO L. A linear programming approach for hybrid flexible flow shop with sequence-dependent setup times to minimise total tardiness [J]. IFAC-PapersOnLine, 2021, 54(1): 1162-1167.

[6] 轩华, 赵一航, 李冰. 带阶段跳跃和恶化效应的炼钢-连铸调度的混合启发式算法[J]. 工业工程与管理, 2024, 29(6): 82-92.
XUAN H, ZHAO Y H, LI B. Hybrid heuristic algorithm for steelmaking-continuous casting scheduling with stage skipping and deterioration effect[J]. Industrial Engineering and Management, 2024, 29(6): 82-92.

[7] YU Y, PAN Q K, PANG X F, et al. An attribution feature-based memetic algorithm for hybrid flowshop scheduling problem with operation skipping[J]. IEEE Transactions on Automation Science and Engineering, 2025, 22: 99-114.

[8] 黄辉, 李梦想, 严永. 考虑序列设置时间的混合流水车间多目标调度研究[J]. 运筹与管理, 2020, 29(12): 215-221.
HUANG H, LI M X, YAN Y. Research on multi-objective scheduling of hybrid flow production shop considering sequence setting time[J]. Operations Research and Management Science, 2020, 29(12): 215-221.

[9] 李浩平, 朱成彪, 陈心怡, 等. 带忽略工序的多目标批量流混合流水车间调度[J]. 计算机集成制造系统, 2025, 31(1): 89-101.
LI H P, ZHU C B, CHEN X Y, et al. Research on multi-objective lot streaming hybrid flowshop scheduling with missing operations[J]. Computer Integrated Manufacturing Systems, 2025, 31(1): 89-101.

[10] CHEN W C, LI J, MA W T. Hybrid flow shop rescheduling algorithm for perishable products subject to a due date with random invalidity to the operational unit[J]. The International Journal of Advanced Manufacturing Technology, 2017, 93(1): 225-239.

[11] MARCHESANO M G, GUIZZI G, POPOLO V, et al. Dynamic scheduling of a due date constrained flow shop with deep reinforcement learning[J]. IFAC-PapersOnLine, 2022, 55(10): 2932-2937.

[12] JANAKI E, ISMAIL A M. Using the genetic algorithm to reduce tardiness by tightening the deadline date for stochastic processing[J]. Soft Computing, 2023: 1-6.

[13] 宋存利. 禁止拖期交付的无等待流水车间调度问题算法研究[J]. 大连交通大学学报, 2018, 39(6): 100-105.
SONG C L. Research on modeling and algorithm of no-wait flow shop scheduling problem with prohibited tardiness[J]. Journal of Dalian Jiaotong University, 2018, 39(6): 100-105.

[14] YING K C, POURHEJAZY P, SUNG C E. Two-agent proportionate flowshop scheduling with deadlines: polynomial-time optimization algorithms[J]. Annals of Operations Research, 2024, 343(1): 543-558.

[15] SHI S Y, XIONG H G, LI G F. A no-tardiness job shop scheduling problem with overtime consideration and the solution approaches[J]. Computers & Industrial Engi-

neering, 2023, 178: 109115.

[16] 史双元, 熊禾根. 考虑加班的无拖期作业车间调度问题多目标金鹰优化算法研究[J]. 中国机械工程, 2023, 34(17): 2077-2088.
SHI S Y, XIONG H G. Research on multi-objective golden eagle optimizer for no-tardiness job shop scheduling problems with overtime consideration[J]. China Mechanical Engineering, 2023, 34(17): 2077-2088.

[17] 史双元, 熊禾根. 考虑外协的作业车间无拖期调度问题多目标差分进化算法[J]. 计算机集成制造系统, 2024, 30(12): 4352-4368.
SHI S Y, XIONG H G. Multi-objective differential evolution algorithm for no-tardiness job shop scheduling problem with outsourcing option[J]. Computer Integrated Manufacturing Systems, 2024, 30(12): 4352-4368.

[18] 崔琪, 吴秀丽, 余建军. 变邻域改进遗传算法求解混合流水车间调度问题[J]. 计算机集成制造系统, 2017, 23(9): 1917-1927.
CUI Q, WU X L, YU J J. Improved genetic algorithm variable neighborhood search for solving hybrid flow shop scheduling problem[J]. Computer Integrated Manufacturing Systems, 2017, 23(9): 1917-1927.

[19] 轩华, 樊银格, 李冰. 改进离散候鸟优化算法求解带缺失阶段的柔性流水车间问题[J]. 工业工程与管理, 2023, 28(1): 98-109.
XUAN H, FAN Y G, LI B. Improved discrete migrating birds optimization algorithm for flexible flowshop problem with missing stages[J]. Industrial Engineering and Management, 2023, 28(1): 98-109.

[20] 轩华, 樊银格, 李冰. 含忽略工序和不相关机的混合流水车间调度[J]. 智能系统学报, 2022, 17(3): 459-470.
XUAN H, FAN Y G, LI B. Hybrid flowshop scheduling with missing operations and unrelated machines[J]. CAAI Transactions on Intelligent Systems, 2022, 17(3): 459-470.

[21] 轩华, 李文婷, 李冰. 混合离散人工蜂群算法求解含不相关并行机的分布式柔性流水线调度[J]. 控制与决策, 2023, 38(3): 779-789.
XUAN H, LI W T, LI B. Hybrid discrete artificial bee colony algorithm for distributed flexible flowline scheduling with unrelated parallel machines[J]. Control and Decision, 2023, 38(3): 779-789.

Hybrid Flexible Flowline Scheduling with Deadline Constraints

XUAN Hua[1], LI Kunbo[1], CAO Ying[2]

(1. School of Management, Zhengzhou University, Zhengzhou 450001, China; 2. School of Civil Engineering and Architecture, Henan University of Science and Technology, Luoyang 471000, China)

Abstract: For the hybrid flexible flowline problem with unrelated parallel machines at each stage, with constraints on deadline and transportation time, an integer programming model was established to minimize the total weighted completion time. A hybrid algorithm of artificial bee colony algorithm and whale optimization algorithm (ABC-WOA) was proposed by combining improved genetic algorithm and neighborhood search strategy to obtain near optimal solutions. The algorithm utilized encoding based on job numbers and the NEH heuristic method to generate an initial set of job sequences. In the employed bee phase, an improved genetic algorithm was introduced to produce higher-quality job sequences. In the onlooker bee phase, five neighborhood search strategies were utilized to obtain better neighboring sequences. In the scout bee phase, a whale optimization algorithm based on the worst solution was designed to enhance the search capabilities of the algorithm. Simulation experiments were conducted to test the effectiveness of the improvements within the hybrid ABC-WOA algorithm, as well as to examine instances of varying sizes. The experimental results showed that the proposed hybrid algorithm performed very well.

Keywords: hybrid flexible flowline; deadlines; ABC-WOA hybrid algorithm; NEH heuristic approach