

文章编号: 1006-3080(2025)02-0228-14

DOI: 10.14135/j.cnki.1006-3080.20240411001

多种群混合进化算法求解带工序跳跃的分布式 异构批量流混合流水车间调度问题

陈三燕, 王学武, 王 烨, 顾幸生

(华东理工大学能源化工过程智能制造教育部重点实验室, 上海 200237)

摘要:针对以最小化最大完工时间、总流经时间、机器等待时间以及总加权提前时间和延迟时间为优化目标的带工序跳跃的分布式异构批量流混合流水车间调度问题, 提出了一种多种群混合进化算法 (Multi-Population Hybrid Evolutionary Algorithm, MPHEA)。首先, 给出每个优化目标的计算方式, 并制定合适的编码方案。其次, 设计了总种群与4个子种群之间的并行协同进化策略。在对总种群执行混合交叉算子和变异操作后, 依据快速非支配排序和拥挤距离更新种群, 从而确保总种群的多样性和优质解的保留。同时, 从总种群中提取出部分解, 形成了4个子种群, 每个子种群专注于优化一个特定目标。总种群与子种群并行进化的方式使得子种群在各自方向上进化时能够避免过分偏重于某一个目标, 从而实现多个目标的均衡优化。考虑到工序跳跃的操作对调度问题的影响, 采用了相应的工序跳跃启发式规则。最后, 通过仿真实验验证了MPHEA解决该调度问题的有效性和优越性。

关键词: 工序跳跃; 分布式车间调度; 批量流; 异构工厂; 高维多目标优化

中图分类号: TB497

文献标志码: A

随着制造行业的全球化步伐加快, 生产制造的
范围已经扩展至全国乃至全球, 实现了从集中向分散的转变, 从而出现了分布式制造的模式^[1]。企业为了保持长久的竞争力并持续发展, 必须灵活适应快速变化的市场需求, 在资源有限、环境受限等情况下, 协同各道生产工序对生产过程进行合理的优化与调度, 以确保满足不同市场客户的订单要求。分布式混合流水车间调度问题 (Distributed Hybrid Flowshop Scheduling Problem, DHFSP) 是混合流水车间调度问题 (Hybrid Flowshop Scheduling Problem, HFSP) 的扩展。与HFSP相比, DHFSP在调度过程中涉及多个工厂的协调, 从而增加了问题的复杂性。在真实制造系统中, 由于不同工厂的同一位置可能配置性能各异的机器, 并且这些机器的数量也可能

因工厂而异, 这就导致了各个生产基地的生产线呈现出多样性^[2]。因此, 研究异构工厂的DHFSP十分有必要。

在各种制造环境如电子、机械行业中, 通常都允许对生产批次进行分割。这意味着不必等到一个批次的所有产品都完成加工后再进行下一步操作, 而是可以将这些产品分成更小的子批次 (该过程被称为批量流), 进行逐步加工, 这样能够显著提高生产速度, 也有助于提升机器的利用效率^[3]。国内外已有一些学者在分布式流水车间调度和批量流集成优化方面进行了相关研究^[4-8], 基本上考虑的是单目标, 而大多数调度场景本质上是 (高维) 多目标。研究 (高维) 多目标调度问题, 寻找一组帕累托最优解集, 为企业决策者提供更多供选择的调度方案, 具有更重

收稿日期: 2024-04-11

基金项目: 国家自然科学基金 (61973120, 62076095)

作者简介: 陈三燕 (1992—), 女, 湖北崇阳人, 博士生, 主要研究方向为多目标优化、分布式车间调度。E-mail: sanyanchen@mail.ecust.edu.cn

通信联系人: 顾幸生, E-mail: xsgu@ecust.edu.cn

引用本文: 陈三燕, 王学武, 王 烨, 等. 多种群混合进化算法求解带工序跳跃的分布式异构批量流混合流水车间调度问题 [J]. 华东理工大学学报 (自然科学版), 2025, 51(2): 228-241.

Citation: CHEN Sanyan, WANG Xuewu, WANG Ye, *et al.* Multiple-Population Hybrid Evolutionary Algorithm for Solving Distributed Heterogeneous Lot-Streaming Hybrid Flowshop Scheduling Problem with Missing Operations[J]. Journal of East China University of Science and Technology, 2025, 51(2): 228-241.

要的研究意义。

在传统的混合流水车间调度中,每个批次均需所有阶段上加工。然而,许多实际的生产系统经常会遇到工序跳跃的情况,即并不是所有的批次都需要经过每个阶段。比如一种不锈钢工厂的生产系统生产各种不锈钢接头,包括T形、L形和P形接头。其中,T形接头和L形接头需要经过倒角机和螺纹攻牙机两个阶段,而P形接头只需要经过螺纹攻牙机一个阶段。再比如在波纹管的生产中,一般需要经过波纹管成型、试压检漏、下料、调波、去应力预处理、焊接、干式检漏、装配网套、包塑、扣压、包装这几个阶段。有些波纹管需要装配网套(将由相互交叉的若干金属丝编织而成的网套按确定的角度套在波纹管外表面),有些波纹管需要包塑(根据生产规格对波纹管进行塑封包覆处理),有些波纹管既不需要装配网套也不需要包塑。目前,文献[9-13]研究了带工序跳跃的HFSP,而鲜有针对分布式和批量流约束下的带工序跳跃的HFSP的研究。

基于上述研究背景,本文研究了带工序跳跃的高维多目标分布式异构批量流混合流水车间调度问题(Distributed Heterogeneous Hybrid Flowshop Lot-Streaming Scheduling Problem with Missing Operations, DHHFLSPMO)。进化算法是求解各种优化问题的工具,特别地,遗传算法已经在求解离散优化问题上表现出强大的效率^[14]。标准的遗传算法由于进化过程本身的局限性,种群特性在经过长时间演化后往往难以产生大的突破。相比之下,多种群通过并行进化成多个不同种群,并在进化过程中实施个体交换,打破了种群内部的稳定状态,有助于逃离局部最优的困境。为了有效地解决高维多目标DHHFLSPMO,本文提出了一种多种群混合进化算法(MPHEA),同时优化最大完工时间、总流经时间、

机器等待时间以及总加权提前时间和延迟时间。在MPHEA中,为了搜索不同区域以提高解集的多样性,利用混合交叉方法实现总种群和4个子种群的并行协同进化。具体地,对总种群基于帕累托支配关系和拥挤距离更新种群,并依据目标个数从总种群中提取部分解形成多个子种群,各个子种群分别重点优化每个目标。同时针对工序跳跃的操作,采用了相应的工序跳跃启发式规则。最后,验证了MPHEA求解DHHFLSPMO的优越性。

1 DHHFLSPMO

DHHFLSPMO可描述如下:存在 f 个异构工厂,每个工厂被构造为具有 s 个加工阶段的混合流水车间。工厂 k 中的阶段 h 有 $m_{k,h}$ 台相同的机器,其中至少一个阶段有 $m_{k,h} > 1$ 。每个批次都可以分配给任何工厂,并在所有阶段遵循一致的加工路线。但是,某些工序可能会被排除在外,即存在工序跳跃的情况。每个批次 j 有 Lb_j 个相同的项目,并且可以分成 l_j 个大小相等的子批。加工批次 j 的第一个子批需要的准备时间为 $st1_{j,h}$,而加工批次 j 的其他子批需要的准备时间为 $st2_{j,h}$ 。同一批次产生的子批在一台机器上连续加工。

该调度问题的目标是找到工厂分配的作业、每个作业的子项数和每个阶段的作业顺序,从而使最大完工时间($C_{\max}$)、总流经时间(F_{tot})、机器等待时间(M_{idl})以及总加权提前时间和延迟时间(T_{wgt})最小化。4个目标函数的计算如式(1~4)所示^[8,10]:

$$\min C_{\max} = \min(\max(CT_1, CT_2, \dots, CT_f)) \quad (1)$$

$$\min F_{\text{tot}} = \sum_{k=1}^f \sum_{w_{k,s}=1}^{G_k} C_{w_{k,s},s,k,z_{w_{k,s},s,k}} \quad (2)$$

$$\min M_{\text{idl}} = \sum_{k=1}^f \sum_{h=1}^s \left(\sum_{i=1}^{m_{k,h}} CQ_{n_{i,h,k},i,h,k,z_{n_{i,h,k},i,h,k}} - \sum_{w_{k,h}=1}^{G_k} \sum_{j=1}^n x_{j,w_{k,h},k} (st1_{j,h} + st2_{j,h}(l_j - 1) + P_{j,h}Lb_j) \right) \quad (3)$$

$$\begin{aligned} \min T_{\text{wgt}} = & \sum_{k=1}^f \sum_{w_{k,s}=1}^{G_k} \max \left\{ \left(\sum_{j=1}^n x_{j,w_{k,s},k} WE_j D_j - C_{w_{k,s},s,k,z_{w_{k,s},s,k}} \sum_{j=1}^n x_{j,w_{k,s},k} WE_j \right), 0 \right\} + \\ & \sum_{k=1}^f \sum_{w_{k,s}=1}^{G_k} \max \left\{ \left(C_{w_{k,s},s,k,z_{w_{k,s},s,k}} \sum_{j=1}^n x_{j,w_{k,s},k} WT_j - \sum_{j=1}^n x_{j,w_{k,s},k} WT_j D_j \right), 0 \right\} \end{aligned} \quad (4)$$

式中: f 为工厂总数; s 为阶段总数; n 为批次总数; CT_k 为工厂 k 的最大完工时间; G_k 为分配给工厂 k 的批次数量; $w_{k,h}$ 为工厂 k 中阶段 h 的批次的位置索引; $z_{w_{k,h},k}$ 为工厂 k 中阶段 h 的位置 w 的子批数量; $C_{w_{k,h},h,k,g}$ 为工厂 k 中安排在阶段 h 位置 $w_{k,h}$ 的批次的

第 g 个子批的结束时间; $z_{q,i,h,k}$ 为安排在工厂 k 中阶段 h 机器 i 上的位置 q 的批次的子批数量; $n_{i,h,k}$ 为分配给工厂 k 中阶段 h 的机器 i 上的批次数量; $CQ_{q,i,h,k,g}$ 为工厂 k 中安排在阶段 h 机器 i 上的位置 q 的批次的第 g 个子批的结束时间; $x_{j,w_{k,h},k}$ 表示如果

批次 j 安排在工厂 k 中的位置为 $w_{k,h}$, 则值为 1, 否则值为 0; P_{jh} 为批次 j 的单个子批在阶段 h 的加工时间; WE_j 为批次 j 的提前完工权重; WT_j 为批次 j 的延期完工权重; D_j 为批次 j 的交货期。

2 MPHEA

通过多种群并行搜索可以保持多样性, 从而避免过早收敛, 此外, 几种不同交叉算子的合作可以搜索不同的方向以此提高求解质量。因此, 本文提出了可实现总种群和 4 个子种群的并行协同进化的 MPHEA, 以解决 DHHFLSPMO 问题。具体地, 对总种群执行混合交叉算子和变异后, 再基于快速非支

配排序和拥挤距离更新种群, 并且从总种群中提取部分解形成 4 个子种群分别专注 4 个目标的优化。此外, 针对工序跳跃的操作, 采用了相应的工序跳跃启发式规则。

2.1 编码及种群初始化

在 DHHFLSPMO 问题中, 需要同时对工厂批次分配、每个批次的子批数量、每个阶段机器的选择和每个阶段批次的加工顺序进行优化。为此, 本文设计一种包含这 4 种信息的 3 层编码机制, 如式 (5) 所示。

$$P_r = \{\mathbf{FV}_r, \mathbf{LV}_r, \mathbf{JM}_r\} \quad (5)$$

其中,

$$\mathbf{LV}_r = [\mathbf{lv}_{r,1}, \mathbf{lv}_{r,2}, \dots, \mathbf{lv}_{r,j}, \dots, \mathbf{lv}_{r,n}], \mathbf{lv}_{r,j} \in \{l_{\min}, l_{\min+1}, \dots, l_{\max}\} \mathbf{JM}_r =$$

$$\begin{bmatrix} \mathbf{jv}_{r,1,1} & \mathbf{jv}_{r,1,2} & \dots & \mathbf{jv}_{r,1,h} & \dots & \mathbf{jv}_{r,1,s} \\ \mathbf{jm}_{r,2,1} & \mathbf{jm}_{r,2,2} & \dots & \mathbf{jm}_{r,2,h} & \dots & \mathbf{jm}_{r,2,s} \\ \vdots & \vdots & & \vdots & & \vdots \\ \mathbf{jm}_{r,j+(k-1)n,1} & \mathbf{jm}_{r,j+(k-1)n,2} & \dots & \mathbf{jm}_{r,j+(k-1)n,h} & \dots & \mathbf{jm}_{r,j+(k-1)n,s} \\ \vdots & \vdots & & \vdots & & \vdots \\ \mathbf{jm}_{r,n+(f-1)n,1} & \mathbf{jm}_{r,n+(f-1)n,2} & \dots & \mathbf{jm}_{r,n+(f-1)n,h} & \dots & \mathbf{jm}_{r,n+(f-1)n,s} \end{bmatrix}$$

式 (5) 表示每个进化个体 r 的编码集合 P_r 包含每个批次的工厂分配向量 $\mathbf{FV}_r$ 、批次分割向量 $\mathbf{LV}_r$ 和批次排列矩阵 $\mathbf{JM}_r$ 。其中, 工厂分配向量 $\mathbf{FV}_r$ 由工厂变量 $\mathbf{fv}_{r,j}$ 组成, 并且每个工厂变量 $\mathbf{fv}_{r,j}$ 为整数, 表示每个进化个体 r 的批次 j 的工厂编号。批次分割向量 $\mathbf{LV}_r$ 由分割变量 $\mathbf{lv}_{r,j}$ 组成, 每个分割变量 $\mathbf{lv}_{r,j}$ 为整数, 表示每个进化个体 r 的批次 j 的分割数。批次排列矩阵 $\mathbf{JM}_r$ 是一个 $(n+(f-1)n) \times s$ 的矩阵, 由排列变量 $\mathbf{jm}_{r,j+(k-1)n,h}$ 组成, 每个排列变量 $\mathbf{jm}_{r,j+(k-1)n,h}$ 为 1 到 $m_{k,h} + 1$ 之间的小数, 整数部分表示加工批次选择的机器编号, 小数部分的大小表示加工批次在机器上的加工顺序 (越小越先加工)。

为了说明此 3 层编码机制, 这里给出 2 个工厂、5 个批次、3 个阶段的例子, 其中 $m_{1,1} = 2$, $m_{1,2} = 1$, $m_{1,3} = 3$, $m_{2,1} = 1$, $m_{2,2} = 3$ 和 $m_{2,3} = 2$, 1 个示例解的编码表示如图 1 所示。工厂分配向量表明批次 1、批次 4 和批次 5 分配给工厂 1, 批次 2 和批次 3 分配给工厂 2。批次分割向量表明批次 1~5 的子批数量分别是 8、7、8、3 和 6。对于批次排列矩阵, 由于 $m_{1,1} = 2$, 所以对于工厂 1 中第 1 阶段的批次, 生成 1 到 $m_{1,1} + 1$ 之间的实数, 整数部分表示在第 1 阶段处理批次的机器。如果整数部分相同, 则根据小数部分的高低确定批次的加工顺序。工厂 1 中, 批次 1~5 在第 1 阶段生成的数分别为 2.36、1.78、1.29、

-1.00、2.78, 从整数部分来看, 批次 2 和 3 在第 1 阶段的第 1 台机器上加工; 从小数部分来看, 加工顺序为 3→2 (因为 $0.29 < 0.78$), 而批次 1 和 5 在第 1 阶段的第 2 台机器上加工, 加工顺序为 1→5。以此类推,

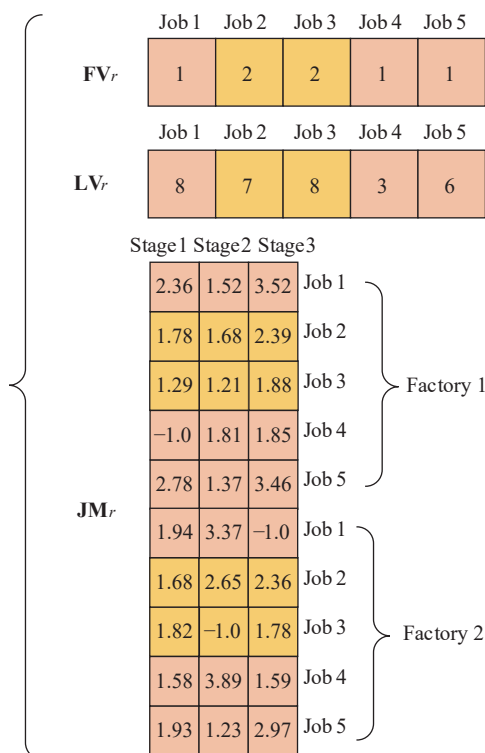


图 1 示例解的编码表示

Fig. 1 Solution representation of an example

可以得到各工厂在各阶段的批次加工顺序矩阵。图中的-1.0代表工序跳跃,即批次4在工厂1的第1阶段不需要加工,批次1在工厂2的第3阶段不需要加工,批次3跳过了工厂2的第2阶段。

算法中,所有初始解都是随机产生。对于工厂分配部分,取一个随机整数作为每个加工批次的工厂编号,该整数均匀分布于 $[1, f]$ 之间。对于批次分割部分,在规定范围内生成一个随机整数作为每个加工批次的子批数。关于批次排列矩阵部分,对工厂 k 中第 h 阶段的批次,生成1到 $m_{k,h}+1$ 之间的小数,工序跳跃的操作用-1.0表示。

2.2 混合交叉算子

混合交叉能够增加种群的多样性。不同的交叉算子在交叉操作中会采用不同的策略,从而产生不同类型的后代个体。这有助于避免算法过早收敛到局部最优解,能够在更广泛的解空间中进行搜索,提高算法的全局搜索能力。因此,本文集成了两种交叉算子,以生成更具有竞争力的子代。工厂分配、批次分割和批次排列均以相等的概率执行两点交叉和均匀交叉^[15],具体地说,产生一个随机数 $r_1 \in [0, 1]$,如果 $r_1 < 0.5$,以交叉概率 P_c 执行两点交叉;如果 $r_1 \geq 0.5$,以交叉概率 P_c 执行均匀交叉。关于两点交叉,对于工厂分配向量和批次分割向量,首先,在父代的个体中随机选择两个点。然后,两个点之间的一些序列被交换以产生后代。对于批次排列矩阵,在父代的个体中随机选择两列,然后,两列之间的一些序列被交换以产生后代。关于均匀交叉,对于工厂分配向量、批次分割向量和批次排列矩阵,对每一个基因位产生一个0~1之间的小数,小于或等于0.5的基因位保持不变,大于0.5的基因位被交换以产生子代。混合交叉的伪代码如下:

Procedure Hybrid_Crossover(P, P_c)

Input: P (Population), P_c (Crossover probability)

Output: P' (Population)

Generate a random number r_1 in the interval of $[0, 1]$

If $r_1 < 0.5$

Generate a random number r_2 in the interval of $[0, 1]$

If $r_2 < P_c$

$P' = \text{Perform two-point crossover on } P$

End If

Else

Generate a random number r_3 in the interval of $[0, 1]$

If $r_3 < P_c$

$P' = \text{Perform uniform crossover on } P$

End If

End If

Return: P'

对于总种群,在每一代进化中,采用二元锦标赛选择操作从总种群中随机选择两个个体,然后比较这两个个体的非支配级别和拥挤距离。具有更低非支配级别的个体将被选为优秀个体(非支配级别一样时,选择拥挤距离更大的个体),以进行后续的交叉操作。对于子种群,在每一次进化中,每个个体与该子种群中随机选择的一个个体进行交叉操作。

2.3 变异算子

为了有效避免陷入局部最优,对工厂分配、批次分割和批次排列矩阵进行变异操作;对于工厂分配部分,更改与随机选择的加工批次相对应的工厂;对于批分割部分,加工批次是随机选择的,其子批数量在其设置范围内随机调整;对于批次排列部分,选择一个工序不跳跃的基因位,对工厂 k 中第 h 阶段的批次,随机生成1到 $m_{k,h}+1$ 之间的小数。

2.4 工序跳跃启发式规则

针对工序跳跃的操作,采取了一种工序跳跃启发式规则^[12],其核心思想是将带工序跳跃的批次(在第 h 阶段)重新分配至第 $h+1$ 阶段机器的开头立即处理,以减少机器的空转时间。这一规则旨在有效减少机器的空转时间,提高生产效率。在算法进化一次后,若批次 j 在第 h 阶段无需加工,立即将其安排至第 $h+1$ 阶段机器的开头,以此减少该阶段机器的空转时间,从而可以减少最大完工时间和总流经时间,因此本文针对第1个和第3个优化目标采用了此工序跳跃启发式规则。

以一个工厂为例说明工序跳跃启发式规则的过程。每个批次的产品数量、子批数量和各阶段的单位加工时间如表1所示,可以看到批次3和批次5在第1阶段的加工时间为0,这表明批次3和批次5在第1阶段不需要加工,即存在工序跳跃的情况。图2(a)是没有加入工序跳跃启发式规则的甘特图,最大完工时间是41,总流经时间为 $14+25+41+22+33=135$ 。如图2(b)所示,当把批次3和批次5立即安排在第2阶段机器的开头,最大完工时间减到37,总流经时间为 $11+20+37+11+22=101$ 。因此,工序跳跃启发式规则比没有加入工序跳跃启发式规则获得的解要好。

2.5 基于FNS和拥挤距离的选择策略

基于帕累托支配关系的多目标优化方法是最具影响力和应用最广泛的方法之一,本文采用 Deb

表 1 例子的相关数据
Table 1 Relevant data of the example

Job	Size	Number of subplot	Unit processing time	
			Stage 1	Stage 2
1	6	3	1	2
2	4	2	2	1.5
3	4	2	0	2
4	5	2	1	1
5	4	2	0	2

等^[16]提出的非支配排序遗传算法-II(Non-dominated Sorting Genetic Algorithm II, NSGA-II)来选择下一代种群,快速非支配排序(Fast Non-dominated Sorting, FNS)和拥挤距离计算是 NSGA-II 的两大核心步骤,它们分别保证了种群的收敛性和多样性,使得算法能够逼近问题的帕累托最优前沿。

在 FNS 中,首先会将种群中的个体按照非支配关系进行分类。非支配关系是指一个个体在至少一个目标上的表现优于另一个个体,而在其他目标上不劣于另一个个体。然后根据这些非支配关系,将种群中的个体划分为不同层级的帕累托前沿,每个层级中的个体相对于其他个体都是非支配的。

拥挤距离用于评估种群中个体在目标空间中的分布密度,计算步骤如下:

(1) 对于种群中的每一个个体,将其拥挤距离初始化为 0。

(2) 对于每个目标函数,将种群中的个体按照该目标函数的值进行排序。

(3) 对于排序后的每个目标函数的两端个体(即在该目标函数上表现最好和最差的个体),赋予一个较大的拥挤距离值,一般设为无穷大。

(4) 对于排序后的每个目标函数中的非边界个体,其拥挤距离等于相邻两个个体在该目标函数上的差值。

(5) 将个体在所有目标函数上的拥挤距离进行累加,得到该个体的总拥挤距离。

基于 FNS 和拥挤距离的选择策略原理如图 3 所示,其中 P_t 是当前代的种群, Q_t 是交叉操作后的子代种群,种群大小均为 PS。首先,将第 t 代产生的子代种群 Q_t 与亲代 P_t 合并形成 R_t ,种群大小均为 $2 \times PS$ 。然后,对 R_t 进行 FNS,得到不同层级的帕累托前沿 $F_1、F_2、F_3、\dots$,同时计算每个个体的拥挤距离。按照帕累托前沿的等级依次将 F_i 放入 P_{t+1} 中,直到 P_{t+1} 的种群大小大于或等于 PS。当加入 F_3 时,种群的大小超过 PS,然后选择 F_3 中拥挤距离

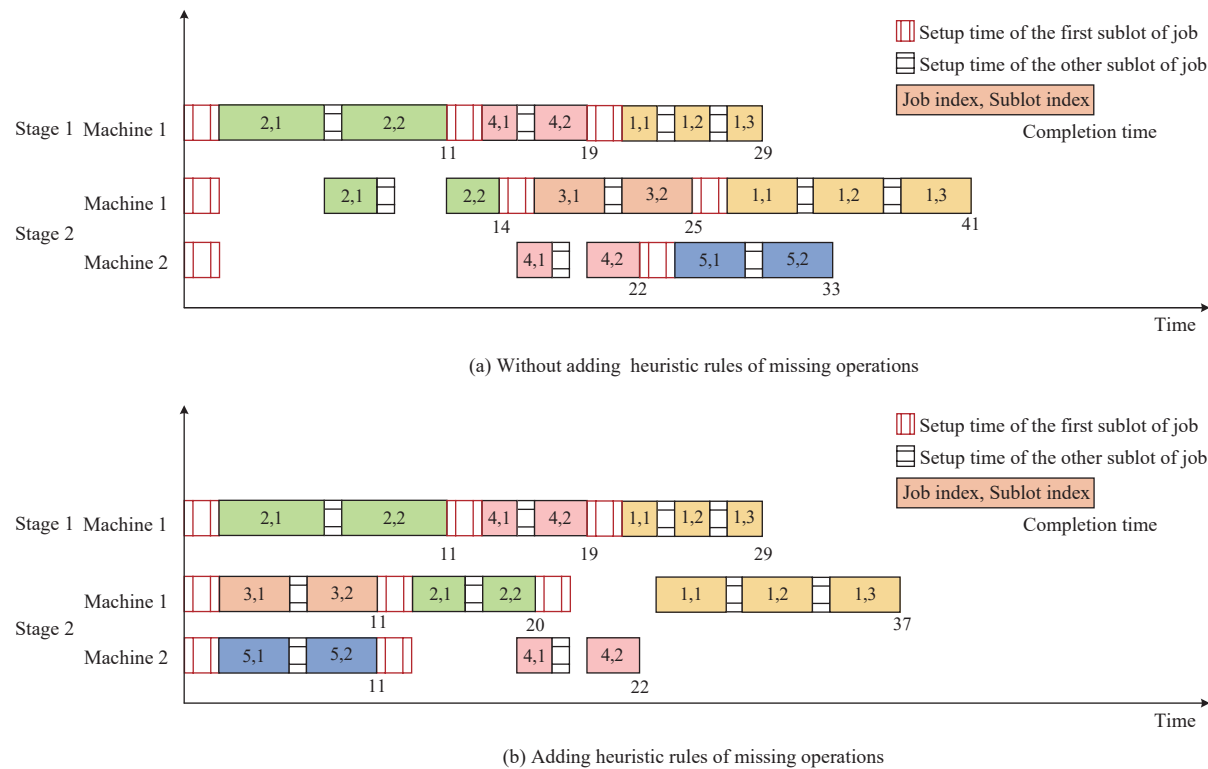


图 2 加入工序跳跃启发式规则前后的甘特图
Fig. 2 Gantt chart before and after adding heuristic rules of missing operations

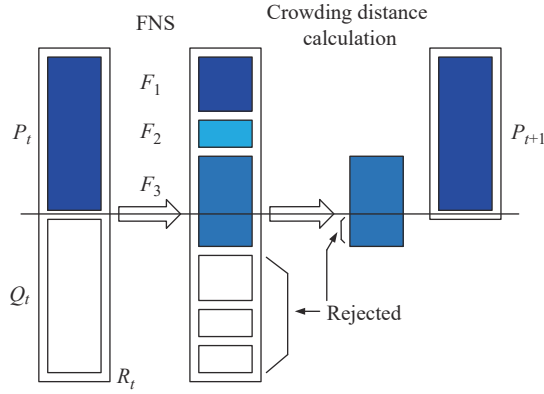


图3 基于FNS和拥挤距离的选择策略

Fig. 3 Selection strategy based on FNS and crowding distance

较大的解,使 P_{t+1} 中的解个数达到 PS。

2.6 多种群混合进化策略

针对高维多目标 DHHFLSPMO 问题,设计了一种总种群与子种群并行进化的多种群混合进化策略,即总种群在进化的同时,四个子种群分别重点优化每个目标,并结合总种群子代与子种群子代来选择优秀个体进入下一代进化。由于每个子种群专注于一个特定的目标进行优化,它们能够更快速地探索与该目标相关的解空间。同时,总种群的进化则能够保持全局搜索的能力,避免陷入局部最优解(使得子种群在各自的方向上进化时能够避免过分偏重于某一个目标)。这种总种群与子种群并行进化的策略使得算法能够在多个目标之间进行有效权衡和折衷,从而提高整体优化效果。多种群混合进化策略的流程如图4所示,详细步骤如下:

(1) 总种群 P 进化。首先对总种群 P 进行混合

交叉操作和变异操作,得到子代 P_a 。然后合并总种群 P 和子代 P_a ,并基于 FNS 和拥挤距离的选择策略更新种群,得到子代 P_b 。

(2) 子种群 1 进化。根据第 1 个优化目标,对总种群 P 进行升序排序,然后选择前 1/4 的解,形成子种群 1,标记为 PS_1 。再对 PS_1 进行混合交叉操作,然后执行工序跳跃启发式规则,得到子代 PS_{11} 。

(3) 子种群 2 进化。根据第 2 个优化目标,对总种群 P 进行升序排序,然后选择前 1/4 的解,形成子种群 2,标记为 PS_2 。再对 PS_2 进行混合交叉操作,得到子代 PS_{22} 。

(4) 子种群 3 进化。根据第 3 个优化目标,对总种群 P 进行升序排序,然后选择前 1/4 的解,形成子种群 3,标记为 PS_3 。再对 PS_3 进行混合交叉操作。然后执行工序跳跃启发式规则,得到子代 PS_{33} 。

(5) 子种群 4 进化。根据第 4 个优化目标,对总种群 P 进行升序排序,然后选择前 1/4 的解,形成子种群 4,标记为 PS_4 。再对 PS_4 进行混合交叉操作,得到子代 PS_{44} 。

(6) 种群融合与环境选择。将总种群 P 的子代 P_b 、子种群 1 的子代 PS_{11} 、子种群 2 的子代 PS_{22} 、子种群 3 的子代 PS_{33} 和子种群 4 的子代 PS_{44} 合并在一起。然后,基于 FNS 和拥挤距离的选择策略选择一定规模的优秀个体进行下一代进化。

2.7 MPHEA 算法

算法首先初始化种群 P ,再对 P 执行混合交叉操作和变异操作,得到子代 P_a 。然后将 P 和 P_a 合并,并基于 FNS 和拥挤距离的选择策略更新种群,得

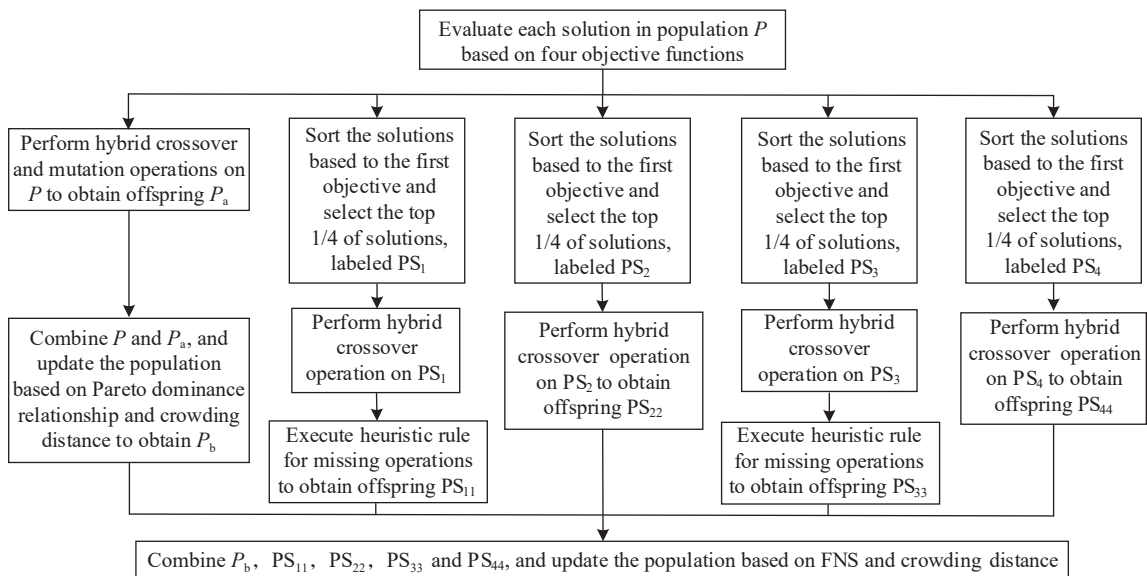


图4 多种群混合进化策略的流程图

Fig. 4 Flowchart of the multiple-population hybrid evolutionary strategy

到子代 P_b 。接着,从总种群中提取出部分解,形成 4 个子种群 PS_1 、 PS_2 、 PS_3 和 PS_4 ,每个小种群专注于优化一个特定的目标。这种大种群与小种群并行进化的方式,使得小种群在各自的方向上进化时能够避免过分偏重于某一个目标,从而实现了多个目标的均衡优化。考虑到工序跳跃的操作对调度问题的影响,在优化第 1 个和第 3 个子种群时,采用相应的工序跳跃启发式规则。通过基于 FNS 和拥挤距离的选择策略更新总种群和子种群的子代,选择 PS 个优秀个体进入到下一代种群,完成一次进化。MPHEA 算法的伪代码如下:

Procedure Framework_MPHEA(PS, P_c, P_m)

Input: PS (Population size)

Output: Non-dominated solutions

Initialize population P

While termination criterion is not fulfilled do

$P_a = \text{Hybrid_Crossover}(P)$

Through FNS and crowding distance calculation, select the best PS individuals from P and P_a , denoted as P_b

min1 = Sort the solutions according to the first objective

min2 = Sort the solutions according to the second objective

min3 = Sort the solutions according to the third objective

min4 = Sort the solutions according to the fourth objective

$PS_1 = \text{min1}(1:\text{round}(PS/4))$

$PS_2 = \text{min2}(1:\text{round}(PS/4))$

$PS_3 = \text{min3}(1:\text{round}(PS/4))$

$PS_4 = \text{min4}(1:\text{round}(PS/4))$

PS_{11} = Execute hybrid crossover and heuristic rules of missing operations on PS_1

PS_{22} = Execute hybrid crossover on PS_2

PS_{33} = Execute hybrid crossover and heuristic rules of missing operations on PS_3

PS_{44} = Execute hybrid crossover on PS_4

Through FNS and crowding distance calculation, Select PS excellent individuals from P_b , PS_{11} , PS_{22} , PS_{33} and PS_{44} as the next generation population

End While

Return: Non-dominated solutions

3 仿真实验与分析

为了验证 MPHEA 在解决 DHHFLSPMO 方面的有效性,将本文提出的 MPHEA 与 MPMOGA^[14]、NSGA-II^[16]、mMaOEA/AS^[17] 和 MaOEA/AS^[18] 进行了比较。MPMOGA 采用了多种群框架,NSGA-II 是应用广泛的多目标优化算法,mMaEA/AS 用于解决分布式异构批量流水车间调度问题,而 MaOEA/AS 是最近提出的一种高维多目标优化算法,所有算法均采用 C 语言编写。

3.1 实验设置

在实验中,使用 400 个算例来评估 MPHEA 的性能。根据文献 [3] 和文献 [8],这些算例由 f 个工厂、 s 个阶段和 n 个批次构成,其中 $f = \{2,3,4,5\}$, $s = \{5,10\}$, $n = \{20,40,60,80,100\}$,工序的跳跃率设为 20% 和 40%。每个组合都创建 5 个算例。因此,算例总数是 $4 \times 2 \times 5 \times 2 \times 5 = 400$ 。表 2 给出了其他相关数据的一些细节,其中 P_j 表示批次 j 的处理时间,rand 表示区间 $[0,1]$ 上的均匀分布的随机数。

表 2 相关数据的信息

Table 2 Information of the related data

Parameter	Value
Quantities of units Lb_j	$U[30, 60]$
Unit processing time $P_{j,h}$	$U[5, 20]$
Number of machines $m_{k,h}$	$U[2, 5]$
Number of sublots l_j	$U[1, 10]$
Setup time $st1_{j,h}$	$U[5, 20]$
Setup time $st2_{j,h}$	$U[5, 10]$
Due date D_j	$P_j \times (1 + \text{rand} \times 3)$

3.2 评价指标

本文使用 2 个性能指标来评估解集的收敛性和多样性,分别是:反世代距离 (IGD)^[19] 和超体积 (HV)^[20]。IGD 和 HV 指标能够衡量得到的解的收敛性和多样性。

IGD 的计算方法如下:

$$\text{IGD}(A, S) = \frac{1}{|S|} \sum_{i=1}^{|S|} \min_{j=1}^{|A|} d(s_i, a_j) \quad (6)$$

其中, $A = \{a_1, a_2, \dots, a_{|A|}\}$, 表示求解目标空间中得到的解集, $|\cdot|$ 代表集合 $\cdot$ 中的元素个数; $S = \{s_1, s_2, \dots, s_{|S|}\}$, 表示真实帕累托前沿采集的一组解集; $d(s_i, a_j) =$

$\|s_i - a_j\|$ 是 s_i 到 a_j 的欧氏距离。值得指出的是,较小的IGD值表明算法求得的解集更接近真实的帕累托前沿和具有更好的多样性。

HV 指标表示非支配解与参考点 $s^r = (s_1^r, s_2^r, \dots, s_M^r)$ 之间空间的超体积。计算公式如下:

$$HV(\Omega) = \text{Leb} \left(\bigcup_{\chi \in \Omega} [f_1(\chi), s_1^r] \times [f_2(\chi), s_2^r] \times \dots \times [f_M(\chi), s_M^r] \right) \quad (7)$$

式中: χ 为决策变量; M 为目标个数; Ω 为决策空间; $f_1(\chi), f_2(\chi), \dots, f_M(\chi)$ 为目标函数。最终解集中的目标向量按 $1.1 \times (\max f_1(\chi), \max f_2(\chi), \dots, \max f_M(\chi))$ 归一化, s^r 设置为 $(1.0, 1.0, \dots, 1.0)$ 。注意,由参考点支配的解不涉及 HV 的计算。

3.3 对比算法

本文扩展了 MPMOA^[14]、NSGA-II^[16]、mMaOEA/AS^[17] 和 MaOEA/AS^[18] 这 4 种高级算法来解决 DHHFLSPMO 问题。在 MPMOGA、NSGA-II、mMaOEA/AS 和 MaOEA/AS 中使用的交叉算子为两点交叉,而突变算子与 2.3 节中讨论的相同。采用田口实验设计法对比较算法的参数设置进行了标定。为简单起见,表 3 仅提供最终的参数设置。

3.4 参数设置

所有算法的终止条件都设置为 $100 \times f \times s \times n(\text{ms})$ 。对比算法的参数设置如表 3 所示。MPHEA 包含 3 个参数,即种群大小 PS、交叉概率 P_c 和突变概率 P_m 。采用广泛使用的田口实验设计方法^[21],在随机选择的 $f = 5$ 、 $n = 60$ 、 $s = 10$ 和 20% 跳跃率的算例上来校正合适的参数。PS 的水平设置为: $PS \in \{30, 56, 84, 104\}$, P_c 和 P_m 的水平分别设置为: $P_c \in \{0.7, 0.8, 0.9, 1.0\}$ 和 $P_m \in \{0.1, 0.2, 0.3, 0.4\}$ 。本实验选用正交矩阵 $L16(4^3)$, 并且采用 HV 作为响应指标。表 4 列出了每个参数的平均 HV 值和显著性等级,其中 Delta 显示了每个参数不同级别之间 HV 的最大差异。根据表 4, PS 是最显著的参数,其次分别是 P_m 和 P_c 。从表 4 的结果可以看出, PS 设置为 104, $P_c = 0.9$, $P_m = 0.4$ 。

3.5 策略的有效性分析

为了验证工序跳跃启发式规则和混合交叉算子的有效性,将 3 个 MPHEA 变体与 MPHEA 进行了比较。在变体 1 中,删除混合交叉算子,均使用两点交叉,其他的过程和 MPHEA 一样;在变体 2 中,删除混合交叉算子,均使用均匀交叉,其他的过程和 MPHEA 一样;在变体 3 中,删除工序跳跃启发式规则,其他的过程和 MPHEA 一样。为了避免随机性,所有算法在 400 个测试实例上独立运行 20 次。按

表 3 对比算法的参数设置

Algorithm	PS	P_c	P_m
MPMOGA	200	1.0	0.1
NSGA-II	104	1.0	0.3
mMaOEA/AS	104	1.0	0.3
MaOEA/AS	104	1.0	0.1

表 4 平均 HV 值和各参数的排序

Level	Average HV		
	PS	P_c	P_m
1	0.072 71	0.080 44	0.079 34
2	0.079 73	0.080 07	0.080 11
3	0.085 07	0.081 61	0.080 90
4	0.085 08	0.080 48	0.082 24
Delta	0.012 37	0.001 54	0.002 90
Rank	1	3	2

f_n_s 分组的每个实例的最大、平均和最小 HV 结果如表 5 所示,其中最优 HV 以粗体标记。通过 Wilcoxon 秩和检验^[22]进一步分析了 MPHEA 及其变体之间的性能差异,其中:“+”表示经 Wilcoxon 秩和检验,比较算法的性能显著优于 MPHEA,显著性水平为 0.05;“-”表示 MPHEA 的性能明显优于比较算法;“=”表示比较结果之间无显著差异(表 6、表 7 中的“+”“-”“=”意义相同)。表 5 清楚地表明 MPHEA 在大部分测量算例中都达到了最佳值,并且和变体 1 与变体 3 的性能差异显著。MPHEA 在 3_80_10 和 3_100_10 显著优于变体 2,在大部分算例上和变体 2 没有显著性差异。

3.6 与其他算法的对比与分析

为了验证 MPHEA 在解决 DHHFLSPMO 方面的有效性,将其与 MPMOGA^[14]、NSGA-II^[16]、mMaOEA/AS^[17] 和 MaOEA/AS^[18] 进行比较,所有算法在每个算例上均进行 20 次独立运行。当跳跃率为 20% 时,5 种算法在 40 组算例上的平均 HV 和 IGD 结果如表 6 所示。当跳跃率为 40% 时,5 种算法在 40 组算例上的平均 HV 和 IGD 结果如表 7 所示。

如表 6 所示,当 $f = 2$ 、跳跃率为 20% 时,MPHEA 获得了 8 组实例的最优 HV 值和 7 组实例的最优 IGD 值。MPHEA 在 2_40_10 上的 HV 值和 NSGA-II 没有显著性差异,但是 MPHEA 在 2_40_10 上的 IGD 值不如 NSGA-II,这表明 MPHEA 在

表 5 40 组测试算例中 MPHEA 及其变体的 HV 值(20% 跳跃率)

Table 5 HV values for MPHEA and its variants on 40 groups of test instances (20% missing operations)

f_n_s	HV			
	Variant 1 of MPHEA	Variant 2 of MPHEA	Variant 3 of MPHEA	MPHEA
2_20_5	0.202 96(=)	0.205 66(=)	0.202 81(=)	0.206 15
2_20_10	0.210 19(-)	0.213 89(=)	0.207 98(-)	0.216 11
2_40_5	0.061 86(-)	0.068 97(=)	0.056 81(-)	0.069 91
2_40_10	0.060 79(=)	0.063 91(=)	0.054 89(-)	0.064 19
2_60_5	0.017 01(-)	0.021 38(=)	0.014 71(-)	0.022 78
2_60_10	0.012 92(-)	0.013 47(=)	0.009 65(-)	0.014 51
2_80_5	0.005 74(-)	0.008 08(=)	0.004 48(-)	0.008 14
2_80_10	0.001 43(=)	0.001 70(=)	0.000 50(-)	0.001 65
2_100_5	0.000 22(-)	0.000 75(=)	0.000 08(-)	0.000 61
2_100_10	0.000 02(-)	0.000 08(=)	0(-)	0.000 10
3_20_5	0.238 83(=)	0.241 43(=)	0.240 34(=)	0.241 87
3_20_10	0.252 16(-)	0.256 96(=)	0.250 82(-)	0.257 45
3_40_5	0.132 83(=)	0.136 37(=)	0.124 99(-)	0.137 58
3_40_10	0.128 46(=)	0.131 51(=)	0.120 16(-)	0.131 21
3_60_5	0.056 44(-)	0.060 78(=)	0.052 08(-)	0.062 35
3_60_10	0.046 27(-)	0.049 68(=)	0.041 10(-)	0.050 34
3_80_5	0.030 51(-)	0.034 25(=)	0.026 90(-)	0.036 22
3_80_10	0.011 83(-)	0.012 60(-)	0.008 57(-)	0.013 89
3_100_5	0.006 49(-)	0.008 89(=)	0.005 24(-)	0.009 83
3_100_10	0.002 67(-)	0.002 71(-)	0.001 23(-)	0.003 46
4_20_5	0.259 50(-)	0.261 59(=)	0.259 44(-)	0.262 99
4_20_10	0.269 61(=)	0.273 72(=)	0.267 97(-)	0.272 60
4_40_5	0.178 06(=)	0.179 46(=)	0.171 69(-)	0.180 50
4_40_10	0.173 63(-)	0.176 04(=)	0.167 63(-)	0.176 99
4_60_5	0.105 09(-)	0.106 92(=)	0.096 95(-)	0.108 93
4_60_10	0.090 28(-)	0.091 62(=)	0.083 94(-)	0.094 81
4_80_5	0.061 84(-)	0.064 96(=)	0.055 40(-)	0.066 65
4_80_10	0.032 02(-)	0.034 96(=)	0.027 22(-)	0.036 38
4_100_5	0.019 07(-)	0.023 62(=)	0.016 90(-)	0.024 33
4_100_10	0.014 55(-)	0.015 62(=)	0.011 30(-)	0.016 51
5_20_5	0.261 56(=)	0.263 38(=)	0.259 54(=)	0.262 06
5_20_10	0.266 41(=)	0.271 60(=)	0.263 50(-)	0.269 66
5_40_5	0.195 79(=)	0.196 21(=)	0.191 02(-)	0.197 48
5_40_10	0.204 76(-)	0.207 51(=)	0.199 71(-)	0.207 87
5_60_5	0.126 38(-)	0.127 39(=)	0.118 29(-)	0.130 18
5_60_10	0.115 08(=)	0.112 60(=)	0.103 76(-)	0.115 20
5_80_5	0.089 17(-)	0.090 00(=)	0.081 62(-)	0.091 48
5_80_10	0.055 40(=)	0.056 33(=)	0.046 76(-)	0.058 22
5_100_5	0.049 34(-)	0.051 85(=)	0.043 68(-)	0.053 78
5_100_10	0.024 33(-)	0.025 84(=)	0.020 38(-)	0.026 86

2_40_10 上获得的解集的收敛性不如 NSGA-II, 而解集的多样性和 NSGA-II 不相上下。当 $f=3$ 、跳跃率为 20% 时, MPHEA 获得了 7 组实例的最优 HV 值和 6 组实例的最优 IGD 值。当 $f=4$ 、跳跃率为 20% 时, 可以看出 NSGA-II 在 4_60_10 和 4_80_10 上的 IGD

值显著优于 MPHEA, 在 4_100_10 上的 HV 值和 IGD 值均显著优于 MPHEA。当 $f=5$ 、跳跃率为 20% 时, MPHEA 获得了 8 组实例的最优 HV 值和 7 组实例的最优 IGD 值。

当跳跃率为 40% 时, 所有算法的 HV 和 IGD

表 6 5 种算法在 40 组测试算例上 HV 和 IGD 结果的比较(跳跃率 20%)

Table 6 HV and IGD results for the five algorithms on 40 groups of test instances (20% missing operations)

f_n_s	MPMOGA		NSGA-II		mMaOEA/AS		MaOEA/AS		MPHEA	
	HV	IGD	HV	IGD	HV	IGD	HV	IGD	HV	IGD
2_20_5	0.1789(-)	0.1260(-)	0.2011(-)	0.0803(=)	0.1463(-)	0.1705(-)	0.1590(-)	0.1688(-)	0.2062	0.0781
2_20_10	0.1989(-)	0.0971(-)	0.2034(-)	0.0753(-)	0.1479(-)	0.1663(-)	0.1531(-)	0.1746(-)	0.2161	0.0702
2_40_5	0.0290(-)	0.3377(-)	0.0612(-)	0.1095(-)	0.0208(-)	0.3633(-)	0.0321(-)	0.2486(-)	0.0699	0.0918
2_40_10	0.0342(-)	0.2622(-)	0.0630(=)	0.0836(+)	0.0237(-)	0.2977(-)	0.0320(-)	0.2219(-)	0.0642	0.0895
2_60_5	0.0013(-)	0.5458(-)	0.0190(-)	0.1166(-)	0.0022(-)	0.5395(-)	0.0071(-)	0.2784(-)	0.0228	0.0907
2_60_10	0.0008(-)	0.7157(-)	0.0169(+)	0.1062(+)	0.0018(-)	0.5719(-)	0.0070(-)	0.2800(-)	0.0145	0.1451
2_80_5	0(-)	0.7474(-)	0.0066(-)	0.1261(-)	0.0002(-)	0.6636(-)	0.0017(-)	0.2838(-)	0.0081	0.1003
2_80_10	0(-)	1.0390(-)	0.0028(+)	0.0886(+)	0(-)	0.7939(-)	0.0004(-)	0.2732(-)	0.0017	0.1304
2_100_5	0(-)	1.0931(-)	0.0003(-)	0.1468(-)	0(-)	0.9689(-)	0(-)	0.4091(-)	0.0006	0.1210
2_100_10	0(-)	1.4930(-)	0.0004(+)	0.0933(+)	0(-)	1.2288(-)	0(-)	0.4304(-)	0.0001	0.1852
3_20_5	0.2212(-)	0.1080(-)	0.2307(-)	0.0857(-)	0.1983(-)	0.1520(-)	0.2096(-)	0.1615(-)	0.2419	0.0794
3_20_10	0.2433(-)	0.0856(-)	0.2435(-)	0.0783(-)	0.2090(-)	0.1515(-)	0.2128(-)	0.1754(-)	0.2575	0.0730
3_40_5	0.1001(-)	0.1960(-)	0.1253(-)	0.0913(-)	0.0831(-)	0.2133(-)	0.0902(-)	0.2075(-)	0.1376	0.0839
3_40_10	0.1071(-)	0.1572(-)	0.1265(-)	0.0807(=)	0.0822(-)	0.2085(-)	0.0906(-)	0.2057(-)	0.1312	0.0815
3_60_5	0.0267(-)	0.3532(-)	0.0581(-)	0.0965(=)	0.0257(-)	0.2883(-)	0.0372(-)	0.1946(-)	0.0624	0.0932
3_60_10	0.0250(-)	0.3153(-)	0.0508(=)	0.0879(+)	0.0230(-)	0.2726(-)	0.0310(-)	0.1947(-)	0.0503	0.0993
3_80_5	0.0074(-)	0.4868(-)	0.0343(=)	0.1045(=)	0.0086(-)	0.3952(-)	0.0216(-)	0.2034(-)	0.0362	0.0991
3_80_10	0.0010(-)	0.6599(-)	0.0150(+)	0.1045(+)	0.0028(-)	0.5107(-)	0.0095(-)	0.1811(-)	0.0139	0.1265
3_100_5	0.0001(-)	0.7836(-)	0.0083(-)	0.1380(-)	0.0005(-)	0.7105(-)	0.0050(-)	0.2184(-)	0.0098	0.1133
3_100_10	0(-)	0.9829(-)	0.0050(+)	0.0986(+)	0.0001(-)	0.7819(-)	0.0024(-)	0.1818(-)	0.0035	0.1493
4_20_5	0.2426(-)	0.1032(-)	0.2501(-)	0.0859(-)	0.2248(-)	0.1442(-)	0.2329(-)	0.1618(-)	0.2630	0.0770
4_20_10	0.2594(-)	0.0853(-)	0.2591(-)	0.0791(-)	0.2398(-)	0.1411(-)	0.2407(-)	0.1726(-)	0.2726	0.0698
4_40_5	0.1497(-)	0.1373(-)	0.1665(-)	0.0921(-)	0.1293(-)	0.1881(-)	0.1355(-)	0.2047(-)	0.1805	0.0818
4_40_10	0.1547(-)	0.1227(-)	0.1666(-)	0.0856(-)	0.1265(-)	0.1889(-)	0.1339(-)	0.2073(-)	0.1770	0.0799
4_60_5	0.0741(-)	0.2302(-)	0.1033(-)	0.0946(-)	0.0648(-)	0.2426(-)	0.0745(-)	0.2417(-)	0.1089	0.0897
4_60_10	0.0730(-)	0.1956(-)	0.0943(=)	0.0866(+)	0.0568(-)	0.2360(-)	0.0662(-)	0.2223(-)	0.0948	0.0939
4_80_5	0.0342(-)	0.3112(-)	0.0646(=)	0.0930(=)	0.0357(-)	0.2614(-)	0.0464(-)	0.2104(-)	0.0667	0.0930
4_80_10	0.0138(-)	0.3968(-)	0.0362(=)	0.1115(+)	0.0156(-)	0.3157(-)	0.0274(-)	0.1868(-)	0.0364	0.1265
4_100_5	0.0029(-)	0.4781(-)	0.0235(=)	0.0980(=)	0.0055(-)	0.3670(-)	0.0172(-)	0.1659(-)	0.0243	0.0985
4_100_10	0.0019(-)	0.6244(-)	0.0199(+)	0.1034(+)	0.0037(-)	0.4646(-)	0.0147(-)	0.1473(-)	0.0165	0.1527
5_20_5	0.2398(-)	0.1015(-)	0.2490(-)	0.0825(-)	0.2301(-)	0.1430(-)	0.2378(-)	0.1593(-)	0.2621	0.0736
5_20_10	0.2528(-)	0.0894(-)	0.2554(-)	0.0806(-)	0.2438(-)	0.1383(-)	0.2472(-)	0.1648(-)	0.2697	0.0721
5_40_5	0.1707(-)	0.1233(-)	0.1861(-)	0.0910(-)	0.1533(-)	0.1738(-)	0.1605(-)	0.1990(-)	0.1975	0.0830
5_40_10	0.1863(-)	0.1087(-)	0.1944(-)	0.0907(-)	0.1633(-)	0.1880(-)	0.1689(-)	0.2172(-)	0.2079	0.0789
5_60_5	0.0987(-)	0.1796(-)	0.1228(-)	0.0941(-)	0.0881(-)	0.2213(-)	0.0950(-)	0.2327(-)	0.1302	0.0858
5_60_10	0.0958(-)	0.1611(-)	0.1101(-)	0.0986(-)	0.0775(-)	0.2379(-)	0.0869(-)	0.2402(-)	0.1152	0.0919
5_80_5	0.0579(-)	0.2616(-)	0.0890(-)	0.0977(=)	0.0572(-)	0.2490(-)	0.0682(-)	0.2343(-)	0.0915	0.0961
5_80_10	0.0364(-)	0.2883(-)	0.0583(=)	0.1088(+)	0.0326(-)	0.2515(-)	0.0459(-)	0.1918(-)	0.0582	0.1203
5_100_5	0.0241(-)	0.3513(-)	0.0535(=)	0.1019(+)	0.0293(-)	0.2591(-)	0.0412(-)	0.1989(-)	0.0538	0.1104
5_100_10	0.0099(-)	0.4279(-)	0.0299(+)	0.1102(+)	0.0126(-)	0.3079(-)	0.0233(-)	0.1606(-)	0.0269	0.1508

的结果如表 7 所示。虽然 NSGA-II 在 2_60_10、2_80_10、3_100_5、3_100_10 和 4_100_10 上获得了最优的 HV 值,但是 NSGA-II 和 MPHEA 的 HV 值在这些算例上没有显著性差异。因此,MPHEA 在大部分算例上 HV 值均优于对比算法。关于 IGD 值,

NSGA-II 在 2_40_10、2_80_10、4_80_10 和 5_100_10 上获得了最优的 IGD 值,然而 Wilcoxon 秩和检验结果表明 NSGA-II 和 MPHEA 的 IGD 值在这些算例上没有显著性差异。在统计意义上,MPHEA 除在 2_60_10、2_100_10、3_60_10、3_80_10、3_100_5、

表 7 5 种算法在 40 组测试算例上 HV 和 IGD 结果的比较(跳跃率 40%)

Table 7 HV and IGD results for the five algorithms on 40 groups of test instances (40% missing operations)

f_n_s	MPMOGA		NSGA-II		mMaOEA/AS		MaOEA/AS		MPHEA	
	HV	IGD	HV	IGD	HV	IGD	HV	IGD	HV	IGD
2_20_5	0.1954(-)	0.1151(-)	0.2100(-)	0.0811(=)	0.1638(-)	0.1624(-)	0.1789(-)	0.1632(-)	0.2184	0.0796
2_20_10	0.2203(-)	0.0991(-)	0.2299(-)	0.0763(-)	0.1879(-)	0.1535(-)	0.1931(-)	0.1774(-)	0.2413	0.0700
2_40_5	0.0750(-)	0.2284(-)	0.1084(-)	0.0921(=)	0.0642(-)	0.2406(-)	0.0795(-)	0.1948(-)	0.1127	0.0896
2_40_10	0.0928(-)	0.1871(-)	0.1166(-)	0.0832(=)	0.0666(-)	0.2347(-)	0.0764(-)	0.2081(-)	0.1209	0.0833
2_60_5	0.0185(-)	0.3911(-)	0.0505(-)	0.1076(-)	0.0171(-)	0.3875(-)	0.0314(-)	0.2186(-)	0.0559	0.0950
2_60_10	0.0165(-)	0.3974(-)	0.0488(=)	0.0884(+)	0.0153(-)	0.3645(-)	0.0273(-)	0.2182(-)	0.0468	0.1065
2_80_5	0.0079(-)	0.4695(-)	0.0353(=)	0.1059(-)	0.0080(-)	0.4565(-)	0.0210(-)	0.2199(-)	0.0367	0.0990
2_80_10	0.0006(-)	0.7517(-)	0.0144(=)	0.0983(=)	0.0008(-)	0.5993(-)	0.0070(-)	0.2067(-)	0.0137	0.1094
2_100_5	0(-)	0.7841(-)	0.0076(=)	0.1396(-)	0(-)	0.7264(-)	0.0033(-)	0.2460(-)	0.0084	0.1233
2_100_10	0(-)	1.0857(-)	0.0046(+)	0.1026(+)	0(-)	0.8977(-)	0.0013(-)	0.2401(-)	0.0036	0.1395
3_20_5	0.2322(-)	0.1002(-)	0.2397(-)	0.0845(-)	0.2105(-)	0.1506(-)	0.2221(-)	0.1593(-)	0.2523	0.0783
3_20_10	0.2543(-)	0.0891(-)	0.2582(-)	0.0785(-)	0.2318(-)	0.1421(-)	0.2382(-)	0.1629(-)	0.2703	0.0741
3_40_5	0.1293(-)	0.1582(-)	0.1523(-)	0.0888(-)	0.1153(-)	0.1872(-)	0.1244(-)	0.1953(-)	0.1614	0.0836
3_40_10	0.1566(-)	0.1269(-)	0.1722(-)	0.0841(-)	0.1292(-)	0.1988(-)	0.1377(-)	0.2183(-)	0.1809	0.0793
3_60_5	0.0695(-)	0.2534(-)	0.1027(-)	0.0940(-)	0.0640(-)	0.2378(-)	0.0774(-)	0.2114(-)	0.1076	0.0890
3_60_10	0.0708(-)	0.2281(-)	0.0993(=)	0.0882(+)	0.0600(-)	0.2434(-)	0.0699(-)	0.2267(-)	0.1004	0.0937
3_80_5	0.0374(-)	0.3333(-)	0.0738(=)	0.0947(=)	0.0388(-)	0.2909(-)	0.0554(-)	0.2164(-)	0.0763	0.0928
3_80_10	0.0150(-)	0.3942(-)	0.0413(=)	0.0932(+)	0.0168(-)	0.2969(-)	0.0284(-)	0.1721(-)	0.0415	0.1032
3_100_5	0.0072(-)	0.4528(-)	0.0384(=)	0.1075(+)	0.0111(-)	0.3896(-)	0.0289(-)	0.1840(-)	0.0381	0.1138
3_100_10	0.0024(-)	0.5758(-)	0.0230(=)	0.1091(=)	0.0044(-)	0.4384(-)	0.0165(-)	0.1746(-)	0.0223	0.1283
4_20_5	0.2370(-)	0.1030(-)	0.2489(-)	0.0807(-)	0.2208(-)	0.1498(-)	0.2322(-)	0.1671(-)	0.2588	0.0799
4_20_10	0.2543(-)	0.0906(-)	0.2612(-)	0.0776(-)	0.2411(-)	0.1453(-)	0.2473(-)	0.1704(-)	0.2740	0.0714
4_40_5	0.1720(-)	0.1154(-)	0.1886(-)	0.0878(-)	0.1583(-)	0.1818(-)	0.1637(-)	0.2216(-)	0.1963	0.0817
4_40_10	0.1820(-)	0.1191(-)	0.1957(-)	0.0867(-)	0.1646(-)	0.1818(-)	0.1734(-)	0.2088(-)	0.2045	0.0824
4_60_5	0.1074(-)	0.1845(-)	0.1367(-)	0.0937(-)	0.0985(-)	0.2202(-)	0.1087(-)	0.2400(-)	0.1422	0.0876
4_60_10	0.1248(-)	0.1510(-)	0.1470(=)	0.0897(-)	0.1049(-)	0.2409(-)	0.1161(-)	0.2501(-)	0.1497	0.0845
4_80_5	0.0773(-)	0.2159(-)	0.1084(-)	0.0947(-)	0.0733(-)	0.2456(-)	0.0857(-)	0.2450(-)	0.1125	0.0895
4_80_10	0.0530(-)	0.2638(-)	0.0782(=)	0.1045(=)	0.0485(-)	0.2480(-)	0.0632(-)	0.2167(-)	0.0813	0.1048
4_100_5	0.0286(-)	0.3731(-)	0.0615(=)	0.1063(+)	0.0335(-)	0.2738(-)	0.0488(-)	0.1922(-)	0.0616	0.1147
4_100_10	0.0211(-)	0.3830(-)	0.0490(=)	0.1116(+)	0.0237(-)	0.2904(-)	0.0384(-)	0.1894(-)	0.0487	0.1265
5_20_5	0.2451(-)	0.1040(-)	0.2622(-)	0.0795(-)	0.2308(-)	0.1534(-)	0.2433(-)	0.1645(-)	0.2742	0.0730
5_20_10	0.2534(-)	0.0892(-)	0.2590(-)	0.0781(-)	0.2476(-)	0.1398(-)	0.2534(-)	0.1612(-)	0.2703	0.0713
5_40_5	0.1896(-)	0.1101(-)	0.2055(-)	0.0898(-)	0.1758(-)	0.1914(-)	0.1810(-)	0.2296(-)	0.2156	0.0822
5_40_10	0.2107(-)	0.1021(-)	0.2214(-)	0.0868(-)	0.1952(-)	0.1772(-)	0.2034(-)	0.2109(-)	0.2300	0.0814
5_60_5	0.1349(-)	0.1523(-)	0.1590(-)	0.0968(-)	0.1247(-)	0.2110(-)	0.1338(-)	0.2355(-)	0.1672	0.0847
5_60_10	0.1404(-)	0.1383(-)	0.1563(-)	0.0977(-)	0.1224(-)	0.2289(-)	0.1340(-)	0.2485(-)	0.1645	0.0848
5_80_5	0.0916(-)	0.1942(-)	0.1223(-)	0.0996(-)	0.0895(-)	0.2411(-)	0.1010(-)	0.2569(-)	0.1263	0.0921
5_80_10	0.0862(-)	0.1777(-)	0.1062(-)	0.1022(-)	0.0722(-)	0.2603(-)	0.0864(-)	0.2600(-)	0.1110	0.0933
5_100_5	0.0629(-)	0.2416(-)	0.0945(=)	0.1095(=)	0.0640(-)	0.2607(-)	0.0777(-)	0.2649(-)	0.0955	0.1082
5_100_10	0.0521(-)	0.2615(-)	0.0757(=)	0.1120(=)	0.0465(-)	0.2586(-)	0.0618(-)	0.2165(-)	0.0774	0.1153

3_100_10、4_100_5 和 4_100_10 上的 IGD 值不如 NSGA-II 的外,在大部分的算例上均优于对比算法。

综上所述,MPHEA 在 80 组算例上表现出令人满意的结果,显著优于其对比算法,即 MPHEA 获得

的最终解集的收敛性和多样性显著优于对比算法。

图 5 显示了 MPHEA 及其对比算法在一个工序跳跃率为 20% 的 5_100_10 算例中的最终解集,分别显示了 $C_{\max}$ 和 F_{tot} 、 $C_{\max}$ 和 M_{idl} 、 $C_{\max}$ 和 T_{wgt} 、 F_{tot}

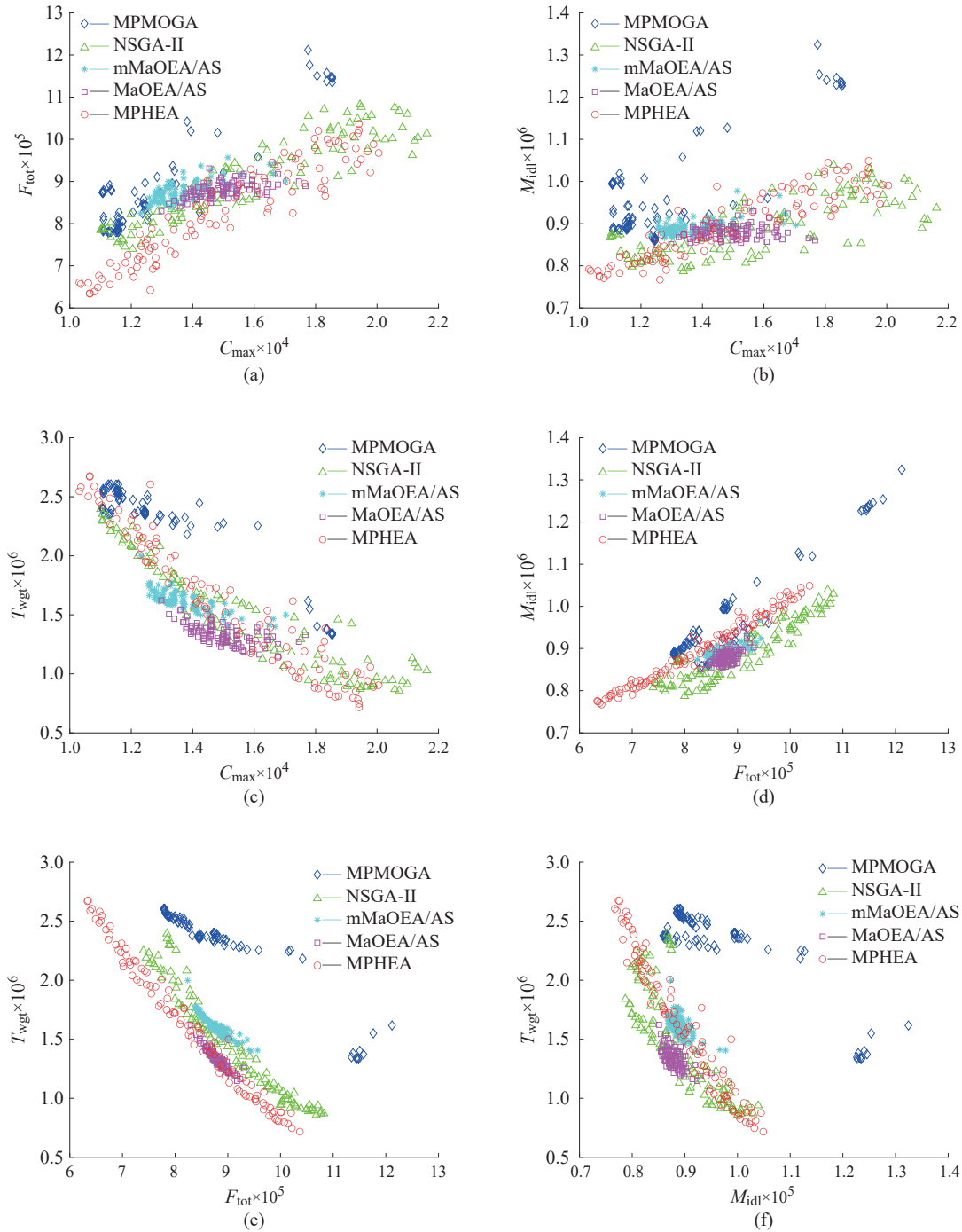


图5 关于二维目标空间的5_100_10算例上所有算法的最终解

Fig. 5 Final solutions of all algorithms on 5_100_10 instance regarding the 2-D objective space

和 M_{idl} , F_{tot} 和 T_{wgt} 以及 M_{idl} 和 T_{wgt} 的二维视图。可以看出 MPHEA 分别在 4 个目标上找到的解均优于其他算法, 在收敛性和多样性方面, 也可以求得令人满意的非支配解。

4 结 论

针对高维多目标 DHHFLSPMO 问题, 提出了一种 MPHEA 算法, 用于优化问题中的最大完工时间、

总流经时间、机器等待时间以及总加权提前时间和延迟时间 4 个目标。首先, 给出各个优化目标的计算方式, 并制定了编码方式。为增强解集多样性, 利用两点交叉和均匀交叉两种不同的交叉策略进行混合交叉。然后, 设计了总种群与 4 个子种群的并行协同进化以搜索不同的解空间区域。此外, 根据优化目标的特点, 采用了相应的工序跳跃启发式规则。最后, 在 400 个算例上进行了实验, 比较了不同的知名算法的结果, 实证结果表明 MPHEA 在求解

DHHFLSPMO 上优于其他算法。

参考文献:

- [1] 黄佳琳, 张丫丫, 顾幸生. 基于改进生物地理学优化算法的分布式装配换流水车间调度问题[J]. 华东理工大学学报(自然科学版), 2020, 46(6): 758-769.
- [2] LUO C, GONG W, LI R, LU C. Problem-specific knowledge MOEA/D for energy-efficient scheduling of distributed permutation flow shop in heterogeneous factories[J]. Engineering Applications of Artificial Intelligence, 2023, 123: 106454.
- [3] MENG T, PAN Q, LI J, *et al.* An improved migrating birds optimization for an integrated lot-streaming flow shop scheduling problem[J]. Swarm and Evolutionary Computation, 2018, 38: 64-78.
- [4] DUAN J, MENG T, CHEN Q, *et al.* An effective artificial bee colony for distributed lot-streaming flowshop scheduling problem[C]//International Conference on Intelligent Computing ICIC 2018: Intelligent Computing Methodologies. Cham, Switzerland: Springer International Publishing, 2018: 795-806.
- [5] PAN Y, GAO K, LI Z, *et al.* Solving biobjective distributed flow-shop scheduling problems with lot-streaming using an improved Jaya algorithm[J]. IEEE Transactions on Cybernetics, 2023, 53(6): 3818-3828.
- [6] PAN Y, GAO K, LI Z, *et al.* Improved meta-heuristics for solving distributed lot-streaming permutation flow shop scheduling problems[J]. IEEE Transactions on Automation Science and Engineering, 2023, 20(1): 361-371.
- [7] MENG T, PAN Q. A distributed heterogeneous permutation flowshop scheduling problem with lot-streaming and carryover sequence-dependent setup time[J]. Swarm and Evolutionary Computation, 2021, 60: 100804.
- [8] SHAO W, SHAO Z, PI D. Modelling and optimization of distributed heterogeneous hybrid flow shop lot-streaming scheduling problem[J]. Expert Systems With Applications, 2023, 214: 119151.
- [9] LEISTEN R, KOLBE M. A note on scheduling jobs with missing operations in permutation flow shops[J]. International Journal of Production Research, 1998, 36(9): 2627-2630.
- [10] DIOS M, FERNANDEZ-VIAGAS V, FRAMINAN J M. Efficient heuristics for the hybrid flow shop scheduling problem with missing operations[J]. Computers & Industrial Engineering, 2018, 115: 88-99.
- [11] 江国焕. 允许工序跳跃的混合流水线调度优化研究[D]. 福州: 福州大学, 2019.
- [12] TSENG C, LIAO C, LIAO T. A note on two-stage hybrid flowshop scheduling with missing operations[J]. Computers & Industrial Engineering, 2008, 54: 695-704.
- [13] MARICHEL VAM M. Performance evaluation of an improved hybrid genetic scatter search (IHGSS) algorithm for multistage hybrid flow shop scheduling problems with missing operations[J]. International Journal of Industrial and Systems Engineering, 2014, 16(1): 120-141.
- [14] LIU S, CHEN Z, ZHAN Z, *et al.* Many-objective job-shop scheduling: A multiple populations for multiple objectives-based genetic algorithm approach[J]. IEEE Transactions on Cybernetics, 2023, 53(3): 1460-1474.
- [15] SYSWERDA G. Uniform crossover in genetic algorithms[C]//Proceedings of the 3rd International Conference on Genetic Algorithms. San Francisco: Morgan Kaufmann, 1989.
- [16] DEB K, PRATAP A, AGARWAL S, *et al.* A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197.
- [17] CHEN S, WANG X, WANG Y, *et al.* A modified adaptive switching based many-objective evolutionary algorithm for distributed heterogeneous flowshop scheduling with lot-streaming[J]. Swarm and Evolutionary Computation, 2023, 81: 101353.
- [18] CHEN S, WANG X, GAO J, *et al.* An adaptive switching-based evolutionary algorithm for many-objective optimization[J]. Knowledge-Based Systems, 2022, 248: 108915.
- [19] COELLO C A C, CORTES N C. Solving multiobjective optimization problems using an artificial immune system[J]. Genetic Programming and Evolvable Machines, 2005, 6(2): 163-190.
- [20] ZITZLER E, THIELE L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach[J]. IEEE Transactions on Evolutionary Computation, 1999, 3(4): 257-27.
- [21] MONTGOMERY D. Design and Analysis of Experiments[M]. Hoboken, NJ: John Wiley & Sons, 2005.
- [22] DERRAC J, GARCA S, MOLINA D, *et al.* A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms[J]. Swarm and Evolutionary Computation, 2011, 1(1): 3-18.

Multiple-Population Hybrid Evolutionary Algorithm for Solving Distributed Heterogeneous Lot-Streaming Hybrid Flowshop Scheduling Problem with Missing Operations

CHEN Sanyan, WANG Xuewu, WANG Ye, GU Xingsheng

(Key Laboratory of Smart Manufacturing in Energy Chemical Process, Ministry of Education, East China University of Science and Technology, Shanghai 200237, China)

Abstract: A Multi-Population Hybrid Evolutionary Algorithm (MPHEA) was proposed for the distributed heterogeneous lot-streaming hybrid flowshop scheduling problem with missing operations, aiming to minimize the makespan, total flow time, idle time of machines, and total weighted earliness and tardiness. Firstly, the calculation method for each optimization objective was provided, and an appropriate coding scheme was developed. Then, a parallel coevolutionary strategy was designed between the total population and the four subpopulations. After performing hybrid crossover and mutation operations on the population, the population was updated based on Pareto dominance and crowding distance to ensure diversity and preservation of high-quality solutions. Furthermore, partial solutions were extracted from the population to form four subpopulations, each focusing on optimizing a specific objective. The parallel evolution of the population and subpopulations allows subpopulations to avoid overly focusing on a particular objective when evolving in their respective directions, thereby achieving balanced optimization of multiple objectives. Considering the impact of missing operations on scheduling, corresponding heuristic rules of missing operations was designed. Finally, the effectiveness of MPHEA in solving the scheduling problem was verified through 400 instances. The results of four well-known algorithms were compared and the empirical results showed that MPHEA outperformed the compared algorithms.

Key words: missing operation; distributed shop scheduling; lot-streaming; heterogeneous factory; many-objective optimization

(责任编辑: 刘亚萍)