

DOI:10.3969/j.issn.1003-5060.2025.12.015

考虑机器故障的分布式异构柔性流水车间重调度

轩 华, 熊梦莹, 李 冰

(郑州大学 管理学院, 河南 郑州 450001)

摘 要:为解决考虑机器故障的分布式异构柔性流水车间重调度问题(distributed heterogeneous flexible flowshop rescheduling problem, DHFFRP),文章建立以最小化最大完工时间为目标的整数规划模型,提出采用一种融合遗传算法、变邻域搜索策略和模拟退火思想的混合人工蜂群算法(hybrid artificial bee colony algorithm, HABCA)进行求解。为应对动态变化的工序状态,根据故障时间、故障机器区分不同状态的工序,从而将上述动态问题转化为一系列阶段性的静态问题。在 HABCA 中设计 3 层编码方式和插入式贪婪解码机制;在雇佣蜂搜索过程中引入交叉和变异策略优化种群的蜜源;在跟随蜂搜索过程中采用基于最小完工时间规则的局域搜索算子;在侦查蜂搜索过程中应用 4 种变邻域搜索算子、模拟退火操作进行解的进化。通过大量的实例测试,验证重调度的性能相较于初始调度有显著的提高;将所提算法与经典人工蜂群算法、已有算法进行比较,结果表明 HABCA 的有效性和优越性。

关键词:分布式异构柔性流水车间;机器故障;重调度;混合人工蜂群算法(HABCA);模拟退火操作

中图分类号:TB49

文献标志码:A

文章编号:1003-5060(2025)12-1678-11

Rescheduling of distributed heterogeneous flexible flowshop considering machine breakdown

XUAN Hua, XIONG Mengying, LI Bing

(School of Management, Zhengzhou University, Zhengzhou 450001, China)

Abstract: In order to solve the distributed heterogeneous flexible flowshop rescheduling problem(DHFFRP) considering machine breakdown, an integer planning model with the objective of minimizing the maximum completion time is established, and a hybrid artificial bee colony algorithm(HABCA), which incorporates genetic algorithms, variable neighborhood search strategies, and the idea of simulated annealing, is proposed for problem-solving. In order to cope with the dynamically changing process states, the above dynamic problem is transformed into a series of static problems with phases by distinguishing different states of the process according to the failure time and the faulty machine. In HABCA, a three-layer coding method and an insertion-based greedy decoding mechanism are designed; in the employed bees search process, crossover and mutation strategies are introduced to optimize the nectar sources in the population; in the onlooker bees search process, a local search operator based on the minimum completion time rule is used; and in the scout bees search process, four kinds of variable neighborhood search operators and simulated annealing operations are applied for the evolution of the solution. The performance improvement between initial scheduling and rescheduling is verified through a large number of example tests; the proposed algorithm is compared with the classical artificial bee colony algorithm and existing algorithms, and the results show the effectiveness and superiority of HABCA.

Key words: distributed heterogeneous flexible flowshop; machine breakdown; rescheduling; hybrid artificial bee colony algorithm(HABCA); simulated annealing operation

收稿日期:2024-06-21;修回日期:2024-08-04

基金项目:国家自然科学基金青年科学基金资助项目(72201251);河南省自然科学基金面上资助项目(252300421945)和河南省科技研发计划联合基金资助项目(242103810046)

作者简介:轩 华(1979—),女,河南睢县人,博士,郑州大学教授,硕士生导师,通信作者,E-mail:hxnan@zzu.edu.cn.

0 引 言

分布式柔性流水车间调度涉及到多个工厂之间的协同作业,旨在优化工件在工厂间的分配及在每个工厂内的加工顺序,从而达到调度指标的最优化^[1]。目前,全球产业的发展趋势朝着多品种、小批量的方向发展,分布式柔性流水车间调度的特征恰好契合这一转变。然而,在实际制造环境中,受到众多因素的影响,各工厂的内部组成难以保持完全一致。鉴于此,针对分布式异构柔性流水车间问题(distributed heterogeneous flexible flowshop problem, DHFFP)的研究更贴近真实的制造过程,具有重要的实践意义。同时,难以预测的扰动因素时常发生,如紧急插单、机器故障等,考虑到机器故障较为常见,本文主要研究考虑机器故障的分布式异构柔性流水车间重调度问题(distributed heterogeneous flexible flowshop re-scheduling problem, DHFFRP),以确保生产流程的顺畅运行和调度指标的优化。因为 DHFFP 已被证明是 NP-hard 问题^[2],所研究的更复杂的 DHFFRP 也是 NP-hard 问题,所以研究该类问题的有效求解方法成为急待解决的理论问题。

近年来,很多学者对 DHFFP 进行了深入研究,为求解以最小化最大完工时间为目标的调度问题,文献[3]针对 DHFFP 提出迭代贪婪算法;文献[4]提出改进的离散人工蜂群算法求解 DHFFP;文献[5]针对部分工件阶段可跳跃的 DHFFP,提出分布式 NEH 算法以及多邻域迭代贪心算法;文献[6]在 DHFFP 中考虑相邻工序间的运输时间,采用改进的布谷鸟算法求解问题。为求解多目标问题,文献[7]提出改进的蛙跳算法求解考虑顺序相关准备时间的 DHFFP,以同时最小化拖后工件数和最大完工时间;文献[8]为实现最小化最大完工时间和总电力成本提出自学习机制的多目标帝王蝶优化算法,求解带分时电价约束的 DHFFP;文献[9]为 DHFFP 提出一种优化人工蜂群算法,目标是同时最小化最大完工时间和总能耗。

围绕机器故障问题,针对柔性流水车间重调度,文献[10]以最小化最大完工时间为目标,利用融合帝国竞争算法和完全重调度策略的方法进行求解;文献[11]提出基于局部修复的重调度算法以最小化最大完工时间;文献[12]利用非支配排序遗传算法求解机器故障约束下机器负荷失衡率和最大完工时间同时最小化问题;文献[13]采用

候鸟优化算法求解考虑机器故障和作业取消的重调度问题,目标是最大完工时间、作业开始时间偏差和机器分配差同时达到最小化。针对柔性作业车间重调度,文献[14]考虑机器故障、紧急插单和模糊质检时间,提出改进遗传算法(genetic algorithm, GA)以最小化最大完工时间;文献[15]在柔性作业车间中考虑新工件到来、机器故障、作业取消以及变化工序处理时间 4 种情况,设计一种基于蒙特卡洛树搜索算法的重调度方法以最小化最大完工时间;文献[16]考虑紧急订单、截止日变更机械故障、订单取消,提出一种贪婪随机自适应搜索算法求解问题,根据词典法对最小化平均迟到率、进度不稳定性、完工时间、平均流程时间进行评估。

虽然关于重调度问题已经受到一定程度的关注,但针对 DHFFP 的研究主要还是集中于静态调度,缺乏对实际生产过程中实时发生的不确定事件(如机器故障)导致的重调度问题的研究。因此,本文采用混合人工蜂群算法(hybrid artificial bee colony algorithm, HABCA)求解 DHFFRP 在机器故障约束条件下的优化,根据故障发生时间和故障机器对工序进行细致划分,针对不同情况的工序采取差异化的重调度策略,以实现最大完工时间的最小化。

1 考虑机器故障的 DHFFRP 描述及模型

1.1 考虑机器故障的 DHFFRP 描述

本文研究的 DHFFRP 可描述如下: J 个工件可在 F 个具有相同加工流程但各工序的机器数量和处理时长不同的柔性流水车间中的任意一个车间进行加工;所有工件在 0 时刻进入生产车间,然后按相同的加工方向访问 I 道工序,每道工序 i 上有 $M_{f,i}$ 台不相关并行机,且至少存在一道工序的 $M_{f,i} > 1$;一些机器可能会发生故障,其不可用时间段事先已知,即故障机器 g 在故障开始时间 T_s 不可用,此时该机器将停止运作进行维修,在机器故障的结束时间 T_c 重新投入使用;若其他机器在 T_s 时正在处理工件,则工件在其他机器的当前工序可继续加工,无须重调度;调度的目标是最小化最大完工时间。

假设条件如下:

- 1) 每台机器一次只能处理一个工件,每个工件一次至多在一台机器上加工。
- 2) 不考虑工序间的运输时间。
- 3) 每道工序不允许中断。

1.2 基于故障机器的重调度策略划分

根据故障发生的信息,假设 g 为故障机器、 i 为 g 所在工序, $[T_s, T_c)$ 为发生故障的时间区间, 确定工序所处状态, 进而划分不同的工序集以制定差异化重调度策略。

1) 若工件 j 在机器故障开始时间 T_s 时已经完成工序 i 的加工或正在非故障机器上加工, 则将相应工序 $O_{j,i}$ 归入稳定调度工序集 Ω_1 , 这类工序将保持原有调度方案。

2) 若工件 j 在故障发生时, 正在故障机器 g 上加工, 则判断该故障机器是否存在可用并行机, 若不存在, 则将 $O_{j,i}$ 划入右移重调度工序集 Ω_2 ; 否则比较工件在可用并行机上的加工时间和在当前故障机器 g 上的加工时间与维修时间之和, 若工序 $O_{j,i}$ 在当前故障机器 g 的占用时间较短, 则将 $O_{j,i}$ 归入 Ω_2 。这类工序将在故障修复结束后仍留在当前机器上加工。若工序 $O_{j,i}$ 在可用并行机上加工时间较短, 则将 $O_{j,i}$ 归入并行机重调度工序集 Ω_3 。这类工序将移至当前工序的具有较短加工时间的并行机上加工。

3) 若工件 j 的开始加工时间在机器故障的开始时间 T_s 之后, 则将 $O_{j,i}$ 划入完全重调度工序集 Ω_4 。这类工序将重新进行调度安排以合理地应对故障问题。

机器故障处理流程如图 1 所示。

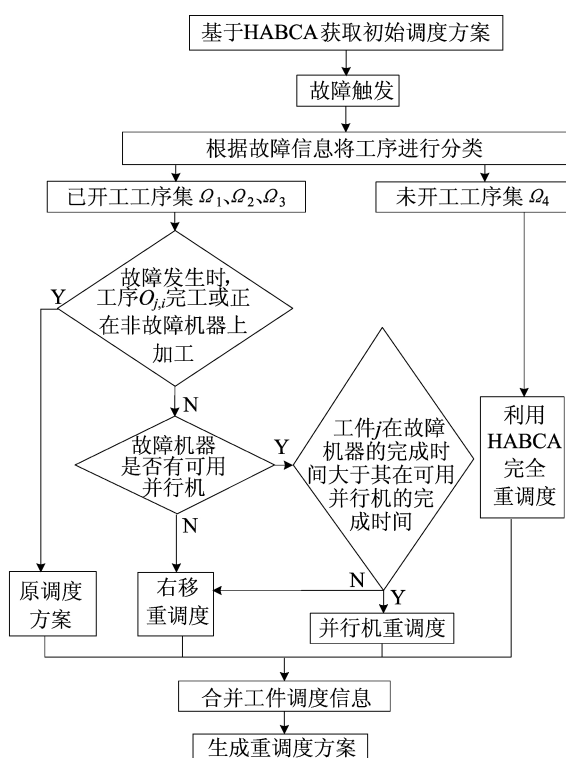


图 1 机器故障处理流程

1.3 符号定义

参数设置如下: f 为工厂标号, $f=1, 2, \dots$, F, F 为工厂数; j 为工件标号, $j=1, 2, \dots, J, J$ 为工件数; i 为工序标号, $i=1, 2, \dots, I, I$ 为工序数; g, g' 为机器标号, $g, g'=1, 2, \dots, M_{f,i}, M_{f,i}$ 为工序 i 的不相关并行机数, $g' \neq g$; $O_{j,i}$ 为工件 j 的第 i 道工序; T_s 为机器故障的开始时间; T_c 为机器故障修复的结束时间; $P_{f,j,i,g}$ 为工件 j 在工厂 f 内工序 i 的机器 g 的加工时间; L 为一个很大的数。

决策变量如下: $S_{j,i}, C_{j,i}$ 分别为初始调度方案中工件 j 在工序 i 的开始时间、完工时间; $S_{j,i}^*, C_{j,i}^*$ 分别为重调度方案中工件 j 在工序 i 的开始时间、完工时间; $X_{f,j}/X_{f,j}^*$ 为二元变量, 若初始调度/重调度方案中工件 j 分配到工厂 f , 则其值为 1, 否则为 0; $Y_{f,j,i,g}/Y_{f,j,i,g}^*$ 为二元变量, 若初始调度/重调度方案中工件 j 分配到工厂 f 内工序 i 的机器 g 上处理, 则其值为 1, 否则为 0; $W_{f,j',j,i,g}$ 为二元变量, 若工件 j', j 均分配至工厂 f 内工序 i 的机器 g 上且工件 j' 是工件 j 的紧前工件, 则 $W_{f,j',j,i,g}=1$, 否则 $W_{f,j',j,i,g}=0$ 。

1.4 模型

1.4.1 静态调度

以最小化最大完工时间为目标, 建立 DHF-FRP 的静态调度模型, 即

$$\min C_{\max} = \min \max\{C_{j,I}\} \quad (1)$$

$$\text{s. t.} \quad \sum_{f=1}^F X_{f,j} = 1, \quad \forall j \quad (2)$$

$$X_{f,j} = \sum_{g=1}^{M_{f,i}} Y_{f,j,i,g}, \quad \forall f, j, i \quad (3)$$

$$C_{j,i} \leq S_{j,(i+1)}, \quad \forall j, i \quad (4)$$

$$C_{j,i} = S_{j,i} + \sum_{f=1}^F \sum_{g=1}^{M_{f,i}} Y_{f,j,i,g} P_{f,j,i,g}, \quad \forall j, i \quad (5)$$

$$W_{f,j',j,i,g} + W_{f,j,j',i,g} \leq 1, \quad \forall f, j', j, i, g; j' \neq j \quad (6)$$

$$C_{j,i} \geq C_{j',i} + \sum_{f=1}^F \sum_{g=1}^{M_{f,i}} Y_{f,j,i,g} P_{f,j,i,g} - L(3 - Y_{f,j',i,g} - Y_{f,j,i,g} - W_{f,j',j,i,g}), \quad \forall j', j, i; j' \neq j \quad (7)$$

$$S_{j,i}, C_{j,i} \geq 0, \quad \forall j, i \quad (8)$$

$$X_{f,j}, Y_{f,j,i,g} \in \{0, 1\}, \quad \forall f, j, i, g \quad (9)$$

式(1)为所研究问题的目标, 即最小化最大完工时间; 约束式(2)确保每个工件只能在一个工厂加工; 约束式(3)表示每个工件在任一道工序只能

在一台机器上加工;约束式(4)定义了同一工件的相邻两道工序之间的优先级关系;约束式(5)表示同一工件的开完工时间关系;约束式(6)、式(7)确保一台机器一次最多只能处理一个工件;约束式(8)、式(9)说明决策变量的取值范围。

1.4.2 重调度

在 DHFFRP 中考虑机器故障,建立重调度模型。

模型 1 $O_{j,i} \in \Omega_1$

$$S_{j,i}^* = S_{j,i}, \quad \forall O_{j,i} \in \Omega_1 \quad (10)$$

$$C_{j,i}^* = C_{j,i}, \quad \forall O_{j,i} \in \Omega_1 \quad (11)$$

$$Y_{f,j,i,g}^* = Y_{f,j,i,g}, \quad \forall f,g; O_{j,i} \in \Omega_1 \quad (12)$$

$$X_{f,j}^* = X_{f,j}, \quad \forall f; O_{j,i} \in \Omega_1 \quad (13)$$

式(10)~(13)表明属于稳定调度工序集 Ω_1 的工序维持其在初始调度方案的工厂和机器分配结果,开工、完工时间与原先一致。

模型 2 $O_{j,i} \in \Omega_2, O_{j,i} \in \Omega_3$

$$S_{j,i}^* = S_{j,i} + T_C - T_S, \quad \forall O_{j,i} \in \Omega_2 \quad (14)$$

$$C_{j,i}^* = C_{j,i} + T_C - T_S, \quad \forall O_{j,i} \in \Omega_2 \quad (15)$$

$$Y_{f,j,i,g}^* = Y_{f,j,i,g}, \quad \forall f,g; O_{j,i} \in \Omega_2 \quad (16)$$

$$X_{f,j}^* = X_{f,j}, \quad \forall f; O_{j,i} \in \Omega_2 \quad (17)$$

式(14)~(17)表明属于右移重调度工序集 Ω_2 的工序仍不改变其在初始调度方案的工厂和机器分配结果,但其开工、完工时间需要右移至故障机器 g 的修复时间之后。

$$S_{j,i}^* \geq T_S, \quad \forall O_{j,i} \in \Omega_3 \quad (18)$$

$$C_{j,i}^* = S_{j,i}^* + \min_{1 \leq g' \leq M_{f,i}} \{Y_{f,j,i,g'}^* P_{f,j,i,g'}\}, \quad \forall f; O_{j,i} \in \Omega_3 \quad (19)$$

$$X_{f,j}^* = X_{f,j}, \quad \forall f; O_{j,i} \in \Omega_3 \quad (20)$$

式(18)表明属于并行机重调度工序集 Ω_3 的开始时间限制;式(19)表明工序集 Ω_3 内工序的开、完工时间关系以及并行机存在时的选择规则;式(20)表明工序集 Ω_3 内工序维持其在初始调度方案的工厂分配结果。

模型 3 $O_{j,i} \in \Omega_4$

$$S_{j,i}^* \geq T_S, \quad \forall O_{j,i} \in \Omega_4 \quad (21)$$

$$C_{j,i}^* = S_{j,i}^* + \sum_{f=1}^F \sum_{g=1}^{M_{f,i}} Y_{f,j,i,g}^* P_{f,j,i,g}, \quad \forall O_{j,i} \in \Omega_4 \quad (22)$$

$$X_{f,j}^* = \sum_{g=1}^{M_{f,i}} Y_{f,j,i,g}^*, \quad \forall f; O_{j,i} \in \Omega_4 \quad (23)$$

式(21)表明完全重调度工序集 Ω_4 中工序的开始时间限制;式(22)表明工序集 Ω_4 中工序的开始时间与完工时间的关系;式(23)表明工序集

Ω_4 中工序的机器约束。

$$C_{j,i}^* \leq S_{j,i+1}^*, \quad \forall j,i \quad (24)$$

$$S_{j,i}^*, C_{j,i}^* \geq 0, \quad \forall j,i \quad (25)$$

$$X_{f,j}^*, Y_{f,j,i,g}^* \in \{0,1\}, \quad \forall f,j,i,g \quad (26)$$

式(24)表明重调度方案中同一工件的相邻两道工序之间的优先级关系;式(25)、式(26)表明决策变量的取值范围。

2 混合人工蜂群算法

混合人工蜂群算法是一种元启发式算法,模仿蜂群的觅食行为,算法由雇佣蜂、跟随蜂、侦查蜂 3 个搜索过程组成,对各部分进行改进操作,具体步骤如下。

2.1 编码与解码

2.1.1 编码

为了说明 DHFFRP 涉及的工件、工厂、机器信息,设计 3 层整数编码方式,以 $V_Q = [jV_Q, fV_Q, gV_Q]$ 表示一个调度,其中,各层内元素一一对应,每层的染色体长度 n 均等于所有工件的工序之和,即 $n = JI$ 。

以 $F=2, J=I=3, M_{1,1}=M_{1,3}=M_{2,2}=3$ 和 $M_{1,2}=M_{2,1}=M_{2,3}=2$ 的 DHFFRP 为例,一个调度解的编码表述如图 1 所示。

	$O_{2,1}$	$O_{3,1}$	$O_{1,1}$	$O_{3,2}$	$O_{2,2}$	$O_{1,2}$	$O_{1,3}$	$O_{3,3}$	$O_{2,3}$
jV_Q	2	3	1	3	2	1	1	3	2
fV_Q	2	1	1	1	2	1	1	1	2
gV_Q	1	1	2	2	3	1	2	3	2

图 2 3 层编码表述

1) 工件工序层 jV_Q 。每个元素表示一个工件号,重复出现的工件号依次对应着该工件的第 i 道工序,图 2 中第 2、4、8 个元素分别对应工件 3 的第 1 道工序 $O_{3,1}$ 、第 2 道工序 $O_{3,2}$ 、第 3 道工序 $O_{3,3}$ 。

2) 工厂分配层 fV_Q 。每个元素表示层 jV_Q 内各工件的工厂分配信息,如工件 3 的 3 道工序 $O_{3,1}$ 、 $O_{3,2}$ 、 $O_{3,3}$ 所分配的工厂为工厂 1。

3) 机器分配层 gV_Q 。每个元素表示工件在所分配工厂内各道工序选择的机器号,如工序 $O_{3,1}$ 所对应的机器号为 1,表明该工序在工厂 1 内分配到第 1 台机器加工。

2.1.2 解码

为避免机器空闲的浪费,采用插入式贪婪解码

方式使工序紧凑排列,从而提高机器利用率、减少资源浪费。解码时,工件的各道工序按层 jV_Q 从左至右的顺序依次安排至对应工厂所分配的机器上,从而得到各工件的工厂和机器分配结果以及在同一台机器的工件加工顺序,进而基于机器空闲原则得到各工件的开工、完工时间。具体步骤如下:

1) 基于层 jV_Q 逐一读取每道工序 $O_{j,i}$, 从层 fV_Q 、 gV_Q 得到该工序分配的工厂 f 、加工机器 $h_{f,j,i}$ 和 $O_{j,i}$ 的紧前工序 $O_{j,i-1}$ 的加工机器 $h_{f,j,i-1}$ 。

2) 对于各台机器上待加工的首个工件,若其为第 1 道工序 $O_{j,1}$, 则令其开始加工时间 $S_{j,1}=0$; 否则该工序从其紧前工序 $O_{j,i-1}$ 的完工时间 $C_{j,i-1}$ 开始加工, 即 $S_{j,i}=C_{j,i-1}, i=2, \dots, I$ 。以此更新所使用的各台机器的空闲时间。

3) 对于其他工序, 记录其加工机器 $h_{f,j,i}$ 上已完工的最后一道工序之前的空闲时间段 $[T_s^h, T_c^h]$ 。应用插入操作, 在满足式(4)或式(23)以及 $T_c^h - T_s^h \geq P_{f,j,i,g}$ 的条件下, 将工序 $O_{j,i}$ 安排到最早可利用的机器空闲时间段内。若在上述机器空闲时间段 $[T_s^h, T_c^h]$ 内未找到符合要求的加工时间段, 则将该工序插入到机器 $h_{f,j,i}$ 上已完工的最后一道工序之后。

2.2 初始解集的生成

初始解集的质量对算法性能有较大影响。考虑到由混沌原理产生的随机数具有高度的随机性和均匀性, 有助于减少陷入局部最优的可能性以及个体在解空间内的均匀分布; 而且 Tent 混沌映射在随机性和均匀性表现更佳^[17], 因此, 引入

Tent 混沌映射生成初始解集。首先, 利用 Tent 混沌映射函数生成 p 个包含 $3 \times J \times I$ 个基因的 3 层向量, 构成混沌种群; 然后, 将混沌种群中的混沌值 $x_{e,q}$ 映射到解空间以得到初始解集。Tent 混沌映射函数为:

$$x_{e,q+1} = \begin{cases} x_{e,q}, & 0 \leq x_{e,q} \leq \gamma; \\ \frac{1-x_{e,q}}{1-\gamma}, & \gamma < x_{e,q} \leq 1 \end{cases} \quad (27)$$

其中: $x_{e,q}$ 为染色体 e 的第 q 个基因对应的混沌数, $x_{e,q} \in [0, 1]$; γ 为映射参数。经多次实验, 设置 $\gamma=0.499$ 。

2.3 雇佣蜂

雇佣蜂对解空间进行搜索以寻找更优质的蜜源(染色体), 引入改进遗传算法, 通过选择、交叉、变异算子生成新解, 以提高搜索效率。

2.3.1 三元锦标赛选择

采用三元锦标赛选择机制筛选蜜源。从当前解集内随机取出 3 个蜜源 $\{V_{Q1}, V_{Q2}, V_{Q3}\}$, 计算各自适应度值, 从中选择适应度值最大的最佳染色体用于后续的进化操作, 采用放回抽样重复操作得到 p 个染色体。适应度值计算公式为:

$$f_{it} = 1/C_{\max} \quad (28)$$

2.3.2 交叉算子

由于 3 层编码方式中各层间有关联, 为避免交叉产生非法解, 采用基于工件层和机器层的分层交叉方式。

在 jV_Q 层采用 POX 交叉方式, 如图 3 所示。

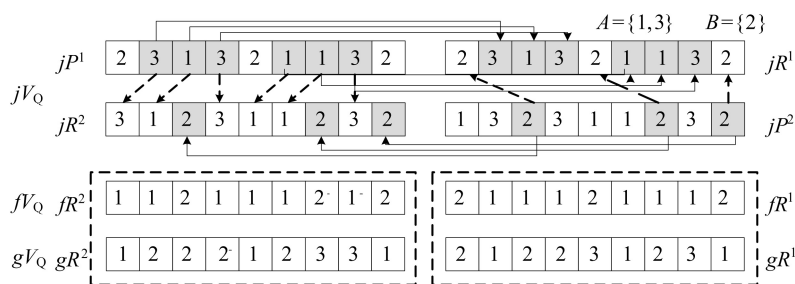


图 3 POX 交叉

首先, 从前述得到的解集内随机选择 2 个染色体作为父代 jP^1 、 jP^2 ; 然后, 将工件集 $\{1, 2, \dots, J\}$ 随机划分为 2 个非空子集 A 、 B ; 最后, 保留父代 jP^1 中属于工件集 A 的工序原有位置及其基因值, 复制到 jR^1 中, 将父代 jP^1 中属于工件集 B 的工序依次填补到 jR^1 中的剩余位置, 从而得到子代 jR^1 ; 同理, 生成 jR^2 。

在上述过程中, jV_Q 层内的各基因变换时, 对应的 fV_Q 、 gV_Q 层元素也随之调整, 即不改变各工序所分配的工厂和机器。

在 gV_Q 层采用 UX 均匀交叉方式, 如图 4 所示。首先, 随机产生长度为 n 的由 0、1 组成的向量; 其次, 记录该向量内元素为 0 的位置, 将父代 gR^1 中该位置的基因值赋于子代 $gR^{1'}$, 父代 gR^2

中该位置的基因值赋予子代 $gR^{2'}$; 然后, 将向量内元素为 1 的位置所对应的父代 gR^2 中基因值赋值于子代 $gR^{1'}$, 父代 gR^1 中的赋值于子代 $gR^{2'}$; 最后, 为防止非法解的产生, 对子代 $gR^{1'}$ 、

$gR^{2'}$ 进行修正, 若机器号超出该工序的可选并行机数, 则从其可选并行机中随机选择一个机器号进行替换。需要注意在交叉过程中 jV_Q 、 fV_Q 层保持不变。

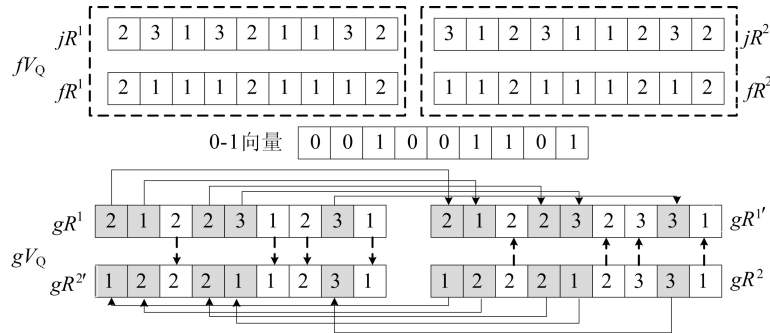
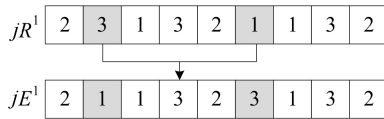


图 4 UX 均匀交叉

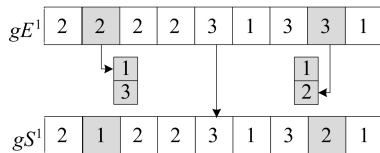
2.3.3 变异算子

对工件的工序层、机器分配层分别应用互换变异、随机变异方式, 保持工厂分配层不变, 从而产生新蜜源。

对 jV_Q 层采用互换变异方式, 如图 5 所示。随机选择 2 个基因位 $\alpha, \beta (\alpha, \beta \in [1, n])$ 对应的工序进行互换, fV_Q 、 gV_Q 层对应的元素也随之调整。

图 5 jV_Q 层互换变异

对 gV_Q 层采用随机变异方式, 如图 6 所示。随机生成 $\tau (\tau \leq n)$ 个基因位, 对其中每个基因位 α , 从对应工序的其他并行机中随机选择一台替换原有机器号。

图 6 gV_Q 层随机变异

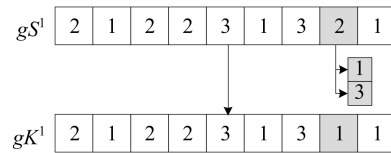
2.4 跟随蜂

2.4.1 轮盘赌选择

跟随蜂通过轮盘赌选择机制确定要保留的蜜源并继续探索。被保留的概率与适应度值成正比, 适应度值越高的蜜源被选中的可能性越大。

2.4.2 局部搜索算子

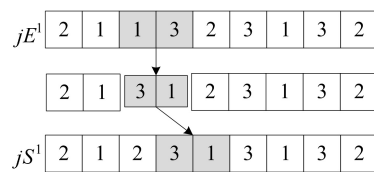
保留的蜜源对机器分配层执行基于最小完工时间规则的局域搜索。对 gV_Q 层随机生成一个基因位 $\alpha (\alpha \leq n)$, 将对应机器号替换成所有可能的并行机, 比较每种机器分配方案的完工时间, 选择具有最小完工时间的机器分配方案作为更新后的机器分配层, 如图 7 所示。

图 7 gV_Q 层局部搜索

2.5 侦查蜂

为充分利用现有蜜源信息, 对连续 l 次迭代未更新的蜜源, 引入 4 种变邻域搜索算子、模拟退火操作优化蜜源。

1) 基于工件的逆序插入。随机选取蜜源内 2 个连续基因位 $\alpha, \alpha+1 (\alpha < n)$, 将对应的连续基因段进行逆序排列, 然后把该基因段随机插入至蜜源的其他位置, 如图 8 所示。

图 8 jV_Q 层基于工件的逆序插入

2) 基于工件的工序段插入。首先随机选择 $\tau (\tau < n)$ 个基因, 将其从工序层移除; 然后将这些

被删除的基因按原顺序整合;最后将整合后的连续基因段随机插入到工序层内任一位置,如图 9 所示。

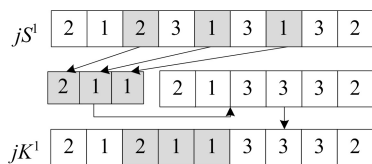


图 9 jV_Q 层基于工件的工序段插入

3) 基于机器的变异。从 jV_Q 层随机选择一道工序,比较对应工件在所分配机器的其他并行机的加工时间,将具有最短加工时间的机器号取代该工序原有分配机器,如图 10 所示。

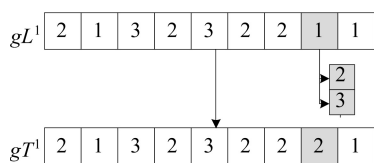


图 10 gV_Q 层基于机器的变异

4) 基于工件-工厂的交换。将所有工厂按完工时间的增序排列,分别从完工时间最长和最短的工厂 f_1 、 f_2 中随机选择工件 N_{f_1} 、 N_{f_2} , 2 个工件的交换如图 11 所示。

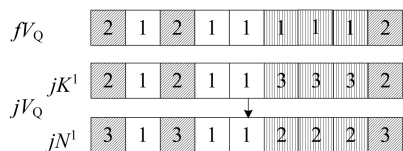


图 11 基于工件-工厂的交换

对当前解 $C_{\max}^{\text{old}}$ 依次执行上述 4 种变邻域搜索算子,产生新解 $C_{\max}^{\text{new}}$ 。

若 $C_{\max}^{\text{new}} < C_{\max}^{\text{old}}$, 则接受新解;否则,借鉴模拟退火中以一定概率 p_r 接受最差解的思想^[18],设计模拟退火操作,根据式(29)决定是否接受此解,即

$$p_r = \exp\left(-\frac{C_{\max}^{\text{new}} - C_{\max}^{\text{old}}}{T_r}\right) \quad (29)$$

其中: p_r 为第 r 次迭代的接受较差解概率; T_r 为第 r 次迭代的温度, $T_{r+1} = \lambda T_r$, λ 为冷却系数。从 $(0, 1)$ 随机产生一个数 r' , 若 $p_r > r'$, 则接受新解。

2.6 HABCA 流程

基于以上描述得到 HABCA 的总体流程,如

图 12 所示。

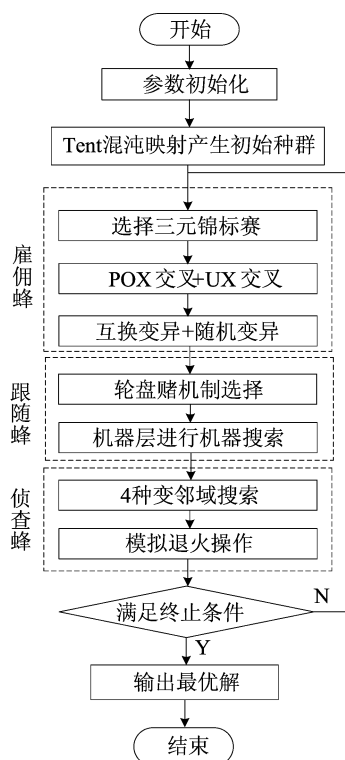


图 12 HABCA 流程

3 实验仿真与分析

为了分析所提算法 HABCA 求解 DHFFRP 的性能,对比 HABCA 在初始调度与重调度方案之间的效果差异,将其与经典人工蜂群算法(ABCA)、改进布谷鸟算法(ICA)^[6]、改进鲸鱼算法(SWOA)^[19]进行仿真测试和比较。采用 MATLAB R2021a 作为编程工具,并在 CPU 为 AMD R5-6600H、内存为 16 GiB、主频为 3.3 GHz 的计算机上实现。

3.1 算例生成

由于目前缺乏针对包含机器故障的 DHFFRP 的标准测试算例,拓展文献[20-21]得到本文所使用的算例规模,即 $J \in \{10, 20, 40, 60, 80\}$ 、 $F \in \{2, 3, 4\}$ 、 $I \in \{2, 3, 4, 5, 6\}$ 、 $M_{f,i} \in \{1, 2, 3\}$ 。由此产生 34 种不同的算例组合,包括 19 种小规模算例、15 种中大规模算例;每种算例规模随机运行 10 次,因此共产生和测试 340 组算例。考虑到 $P_{f,j,i,g}$ 受机器磨损和工件特性的影响,参考文献[22-23],令其满足区间 $[10, 20]$ 的均匀分布。简单起见,假定 $T_s = 20$ s 时工厂 1 内的第 2 台机器 b_m 出现故障,修复故障耗时 20 s^[11,24-25],即 $T_c = 40$ s,但所提算法也能求解不同故障机器和

故障时间。为了确保各算法能够在合理的时间内完成,同时为不同规模算例提供足够的运行时间,将 HABCA 运行 100 代,以最大 CPU 时间不超过 1 600 s 所需的 CPU 时间作为其他 3 种算法的停止时间。

3.2 参数设置

为了公平比较所有算法,采用正交试验法对各算法的参数进行优化设置以确保比较的精确度和有效性。由该方法得到的最佳参数见表 1 所列。

表 1 各算法的最佳参数

算法	参数
HABCA	$p=100, p_c=0.8, p_m=0.3, l=15, T_0=100\text{ }^{\circ}\text{C}, \lambda=0.96$
ABCA	$p=100, l=15$
ICA	$p=80, p_c=0.8, p_m=0.3$, 原巢主发现概率为 0.3
SWOA	$p=50, p_c=0.5$

表 1 中: p_c 、 p_m 分别为交叉、变异概率; T_0 为初始温度。

3.3 基于 HABCA 的初始调度与重调度方案

以 $F=2, J=20, I=4, M_{1,1}=M_{1,3}=M_{2,4}=1$ 和 $M_{1,2}=M_{1,4}=M_{2,1}=M_{2,2}=M_{2,3}=2$ 算例为例,运行 HABCA 得到初始调度和重调度方案,初始调度甘特图如图 13 所示, $C_{\max}=179\text{ s}$, 图 13 中, $j-i$ 表示工件 j 的第 i 道工序。

从图 13 可以看出, $\Omega_1 = \{O_{1,1}, O_{3,1}, O_{6,1}, O_{8,1}, O_{8,2}, O_{16,1}, O_{16,2}, O_{17,1}\}$, $\Omega_4 = \{O_{1,3}, O_{1,4}, O_{2,1}, \dots, O_{2,4}, O_{3,2}, \dots, O_{3,4}, O_{4,1}, \dots, O_{4,4}, O_{5,1}, \dots, O_{5,4}, O_{6,3}, O_{6,4}, O_{7,1}, \dots, O_{7,4}, O_{8,3}, O_{8,4}, O_{9,1}, \dots, O_{9,4}, O_{10,1}, \dots, O_{10,4}, O_{11,1}, \dots, O_{11,4}, O_{12,1}, \dots, O_{12,4}, O_{13,1}, \dots, O_{13,4}, O_{14,1}, \dots, O_{14,4}, O_{15,1}, \dots, O_{15,4}, O_{16,3}, O_{16,4}, O_{17,2}, \dots, O_{17,4}, O_{18,1}, \dots, O_{18,4}, O_{19,1}, \dots, O_{19,4}, O_{20,1}, \dots, O_{20,4}\}$ 。

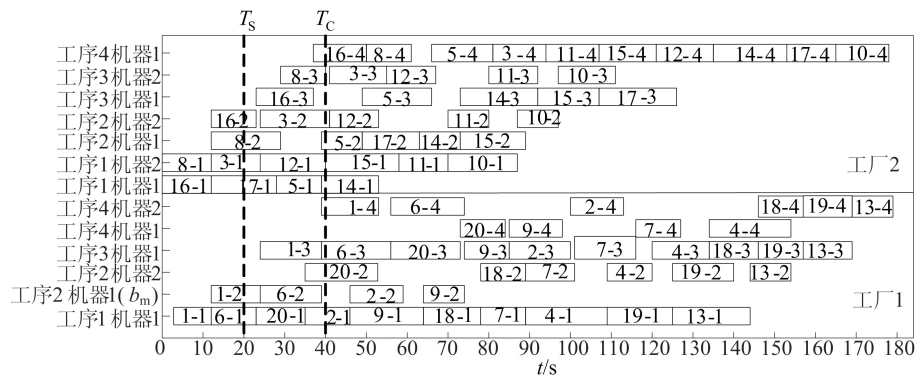


图 13 初始调度甘特图

当故障发生时,故障机器 b_m 有一台可用并行机,因此对策略 2 内工序 $\{O_{1,2}, O_{6,2}\}$ 进行判断,比较其各自在当前机器上的完工时间与在并行机的完工时间。

对于 $O_{1,2}$, $T_C + P_{1,1,2,1} = 40 + 12 = 52$, $T_S + P_{1,1,2,2} = 20 + 16 = 36$; 对于 $O_{6,2}$, $T_C + P_{1,6,2,1} =$

$40 + 14 = 54$, $T_S + P_{1,6,2,2} = 20 + 18 = 38$, 因此这 2 个工序均选择移至并行机上继续加工,即 $\Omega_2 = \emptyset$, $\Omega_3 = \{O_{1,2}, O_{6,2}\}$ 。对 Ω_3 、 Ω_4 进行重调度,重调度甘特图如图 14 所示, $C_{\max}=183\text{ s}$ 。

从图 14 可以看出, Ω_4 内多数分配与初始调度方案相比也发生变化。

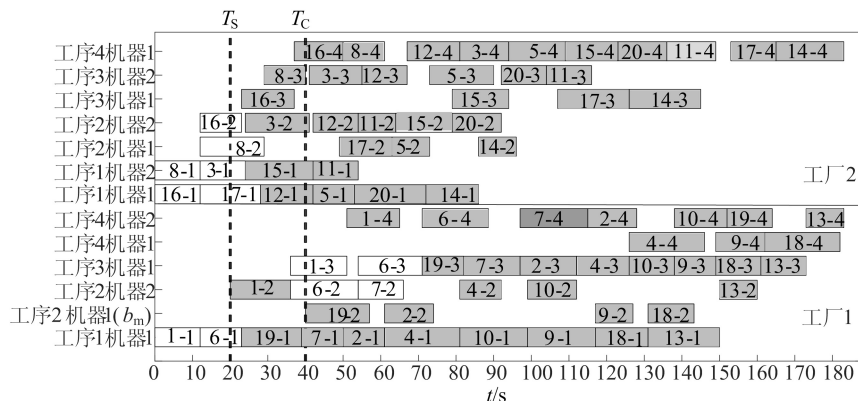


图 14 重调度甘特图

3.4 改进策略有效性分析

为验证 Tent 混沌映射初始化方法和变邻域搜索对算法性能的提升,本节从前述不同规模中各选择 2 组规模,每组随机运行 10 次得到平均目标值 $\bar{C}_{\max}$,将 HABCA 算法及其变体算法与其他算法进行比较。

3.4.1 Tent 映射初始化方法有效性分析

为验证本文所引入的 Tent 混沌映射初始化方法的有效性,比较 HABCA 算法与 ABCA、ICA、SWOA 算法在初始解质量上的差异,初始解目标值结果见表 2 所列。从表 2 可以看出,本文所采用的 Tent 映射初始解优于其他对比算法。通过将 Tent 混沌映射的全局遍历性、随机性、内有序性融入传统优化算法,可以有效规避局部最优的缺陷,以实现全局最优。

表 2 初始解目标值

$F \times J \times I$	算法			
	HABCA	ABCA	ICA	SWOA
$2 \times 20 \times 3$	152.3	184.2	202.1	179.3
$3 \times 40 \times 4$	279.8	319.9	340.2	324.7
$3 \times 60 \times 5$	409.2	519.4	581.8	476.8
$4 \times 80 \times 6$	489.1	568.6	696.4	552.3

3.4.2 变邻域搜索策略有效性分析

为验证本文所引入的 4 种变邻域搜索策略的

有效性,在 HABCA 算法的基础上设计一种变体算法即不包含变邻域搜索的 HABCA 算法 HABCA_nV。将 HABCA 与 HABCA_nV、ABCA、ICA、SWOA 算法进行比较,结果见表 3 所列。

从表 3 可以看出,HABCA 算法获得了最佳目标值,说明本文所引入的 4 种变邻域搜索策略拓宽了算法的搜索空间,实现更深层次的搜索,能够较好地跳出局部最优解,提高整体搜索效率和性能。

表 3 不同搜索策略改进效果对比

$F \times J \times I$	算法				
	HABCA	ABCA	ICA	SWOA	HABCA_nV
$2 \times 20 \times 3$	138.5	152.7	172.5	148.9	141.1
$3 \times 40 \times 4$	238.9	267.3	306.8	273.5	242.7
$3 \times 60 \times 5$	381.6	398.1	563.7	407.2	385.4
$4 \times 80 \times 6$	443.9	473.8	602.3	496.2	448.3

3.5 与其他算法的对比分析

采用相对百分比偏差 R_{PD} 衡量解的质量,即

$$R_{PD} = \frac{\bar{C}_{\max}^{\omega} - \bar{C}_{\max}^{HABCA}}{\bar{C}_{\max}^{HABCA}} \times 100\% \quad (30)$$

其中: $\bar{C}_{\max}^{\omega}$ 为当前算法 ω ($\omega \in \{ABCA, ICA, SWOA\}$) 求解一种算例规模得到的平均目标值; $\bar{C}_{\max}^{HABCA}$ 为由 HABCA 获得的平均目标值。

不同规模算例测试结果见表 4、表 5 所列。

表 4 小规模算例测试结果

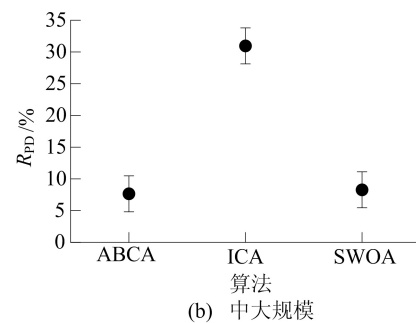
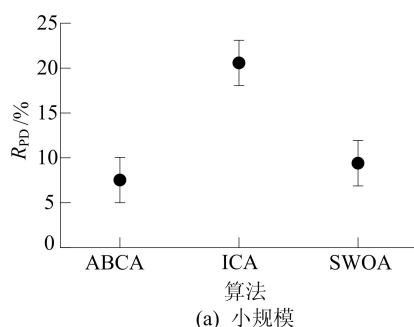
$F \times J \times I$	HABCA	ABCA		ICA		SWOA		运行时间/s
		$\bar{C}_{\max}$	$R_{PD}/\%$	$\bar{C}_{\max}$	$R_{PD}/\%$	$\bar{C}_{\max}$	$R_{PD}/\%$	
$2 \times 10 \times 2$	61.8	63.3	2.43	72.5	17.31	69.2	11.97	45.3
$2 \times 10 \times 3$	78.6	77.5	-1.40	98.8	25.70	89.4	13.74	73.2
$2 \times 10 \times 4$	114.5	115.8	1.14	132.2	15.46	116.2	1.48	105.8
$2 \times 20 \times 2$	106.7	109.5	2.62	128.1	20.06	118.3	10.87	113.9
$2 \times 20 \times 3$	136.1	145.2	6.69	168.7	23.95	153.1	12.49	173.2
$2 \times 20 \times 4$	181.8	189.2	4.07	212.8	17.05	212.1	16.67	192.4
$2 \times 20 \times 5$	202.1	209.9	3.86	241.7	19.59	217.8	7.77	236.8
$2 \times 40 \times 2$	204.4	223.7	9.44	246.5	20.60	224.5	9.83	221.5
$2 \times 40 \times 3$	248.5	264.4	6.40	306.8	23.46	258.6	4.06	257.9
$2 \times 40 \times 4$	330.3	362.3	9.69	381.3	15.44	352.3	6.66	412.7
$2 \times 40 \times 5$	362.9	395.1	8.87	432.2	19.10	385.6	6.26	458.7
$3 \times 20 \times 2$	81.3	93.3	14.76	103.5	27.31	89.2	9.72	117.4
$3 \times 20 \times 3$	109.4	123.2	12.61	137.6	25.78	119.3	9.05	154.6
$3 \times 20 \times 4$	136.7	154.4	12.95	164.2	20.12	148.6	8.71	254.9
$3 \times 20 \times 5$	150.3	178.6	18.83	187.6	24.82	162.8	8.32	281.3
$3 \times 40 \times 3$	193.8	209.6	8.15	223.5	15.33	217.5	12.23	341.3
$3 \times 40 \times 4$	248.3	256.4	3.26	288.8	16.31	268.5	8.14	450.7
$3 \times 40 \times 5$	269.2	301.7	12.07	332.5	23.51	295.4	9.73	672.8
$3 \times 40 \times 6$	310.4	330.9	6.60	372.9	20.14	344.3	10.92	734.6
平均值	185.6	200.2	7.53	222.7	20.58	202.2	9.40	291.9

表 5 中大规模算例测试结果

$F \times J \times I$	HABCA	ABCA		ICA		SWOA		运行时间/s
		$\bar{C}_{\max}$	$R_{PD}/\%$	$\bar{C}_{\max}$	$R_{PD}/\%$	$\bar{C}_{\max}$	$R_{PD}/\%$	
3×60×4	338.7	368.7	8.86	404.4	19.40	360.8	6.52	756.9
3×60×5	375.8	404.5	7.64	471.8	25.55	398.1	5.93	943.1
3×60×6	423.4	448.1	5.83	579.6	36.89	456.4	7.79	1 178.6
3×80×4	443.6	473.3	6.70	569.7	28.43	463.5	4.49	1 267.9
3×80×5	487.2	521.2	6.98	672.5	38.03	541.6	11.17	1 578.2
3×80×6	542.7	568.3	4.72	755.3	39.17	589.3	8.59	1 600.0
4×40×4	183.3	209.5	14.29	244.8	33.52	207.4	13.15	437.6
4×40×5	224.5	239.8	6.82	256.3	14.16	241.2	7.44	613.5
4×40×6	242.5	274.3	13.11	332.5	37.11	269.4	11.09	615.7
4×60×4	281.3	298.7	6.19	349.6	24.28	301.2	7.07	657.4
4×60×5	310.3	338.7	9.15	393.2	26.72	331.7	6.90	816.4
4×60×6	319.1	334.3	4.76	457.2	43.28	365.3	14.48	1 053.5
4×80×4	359.8	387.6	7.73	458.3	27.38	367.2	2.06	1 167.9
4×80×5	393.4	417.3	6.08	516.8	31.37	412.6	4.88	1 600.0
4×80×6	437.1	462.8	5.88	607.6	39.01	492.4	12.65	1 600.0
平均值	357.5	383.1	7.65	471.3	30.95	386.5	8.28	1 059.1

由表 4、表 5 可知,在求解小规模算例时,在平均运行时间为 291.9 s 内,HABCA 在 19 次测试中有 18 次得到了最佳平均目标值。就平均 R_{PD} 而言,HABCA 相较 ABCA、ICA、SWOA 分别提升了 7.53%、20.58%、9.40%,虽然个别算例(如 $2 \times 10 \times 3$)HABCA 的目标值略差于 ABCA,但随着算例规模的增大,HABCA 一致优于其他 3 种算法。对于中大规模算例,平均运行时间为 1 059.1 s。HABCA 在 15 次测试中全部得到了最佳平均目标值,相较于 ABCA、ICA、SWOA 分别提升了 7.65%、30.95%、8.28%。因此,对于所有规模算例,在相同的运行时间下,HABCA 在解的质量方面的表现优于其他 3 种算法。

为了验证测试结果的可靠性和统计学上的显著性,在 95% 的置信水平下进行统计检验,对 ABCA、ICA、SWOA 3 种算法在各算例测试下进行 R_{PD} 比较分析,从而确保结论具有科学依据。3 种算法的 R_{PD} 区间如图 15 所示。

图 15 3 种算法的 R_{PD} 区间

从图 15 可以看出:这些算法在性能上有显著差异,表明算法性能差异是真实存在的,而非由随机误差或样本偏差导致,因此测试结果可靠;HABCA 算法在各个算例规模中相较于其他 3 种算法有明显改进。

4 结 论

本文研究了 DHFFRP,聚焦于机器故障对生产效率的影响,以最小化最大完工时间作为优化目标,建立整数规划模型,提出一种 HABCA 得到近优解。该算法采用 3 层编码与插入式贪婪解码,引入 Tent 混沌映射改善初始解集的质量。设计改进 GA 算法、变邻域搜索算子、模拟退火操作分别用于雇佣蜂、跟随蜂、侦查蜂的搜索。通过在不同规模算例的仿真实验,测试结果表明 HABCA 在求解性能上具有明显优势。

由于 HABCA 求解考虑机器故障的 DHFFRP 表现出的良好性能,未来研究可探讨将该算

法应用于重调度领域中的其他突发问题,如紧急订单插入、订单取消等,还可进一步改进算法以提高求解能力。

[参 考 文 献]

- [1] 王凌,邓瑾,王圣尧. 分布式车间调度优化算法研究综述[J]. 控制与决策,2016,31(1):1-11.
- [2] 酆仕云,杨孟平,易文超,等. 基于混合离散差分进化算法的分布式异构混合流水车间调度[J]. 计算机集成制造系统,2025,31(7):2515-2527.
- [3] YING K C, LIN S W. Minimizing makespan for the distributed hybrid flowshop scheduling problem with multiprocessor tasks[J]. Expert Systems with Application, 2018, 92: 132-141.
- [4] LI Y L, LI F, PAN Q K, et al. An artificial bee colony algorithm for the distributed hybrid flowshop scheduling problem[J]. Procedia Manufacturing, 2019, 39: 1158-1166.
- [5] SHAO W S, SHAO Z S, PI D C. Modeling and multi-neighborhood iterated greedy algorithm for distributed hybrid flow shop scheduling problem[J]. Knowledge-Based Systems, 2020, 194(22): 105527.
- [6] 唐红涛,刘家毅. 改进的布谷鸟算法求解考虑运输时间的分布式柔性流水车间调度问题[J]. 运筹与管理,2021,30(11):76-83.
- [7] 雷德明,王甜. 基于改进蛙跳算法的分布式两阶段混合流水车间调度[J]. 控制与决策,2021,36(1):241-248.
- [8] 许天鹏,赵付青,张建林,等. 带分时电价约束的分布式柔性流水车间调度问题及其求解算法[J]. 计算机集成制造系统,2025,31(4):1299-1313.
- [9] 李颖俐,刘翔,邓旭东. 分布式异构混合流水车间生产与运输集成调度[J]. 计算机集成制造系统,2024,30(11):4088-4098.
- [10] 黄家文,周昭龙,陶涛,等. 基于机器故障下板式定制家具混合流水车间动态调度研究[J]. 林产工业,2023,60(11):66-71,77.
- [11] 李铁克,肖拥军,王柏林. 基于局部性修复的 HFS 机器故障重调度[J]. 管理工程学报,2010,24(3):45-49,32.
- [12] 苏建涛,董绍华,朱诗敏. 多目标混合流水车间机器故障重调度问题研究[J]. 机械工程学报,2024,60(4):438-448.
- [13] ZHANG B, PAN Q K, GAO L, et al. A multi-objective migrating birds optimization algorithm for the hybrid flowshop rescheduling problem[J]. Soft Computing, 2019, 23(17):8101-8129.
- [14] 张晓楠,姜帅,王陆宇,等. 考虑模糊质检时间的柔性作业车间动态调度问题[J]. 计算机应用研究,2024,41(8):2351-2359.
- [15] LI K X, DENG Q W, ZHANG L K, et al. An effective MCTS-based algorithm for minimizing makespan in dynamic flexible job shop scheduling problem[J]. Computers & Industrial Engineering, 2021, 155(1):107211.
- [16] BAYKASOGLU A, MADENOGLU F S, HAMZADAYI A. Greedy randomized adaptive search for dynamic flexible job-shop scheduling[J]. Journal of Manufacturing Systems, 2020, 56: 425-451.
- [17] 单梁,强浩,李军,等. 基于 Tent 映射的混沌优化算法[J]. 控制与决策,2005,20(2):179-182.
- [18] 阎哲,汪民乐,汪江鹏,等. 基于混合遗传算法的海军航空兵场站物资配送车辆调度智能优化[J]. 系统工程与电子技术,2023,45(12):3908-3914.
- [19] 张斯琪,倪静. 混合鲸鱼算法在柔性作业车间系统中的应用[J]. 系统科学学报,2020,28(1):131-136.
- [20] 尤一琛,王艳,纪志成. 基于博弈论的柔性作业车间动态调度研究[J]. 系统仿真学报,2021,33(11):2579-2588.
- [21] 张国辉,陆熙熙,胡一凡,等. 基于改进帝国竞争算法的柔性作业车间机器故障重调度[J]. 计算机应用,2021,41(8):2242-2248.
- [22] JIANG E D, WANG L, WANG J J, et al. Decomposition-based multi-objective optimization for energy-aware distributed hybrid flow shop scheduling with multiprocessor tasks[J]. Tsinghua Science and Technology, 2021, 26(5): 646-663.
- [23] WANG J J, WANG L. A cooperative memetic algorithm with learning-based agent for energy-aware distributed hybrid flow-shop scheduling[J]. IEEE Transactions on Evolutionary Computation, 2021, 26(3): 461-475.
- [24] 刘爱军,杨育,邢青松,等. 柔性作业车间多目标动态调度[J]. 计算机集成制造系统,2011,17(12):2629-2637.
- [25] 吉卫喜,蔡西勇,张朝阳,等. 异常事件驱动的离散制造车间重调度决策[J]. 系统仿真学报,2018,30(11):4043-4052.

(责任编辑 张 镅)