基于学习机制的分布式混合流水车间调度算法研究

陈学锋，左　阳
（河南工学院 电子信息工程学院，河南 新乡 453003）

摘要：在现代制造业中，DHFSP 对提高生产效率和降低成本具有重要意义。提出了一种基于迭代贪婪的自适应学习算法，旨在有效地最小化最大完工时间。首先，通过一种初始化策略构建初始解；其次，引入一个基于历史反馈的动态学习机制，该机制能够根据累积的优化经验，智能选择扰动策略，以逐步提升解的优化质量；最后，通过轮盘赌选择策略，进一步增强了算法对全局最优解的探索能力。实验结果显示，所提出的算法在求解分布式混合流水车间调度问题时，相较于现有的对比算法，展现出了显著的性能优势。

关键词：DHFSP；自适应；轮盘赌；学习机制
中图分类号：TP301　　**文献标志码**：A　　**文章编号**：2096-7772-(2024)04-0025-07

0　引言

在大规模半导体制造企业中，半导体的制造包含了晶圆加工、氧化、刻蚀、光刻、薄膜沉淀、互连、测试和封装等过程；每一个加工过程包含多个工艺步骤，比如光刻过程中的涂覆光刻胶、曝光和显影等；每一个工艺步骤还涉及使用多台加工设备，这种需要跨越多个生产车间及地理位置分散的生产过程的调度问题被认为是典型的分布式混合流水车间调度问题（Distributed Hybrid Flow Shop Scheduling Problem，DHFSP）。通过优化求解 DHFSP，可以有效地优化生产调度的过程，进而提升生产效率。

DHFSP 主要表现在如何高效地安排和协调跨不同地理位置的生产车间或工厂中的多个工艺流程，以实现生产资源的最优分配和生产效率的最大化。在这种分布式生产模式下，求解 DHFSP 的关键是通过高效的算法将半导体工件分配至最合适的生产工厂。此外，还需对工件在各工厂的加工顺序进行优化，制定决策规则，选择每个生产阶段的加工机器。实施这些策略可以有效提高半导体生产的效率，降低生产成本[1,2]。

近年来，DHFSP 因其融合了分布式流水车间调度和并行机器调度的特点而受到广泛关注。针对不同情况，已有文献提出解决 DHFSP 的各种算法。例如，Cai 等[3]提出了一种动态洗牌蛙跳算法，以最小化具有多处理器任务的 DHFSP 中的完工时间。Shao 等[4]专注于分布式阻塞流水车间调度问题，并提出了一种混合增强离散果蝇优化算法，以在分布式环境中最小化完工时间。Zheng 等人[5]解决了多目标模糊分布式混合流水车间调度问题，并通过协同进化算法，在制造系统中引入了不确定性。Wang 等人[6]也处理了模糊分布式混合流水车间调度问题，并特别关注异构工厂和无关并行机器，引入了一种带有多个搜索策略协作的洗牌蛙跳算法。

在能源效率方面，Lu 等人[7]讨论了汽车行业中具有异构工厂的分布式流水车间的节能调度问题。Geng 等人[8]针对时段用电价格约束下的节能分布再进入混合流水车间调度问题提出了一种记忆算法。Wang 等人[9]为具有异构工厂的 DHFSP 提出了一种双群体协作记忆算法，以最小化完工时间。

尽管现有研究在求解 DHFSP 方面取得了显著进展，开发了多种算法和模型，但利用历史经验来提升算法性能的研究尚不多见。本文提出一种基于迭代贪婪的自适应学习算法（Adaptive Learning Algorithm Based on Iterative Greedy，AIG），该算法可通过累积

收稿日期：2024-04-24
第一作者简介：陈学锋（1980—　），男，河南新乡人，副教授，主要从事信息技术和智能制造研究。

历史经验，动态地选择扰动策略，以提高解决 DHFSP 时的搜索效率。这种方法的核心之处在于其对历史数据的深入挖掘和智能利用，有望为半导体等领域的生产调度提供更加高效的解决方案。

1 问题描述和模型

1.1 问题描述

DHFSP 涉及一个由个工件和个工厂构成的生产场景。在此场景中，每个工件需经历个加工阶段，且在每个阶段，存在台具有相同功能的加工机器。依据预定的分配规则，每个工件被指定至单一工厂进行加工。一旦工件分配完毕，各工厂将依据工件到达的顺序，在可用机器上安排其加工顺序。需特别指出的是，每个工件仅能被分配至一个工厂，且各工件的加工过程均保持独立。在任何特定时刻，每台机器仅能加工一个工件。在加工流程中，提供无限缓冲区，同时在计算工件总完工时间时，需将包括准备时间、转移时间和等待时间在内的所有时间均考虑在内。DHFSP 的核心目标是最小化所有工件中的最大完工时间，即寻求最优的加工调度策略，以确保尽可能早地完成最后一个工件的加工，从而提升整体生产效率和降低成本。

1.2 问题模型

1）符号说明

表 1 符号说明

符号	描述
n	工件的索引，$n \in N$
m	工厂的索引，$m \in M$
s	阶段的索引，$s \in S$
h	机器的索引，$h \in H$
$p_{n,s}$	工件n在第s阶段的加工时间
$C_{n,s}$	工件n在第s阶段的完工时间
L	极大的正数

2）目标函数

$$C_{max} = min\ (max\ C_{n,S}) \tag{1}$$

3）决策变量

$Y_{n,m}$：当工件n被分配到第m个工厂时为 1，否则为 0；

$X_{n,s,h}$：当工件n在第s个阶段的机器h上加工时为 1，否则为 0；

$Z_{n,n',s}$：在第s个阶段加工，工件n是工件n'的前一个工件时为 1，否则为 0。

4）约束条件

$$\sum_{m=1}^{M} Y_{n,m} = 1, \quad n \in N \tag{2}$$

$$\sum_{h=1,s=1}^{H,S} X_{n,h,s} = 1, \quad n \in N \tag{3}$$

$$C_{n,1} - p_{n,1} \geq 0, \quad n \in N \tag{4}$$

$$p_{n,s} \geq 0, \quad n \in N, h \in H \tag{5}$$

$$C_{n,s+1} - C_{n,s} \geq p_{n,s}, \quad n \in N,\ s \in S \tag{6}$$

$$Z_{n,n',s} + Z_{n',n,s} \leq 1, \quad n \neq n' \cap n, n' \in N,\ s \in S \tag{7}$$

$$C_{n',s} - C_{n,s} \geq p_{n,s} - L * \left(5 - Z_{n,n',s} - X_{n,h,s} - X_{n',h,s} - Y_{n,m} - Y_{n',m}\right), \quad n \neq n' \cap n, n' \in N,\ s \in S, h \in H,\ m \in M \tag{8}$$

约束（2）保证了工件分配的工厂唯一性，即每个工件只能被指派到特定的工厂进行加工。约束（3）确保了在任何给定时间点，每台机器仅能加工一个工件，从而维护了加工过程的独占性。约束（4）与约束（5）共同保证了工件的起始加工时间和加工持续时间的非负性，这是符合实际情况的基本要求。约束（6）考虑了工件在各个加工阶段之间的缓冲时间，确保了生产过程的连续性和平稳性。约束（7）引入了工件加工的先后顺序，保证了工件在加工流程中的逻辑顺序不被违反。最后，约束（8）确保了工件在机器上的加工顺序，维护了生产调度的有序性。这些约束条件的综合应用，确保了 DHFSP 的数学模型能够准确地反映实际生产环境的复杂性和多变性。

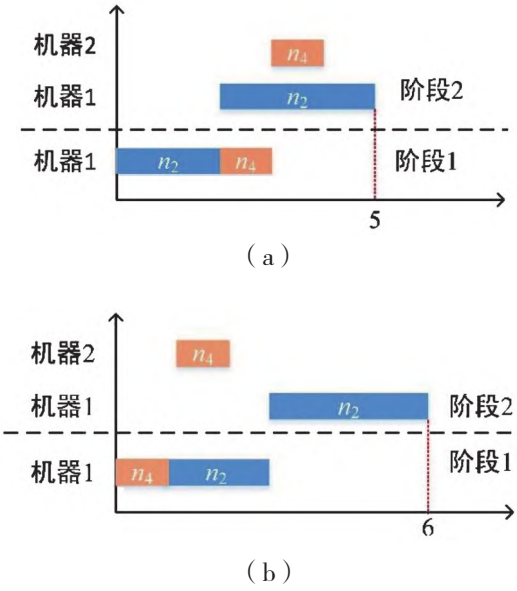
2 编解码

DHFSP 可细分为三个子问题：工件到工厂的分配问题、工厂内工件的排序问题以及工件在机器上的选择问题。本文采用基于序列的编码方法，在每个子问题的编码过程中引入特定的调度规则，以完成整个编码流程。

针对这三个子问题，本文分别采用了分配规则、排序规则和选择规则进行编码。具体而言，本文采用了最长加工时间规则作为分配规则，最短加工时间规则作为排序规则，最早离开和最早空闲规则作为选择规则。在三种规则的指导下，所有工件被安排到相应工厂的机器上进行加工。每个工厂包含一组工件序列π，所有工厂的工件序列共同构成完整的解$\boldsymbol{\varphi}$，其中$\boldsymbol{\varphi} = (\pi_1, \pi_2 ... \pi_m)$。

通过一个包含 4 个工件、2 个工厂和 2 个阶段的简单实例，本文将说明解码过程。4 个工件的加工时间分别为n_1=[1 2]，n_2=[2 3]，n_3=[3 1]，n_4=[1 2]。4 个工件的总加工时间分别为 3、5、4、3。按照最长

加工时间规则，4 个工件的加工时间排序为n_2、n_3、n_1、n_4。因此，n_2分配给工厂 1，n_3分配给工厂 2。因为n_1被分配给工厂 2 的总加工时间少于工厂 1，所以n_1被分配给工厂 2。同理，n_4被分配给工厂 1。此时π_1=[n_2 n_4], π_2=[n_3 n_1]。



（a）

（b）

图 1　排序规则和选择规则示意图

在工厂中，两个阶段的机器数量分别为 1 和 2。工件在工厂内的排序遵循最短作业时间规则，且排序过程同时包含了选择机器的步骤。如图 1 所示，在工厂 1 中，工件的排序过程考虑了两种可能的序列：[n_2 n_4]和[n_4 n_2]，其中[n_2 n_4]的完工时间为 5，而序列[n_4 n_2]的完工时间为 6，因此在工厂 1 中选择序列[n_2 n_4]。

采用类似的方法，可以得到完整的解$\boldsymbol{\varphi}$ =[π_1;π_2] =[n_2 n_4;n_1 n_3]。在两个工厂中，完工时间均为 5，因此四个工件的总完工时间为 5。这一结果体现了调度策略的有效性，即在满足所有工件加工需求的同时，实现了完工时间的最小化。

3　AIG 算法

3.1　算法流程

AIG 算法是一种自适应学习的优化方法，其由四个核心组成部分构成：初始化策略、破坏重建方法、扰动策略和学习机制。如图 2 所示，该算法首先通过初始化策略生成三个高质量的初始解，这些解为整个优化过程提供了一个卓越的起点。接着，算法采用破坏重建方法对初始解进行重组，以促进对解空间的全局探索。

在此基础上，AIG 算法利用学习机制，结合历史经验，通过轮盘赌策略动态地选择扰动策略，对当前解进行更深层次的调整和改进。学习机制是 AIG 算法的核心，它通过不断从历史数据中学习，优化扰动策略的选择，显著提升了算法的搜索效率和解的质量。这种自我优化的能力使得 AIG 算法在处理复杂优化问题时表现出色，尤其是在需要高效利用历史信息以指导搜索过程的情况下。

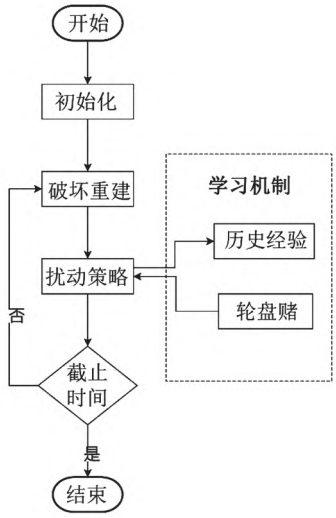


图 2　AIG 算法流程图

3.2　初始化策略

为了构建高质量的多样性初始解，本文采用了三种不同的初始工件序列生成策略。首先，通过随机排列方法生成一个工件序列，作为第一种初始序列；其次，本文运用 NEH（Nawaz–Enscore–Ham）启发式算法生成第二种工件序列；最后，将第二种序列进行反向排列，从而生成第三种工件序列。这种方法确保了初始解的多样性和覆盖范围。

在生成这三种工件序列之后，本文按照预定的分配规则和选择规则，将工件分配到各个工厂中的相应机器上。这一过程确保了工件在工厂内的合理分布，为后续的优化过程提供了一个均衡的起点。通过这种策略，本文旨在提高初始解的质量，从而为整个优化算法的执行打下坚实的基础。

3.3　破坏重建方法

破坏重建方法是一种基于迭代贪婪（Iterated Greedy，IG）的策略，用于探索解空间。这种策略从全局的角度出发，旨在提高解的质量。其具体步骤如下：首先，在工件序列中随机选择b个工件；接着，将这些工件从初始解中删除；最后，按照最短加工时间的规则，将删除的工件重新逐一插入初始解中相应的工厂。为了进一步提升解的质量，插入工厂之后，

每个工厂中的工件序列将按照NEH的方法进行重新排序。通过一系列实验测试，本文发现当b取值为3时，算法的性能达到最优。

3.4 扰动策略

AIG算法将扰动策略作为底层操作，其核心在于通过插入和交换操作来开发解空间。这种策略从局部视角出发，旨在提升解的质量。AIG算法中包含了三种扰动策略，涉及工厂间的操作和工厂内的操作，具体如下：

（1）工厂间交换：该策略允许任意交换两个工厂中的一个工件，通过这种操作，可以打破工厂间的限制，从而增加解的多样性。

（2）工厂内插入：此策略允许从一个工厂中抽取一个工件，并将其插入该工厂其他位置，这种操作有助于在工厂内重新分配工件，从而提高整体生产效率。

（3）工厂内交换：该策略允许在同一工厂内交换两个工件的位置，这种操作有助于优化工厂内的工件序列，从而提高局部解的质量。

通过这三种扰动策略，AIG算法能够在局部操作的基础上，逐步探索和优化解空间，从而为全局优化提供基础。这种方法不仅增强了算法的探索能力，也提高了最终解的质量。

3.5 学习机制

在AIG算法框架中，学习机制扮演着决策层面的核心角色，负责基于累积的历史数据来挑选合适的底层执行策略。该机制由两大部分构成：学习与反馈。学习部分涉及对历史扰动策略成效的系统性积累，旨在提炼出以往成功经验中的共性规律；而反馈环节则利用轮盘赌选择策略，依据各策略的累积成功率来指导下一次扰动策略的选取，从而确保算法在探索与开发之间的平衡。

在学习机制框架内，AIG构建了一个由三种解构成的存档集，该存档集充当智能体的角色。每项扰动策略均被视为潜在的行动方案。智能体的状态转换基于其解质量的改进情况，具体分为“提高”和“不变”两种状态，分别对应于奖励值1和0。AIG采用平均质量作为评价指标，以全面评估存档集的状态，并据此指导后续的策略选择。相应的评估公式表述如下：

$$state = (\sum_{n=1}^{3} newfit_n - oldfit_n)/3 \tag{9}$$

式中：$newfit_n$和$oldfit_n$是第n个工件序列的完工时间，$state$是存档集的状态。根据存档集的状态，奖励如下所示：

$$reward = \begin{cases} 1, state < 0 \\ 0, state \geq 0 \end{cases} \tag{10}$$

AIG采用矩阵形式对存档集的历史经验和相关概率进行系统化记录。该矩阵以存档集的状态作为行索引，每种可能的扰动策略作为列索引。矩阵中的每个单元格记录了对应扰动策略在历史执行中累积获得的奖励值，这些奖励值不仅反映了策略的有效性，也是学习过程中的关键数据。通过这种方法，AIG算法能够将历史经验转化为选择未来扰动策略的概率，从而实现基于经验的自适应决策。学习过程如以下公式：

$$L_{a_{t+1}} = L_{a_t} + reward_{a_t} \tag{11}$$

式中：$reward_{a_t}$指t时刻第a个扰动策略的奖励值，L_{a_t}指第a个扰动策略累积的奖励值。

如果在学习过程中，算法仅选择基于即时奖励最大化的扰动策略可能会导致算法忽略其他策略潜在的知识。换言之，这种做法会失去其他策略探索到最优解的可能性。为了克服这一局限性，AIG中引入了基于轮盘赌的选择机制。该机制通过赋予一定的概率，允许选择那些当前质量不是最优的解，以此保留并利用多样化的策略知识。轮盘赌策略的选择依据，可以通过以下数学表达式进行形式化描述：

$$P_{a_t} = L_{a_t} / \sum_{a=1}^{3} L_{a_t} \tag{12}$$

$$act_{t+1} = a_t,\ P_{a-1_t} \leq GP < P_{a_t} \tag{13}$$

式中：P_{a_t}指t时刻第a个扰动策略的概率值，GP指按照均匀分布产生的[0,1]之间的数；act指选择的扰动策略。式（12）计算t时刻每一个策略被选择的概率，式（13）保证按照高斯分布选择$t+1$时刻的动作。

4 实验与分析

4.1 实验设置

实验涵盖了策略有效性分析与对比实验两个部分。策略有效性分析旨在评估AIG算法中各策略对整体性能的具体贡献，而对比实验则旨在将AIG算法的性能与现有算法进行横向比较。实验的执行环境为Matlab R2023b，所有实验均在配置有AMD Ryzen 5800 CPU、16GB内存的计算机上进行，操作系统为Windows 10。为了确保结果的稳健性，每种算法均在相同的实验设置下重复执行了30次。

鉴于目前缺乏针对 DHFSP 的标准测试集，本文采用随机生成的方法来检验所需的测试集。具体而言，工件数量从集合 {40，60，80} 中逐一选取，工厂数量从集合 {3，4，5} 中随机选取，阶段数量从集合 {5，10，15} 中随机选取。此外，每个工件的加工时间在区间 [1,99] 内随机产生，而每个阶段中的并行机数量则在区间 [1,5] 中随机产生。对于每个确定的规模，本文生成两个独立实例，从而构建了包含 54 个实例的测试集。为了确保算法性能比较的公平性，所有实验均采用统一的截止时间是$N*M*S$毫秒。在评价算法性能时，本文采用平均相对百分比偏差（Average Relative Percentage Deviation，ARPD）作为衡量指标，该方法能够量化算法解与最优解之间的差异，$ARPD$的计算如公式（14）示：

$$ARPD=\frac{1}{Num}\times\sum_{c=1}^{Num}\frac{result_c-result_{best}}{result_{best}}\times 100\% \quad (14)$$

式中：Num指实例的个数，$result_c$和$result_{best}$分别是当前结果和最优结果。

4.2 策略有效性分析

本研究提出的 AIG 算法框架由三个关键策略组成：初始化策略、破坏重建方法和学习机制。为了评估各策略对算法性能的具体贡献，本文设计了一系列消融实验，通过逐一移除每个策略，生成三种 AIG 算法的变体。这些变体在所有的实例上进行了广泛的实验测试，并采用 Tukey 诚实显著性差异检验的置信区间，对实验结果进行统计分析。

三种变体分别是 AIG_1、AIG_2 和 AIG_3。AIG_1 变体通过随机生成工件序列的方式替换了原有的初始化策略，AIG_2 变体去除了破坏重建方法，而 AIG_3 变体则用随机选择扰动策略的方式替代了学习机制。如图 3 所示，与 AIG 算法相比，三种变体的 *ARPD* 值都大，从而证实了每项策略对于算法性能均有积极作用。特别是破坏重建方法对算法性能的贡献最为显著，其次是初始化策略，学习机制的贡献尽管相对较小，但也对算法的整体性能有着不可忽视的影响。

4.3 对比实验

为了全面评估 AIG 算法的性能，本文选取两种近年来针对 DHFSP 的算法：自调迭代贪婪算法（Self-Tuning Iterated Greedy，SIG）[10] 和多邻域迭代贪婪算法（Multi-Neighborhood Iterated Greedy，MIG）[4] 作为对比算法。所有算法针对一系列不同规模的测试实例进行测试，旨在通过严谨的比较分析，揭示 AIG 算法在解决 DHFSP 时的性能表现。

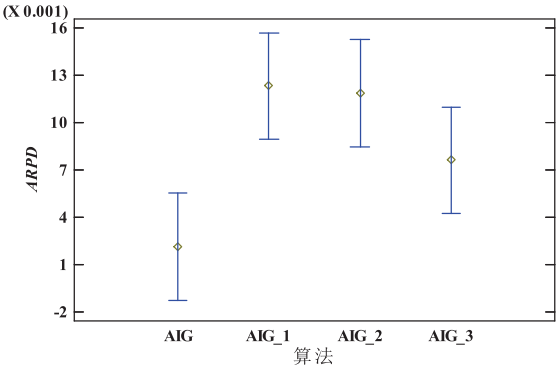


图 3 策略有效性

表 2 算法在所有实例上的*ARPD*值

作业 * 阶段 * 工厂	SIG	MIG	AIG
40*5*3	0.0785	0.0834	0.0054
40*5*4	0.1434	0.1317	0.0079
40*5*5	0.1459	0.1042	0.0000
40*10*3	0.1035	0.0260	0.0045
40*10*4	0.0810	0.0000	0.0000
40*10*5	0.0282	0.0000	0.0000
40*15*3	0.0484	0.0316	0.0032
40*15*4	0.0709	0.0187	0.0054
40*15*5	0.0827	0.0120	0.0039
60*5*3	0.1113	0.1430	0.0109
60*5*4	0.1484	0.1290	0.0129
60*5*5	0.1773	0.0823	0.0118
60*10*3	0.0947	0.1101	0.0032
60*10*4	0.1130	0.1019	0.0060
60*10*5	0.1382	0.0990	0.0065
60*15*3	0.0527	0.0194	0.0031
60*15*4	0.0873	0.0346	0.0042
60*15*5	0.0806	0.0383	0.0043
60*5*3	0.0610	0.0481	0.0025
60*5*4	0.0842	0.0498	0.0073
60*5*5	0.1140	0.0417	0.0063
80*10*3	0.0905	0.0811	0.0008
80*10*4	0.1224	0.0801	0.0048
80*10*5	0.1630	0.0866	0.0115
80*15*3	0.0490	0.0418	0.0040
80*15*4	0.0698	0.0189	0.0010
80*15*5	0.0887	0.0297	0.0083

根据表 2，可以观察到在所有测试实例中，AIG

算法所得结果均优于选定的对比算法。为了深入探究 AIG 算法在处理问题规模变化时的性能稳定性，本研究进一步分析了算法与不同规模实例之间的相互作用，以评估其在多样化条件下的潜在性能表现。

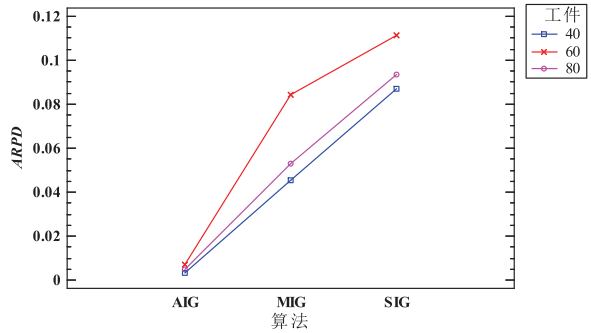


图 4 算法与工件的交互作用图

图 4 展示了算法在不同工件数量下的性能表现。可以看出，随着工件数量的增加，MIG 和 SIG 的性能明显下降。相比之下，AIG 算法在面对不同规模的工件时，始终保持着卓越的性能，且性能波动极小。这不仅凸显了 AIG 算法在处理工件时的稳定性和高效性，其在不同工件数量下的稳健表现，也为解决实际生产中的分布式混合流水车间调度问题提供了有力的支持。

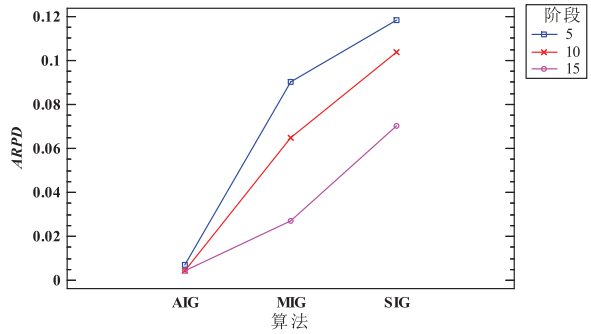


图 5 算法与阶段的交互作用图

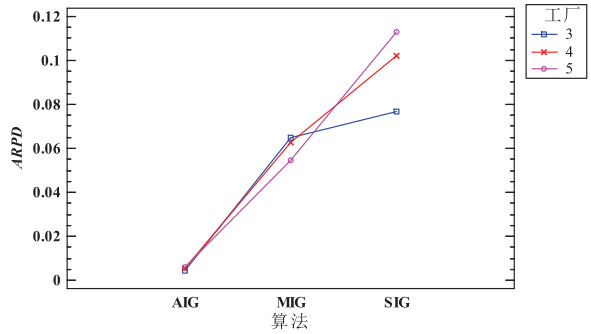


图 6 算法与工厂的交互作用图

图 5 展示了随着阶段数量的增加算法性能的变化趋势。所有算法的性能随着阶段的增加都有所提升，但 MIG 和 SIG 的波动较大。这表明了 AIG 算法在处理复杂问题时具有较高的性能。图 6 进一步展示了随着工厂数量的增加，AIG 算法依然保持了最优的性能。相比之下，MIG 算法的性能随工厂数量的增加而提升，而 SIG 算法的性能却出现了逐级减弱的趋势。

综上所述，在求解 DHFSP 时，AIG 算法的性能优于对比算法。这进一步证实了 AIG 算法在处理复杂生产调度问题时的优越性，为其在实际工业应用中的推广提供了强有力的支持。

5 结论

本文提出了一种基于反馈学习的 AIG 算法。该算法通过学习历史经验，利用知识深入开发解空间，并在演化过程中不断提高求解效率。在 AIG 算法中，学习机制作为高层策略，负责根据历史经验选择合适的扰动策略进行局部开发；而扰动策略则作为底层策略，用于开发解空间。实验结果表明，AIG 在求解 DHFSP 时的性能优于对比算法。

参考文献：

[1] 董君 . 半导体晶圆分布式绿色制造调度问题研究 [D]. 上海 : 上海理工大学，2021.

[2] 李玲 . H 半导体公司生产调度优化研究 [D]. 南昌 : 南昌大学，2023.

[3] CAI J, ZHOU R, LEI D. Dynamic shuffled frog-leaping algorithm for distributed hybrid flow shop scheduling with multiprocessor tasks [J]. Engineering Applications of Artificial Intelligence, 2020, 90: 103540.

[4] 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, 194:105527.

[5] ZHENG J, WANG L, WANG J J. A cooperative coevolution algorithm for multi-objective fuzzy distributed hybrid dlow shop [J]. Knowledge-based systems, 2020, 194: 105536.

[6] WANG L, LI D D. Fuzzy distributed hybrid flow shop scheduling problem with heterogeneous factory and unrelated parallel machine: Ashuffled frog leaping algorithm with collaboration of multiple search strategies [J]. IEEE Access, 2020, 8: 214209-214223.

[7] LU C, GAO L, YI J,et al. Energy-efficient scheduling of distributed flow shop with heterogeneous factories: A real-world case from automobile industry in china [J]. IEEE Transactions on Industrial Informatics, 2021, 17(10): 6687-6696.

[8] GENG K, YE C. A memetic algorithm for energy-efficient distributed re-entrant hybrid flow shop scheduling problem [J]. Journal of Intelligent & Fuzzy Systems, 2021, 41(2): 3951-3971.

[9] WANG J J, WANG L. A bi-population cooperative memetic algorithm for distributed hybrid flow-shop scheduling [J]. IEEE Transactions on Emerging Topics in Computational Intelligence, 2021, 5(6): 947-961.

[10] YING K C, LIN S W. Minimizing makespan for the distributed hybrid flow shop scheduling problem with multiprocessor tasks [J]. Expert Systems With Applications, 2018, 92: 132-141.

A Learning-Based Algorithm for Distributed Hybrid Flow Shop Scheduling

CHEN Xuefeng, ZUO Yang

(School of Electronic Information Engineering, Henan Institute of Technology, Xinxiang 453003, China)

Abstract: The distributed hybrid flow shop scheduling problem holds significant importance in modern manufacturing as it contributes to enhanced productivity and cost reduction. However, the intricate nature of the DHFSP poses a major challenge when seeking solutions. To address this issue, this paper proposes an adaptive learning algorithm based on iterative greedy that incorporates a learning mechanism with the objective of effectively minimizing the maximum completion time. The algorithm comprises three components: firstly, an initialization strategy is employed to construct the initial solution; secondly, the core of the algorithm lies in its dynamic learning mechanism that utilizes historical feedback to intelligently select perturbation strategies based on accumulated optimization experience, thereby progressively enhancing the quality of the solution; and lastly, a roulette selection strategy further enhances the algorithm's ability to explore the global optimal solution. The experimental results demonstrate that the algorithm proposed exhibits significant performance advantages over existing comparison algorithms in addressing the DHFSP.

Key words: DHFSP; adaptive; roulette; learning mechanism

(责任编辑　王　磊)

(上接第 24 页)

Numerical Simulation of Radial-Axial Ring Rolling Process and Combined Control of Rolls

ZHU Xinglin[1,2], SHI Jingru[1], MA Bingxin[1,2], ZHANG Zhiyuan[1,2]

(1.School of Materials Science and Engineering, Henan Institute of Technology, Xinxiang 453003, China;
2.Xinxiang Key Laboratory of Materials Processing Technology and Mould, Xinxiang 453003, China)

Abstract: Ring rolling process is a highly nonlinear and unsteady hot forming process under the action of thermodynamic coupling. The process parameters are numerous and the rolls control are difficult. There is still a lack of efficient and convenient numerical research methods for ring rolling process. Taking the rolling curve as the fundamental variable, the kinematics and dynamics relationships of each roll were established as the theoretical basis. Then based on FORGE software, a finite element model was established according to Φ3000mm CNC ring rolling mill. Through the combined control of rolls, the weak stiffness defects such as eccentricity and ellipse can be predicted successfully. The reliability of the FE model was verified by full size ring rolling test of Inconel718 alloy. The relative error of the predicted result is less than 3%, and this model has been applied to the development of typical parts.

Key words: ring rolling; numerical simulation; roll control; Inconel718 alloy

(责任编辑　吕春红)