分类号：TP391.9　　学校代码：10363
密　级：公　开　　学　　号：2220120117

安徽工程大学
Anhui Polytechnic University

硕士学位论文

题目　改进生物地理学算法在分布式置换流水车间调度问题中的应用研究

论 文 作 者　程龙
指 导 教 师　王雷(教授)/蔡劲草(讲师)
学科（专业）　机械工程
研 究 方 向　智能制造系统

论文提交日期：2025 年 5 月 20 日

分类号：TP391.9　　单位代码：10363

密　级：公　开　　学　　号：2220120117

改进生物地理学算法在分布式置换流水车间调度问题中的应用研究

Research on the Application of Improved Biogeography-based Optimization Algorithm in Distributed Permutation Flowshop Scheduling Problem

学生姓名：程龙

指导教师：王雷(教授)/蔡劲草(讲师)

专　　业：机械工程

研究方向：智能制造系统

论文答辩日期：2025 年 5 月 12 日

安徽工程大学学位论文原创性声明

本人郑重声明：我恪守学术道德，崇尚严谨学风。所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已明确注明和引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品及成果的内容。论文为本人亲自撰写，我对所写的内容负责，并完全意识到本声明的法律后果由本人承担。

学位论文作者签名：

日期：　　年　　月　　日

安徽工程大学学位论文版权使用授权书

学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅或借阅。本人授权安徽工程大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在　　年解密后适用本版权书。

本学位论文属于

不保密□。

学位论文作者签名：　　指导教师签名：

日期：　　年　　月　　日　　日期：　　年　　月　　日

改进生物地理学算法在分布式置换流水车间调度问题中的应用研究

摘要

分布式置换流水车间调度问题具有复杂性和协同性的特点，涉及多个分散工厂协同完成生产任务，工件需在不同车间分配，并在所选车间内按固定工序顺序依次加工。该问题广泛应用于电子制造、汽车零部件生产、半导体制造等行业。研究分布式置换流水车间调度有助于丰富调度优化理论，推动智能优化算法发展，同时优化资源分配、提高生产效率、降低制造成本，并增强制造系统应对市场需求波动和供应链不确定性的能力。基于此，本文对分布式置换流水车间调度问题展开研究，主要研究内容如下：

(1) 研究了以最小化最大完工时间为目标的分布式置换流水车间调度问题。首先，建立分布式置换流水车间调度问题的数学模型，并针对其特性提出了一种改进的生物地理优化算法。通过设计迁移算子和突变算子，提高算法的全局和局部搜索能力。此外，采用田口实验方法优化算法参数，并与现有主流算法进行对比测试，实验结果表明该方法在多个基准测试案例中具有良好的求解性能。

(2) 研究了分布式装配置换流水车间调度问题。以最小化最大完工时间为研究目标并建立相应的数学模型。为了提高算法的初始解质量，设计了一种基于问题特征的启发式方法。进一步优化了路径重联技术和突变方法，提升算法搜索能力。通过引入关键路径邻域搜索算

子和泰勒加速机制，提高了求解效率。实验结果表明，改进的生物地理优化算法在多个分布式装配置换流水车间调度问题测试案例中取得了优异的性能，并刷新了多个基准案例的最优解。

(3) 现实制造环境中，工件加工时间往往受到不确定性因素的影响，如加工延迟或设备故障。针对具有模糊加工时间约束的分布式装配置换流水车间调度问题，本文提出了一种区域生物地理优化算法。该方法基于三角模糊数模型，并将所有栖息地划分为多个区域，通过跨区域迁移和替换过程优化搜索质量。同时，测试了四种不同的迁移率模型，并将区域生物地理优化算法与其他启发式算法进行比较，实验结果表明区域生物地理优化算法在大多数测试实例中优于现有方法。

(4) 结合集成制造系统，研究了分布式柔性装配置换流水车间调度问题。建立了基于集成制造系统的混合整数线性规划模型，并受旅行商问题的启发，引入 Miller-Tucker-Zemlin（MTZ）方法消除冗余约束，使得求解器能够更高效地搜索解空间。此外，提出了基于强化学习的改进策略，以提高算法对环境的感知。通过实验分析不同问题模型的求解能力，并进行敏感性分析，进而获得了一些管理方面的启示。

(5) 研究了绿色低碳场景下的多目标分布式装配置换流水车间调度问题，重点关注调度效率和能耗优化。建立了多目标分布式装配置换流水车间调度问题的混合整数线性规划模型与机器能耗假设模型，采用了三种评价指标对多目标算法进行相应的评估。提出了一种改进的多目标生物地理学优化算法。改进的生物地理优化算法采用双

种群启发式初始化策略，以提升初始解质量。此外，引入新颖的扰动算子，以提高 Pareto 解的质量并加速收敛速度。实验结果表明，改进的生物地理优化算法在多个多目标优化环境下的分布式装配置换流水车间调度问题测试实例中表现出较强的优化能力，为绿色制造环境下的调度优化提供了有效的解决方案。

本文通过数学建模、智能优化算法及启发式方法，为分布式置换流水车间调度问题提供了系统性的研究框架和高效的求解方法。研究成果不仅有助于提升智能制造系统的调度优化能力，还为分布式制造环境中的资源调配和能耗优化提供了新的理论支持和实践指导。

关键词：改进生物地理学优化算法；分布式置换流水车间调度；装配工厂；不确定性因素；混合整数线性规划模型；强化学习；绿色调度

Research on the Application of Improved Biogeography-based Optimization Algorithm in Distributed Permutation Flowshop Scheduling Problem

ABSTRACT

The distributed permutation flowshop scheduling problem is characterized by complexity and collaboration, involving multiple dispersed factories working together to complete production tasks. Jobs need to be allocated to different workshops and processed sequentially in a fixed order within the selected workshop. This problem is widely applied in industries such as electronics manufacturing, automotive parts production, and semiconductor manufacturing. Research on the distributed permutation flowshop scheduling problem helps to enrich scheduling optimization theory, promote the development of intelligent optimization algorithms, optimize resource allocation, improve production efficiency, reduce manufacturing costs, and enhance the system's ability to cope with market demand fluctuations and supply chain uncertainties. Based on this, this paper conducts an in-depth study of the distributed permutation flow shop scheduling problem, with the main content as follows:

(1) The distributed permutation flowshop scheduling problem was studied with the objective of minimizing makespan. First, a mathematical model was established, and an improved biogeography-based optimization algorithm (BBO) was proposed to address its characteristics. By designing migration and mutation operators, the algorithm's global and local search capabilities were enhanced. Additionally, the Taguchi experimental

method was applied to optimize algorithm parameters, and comparative tests were conducted with mainstream algorithms. Experimental results show that the proposed method achieves superior performance in multiple benchmark test cases.

(2) The distributed assembly permutation flowshop scheduling problem (DAPFSP) was examined, also with the objective of minimizing makespan. To improve the initial solution quality, a heuristic method based on problem characteristics was designed. Furthermore, path-relinking techniques and mutation methods were optimized to enhance the algorithm's search capability. The introduction of a critical path neighborhood search operator and a Taylor acceleration mechanism improved the solution efficiency. Experimental results demonstrate that the improved BBO achieved excellent performance across multiple test cases, discovering several new optimal solutions.

(3) Considering real-world manufacturing environments, job processing times are often affected by uncertainties such as processing delays or equipment failures. To address the DAPFSP with fuzzy processing time constraints, this paper proposes a regional biogeography-based optimization algorithm. Based on a triangular fuzzy number model, all habitats were divided into multiple regions, optimizing search quality through cross-regional migration and replacement processes. Moreover, four different migration rate models were tested, and comparative analyses were conducted between the regional biogeography-based optimization algorithm and other heuristic algorithms. Experimental results show that the regional biogeography-based optimization algorithm outperformed existing methods in most test instances.

(4) The distributed flexible assembly permutation flowshop scheduling problem (DFAPFSP) was studied in combination with

integrated manufacturing systems. A mixed-integer linear programming model was established, inspired by the traveling salesman problem, and the Miller-Tucker-Zemlin (MTZ) method was introduced to eliminate redundant constraints, enabling the solver to search the solution space more efficiently. Additionally, an improved reinforcement learning-based strategy was proposed to enhance the solution's adaptability to the environment. Experimental analyses evaluated the model's performance across different problem sizes, and sensitivity analyses provided valuable managerial insights.

(5) The distributed assembly permutation flowshop scheduling problem was further investigated under a multi-objective optimization environment, focusing on scheduling efficiency and energy consumption optimization. A mixed integer linear programming model was established with the objectives of minimizing makespan and total machine energy consumption. To enhance optimization performance, an improved multi-objective biogeography-based optimization algorithm was proposed. The algorithm adopted a dual-population heuristic initialization strategy to improve initial solution quality, incorporating hybrid migration and improved mutation operators to strengthen global and local search capabilities. Additionally, a novel perturbation operator was introduced to enhance Pareto solution quality and accelerate convergence. Experimental results indicate that the improved biogeography-based optimization algorithm demonstrated strong optimization capabilities across various multi-objective test cases, providing effective solutions for scheduling optimization in green manufacturing environments.

Through mathematical modeling, intelligent optimization algorithms, and heuristic methods, this paper provides a systematic research framework and efficient solving methods for the distributed permutation

flowshop scheduling problem. The research results enhance scheduling optimization capabilities in intelligent manufacturing systems while offering new theoretical support and practical guidance for resource allocation and energy optimization in distributed manufacturing environments.

KEY WORDS: Improved Biogeography-Based Optimization Algorithm; Distributed Permutation Flowshop Scheduling; Assembly Factory; Uncertainty Factors; Mixed Integer Linear Programming Model; Reinforcement Learning; Green Scheduling

目录

插图清单

插表清单

第1章 绪论

1.1 课题来源

本课题研究得到了安徽省高校自然科学重点研究项目《基于改进群智能优化算法的柔性作业车间绿色调度问题研究》(项目编号：2022AH050978)、《基于动态群智能的多约束分布式混合流水车间调度研究》(项目编号：2023AH050935)，检测技术与节能装置安徽省重点实验室开放基金项目《基于动态群智能优化的分布式混合流水车间节能调度研究》(项目编号：JCK2022B01)等的资助。

1.2 课题研究背景及意义

制造业作为国民经济的重要支柱产业，对中国经济的长期稳定增长和社会发展具有重要意义。近年来，中国制造业的高速发展在促进就业、提升国民生产总值、扩大国际市场份额等方面发挥了关键作用。随着“中国制造 2025”战略的深入推进，中国制造业正逐步迈向高质量发展阶段，传统生产方式逐渐向智能制造、绿色制造和定制化制造转型。然而，随着全球产业链分工的深化以及技术进步的加速，制造业面临的竞争压力和复杂度不断增加，尤其是在多地协同的分布式制造模式下，生产调度效率与资源协调优化成为当前研究的重点和难点。

分布式制造系统(Distributed Manufacturing System, DMS)是一种面向柔性化、动态化的生产方式，将多个工厂和生产设备通过信息化和网络化手段联结起来，以实现资源共享、生产任务协同和效率最大化。该模式在解决传统集中制造模式中资源分配不均、产能利用率低下等问题方面表现出显著优势。然而，分布式制造的有效实施依赖于各个制造单元之间的高效协调与调度。特别是在当前供应链全球化和客户需求多样化的背景下，如何在分布式制造环境下实现最优的生产调度，已成为提高生产系统灵活性和市场适应力的核心问题。

在分布式制造的环境中，分布式置换流水车间调度问题(Distributed Permutation Flowshop Scheduling Problem, DPFSP)尤为具有挑战性。DPFSP 通过协调多个生产单元间的任务分配、资源调度和加工顺序优化以此解决各生产单元

间的资源不均衡、加工时间不确定等问题。该问题的解决不仅关系到资源利用效率的提升，还将对整个供应链的响应速度和稳定性产生深远影响。同时，在多目标和多约束的实际生产环境中，DPFSP 的有效解决将推动分布式制造朝着绿色化、低耗能、响应灵活的方向迈进。研究 DPFSP 不仅具有现实意义，也为分布式制造的实践提供了理论支撑。

在 DPFSP 相关问题的求解过程中，由于其具有复杂的组合优化特性，传统的优化算法难以在合理时间内找到较优解。因此，采用元启发式算法(Metaheuristic Algorithm, MA)来求解此类问题逐渐成为研究趋势。元启发式算法凭借其随机性和全局搜索能力，可在庞大的问题空间中寻找近似最优解，具有较强的适应性和灵活性。基于生物地理学优化算法(Biogeography-Based Optimization, BBO)作为一种元启发式算法具备对问题的探索与开发能力。BBO 通过模拟物种在不同栖息地的迁移、变异等过程，能够较好地平衡全局搜索与局部开发，有效避免早熟收敛问题，适用于求解复杂多目标调度问题。此外，结合分布式调度的特殊需求，基于生物地理学优化的改进算法(如混合迁移算子、优化异变策略等)能够进一步提高算法的搜索效率与求解质量，从而为分布式制造调度问题提供高效、可行的解决方案。本文利用改进的 BBO 算法对 DMS 进行研究，并将其应用到分布式置换流水车间调度问题中，为生产制造企业带来以下几点优势：

(1) 通过利用改进的 BBO 算法研究 DMS 中的调度问题，能够在分布式环境下有效分配资源，最大限度提升资源利用率，从而提高生产线的整体生产能力和效益，助力企业实现高效运作。

(2) 优化调度的结果提供了详细的工件加工方案，包括各工件的加工机器、开始时间和完成时间等信息，便于企业在分布式生产场景中科学管理生产流程，大幅提高生产效率并降低人工管理成本。

(3) 对不确定性事件进行研究，通过改进 BBO 算法获得的高效调度策略和优化算法，可快速响应生产过程中出现的动态事件，提供科学合理的调度方案，确保生产流程顺畅并减少因突发状况造成的延迟时间。

(4) 基于实际分布式生产环境设计的问题模式，能够有效提高企业的运作效率，在分布式生产的场景下实现对原材料、工件、加工设备和人力资源等资源的

高效管理，全面提升生产的灵活性和响应速度。

综上所述，本文通过改进的 BBO 算法在分布式制造系统中的应用，不仅能提升企业的资源利用率和生产效益，还对制造业领域的转型升级提供了实质性帮助，为企业在竞争激烈的市场中提供了核心竞争力。

1.3 国内外研究现状

1.3.1 分布式置换流水车间调度

DPFSP 是分布式制造领域中最为重要的生产调度问题之一。DPFSP 包含多个生产工厂，每个生产工厂包含多个生产机器，每个工件只能被分配到一个工厂中完成加工任务，工件的每道工序需要依次在这些机器上完成加工。DPFSP 常见的优化目标为最小化完工时间(makespan)。DPFSP 的调度方案不仅需要考虑工件的分配问题还需要考虑工厂内工件的加工顺序，与置换流水车间调度问题(Permutation Flowshop Scheduling Problem, PFSP)相比无疑是增加了问题求解的复杂性[1-4]。

Naderi 等人首次研究了 DPFSP[5]。Naderi 提出了六种 DPFSP 的混合整数线性规划模型(Mixed Integer Linear Programming, MILP)并设计了两种具有建设性的工厂分配方式(NR1,NR2)，这些模型与工厂分配方法被广泛的应用在分布式制造调度问题的模型表达与解码方法中。Jing 等人研究了具有不确定性加工时间约束下的 DPFSP[6]。Han 等人研究了以最小化 makespan 为优化目标的串行分布式置换流水车间调度问题(Serial Distributed Permutation Flowshop Scheduling Problem, SDPFSP)[7]。Han 基于泰勒加速度[8]方法提出了一种新的邻域插入加速度方法，通过这种方法可以有效的减少算法计算的复杂度。

目前，很多国内外学者通过改进各类 MA 解决 DPFSP 并取得了丰富的成果，但这些算法在运行后期容易陷入局部最优且算法结果的随机性比较高。针对此存在的问题，本文使用多种迁移率模型进行实验测试以此为 BBO 选择解决 DPFSP 最佳的迁移率模型，同时提出了一种深度搜索策略用来寻得更优解，大量的仿真实验证明所提方法的有效性。

1.3.2 分布式装配置换流水车间调度

分布式装配置换流水车间调度问题(Distributed Assembly Permutation Flowshop Scheduling Problem, DAPFSP)是由 DPFSP 衍生出的一类分布式生产调度问题。DAPFSP 包含两个子问题，即工件的生产问题与产品的装配问题。DAPFSP 不仅需要考虑优化单个车间内的工序排序，还需要协调多个分布式工厂整合资源和产品装配线进一步提高生产效率和灵活性。通过研究 DAPFSP，可以实现多工厂协同高效生产管理。相比 DPFSP，DAPFSP 具有更多的实际意义。

DAPFSP 概念最早由 Hatami 提出[9]，Hatami 利用 Naderi[5]提出的最小顺序序列的 MILP 模型表达 DAPFSP。Hatami 还提出了两种局部搜索策略即产品搜索(Local Search Product, LSP)与工件搜索(Local Search Job, LSJ)，Hatami 将这两种搜索结构与变邻域搜索算法(Variable Neighborhood Search, VNS)结合与 6 种启发式进行实验结果比较验证了方法的可行性。Hatami 的方法无疑是具有建设性的，但无论是启发式方法还是 VNS 算法在面对复杂度较大的 DAPFSP 时的搜索能力较为薄弱。针对这种问题，Lin[10]提出了一种路径重联技术并将其运用到 BBO 中并提出了混合生物地理学优化算法(Hybrid Biogeography-Based Optimization, HBBO)。HBBO 具有高鲁棒性并且容易跳出局部最优，HBBO 是解决 DAPFSP 的一种有效方法。Hamzadayi[11]在 Hatami[9]提出的最小顺序序列的 MILP 模型基础上提出了一种基于多旅行商问题结构(multiple Travelling Salesman Problem, mTSP)的 MILP 模型，实验结果证明 mTSP 的效果要优于 Hatami 的模型。Zhang[12]提出了一种基于矩阵-立方体的方法求解以最小化 makespan 为优化目标的 DAPFSP 问题。Huang[13]提出了一种群思维迭代贪婪算法(groupthink iterative greedy algorithm, gIGA)来解决以总流程时间(Total Flowtime, TF)为目标的 DAPFSP。IGA 的初始化方法是基于 NEH 法和 Palmer 法获得一批初始解。而 Ferone[14]采用的是一种偏随机化方法生成初始解。Wang[14]提出了一种基于分布式估计的模因算法(Estimation of Distribution Algorithm-based Memetic Algorithm EDA-MA)求解 DAPFSP。

综上所述，无论采用哪种算法求解 DAPFSP，有效的搜索策略都是提高算法深度搜索性能的关键一步。尽管大多数研究人员在对 DAPFSP 的研究中探索了

多种优化策略和解决方案，但在实际求解 DAPFSP 时经常出现计算复杂度高、精度不足等困难。这些问题使得现有的研究成果难以在复杂的工业环境中直接应用。本文将重点提出新的方法和改进策略来应对这些实际挑战，旨在提高 DAPFSP 的效率和实用性。

1.3.3 多种约束下的分布式装配置换流水车间调度

相比传统的分布式装配置换流水车间调度问题，多种约束下的分布式装配置换流水车间调度问题进一步引入了资源、能耗、设备状态等多种约束条件，增强了问题的现实复杂性和挑战性。这类问题在现代制造系统中具有广泛的应用背景，如电子产品、汽车零部件的装配和制造中，都实现了工厂间的高效协同调度，以满足不同工艺、资源和节能等多重约束条件的需求。

Hatami 等人[15]提出了带有序列依赖设定时间的分布式装配置换流水车间调度问题(Distributed Assembly Permutation Flowshop Scheduling Problem with Sequence Dependent Setup Times, DAPFSP-SDST)。Yang 和 Xu [16]提出了一个新颖的分布式装配置换流水车间调度问题，具有灵活装配和批量交付的特点(Distributed Assembly Permutation Flowshop Scheduling Problem with Flexible Assembly and Batch Delivery, DAPFSP-FABD)。Wei 等人[17]提出了一个节能的分布式装配阻塞流水车间问题(Energy-Efficient Scheduling of a Distributed Assembly Blocking Flowshop, EEDABFSP)。Huang 等人[18]提出了一个有效的模仿算法，以解决带有装配机器的分布式流水车间调度问题。Zhang 等人[19]提出了一个分布式估计算法，以解决 DAPFSP，并开发了基于矩阵立方体的概率模型，利用有效的采样机制估计最优解的概率分布，并进行全局探索以寻找有前景的区域。Song 等人[20]使用遗传编程(Genetic Programming Hyper-Heuristic, GP-HH)为从预设计低级启发式方法中生成启发式序列的高级策略集，提出了 GP-HH 算法来解决 DAPFSP-SDST。

然而在实际生产中，各种约束条件通常都可以归结为加工时间的某种不确定性因素。工件的加工时间通常受到设备性能、工人操作水平、材料特性等不确定因素的影响，很难准确预测。因此，模糊加工时间的约束在一定程度上更加贴合真实的生产情境。研究具有模糊加工时间的 DAPFSP 能够更好地模拟现实生产

中的不确定性，为制定更具鲁棒性的调度方案提供理论支持。相比其他约束条件，模糊加工时间的研究能有效提升调度方案的适应性，增强算法在复杂工业环境中的应用潜力，从而更好地满足现代制造系统对高效和灵活性的需求。

1.3.4 基于集成制造系统下的分布式装配置换流水车间调度

集成制造系统(Integrated Manufacturing System, IMS)是通过信息技术将生产、装配、管理等各个环节进行有机结合。自动化和柔性化是 IMS 最大的特点，其实际应用价值在于提高生产效率、降低成本。将 DAPFSP 中的生产和装配环节进行结合后，可以显著提高资源的利用率。通过信息集成和自动化技术，实现工件生产和产品装配环节间的无缝衔接，减少生产切换时间和物流成本，能够提升产品质量和企业自身的竞争力。

在过去的十年里，学者们拓展了有关 DAPFSP 的问题模型，并探索了可能出现的各种生产场景，如鲁棒性生产条件和异构生产机器等[9,10,14,15]。然而，DAPFSP 本质上是一个理想化的生产调度模型[21]。在实际生产环境中，一个生产设备往往需要多个装配工厂来保证充足的装配效率[60,61,62,63]。这种生产模式被描述为协同装配工厂(Collaborative Assembly Factory, CAF)。CAF 为产品的组装路线提供了多种选择，允许产品在任何装配工厂灵活组装，从而提前了产品的交货时间，提高了客户的满意度[22-26]。目前已有学者研究了具有协同装配工厂的分布式装配置换流水车间调度(Collaborative Assembly Factory in Distributed Assembly Permutation Flowshop Scheduling Problem, C-DAPFSP)。Zhang[27]首次研究了具有多个装配机器的分布式装配置换流水车间调度问题，并考虑了工件的准备时间约束。Yang[28]提出了考虑产品批量交付约束下的分布式柔性装配置换流水车间调度问题模型。Wang[29]研究了能量感知的分布式流水车间与灵活装配调度问题，提出了一种带反馈的合作启发式算法。该算法结合了问题特定的启发式方法进行种群初始化，通过反馈机制和局部强化策略实现合作搜索，同时引入了节能策略以降低能耗，将反馈机制与局部强化策略结合，从而显著提升了算法在解决复杂调度问题中的效率和效果，数值对比结果显示该算法在求解能量感知的分布式流水车间与灵活装配调度问题方面优于现有算法。

装配线中装配机器的数量也是影响生产效率的关键因素之一。通常只有多个

装配流水线才能防止出现产品装配可能造成的堵塞情况。为了提高产品装配的灵活性，设置多个装配机器的柔性装配工厂是提高生产效率的关键。在 Hatami[9]的研究基础上，Pan[30]首次提出了将 DAPFSP 的每个生产工厂添加了一条单独的生产线并设计了问题的 MILP 模型，针对问题的特征提出了三种建设性的启发式算法与两种加速方法。Pourhejazy[31]等人研究了分布式两阶段装配流水车间调度。Pourhejazy 将生产工厂与装配流水线组合成一个集成生产环境。与 DAPFSP 不同，DTSAFSP 中每道工件的工序相互独立并且机器不存在空载状态。

根据近年文献调研可知，暂时还没有研究者将生产和装配作为一个集成制造系统并且与柔性装配机器结合进行研究。关这类研究还非常有限，因此，本文研究了以最小化完工时间(makespan)为优化目标的基于集成制造系统的分布式装柔性配置换流水车间调度问题(Distributed Flexible Assembly Permutation Flowshop Scheduling Problem based on Integrated Manufacturing System, IMS-DFAPFSP)。

1.3.5 绿色能耗下的多目标分布式装配置换流水车间调度

目前有关 DAPFSP 的研究主要是提出了各种生产场景，并采用不同的搜索策略以提高生产效率。然而，在实际生产场景中，机器的总能耗是另一个需要关注的调度目标。考虑机器能耗等生态资源的消耗被称为绿色调度问题(Green Scheduling Problem , GSP)[32]。在 DMS 生产模型中，多工厂生产模式的建立导致了大量生产设备的使用，使得能耗较单工厂生产高出数倍。为了平衡生态环境与生产效率之间的关系，研究考虑绿色调度的分布式装配流水车间调度问题具有重要意义。然而，目前相关研究较少。因此，本文研究了考虑机器能耗与最小完工时间的多目标分布式装配流水车间调度问题(Multi-Objective Distributed Assembly Permutation Flowshop Scheduling Problem, MO-DAPFSP)。

DAPFSP 包括两个子问题：生产阶段和装配阶段。DAPFSP 被证明是 NP 难问题，因此多目标 DAPFSP 具有更高的复杂性和更大的求解难度。元启发式算法是解决多目标问题(Multi-Objective Problems, MOPs)最常见的方法。常见的元启发式算法包括遗传算法(Genetic Algorithm, GA) [33]、粒子群优化(Particle Swarm Optimization, PSO)算法[34]、模拟退火(Simulated Annealing, SA)算法[35]、基于生物

地理优化算法[36]、灰狼优化（gray wolf optimization algorithm, GWO）算法[37]、蚁群算法(Ant Colony Optimization Algorithm, ACO)[38]和混合青蛙跳跃算法(Shuffled Frog Leaping Algorithm, SFLA)[39]。

考虑到 MOP 的特点，研究人员在设计 MOA 时会根据问题的特性调整原始的基本算法。Faraji Amiri 和 Behnamian [40] 提出了基于场景估计的数学公式和分布算法(Distribution Algorithm based on Scene Estimation, DA-SE)。Huang 等人[41]设计了一个两阶段进化算法(Two-stage Evolutionary Algorithm, TEA)。在 TEA 的第一阶段，采用了双种群结构，以获取高质量且多样性高的个体。第二阶段使用基于分解的多目标进化算法(Decomposition-based Multi-Objective Evolutionary Algorithm, MOEA/D)进行搜索。Meng 等人[42]解决了具有可控加工时间的柔性车间调度问题(Flexible Job Shop Problem with Controllable Processing Times, FJSP-CPT)，目标是同时最小化完工时间和总能耗。为解决该问题，Meng 等人开发了一个混合整数线性规划模型，并使用 ε 方法求解小规模实例的最优帕累托前沿。对于中大型问题，提出了一种高效的多目标混合青蛙跳跃算法(Multi-Objective Hybrid Shuffled Frog-Leaping Algorithm, MO-HSFLA)，以逼近帕累托前沿。Wei 等人[43]解决了在车间生产中因机器故障等动态事件导致的原调度方案不可行的问题。为此，建立了一个具有机器故障的多目标动态柔性车间调度问题的数学模型，提出了一种基于博弈论的多目标迁徙鸟类优化(Multi-Objective Migrating Birds Optimization MO-MBO)算法。He 等人[44]提出了一种有效的多目标 Jaya(Effective Multi-Objective Jaya, EMOJaya)算法来解决多目标车间调度问题，目标是同时最小化完工时间、总流通时间和平均延迟。

虽然关于分布式装配流水车间调度问题的绿色调度研究文献相对较少，但在绿色调度背景下的分布式多目标调度仍然是多目标调度中研究最多的领域之一。目前，许多学者已研究了绿色调度问题，建立了相关模型并提出了有效的解决方案。Dong 和 Ye[45]提出了一种考虑可再生能源和储能系统的分布式两阶段回流混合流水车间调度模型(Distributed Two-Stage Return Hybrid Flow Shop, DTS-RHFS)。该模型包含多个异构工厂，每个工厂均考虑生产阶段和检验与维护阶段。上层模型优化工件调度以最小化完工时间，而下层模型优化能源分配以最小化总碳排放和总能耗成本。为解决此问题，Dong 和 Ye 提出了一种改进的混合沙丁鱼群算法

(Sardine Swarm Algorithm, SSA)和第三代非支配排序遗传算法(Nondominated Sorting Genetic Algorithm III, NSGA-III)算法，并通过实验验证了使用可再生能源和储能系统在减少碳排放和能耗成本方面的有效性。Hou 等人[46]提出了一个集成的绿色分布式生产与配送调度问题(Integrated Green Production and Distribution Scheduling Problem, IGP-DSP)，其目标是同时最小化总延迟时间和总碳排放量。通过基于 Q 学习的多目标进化算法(Q-learning based Multi-objective Evolutionary Algorithm, Q-MEA)进行数值实验表明，与传统模型相比，该方法可显著减少 13.10%的总延迟时间和 21.02%的总碳排放。Xue 等人[47]研究了具有不相关并行机的绿色调度问题(Unrelated Parallel Machine Green Scheduling Problem, UPMGSP)，其目标函数是最小化完工时间和总碳排放。Xue 等人提出了一种估算分布式进化记忆算法(Estimated Distributed Evolutionary Memetic Algorithm, EDEMA)，通过种群初始化、多目标排序、概率模型、邻域搜索和种群灾难策略，在解决 UPMGSP 方面显著优于第二代非支配排序遗传算法(Nondominated Sorting Genetic Algorithm II, NSGA-II)和分布式进化算法(Distributed Evolutionary Algorithm, DEA)。Li 等人[48]研究了分布式流水车间的能效调度问题(Distributed Permutation Flowshop Energy Efficiency Scheduling Problem, EEDPFSP)，其目标是最小化总加工时间和总能耗。考虑到分布式和多目标优化的复杂性，提出了一种改进的 NSGA-II 算法。

多目标调度问题通常通过多目标元启发式算法来解决。然而，由于该类算法具有内在的随机性，其求解精度往往不稳定。大多数多目标算法(multi-objective algorithms, MOA)容易陷入局部最优，导致解的质量较差。针对这一缺陷，本文采用了生物地理优化算法。BBO 具有较强的全局搜索能力，在具有鲁棒环境问题中可以为算法提供稳定且高质量的解。此外，BBO 可以保持解的多样性，并且易于与其他算法结合，使其非常适合解决多目标问题。

1.3.6 生物地理学优化算法

生物地理学优化算法是一种群智能元启发式算法，其设计灵感来源于生物地理学中的物种迁徙模型[49]。BBO 将物种居住的区域称为栖息地(Habitat, H_r)，栖息地的适宜度通过栖息地适宜性指数(Habitat Suitability Index, HSI)表示。HSI 受

许多因素的影响，如降水量、海拔、地表温度等。这些因素统称为适宜性指数变量(Suitability Index Variables, SIVs)。每个栖息地上生存的物种数量不同，具有高HSI的栖息地资源更多，生存条件更好，因此物种数量较多。相反，低HSI的栖息地不具备适宜的生存条件，环境较差，因此栖息地内的物种数量较少。

BBO 算法主要由物种迁移算子和栖息地变异算子组成。在迁移阶段，每个栖息地 H_r 具有迁入率 λ_r 和迁出率 μ_r。λ_r 和 μ_r 决定了物种是否会迁出栖息地或从其他栖息地迁入该栖息地。移入率和移出率的公式如下：

$$\lambda_r = I(1-\frac{S_r}{S_{\max}}) \tag{1-1}$$

$$\mu_r = E\frac{S_r}{S_{\max}} \tag{1-2}$$

其中，I ($I \in (0,1]$)和 E ($E \in (0,1]$)分别表示栖息地的最大迁入率和最大迁出率。通常认为最大迁入率和最大迁出率相等，即 $I = E$。S 表示栖息地 H_r 中的物种数量，$S_{\max}$ 表示栖息地所能容纳的最大物种数量。图 1-1 展示了当 $I = E = 1$，$S_{\max} = 100$ 时的栖息地 H_r 线性迁移率模型。

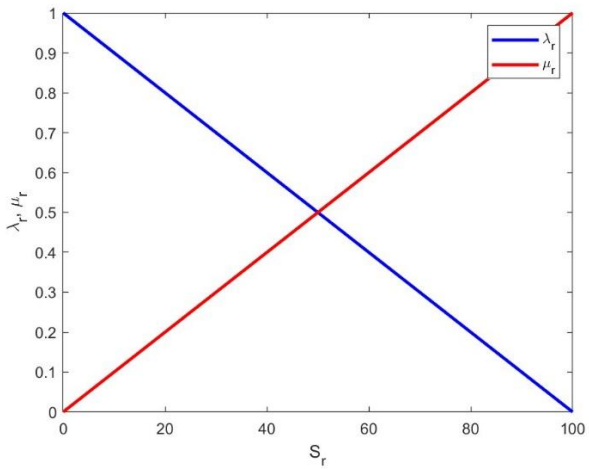

图 1-1 线性迁移率模型示意图

随着迁移的不断进行，物种不断从低HSI的栖息地迁移至高HSI的栖息地。所有栖息地趋向于动态平衡，此时每个栖息地中的物种数量接近均衡，算法面临陷入局部最优的风险，BBO 设计了栖息地突变操作来打破局部最优。经历突变的栖息地其HSI会发生显著变化。栖息地变异的概率公式如下：

$$m_r = m_{\max}(1-\frac{P_r}{P_{\max}}) \tag{1-3}$$

$$\dot{P}_r = \begin{cases} -(\lambda_r+\mu_r)P_r + \mu_{r+1}P_{r+1}, & r=0 \\ -(\lambda_r+\mu_r)P_r + \lambda_{r-1}P_{r-1} + \mu_{r+1}P_{r+1}, & 1\le r\le S_{\max}-1 \\ -(\lambda_r+\mu_r)P_r + \lambda_{r-1}P_{r-1}, & r=S_{\max} \end{cases} \tag{1-4}$$

其中 $m_{\max}$ ($m_{\max}\in(0,1)$)表示最大突变率。P_r 为栖息地 H_r 的物种数概率，$P_{\max}$ 为最大物种数概率。公式(1-4)为物种数量概率的更新公式。图 1-2 展示了当 $m_{\max}=0.1$，$S_{\max}=100$ 时的栖息地突变率模型。

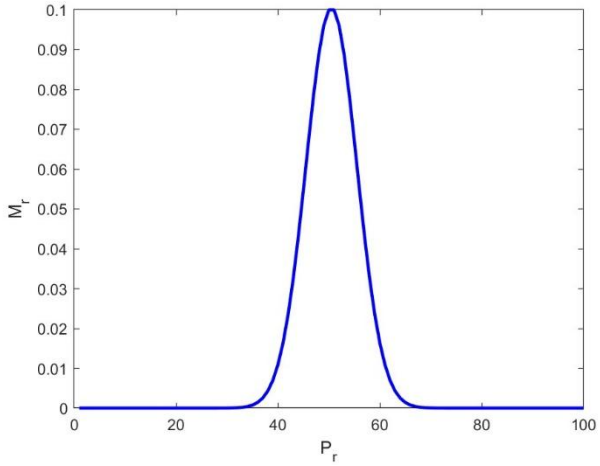

图 1-2 栖息地突变率模型示意图

由图 1-1 可知，当 $S_r = S_{\max}/2$ 时，栖息地内的物种数量达到动态平衡。此时，λ_r 等于 μ_r，根据图 1-2 中发现栖息地变异概率达到最大值。因此可以得出结论，当栖息地中的物种数量接近 $S_{\max}/2$ (即 λ_r 等于 μ_r)时，栖息地发生突变的概率最大。栖息地突变操作打破了栖息地的内部动态平衡，物种开始新一轮的迁移。

生物地理学优化算法已被广泛应用于各种组合优化问题中，表现出较强的全局搜索能力和效率，能够在多次迭代中寻求接近最优解。然而，BBO 在局部搜索方面的能力相对有限，且在搜索过程中容易陷入局部最优，导致早熟收敛，进而影响其效果。为此，本文针对传统 BBO 算法进行改进，以更好地应用于分布式置换流水车间调度的求解。

1.4 论文主要研究内容

本文通过利用改进的生物地理学优化算法对 DPFSP 生产模式进行拓展和深入探索，并且考虑了以轴承加工为生产背景的实际调度环境的模拟方案，具体的研究内容如下所述：

(1) 第 1 章围绕分布式置换流水车间调度问题，分析了该领域在分布式装配置换、多种约束、集成制造系统及多目标优化等方面的国内外研究现状，揭示了当前研究空白。同时，介绍了生物地理学优化算法的基本框架及其优缺点。

(2) 第 2 章对分布式置换流水车间调度问题进行详细描述并建立模型，设计了一种初始化方式用来提升初始解的质量以此来减少冗余解出现的概率。提出了针对 DPFSP 所设计的编码方式、迁移算子与 4 种突变算子。利用田口实验[50]调整 BBO 算法参数使得算法的性能最大化。改进的 BBO 算法在 420 个小型案例和 720 个大型案例中与其他算法进行测试比较，实验结果证明了改进 BBO 算法的可行性。

(3) 第 3 章在 DPFSP 的研究基础上拓展了生产模式引入了产品装配概念，建立了分布式装配置换流水车间调度问题的问题模型。为了避免出现冗余解，基于求解问题的特征设计了一种初始化方式。提出了一种改进的路径重联方法作为算法的全局搜索方式。一种优化的突变策略增强解的多样性以此避免出现局部最优。设计了三种关键路径的深度搜索方式加强算法后期的搜索能力。采用泰勒加速度方式降低了算法计算复杂度。通过消融实验证明了所提出搜索方式的有效性。

(4) 第 4 章考虑了模糊加工时间的分布式装配置换流水车间调度问题，引入了三角模糊数来表示加工时间的不确定性，以更贴近实际生产环境中的不确定性因素。基于此，构建了模糊加工时间下的分布式装配置换流水车间调度问题的数学模型，目标是最小化模糊完工时间。为了高效求解该问题，提出了一种基于区域地理学优化算法的智能优化算法，结合迁移率模型来模拟解空间中的种群迁移行为，增强算法的全局搜索能力。

(5) 第 5 章在集成制造系统背景下，研究了分布式柔性装配置换流水车间调度问题，以应对多工厂协同生产中的复杂性和动态性。首先，基于混合整数线性规划方法，建立了问题的数学模型，目标是最小化总完工时间并优化资源配置。

为了处理问题的高维性和非线性特征，提出了一种结合 SARSA 强化学习的生物地理学优化算法。

(6) 第 6 章在绿色制造背景下，研究了考虑机器能耗的绿色分布式装配置换流水车间调度问题，为了优化产品生产效率的同时最小化机器能源消耗。首先，建立了包含能耗的多目标问题数学模型，目标函数包括最小化总完工时间及最小化总能耗为了高效求解该问题，提出了一种有效的多目标生物地理学优化算法，结合混合迁移算子和改进的突变算子，以增强算法的全局搜索能力和解的多样性。实验结果表明所建立的能耗模型能够准确反映实际生产中的能源消耗情况。结合混合迁移算子和改进突变算子的多目标算法在 Pareto 解集的分布性和收敛性方面表现优异。显著性分析揭示了关键参数对调度结果的显著影响，为绿色制造提供了理论依据。

(7) 第 7 章对全文进行了总结，系统梳理了研究路线与关键成果，明确了各阶段研究的逻辑脉络与创新点。同时，深入探讨了研究中可能存在的局限性，从问题建模、算法设计以及实际应用三个层面进行分析，为后续研究提供了改进方向和参考依据。

1.5 本章小结

本章首先对课题研究背景和意义进行了介绍，以此引出了本文的研究主题：基于改进生物地理学优化算法的分布式置换流水车间调度问题； 随后对具有装配工厂、模糊加工、集成制造系统和绿色多目标分布式置换流水车间调度问题进行国内外现状分析，同时详细介绍了生物地理学优化算法。最后对本文的研究内容及思路进行了叙述。

第 2 章 分布式置换流水车间调度问题研究

随着全球企业的激烈竞争与市场环境的快速变化，制造系统需要更加灵活应对市场需求。分布式置换流水车间调度问题作为一种高效的调度策略，近年来受到了研究人员的广泛关注。本章研究了以最小化完工时间(makespan)为目标的DPFSP，首先描述并建立了 DPFSP 的数学模型。其次，提出了基于 DPFSP 问题特征所设计的迁移算子和突变算子用来对问题进行全局与局部搜索。考虑到改进的 BBO 算法中具有多个参数，因此采用田口实验方法调整算法参数。所设计的算法与目前流行的算法进行案例测试，根据实验结果验证了所提出方法的有效性。

2.1 分布式置换流水车间调度问题

2.1.1 问题描述

DPFSP 可以理解为工序排序问题和工件分配问题的结合。具体而言，DPFSP 将n个工件分配到 f 个工厂中，其中每个工厂包含相同数量的m台不相关机器。每个工件需经历m道加工工序，其每道工序的加工时间均大于零。在加工过程中，工件不能中断，且每个工件的所有工序必须在同一个工厂内完成。每台机器在同一时间只能加工一个工序，且每个工件在任意时刻只能由一台机器加工。为简化问题，本文不考虑机器启动时间和工件在不同工厂之间移动的时间。DPFSP 的目标是同时完成工厂分配和工件排序，使得 makespan 达到最小。

为了完整的表达 DPFSP 的问题模型，本文采用了混合整数线性规划的方法。MILP 在求解 DPFSP 中具有重要作用，因为它能够有效地处理整数和连续决策变量的混合属性，同时能够对复杂约束条件和目标函数进行精确建模。DPFSP 的核心在于在多台机器上对多个工件进行有序调度，这涉及工件排序、机器分配和加工定时等问题，均可通过 MILP 进行优化建模。本文所提出的 MILP 模型为实现最优调度提供了一个系统性的方法，并有助于分析复杂调度问题中的多种因素。

2.1.2 模型建立

本文采用的 MILP 模型是基于最小序列的模型(Minimal Sequence-Mased Model, MSBM)。如表 2-1 所示，在设计 MSBM 之前需要提前定义 DPFSP 的参数、符号和变量，其中变量分为二元变量(Binary Variable, BV)和连续变量(Continuous Variable, CV)。

表 2-1 MSBM 约束模型中的参数、符号和变量

参数	描述
n	工件数量
m	机器数量
f	工厂数量
p_{ij}	工件 j 在机器 i 上加工所需的加工时间
M	一个足够大的正整数
符号	
j,k	表示工件，其中 $j,k=0,1,...,n$，0 为虚拟工件
i	表示机器，其中 $i=1,...,m$
变量	
X_{jk}	二元变量，如果工件 j 为工件 k 的前置加工工件则为 1 否则为 0
C_{ij}	连续变量，工件 j 在机器 i 上的完工时间
$C_{\max}$	目标函数，最大完工时间

目标函数：

$$Minimise C_{\max} \tag{2-1}$$

约束：

$$\sum_{k=0,k\neq j}^{n} X_{kj}=1,\forall j \tag{2-2}$$

$$\sum_{j=0,k\neq j}^{n} X_{kj}=1,\forall k \tag{2-3}$$

$$\sum_{j=1}^{n} X_{0,j}=f \tag{2-4}$$

$$\sum_{k=0}^{n} X_{k0} = f \tag{2-5}$$

$$X_{kj} + X_{jk} \le 1, \forall k, j, k = j \tag{2-6}$$

$$C_{ij} \ge C_{i-1j} + p_{ij}, \forall i, j \tag{2-7}$$

$$C_{ij} \ge C_{ik} + p_{ij} + (X_{kj} - 1) \cdot M, \forall i, j, k, j \ne k \tag{2-8}$$

$$C_{\max} \ge C_{mj}, \forall j \tag{2-9}$$

约束(2-2)表示对于任意工件 j ($j = 0,...,n$)，只存在一个工件 k ($k = 0,...,n$)作为工件 j 的前置加工工件。约束(2-3)表示，对于任意工件 k ($k = 0,...,n$),有且只有一个工件 j ($j = 0,...,n$)为工件 k 的后置加工工件。约束条件(2-4)和(2-5)限制每个工厂至少存在一个工件进行加工，即存在 $n \ge f$ 。约束(2-6)限制了任何工件不能同时作为彼此的前置加工工件和后置加工工件。约束(2-2)至(2-6)为 MSBM 的 BV 约束条件。MSBM 模型简单，BV 数量少，因此最适合描述 DPFSP。MSBM 只有 X_{jk} 一个 BV。 X_{jk} 变量的空间大小为 $n(n+1)$ 。

约束(2-7)限制该工件的完成时间必须大于该工件前一个工序的完成时间加上该工序所需的处理时间。约束(2-8)限制工件的完成时间必须大于前置加工工件的完成时间加上该工件工序所需的处理时间。约束(2-9)表示最大完成时间必须大于任意工件最后一道工序的完成时间。约束条件(2-1)是本文的研究目标，即最小化最大完成时间。MSBM 只有 C_{ij} 一个 CV，其大小为 nm 。

为了更好地理解 DPFSP 生产模式和 MSBM 模型，下面给出一个示例。假设存在一个 DPFSP，该问题工件数量 $n = 12$ ，工厂数量 $f = 2$ ，机器数量 $m = 2$ ，每个工件的加工时间如表 2-1 所示。

假设 X_{jk} 已经给定，即 $X_{0,10} = X_{10,9} = X_{9,7} = X_{7,2} = X_{2,11} = X_{11,5} = X_{5,4} = X_{4,0} = X_{0,3} = X_{3,6} = X_{6,8} = X_{8,12} = X_{12,1} = X_{1,0} = 1$。其余均为 0。在这种情况下，工件 10、9、7、2、11、5、4 分配给同一个工厂，剩下的部分分配给另一个工厂。再根据 BV 条件，可以得到图 2-1 的 DPFSP 案例甘特图。

表 2-2 DPFSP 案例中工序的加工时间

	工件											
机器	1	2	3	4	5	6	7	8	9	10	11	12
1	1	4	4	4	5	2	3	6	3	2	1	6
2	6	3	2	1	2	4	1	3	7	9	3	1

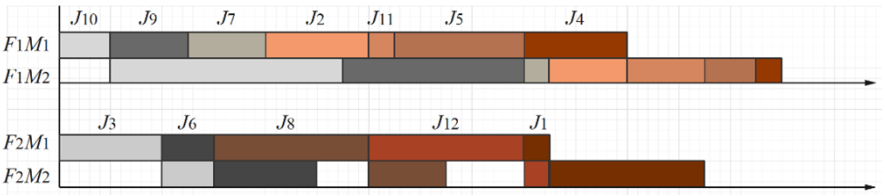

图 2-1 DPFSP 案例甘特图

2.2 改进的生物地理学优化算法

2.2.1 初始化与编码方式

BBO 作为群智能算法需要生成大量的初始栖息地作为算法的初始解，通常采用随机初始化的方式生成初始问题解。然而，随机初始解方式生成的初始解中必定包含的许多冗余解，这些冗余解会使算法的进化速度变得迟缓。因此本文在随机初始化方法的基础上添加了两种工厂分配规则 NR1 和 NR2[5]用来提高初始个体的质量。NR1 和 NR2 的描述如下：

NR1：将工件 j 分配到当前完工时间最小的工厂 f 中完成加工。

NR2：将工件 j 分配到包含工件 j 后完工时间最小的工厂 f 中完成加工。

建立一个一维空数组 Π，将 n 个工件 j 不重复的随机填充到 Π 的末尾端。依次选择 Π 的末尾工件 j，利用 NR1/NR2 将工件 j 填充到二维数组 Π_f 第 f 维的末尾端。直到所有工件被选择，停止填充并返回一个初始解。重复此操作直到满足算法所需初始解的个数。

2.2.2 迁移算子

假设存在两个栖息地，根据迁入率和迁出率公式将这两个栖息地定义为迁出栖息地 $\Pi^E(\Lambda)$ 和迁入栖息地 $\Pi^I(\Lambda)$，其中 Λ 表示的是一个解决方案。

$\Pi^E(\Lambda)$=[{Π_1},{Π_2}]，$\Pi^I(\Lambda)$=[{Π_1},{Π_2}]。$\Pi^E(\Lambda)$和$\Pi^I(\Lambda)$其具体的工件分布如图 2-2 所示。

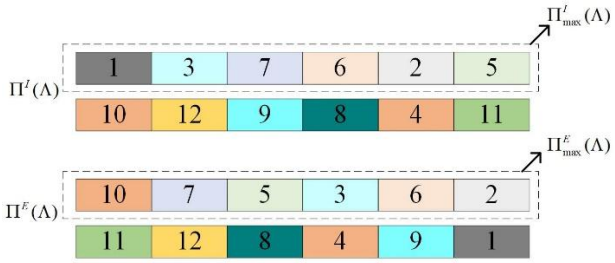

图 2-2 迁入与迁出栖息地

$\Pi^I_{\max}(\Lambda)$与$\Pi^E_{\max}(\Lambda)$分别表示迁入栖息地与迁出栖息地中具有最大完工时间的工厂中工件的加工顺序。将迁出栖息地中部分编码信息迁移到迁入栖息地中以实现优秀编码段的传递。以图 2-2 中的数据为例，具体的迁移操作如下。

操作 1：将工件$\Pi^E_{\max}(\Lambda)-(\Pi^I_{\max}(\Lambda)\cap\Pi^E_{\max}(\Lambda))$保存至$\Pi^E_{save}$中，$\Pi^E_{save}=\{10\}$。如图 2-3 所示。

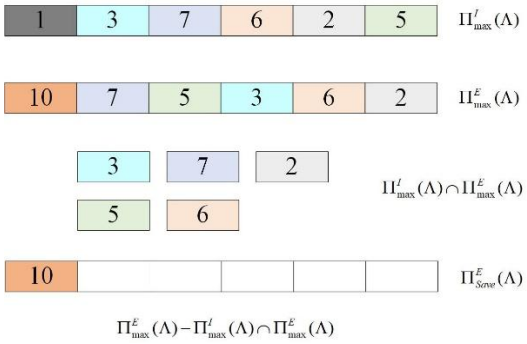

图 2-3 BBO 迁移操作第 1 步

操作 2：将工件$\Pi^I_{\max}(\Lambda)-(\Pi^I_{\max}(\Lambda)\cap\Pi^E_{\max}(\Lambda))$保存至$\Pi^I_{save}$中，$\Pi^I_{save}=\{1\}$。如图 2-4 所示。

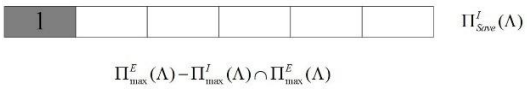

图 2-4 BBO 迁移操作第 2 步

操作 3：删除$\Pi^I(\Lambda)$中Π^E_{save}工件，无论这个工件在什么位置。此时$\Pi^I(\Lambda)$发生变化，如图 2-5 所示。

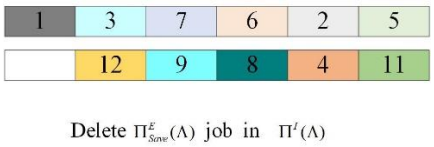

图 2-5 BBO 迁移操作第 3 步

操作 4：将$\Pi^{I}(\Lambda)$中$\Pi^{I}_{\max}(\Lambda)$工件序列用$\Pi^{E}_{\max}(\Lambda)$替代，替代后的$\Pi^{I}(\Lambda)$如图 2-6 所示。

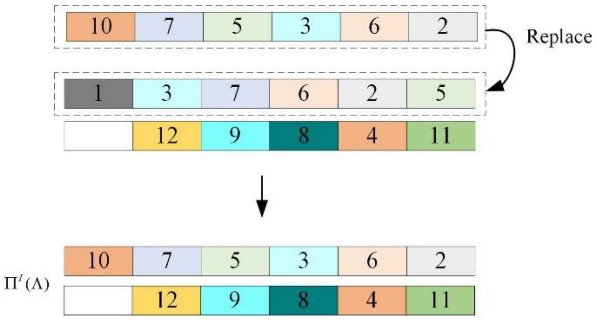

图 2-6 BBO 迁移操作第 4 步

操作 5：从左到右依次在Π^{I}_{save}中选择工件，将被选择的工件插入到除了具有最大完工时间工厂Π_{f}的任意一个工厂中并满足所插入的位置是最佳位置。插入工件后的$\Pi^{I}(\Lambda)$变化如图 2-7 所示。

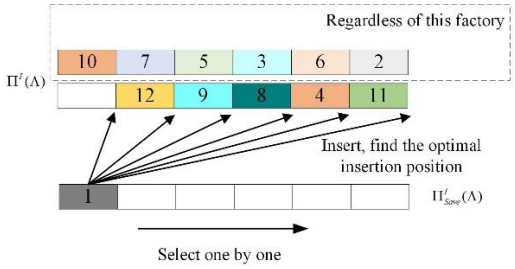

图 2-7 BBO 迁移操作第 5 步

根据以上操作最终可以得到更新后的迁入栖息地，此时的迁入栖息地不仅获得了迁出栖息地中部分优秀的编码段并且还通过插入搜索的方式提高了解的质量。

2.2.3 突变算子

设计了四种突变算子以满足算法在进化过程中应对陷入局部最优等不良情况。四种突变算子的详细操作描述如下。

$LS_{insert}(MF)$：如图 2-8 所示，计算具有最大完工时间工厂Π_{f}中的每一个工

件所需工序时间和，将计算结果按照从小打大进行排序并依次选择该工件插入到该工厂的其余位置中，一旦该工厂的完工时间减少则停止插入搜索否则选择其余工件直到遍历所有工件。

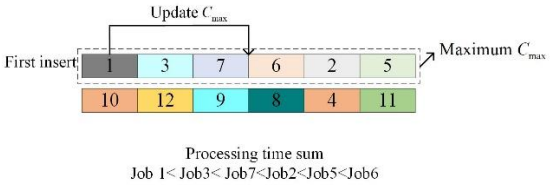

图 2-8 $LS_{insert}(MF)$ 突变过程

$LS_{swap}(MF)$：如图 2-9 所示，从左到右依次选择具有最大完工时间工厂 Π_f 中的每一个工件，将该工件与该工厂其余工件交换位置，一旦该工厂的完工时间减少则停止交换搜索否则选择其余工件直到遍历所有工件。

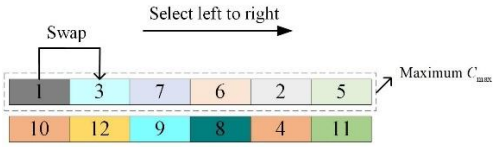

图 2-9 $LS_{swap}(MF)$ 突变过程

$LS_{insert}(OF)$：如图 2-10 所示，计算具有最大完工时间工厂 Π_f 中的每一个工件所需工序时间和，将计算结果按照从小打大进行排序并依次选择该工件随机插入到其余工厂中，一旦该工厂的完工时间减少则停止插入搜索否则选择其余工件直到遍历所有工件。

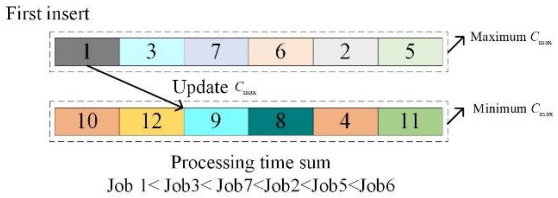

图 2-10 $LS_{insert}(OF)$ 突变过程

$LS_{swap}(OF)$：如图 2-11 所示，从左到右依次选择具有最大完工时间工厂 Π_f 中的每一个工件，将该工件与其余工厂的工件交换位置，一旦该工厂的完工时间减少则停止插入搜索否则选择其余工件直到遍历所有工件。

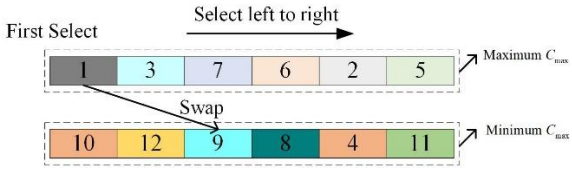

图 2-11 $LS_{swap}(OF)$ 突变过程

2.2.4 算法流程

图 2-12 展示了 BBO 算法的流程图。算法首先对栖息地进行初始化，通过结合两种工厂分配方式的初始化方法，有效减少了随机初始化时出现冗余解的概率，从而提升了算法在早期阶段的运行效率。初始化完成后，对第一批栖息地进行适应度计算，以确保每代更新时都能获得准确的适应度值。在迁移阶段，算法随机选择栖息地并计算其迁入率和迁出率，从而确定需要执行迁移操作的栖息地。迁移操作完成后，再随机选择一个栖息地并计算其突变概率，并将该突变概率与随机概率进行比较：若突变概率大于随机概率，则执行突变操作；否则，直接判断是否满足终止条件。本章设定的终止条件包括最大迭代次数和 CPU 运行时间。若在规定的 CPU 运行时间内达到最大迭代次数，则输出最后一代的最优解；否则，输出 CPU 运行时间结束时当前代的最优解。关于 CPU 运行时间的具体设置将在 2.3 节中详细介绍。

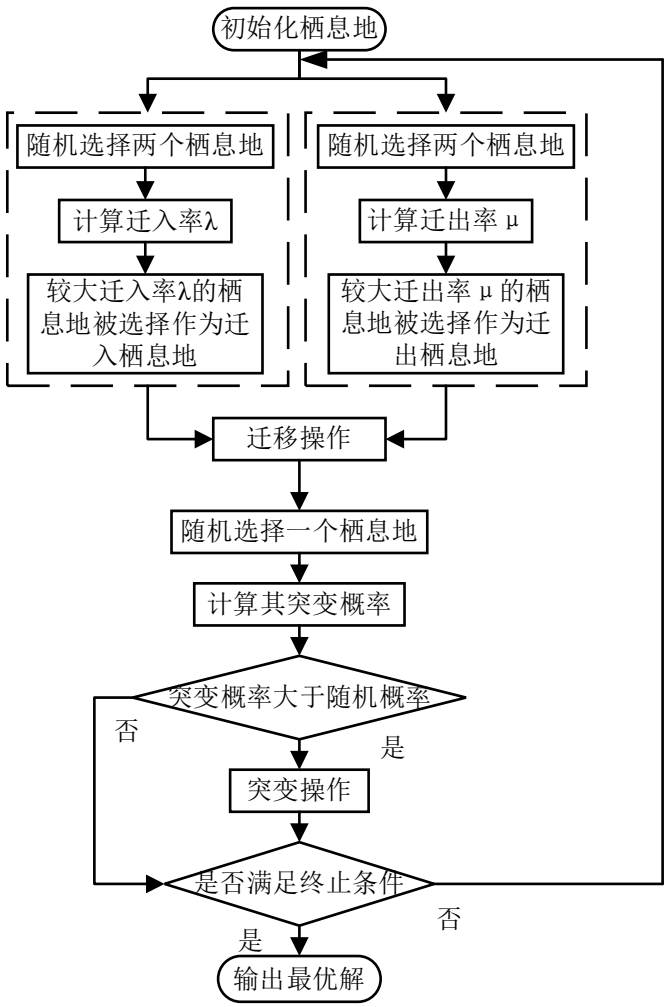

图 2-12 BBO 算法流程

2.3 仿真实验

2.3.1 测试案例与对比准则

BBO 与对比算法的运行环境为 32.0 RAM，3.7GHz CPU 的笔记本，算法测试所使用的仿真程序为 Microsoft Visual C++2022。本节所采用的 DPFSP 测试案例的最优解以及案例数据均可在 http://soa.iti.es 上查询到。该 DPFSP 小型案例的规模为 $n=\{4,6,8,10,12,14,16\}$, $m=\{2,3,4,5\}$, $f=\{2,3,4\}$，每种组合随机生成 5 个案例共有 420 个小型案例。DPFSP 大型案例的规模为 $n=\{20,50,100,200,500\}$，$m=\{5,10,20\}$，$f=\{2,3,4,5,6,7\}$，每种组合随机生成 8 个案例共有 720 个大型案例。工件的加工时间设置在 $[1,99]$，该时间设置区间在调度中较为常用。每个

案例的运行时间控制在 $n\times f\times 0.2$ 秒内。

采用相对百分比差异(Relative Percentage Difference, RPD)来评价算法运行的效果。RPD 的计算公式如 2-10 所示。

$$RPD=\frac{H_{ALG}-H_{OPT}}{H_{OPT}}\times 100 \tag{2-10}$$

其中 H_{ALG} 表示算法在运行过程中出现的最优解，H_{OPT} 为基准案例已知最优解。如果 RPD 为负值代表算法刷新了目前最优解。

2.3.2 参数实验

为了确保 BBO 算法的有效性，采用田口参数实验方法来为 BBO 选择最佳的参数组合。BBO 算法参数包含栖息地数量 N、最大异变率 $M_{\max}$ 以及算法最大迭代次数 $Iter_{\max}$。选择 DPFSP 小型案例进行测试，每个案例运行 5 次保留其平均 RPD 值。表 2-3 展示了不同精度下的 BBO 算法参数，图 2-13 为正交参数实验参数的 PRD 均值主效应图。根据图 2-10 可知，当 N，$M_{\max}$ 和 $Iter_{\max}$ 参数精度为 3，3，4 时 BBO 具有最佳的搜索能力，因此选择该参数组合作为 BBO 的算法参数。

表 2-3 BBO 算法参数与精度等级

	精度等级			
参数	1	2	3	4
N	50	100	150	200
$M_{\max}$	0.05	0.10	0.15	0.2
$Iter_{\max}$	50	100	150	200

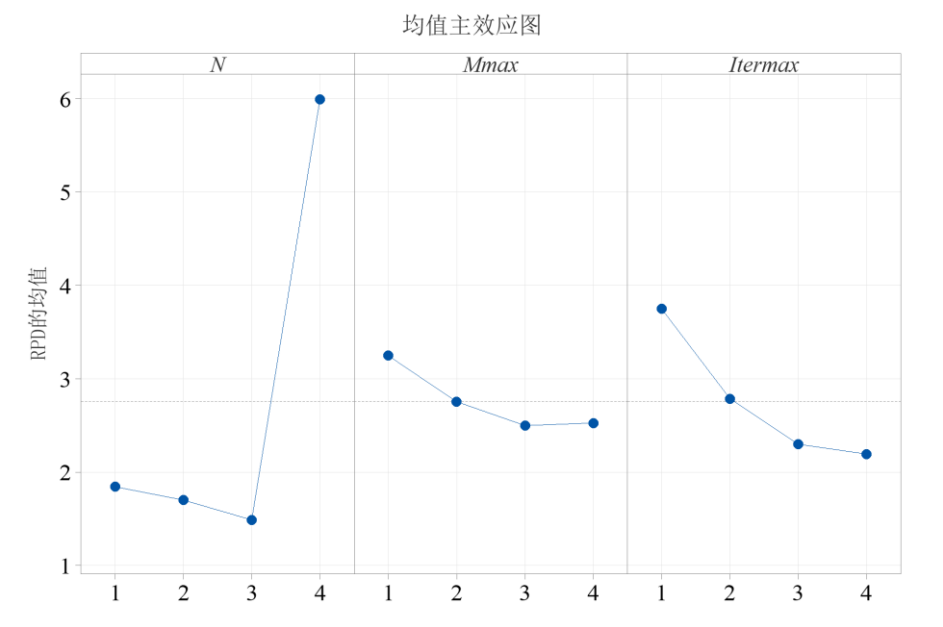

图 2-13 BBO 参数实验的均值主效应图

2.3.3 对比实验

为了验证 BBO 算法在求解 DPFSP 的优越性，本节选择了 12 种启发式算法与 2 种变邻域搜索算法[5]在 DPFSP 小型案例中与 BBO 算法进行仿真比较。在 DPFSP 大型案例中，除了以上对比算法外还选择另外 2 种改进的启发式算法[51]与 5 种群智能进化算法[52]进行比较。DPFSP 小规模与大规模案例仿真结果如下表所示。表 2-4 为 BBO 算法与 12 种启发式算法和变邻域搜索算法在小型 DPFSP 案例中的仿真结果，表 2-5 与表 2-6 为 BBO 算法与启发式算法和智能进化算法在 DPFSP 大型案例中的仿真结果。其中部分数据加粗表示该算法在此案例下的 RPD 值小于或等于 0。

表 2-4 BBO 算法与启发式算法在小型 DPFSP 案例中的仿真结果

$F\times n$	SPT1	LPT1	Johnson1	CDS1	Palmer1	NEH1	VND(a)	SPT2	LPT2	Johnson2	CDS2	Palmer2	NEH2	VND(b)	BBO
2×4	12.95	19.31	6.13	2.69	7.75	2.61	**0.00**	10.02	17.80	3.76	0.72	5.22	0.15	0.15	**0.00**
2×6	13.30	29.53	10.34	7.56	7.57	4.42	1.26	10.38	28.37	8.08	4.60	4.83	1.44	1.44	**0.00**
2×8	17.31	35.43	9.75	12.33	9.97	4.43	2.10	16.16	33.09	7.98	10.33	8.64	2.53	2.52	**0.00**
2×10	21.03	36.93	8.38	9.72	10.18	4.28	3.22	17.49	37.12	7.52	8.38	9.25	3.27	2.89	**0.00**
2×12	20.40	39.36	10.14	11.81	10.77	6.95	3.38	17.23	37.41	8.29	10.49	10.32	4.13	3.90	**0.00**
2×14	17.73	40.00	10.27	6.59	10.57	6.05	2.70	17.34	38.36	9.21	5.24	9.24	3.16	2.82	0.01
2×16	17.11	42.42	12.33	10.53	10.97	6.66	2.92	16.95	41.63	10.33	8.26	9.00	3.84	3.30	**0.00**

3×4	8.87	6.42	5.31	5.01	4.82	0.43	**0.00**	5.36	5.41	1.33	2.55	1.50	0.43	0.00	**0.00**
3×6	21.24	27.63	6.91	8.50	9.40	4.04	0.68	11.93	25.26	5.28	6.99	3.40	1.39	1.39	**0.00**
3×8	17.71	26.87	10.14	9.17	11.84	5.08	1.86	17.93	25.14	8.30	8.67	6.98	2.43	2.43	**0.00**
3×10	24.48	37.98	12.67	13.56	13.97	8.42	2.59	16.23	35.43	10.07	10.81	11.67	3.79	3.63	**0.00**
3×12	25.92	42.12	14.90	14.66	14.39	7.66	3.66	19.55	40.14	10.12	11.00	11.36	5.08	5.08	**0.00**
3×14	23.44	41.00	14.62	16.45	17.97	10.54	4.48	19.95	38.56	14.04	13.47	14.05	4.90	4.46	**0.00**
3×16	25.31	41.71	14.39	15.55	15.68	8.18	3.50	22.83	39.39	10.41	11.30	11.06	3.98	3.81	**-0.19**
4×4	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
4×6	13.34	14.52	7.65	4.99	6.02	3.10	**0.00**	11.31	12.52	5.62	3.39	2.98	0.47	**0.00**	**0.00**
4×8	15.21	16.86	9.20	9.23	10.17	4.19	0.77	13.89	16.44	5.56	8.61	7.04	0.77	0.77	**0.00**
4×10	22.65	34.47	13.19	13.90	16.21	7.07	1.57	16.84	30.64	8.93	9.30	7.23	2.30	2.22	**0.00**
4×12	25.51	38.72	13.61	18.13	15.49	8.58	4.23	20.54	34.06	8.68	14.49	12.39	4.97	4.68	**0.00**
4×14	26.61	42.89	13.50	15.47	16.53	10.94	4.25	22.31	39.79	8.51	11.23	12.25	4.54	4.46	**0.00**
4×16	28.70	44.01	19.12	17.62	19.34	9.79	5.08	14.09	40.67	13.96	13.69	15.84	5.59	5.55	**-0.14**
Average	18.99	34.34	10.60	10.64	11.41	5.88	2.30	15.63	29.39	7.97	8.28	8.30	2.82	2.64	**-0.02**

表 2-5 BBO 算法与启发式算法在大型 DPFSP 案例中的仿真结果

F	SPT1	LPT1	Johnson1	CDS1	Palmer1	NEH1	VND(a)	SPT2	LPT2	Johnson2	CDS2	Palmer2	NEH2	VND(b)	BBO
2	18.71	33.70	12.68	9.29	10.44	2.92	0.10	17.71	32.36	11.58	8.52	9.34	1.21	0.32	**-1.67**
3	19.95	33.05	13.40	10.83	11.72	3.42	0.10	17.58	31.7	11.18	8.92	9.43	1.15	0.35	**-1.42**
4	20.07	33.30	13.48	11.19	11.95	4.21	0.06	17.23	30.42	10.67	8.54	8.90	1.11	0.46	**-1.09**
5	20.04	32.89	13.18	11.29	12.24	4.28	0.11	16.73	29.57	10.24	8.07	8.76	0.92	0.46	**-0.73**
6	20.32	32.58	13.57	11.50	12.70	4.73	0.11	16.07	28.55	9.94	7.83	8.45	0.95	0.51	**-0.21**
7	21.04	32.02	13.61	11.49	12.53	4.86	0.10	15.42	27.14	9.71	7.31	8.21	0.81	0.45	0.24
Average	20.02	32.92	13.35	10.93	11.93	4.07	0.10	16.79	29.93	10.55	8.20	8.85	1.03	0.43	**-0.85**

表 2-6 BBO 算法与群智能进化算法在大型 DPFSP 案例中的仿真结果

F	NEH-d	NEH-f	NEH-df	EM	HGA	IG	SS	TS	BBO
2	1.18	0.95	1.03	4.65	2.49	2.10	0.70	1.42	**-1.67**
3	1.05	0.94	0.88	4.84	2.90	2.34	0.69	1.96	**-1.42**
4	1.01	0.89	0.69	5.11	3.18	2.34	0.90	2.62	**-1.09**
5	0.89	0.86	0.83	5.32	3.52	2.22	1.14	3.28	**-0.73**
6	0.85	1.05	0.95	5.48	3.80	1.94	1.57	3.89	**-0.21**
7	0.60	0.90	0.68	6.09	4.67	1.99	2.14	4.92	0.24
Average	0.93	0.93	0.84	5.25	3.43	2.16	1.19	3.02	**-0.85**

经过大量的仿真实验，得到了 BBO 算法与其他智能算法的仿真结果。通过表 2-4，2-5 和 2-6 可以得出结论：不管是小型还是大型 DPFSP 案例 BBO 算法具

有显著的搜索能力优势，且无论在什么规模下 BBO 算法几乎都可以找到并更新最优解，造成这一现象的原因是算法在初期提供了一批适应度较好的初始种群加快了算法的搜索速度，同时较少的算法参数使得 BBO 算法的空间复杂度降低从而间接提高了算法的搜索效率。2.2.2 和 2.2.3 节所设计的迁移算子与突变算子帮助算法摆脱了陷入局部最优的劣势。表 2-7 为 BBO 算法在小型 DPFSP 基准案例中更新的最优解。

表 2-7 BBO 算法在小型案例中得到的最优解

案例名称	*F*	*n*	*m*	已知最优	BBO	RPD
I_2_16_5_1	2	16	5	526	523	-0.57
I_2_16_5_3	2	16	5	652	650	-0.31
I_3_16_3_1	3	16	3	340	339	-0.29
I_3_16_3_3	3	16	3	350	349	-0.29
I_3_16_3_5	3	16	3	362	361	-0.28
I_3_16_4_2	3	16	4	458	457	-0.22
I_3_16_4_4	3	16	4	430	429	-0.23
I_3_16_4_5	3	16	4	419	414	-1.19
I_3_16_5_1	3	16	5	453	451	-0.44
I_3_16_5_2	3	16	5	500	499	-0.20
I_3_16_5_3	3	16	5	476	475	-0.21
I_3_16_5_4	3	16	5	524	522	-0.38
I_4_16_3_3	4	16	3	312	311	-0.32
I_4_16_4_1	4	16	4	323	319	-1.24
I_4_16_4_2	4	16	4	359	358	-0.28
I_4_16_5_3	4	16	5	365	363	-0.55
I_4_16_5_4	4	16	5	447	441	-1.34

2.4 本章小结

本章主要研究了 BBO 算法解决以最小化完工时间为目标的 DPFSP，设计了一种初始化方式，通过一维随机编码和工厂分配方法提供了一批冗余解较少的初始栖息地。构建的迁移算子与四种突变算子作为算法的全局与局部搜索方式帮助算法跳出局部最优以此寻得更优解。田口参数实验调整了 BBO 的算法参数，大量的仿真实验证明 BBO 算法的强搜索能力。

第 3 章 分布式装配置换流水车间调度问题研究

近年来，分布式装配置换流水车间调度问题在分布式调度问题领域受到了广泛的关注。本章研究了最小化最大完工时间为目标的 DAPFSP。为了解决这一问题，提出了一种改进的基于生物地理的优化算法(Modified Biogeography-Based Optimization, MBBO)并建立了 DAPFSP 的数学模型。为了减少初始化阶段冗余解出现的概率，根据问题的特点设计了一种启发式方法来生成初始解。提出了一种改进的路径重联技术和一种优化的突变方法作为 MBBO 的全局和局部搜索阶段。设计了三种基于关键路径的邻域搜索算子进行深度搜索。采用泰勒加速度方法提高 MBBO 的搜索速率。田口实验作为一种算法参数实验，为 MBBO 选择最佳的参数组合。通过消融实验验证了邻域搜索算子的有效性。将 MBBO 算法与其他常用算法进行比较，在 900 个 DAPFSP 实例中发现了 53 个已知的最优解。

3.1 分布式装配置换流水车间调度问题

3.1.1 问题描述

图 3-1 为 DAPFSP 的原理图。DAPFSP 包括两个子阶段，即工件的生产阶段和产品的装配阶段。在生产阶段中，工厂需要加工若干个工件。每个工厂都可以看作是一个置换流车间调度问题。在工厂完成对所有工件的加工后，将这些工件送到装配线进行产品的组装。无论工件分配到哪个工厂，所有工件都具有相同的加工路径。DAPFSP 的装配线只配备一台装配机器。

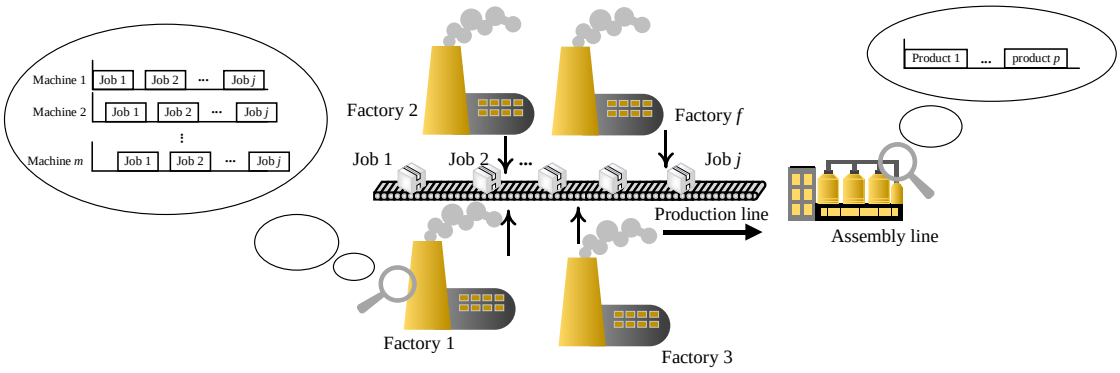

图 3-1 DAPFSP 的原理图

3.1.2 模型建立

DAPFSP 可以描述如下。在生产阶段，工件 j($j=1,...,n$)需要被分配到工厂 g($g=1,...,f$)中进行加工。每一个工件只能被分配到一个工厂中去。所有工厂都有相同的生产机器 i ($i=1,...,m$)。所有工件 j 都需要在机器 i 上进行加工。在同一加工路线上处理的所有作业的操作，即首先在机器 1 上，然后在机器 2 上，以此类推，直到机器 i，这类加工类型被称为流水加工。工件在 j 机器 i 上的加工时间为 pp_{ij}($pp_{ij}>0$)。一旦工件被处理，它就不能被中断。在装配阶段，完成所有工序的工件需要装配成一个产品。每一个工件只属于一个产品 t ($t=1,...,p$)，每一个产品至少包含一个工件，属于产品 t 的工件数量为 Ψ_t，$n=\sum_{t=1}^{p}\Psi_t$。只有当产品 t 中包含的所有工件都被加工完成后，工件才能被发送到装配线进行产品组装。产品 t 的装配时间为 pa_t($pa_t>0$)。产品一旦开始装配，它就不能被中断。

在生产调度领域，混合整数线性规划模型常用于表示调度问题。MILP 通过设置一些约束和变量来确定解的范围，然后通过求解器的连续搜索得到精确解。基于最小顺序序列的 MILP 模型被用来表达 DAPFSP[9]。在提出 DAPFSP 的 MILP 模型之前，需要设计和修改一些参数和变量。

j,k($j,k=0,...,n$)表示工件，其中 0 表示虚拟工件。虚拟工件的功能是表示每个工厂中第一个和最后一个加工的工件。 t,s ($t,s=0,...,p$)表示产品，其中 0 表示虚拟产品。虚拟产品的功能是表示装配线上第一个和最后一个组装的产品。G_{tj} 是已知的二进制变量。 当工件 j 属于产品 t 时，G_{tj} 等于 1。X_{jk} 是一个二进制未知变量。当工件 j 在工件 k 之前加工时， X_{jk} 等于 1（工件 j 和工件 k 是连续加工的）。Y_{ts} 是一个二进制未知变量。当产品 t 在产品 s 之前组装时，Y_{ts} 等于 1（产品 t 和产品 s 是连续组装的）。C_{ij} 是一个未知的连续变量。C_{ij} 表示工件 j 在机器 i 上的完成时间。C_t 是一个未知的连续变量。C_t 表示产品在装配线上组装完成的时间。 M 是一个足够大的正整数。$C_{\max}$ 是 makespan。

最小化最大完工时间为目标的 DAPFSP 目标函数如下。

$$Minimise C_{\max} = \max_{t=1,2,...,p} \{C_t\} \tag{3-1}$$

约束：

$$\sum_{k=0,k\neq j}^{n} X_{kj} = 1, \forall j, j \neq 0 \tag{3-2}$$

$$\sum_{j=0,j\neq k}^{n} X_{kj} = 1, \forall k, k \neq 0 \tag{3-3}$$

$$\sum_{j=1}^{n} X_{0j} = f \tag{3-4}$$

$$\sum_{k=1}^{n} X_{k0} = f \tag{3-5}$$

$$X_{kj} + X_{jk} \leq 1, \forall k, j, k \neq 0, j \neq 0, k \neq j \tag{3-6}$$

$$\sum_{t=0,t\neq s}^{p} Y_{ts} = 1, \forall s, s \neq 0 \tag{3-7}$$

$$\sum_{s=0,s\neq t}^{p} Y_{ts} = 1, \forall t, t \neq 0 \tag{3-8}$$

$$Y_{ts} + Y_{st} \leq 1, \forall t, s, t \neq s \tag{3-9}$$

$$C_{1j} \geq pp_{1j}, \forall j, j \neq 0 \tag{3-10}$$

$$C_{1j} \geq pp_{1j} + C_{1k} + (X_{kj} - 1) \cdot M, \forall j, k, j \neq 0, k \neq 0, j \neq k \tag{3-11}$$

$$C_{ij} \geq C_{i-1j} + pp_{ij}, \forall i, j, i \neq 1, j \neq 0 \tag{3-12}$$

$$C_{ij} \geq pp_{ij} + C_{ik} + (X_{kj} - 1) \cdot M, \forall i, j, k, i \neq 1, j \neq 0, k \neq 0, j \neq k \tag{3-13}$$

$$C_t \geq C_{mj} \cdot G_{tj} + pa_t, \forall j, s, j \neq 0, s \neq 0 \tag{3-14}$$

$$C_t \geq C_s + pa_t + (Y_{st} - 1) \cdot M, \forall t, s \neq 0, t \neq 0, t \neq s \tag{3-15}$$

公式(3-1)是该问题的目标函数。约束(3-2)限制每个工件必须有且仅有一个前置加工工件。约束(3-3)限制每个工件必须有且仅有一个后置加工工件。约束(3-4)至(3-5)表明每个工厂必须加工至少一个工件，并且每个工厂的第一个或最后一个工件必须是虚拟工件。约束(3-6)限制工件不能同时是另一个工件的前置和后置。约束(3-7)限制每个产品必须有且仅有一个前置装配产品。约束(3-8)限制每个产品必须有且仅有一个后置装配产品。约束(3-9)限制产品不能同时是另一个产品的前置和后置。约束(3-10)至(3-11)计算每个工件在首个机器上的完成时间。约束(3-12)至(3-13)计算每个工件在机器 i 上的完成时间。约束(3-14)至(3-15)计算每个产品

的组装时间。

为了更好地理解 MILP 模型如何表示 DAPFSP 调度方案，列举了一个 DAPFSP 的调度案例。该案例的加工数据如表 3-1 所示。该案例由 8 个工件、2 台机器、2 个工厂和 2 个产品组成。工件 1、2、5、8 属于产品 1，工件 3、4、6、7 属于产品 2。假设在该案例中，$X_{0,1}=X_{1,3}=X_{3,6}=X_{6,4}=X_{4,0}=X_{0,8}=X_{8,5}=X_{5,2}=X_{2,7}=X_{7,0}=1$其余值为 0。该案例的甘特图如图 3-2 所示。

表 3-1 DAPFSP 案例的加工数据

	工件							
机器	1	2	3	4	5	6	7	8
机器 1	1	5	7	9	9	3	8	4
机器 2	3	8	5	7	3	4	1	3
	产品 1				产品 2			
pp_l	6				5			

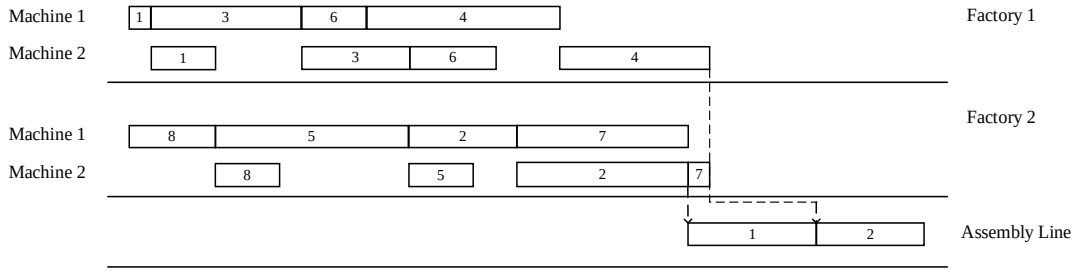

图 3-2 DAPFSP 案例甘特图

3.2 基于深度搜索的生物地理学优化算法

3.2.1 初始化与编码方式

每个栖地 H_r 可以看作是一个 DAPFSP 解。栖息地 H_r 的 HSI 可以看作是 DAPFSP 的目标函数。$\Pi(t)$($\Pi(t)=[\Pi_1(t),...,\Pi_f(t)]$)表示属于产品 t 的工件在工厂 g 中的加工分布情况。$\Pi_g(t)$ 表示属于产品 t 的工件在工厂 g 中的加工顺序。$\Pi_g(t)$ 可能为零。当 $\Pi_g(t)$ 等于零时，产品 t 的工件在不在工厂 g 中加工。$\Pi(\Lambda)$ ($\Pi(\Lambda)=[\Pi(1),...,\Pi(p)]$)表示工件的加工优先级。 $\Pi(\Lambda)$ 中的 $\Pi(t)$ 从右到左优先级逐步提高。这里 Λ 表示一个 DAPFSP 解，即一个栖息地。

提出了一种双种群初始化(Double Habitat Initialization, DHI)方法，该方法在2.2.1 节的初始化基础上进行了一些改动。列举一个 DHI 生成初始栖地的过程示例。DHI 生成初始解过程的示例如图 3-3 所示。假设产品的随机选择顺序为$[2,1,3]$。随机打乱这些产品的工件顺序。$\Pi(1)=[2,5,7]$，$\Pi(2)=[1,3,6]$，$\Pi(3)=[4,8]$。$\Pi(\Lambda)_{NR1}$和$\Pi(\Lambda)_{NR2}$是通过 NR1/NR2 方法得到的。假设只允许交换产品 1 的工件。$\Pi(\Lambda')_{NR1}$和$\Pi(\Lambda')_{NR2}$是通过交换$\Pi(\Lambda)_{NR1}$和$\Pi(\Lambda)_{NR2}$中的$\Pi(1)$后得到的。

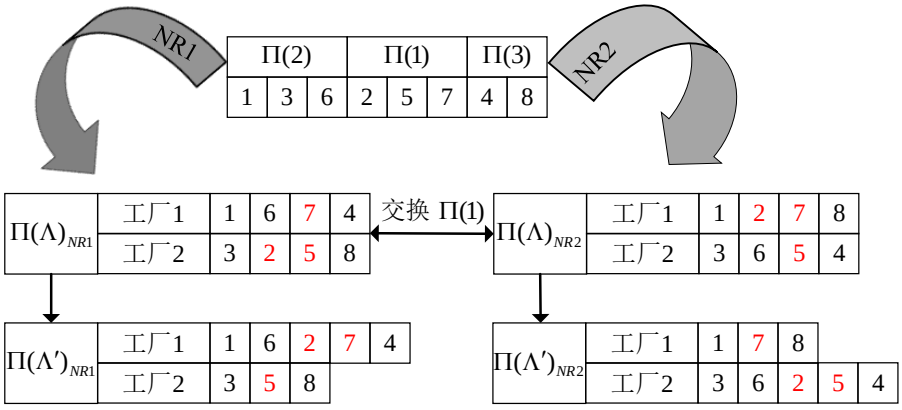

图 3-3 DHI 初始化生成初始解的过程

3.2.2 改进的路径重联

路径重联是一种非常有效的全局搜索方法[53]，通常基于一个解的编码来改进另一个解的质量。Lin[10]首次将路径重联作为 BBO 的迁移模式应用于解决 DAPFSP 问题。Lin 提出的路径重联方法是转移部分编码段。Lin 提出的特殊编码方法中，路径重联需要转移编码段并解码以确定最优位置。如果使用这种方法，可能会导致算法错过一些更好的解。除此之外还发现，在 Lin 设计的路径重链接中，产品是按照固定顺序选择的。例如，如果首先搜索产品 1 的工件且未找到更好的解，则会选择产品 2 的工件进行交换，以此类推。基于上述问题，提出了一种改进的路径重链接方法(Improved Path Relinking, IPR)。由于编码的特殊性，IPR 直接选择已确定的工件加工顺序进行搜索。同时，IPR 不再按固定的产品顺序选择，而是每次随机选择非重复的产品进行转移，从而确保搜索的随机性，增强算法的搜索能力。

IPR 被用作 MBBO 的迁移算子。迁移过程的详细步骤如表 3-2 所示。H_i 和

H_e 根据 λ_r 和 μ_r 被选择。随机选择一个产品 t，在 H_i 中删除 $\Pi(t)$。通过 H_e 的 $\Pi(t)$ 在 H_i 中插入所有可能的位置，确保不重复。直到发现插入后的栖地的 HSI 优于原迁入栖息地 H_i 的 HSI 时，停止操作，并输出更新后的栖息地。

表 3-2 IPR 迁移算子伪代码

```
1: HabitatSize ← N, immigration rate ← λ_ha , emigration rate ← μ_ha
2: for iHabitat = 1 to HabitatSize do
3:     if ( RP < λ_hi )                    %% RP 为随机概率，RP ∈ (0,1]
4:         选择栖息地 iHabitat 作为迁入栖息地
5:     end if
6:     for eHabitat = 1 to HabitatSize do
7:         if ( RP < μ_he )
8:             选择 eHabitat 作为迁出栖息地
9:         end if
10:    IPR (iHabitat, eHabitat)            %% IPR ()为改进的路径重联函数
11:    更新迁入率和迁出率
12: end for
```

3.2.3 优化的突变策略

优化的突变策略是基于原始突变操作所改进的局部搜索策略。在 BBO 的突变操作中，栖息地 H_r 根据其突变概率判断是否允许突变。在 MBBO 中，如果栖地 H_r 不执行突变操作则执行额外的局部搜索策略。该局部搜索策略描述如下：

随机选择一个产品 t，计算属于该产品 t 的所有工件在机器 m 上的完成时间。选择包含该产品的工件中具有最大完成时间的工厂 g 作为目标。将工厂 g 中属于该产品的工件 $\Pi_t(g)$ 复制到一个一维数组 Γ 中。数组 Γ 中的工件从左到右被选中，并将该工件插入到 $\Pi_t(g')$ $(g' \neq g)$。如果插入后栖息地被优化，则停止搜索。否则，不重复选择其他工厂进行搜索。如果在所有工厂选择后栖地没有优化，则直到所有产品都被考虑后，才会继续选择其他产品再次进行搜索。

如果栖息地 H_r 被选择执行突变操作，则随机选择一个产品 t 和一个不包含属于该产品的工件且具有最大完成时间的工厂 g。然后，随机打乱该工厂中 $\Pi_t(g)$

的工件顺序。

3.2.4 关键路径搜索方法

在 MBBO 中，工件在$\Pi(t)$中的顺序不断变化，无论是执行迁移操作还是变异操作。考虑到一种特殊情况，某些工件$\Pi(t)$插入到$\Pi(s)$中可以有效地减少产品的组装时间。基于这一思路，提出了三种邻域结构，分别为$\mathcal{N}_1$,$\mathcal{N}_2$和$\mathcal{N}_3$。这三种搜索结构被称为邻域搜索方法(Neighborhood Search Method, NSM)，用于对栖息地进行深度搜索。NSM 中的三种邻域结构搜索描述如下：

$\mathcal{N}_1$ ：随机选择产品t的关键路径。将关键路径中的最后一个工件与其前置或后置工件进行交换。如果$\mathcal{N}_1$交换后栖息地被优化，则保留该栖息地，否则保持不变。

$\mathcal{N}_2$ ：随机选择产品的关键路径。随机选择关键路径中的末尾工件，将其插入到其他工厂中。如果插入后栖息地被优化，$\mathcal{N}_2$停止操作并保留优化后的栖息地，否则继续选择另一个工厂，直到所有工厂都考虑过。

$\mathcal{N}_3$: ：随机选择产品的关键路径，然后随机选择另一个工厂g。在该工厂g中选择一个连续的工件序列，包含该工厂中所有属于该产品t的工件（即该序列可能包含不属于该产品的工件）。将关键路径中的最后一个工件插入到序列中的所有可能位置。如果插入后栖息地被优化，$\mathcal{N}_3$停止操作并保留优化后的栖息地，否则继续选择另一个工厂，直到所有工厂都考虑过。

图 3-4 至图 3-6 展示了$\mathcal{N}_1$,$\mathcal{N}_2$和$\mathcal{N}_3$的搜索过程，其中包含了案例在每个搜索结构下的析取图以及搜索前后的甘特图。在 NSM 中，产品t的关键路径指的是工件的加工顺序，即从工厂g的第一个加工工件到工件j（属于产品t的工件，g具有最大完成时间的工厂）。以图 3-5(b)中的甘特图为例，假设产品 1 的关键路径为工厂 1 中的工件[3,4,5,1]。图中所述的产品t的最大加工序列指的是一些工件的加工顺序，该序列的第一个和最后一个工件必须属于该产品t。以图 3-6(a)中的析取图为例，[7,6]是最大加工序列。需要注意的是，产品t的最大加工序列允许包含不属于该产品的工件。这种情况由$\mathcal{N}_1$和$\mathcal{N}_2$搜索引起的。这种情

况不仅不会影响 NSM 的操作，且还可能帮助算法找到更优解。

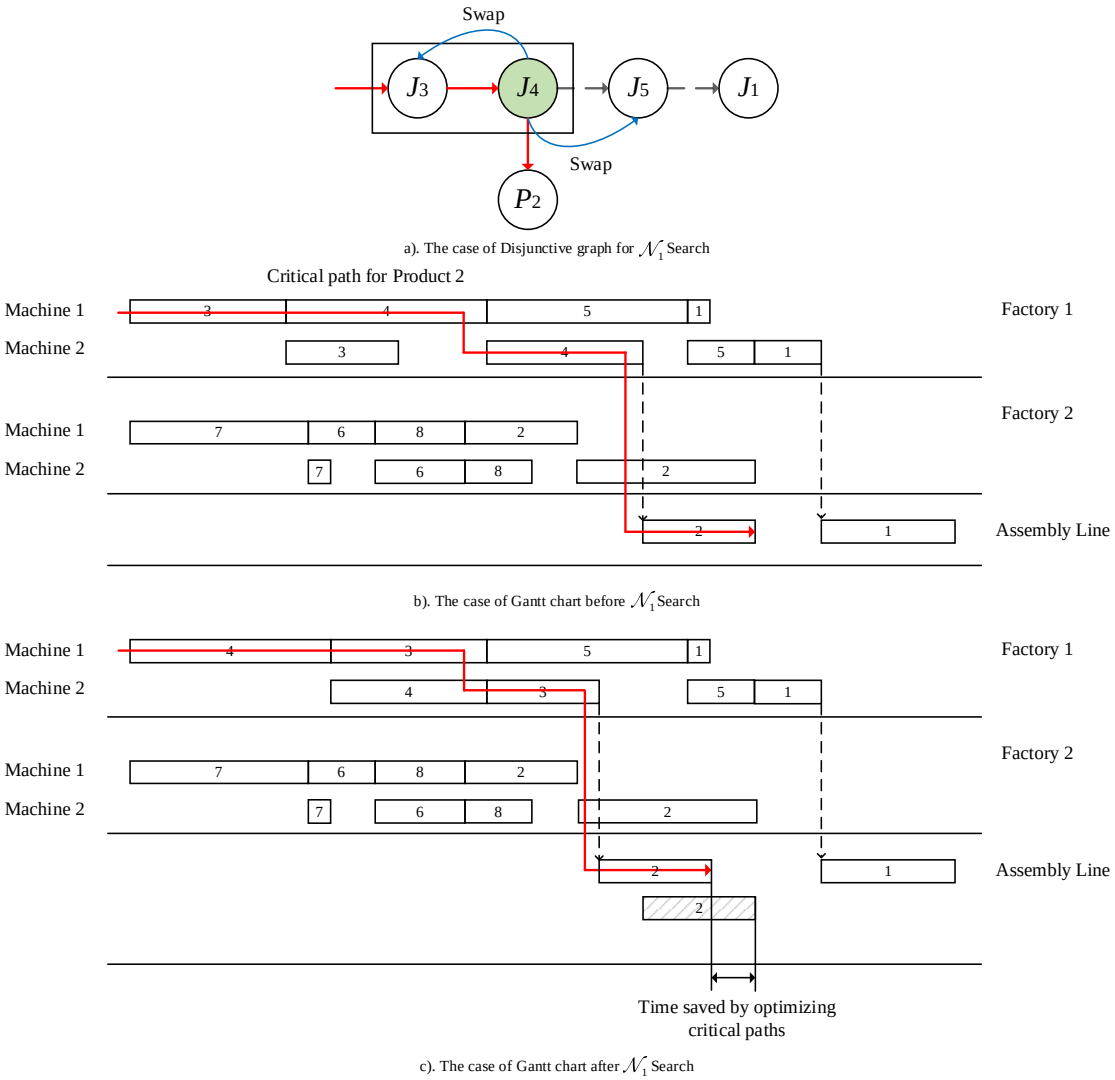

图 3-4 $\mathcal{N}_1$ 邻域搜索过程

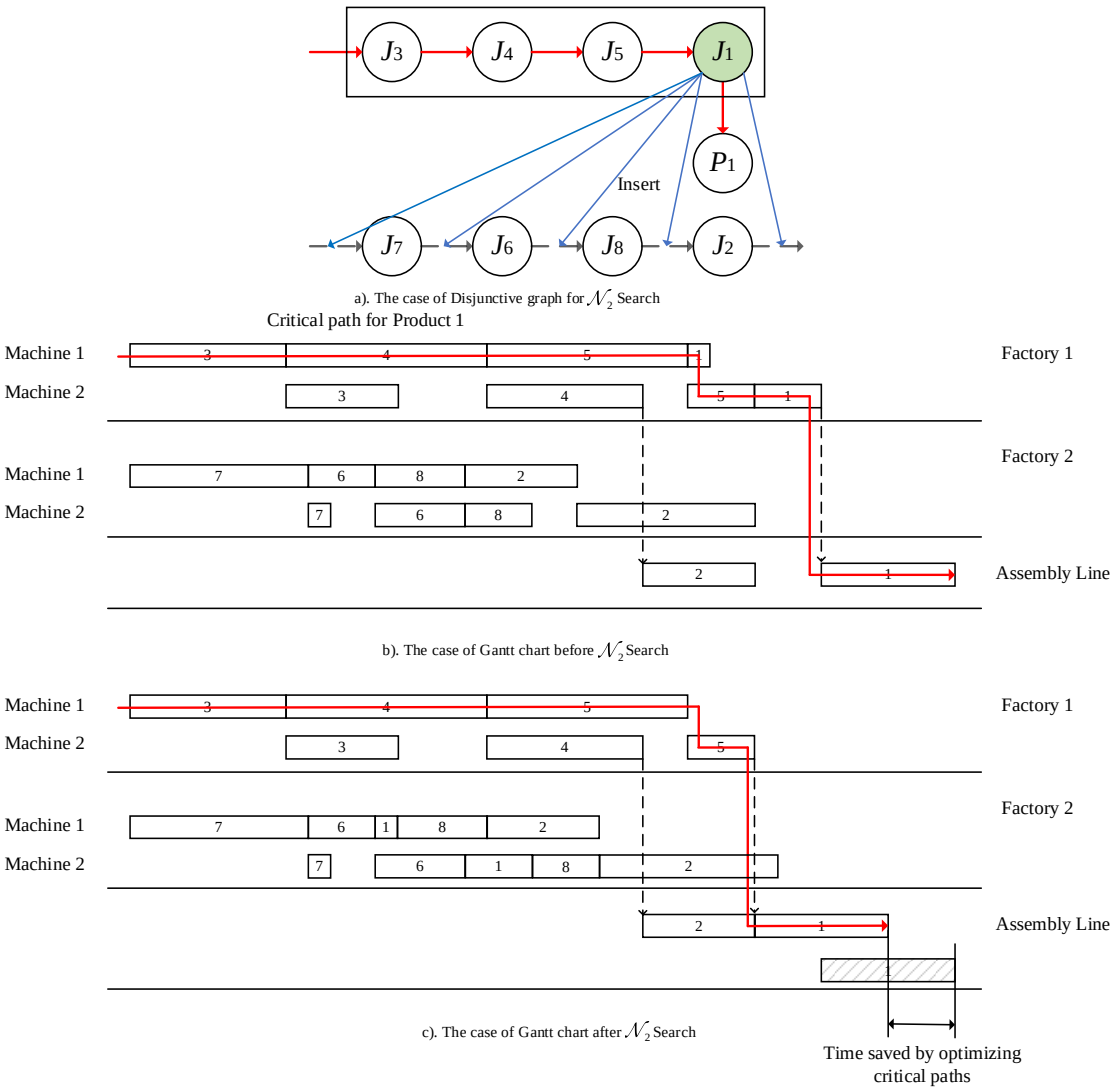

图 3-5 $\mathcal{N}_2$ 邻域搜索过程

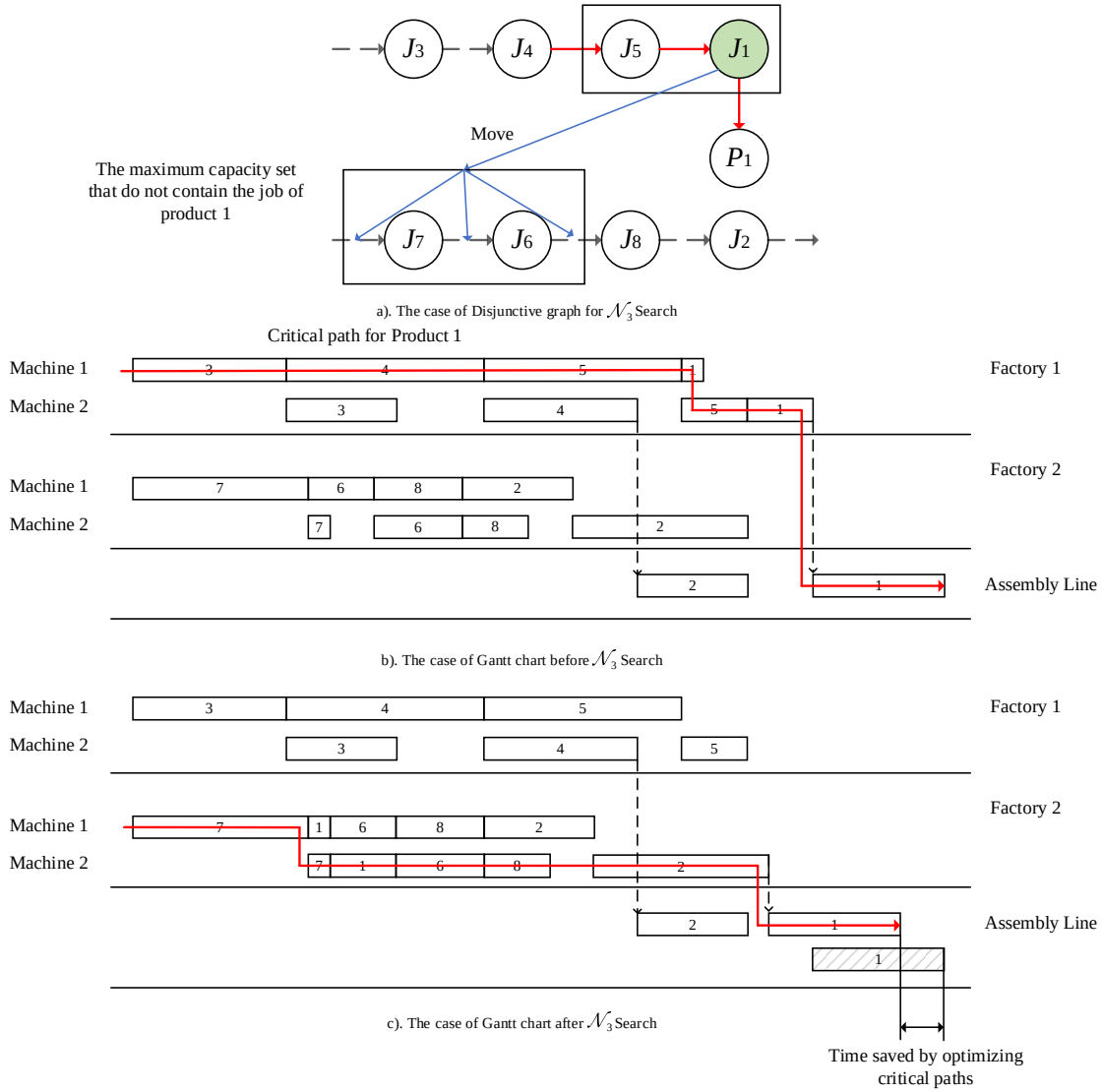

图 3-6 $\mathcal{N}_3$ 邻域搜索过程

3.2.5 泰勒加速度

MBBO 算法涉及许多插入搜索操作，每次工件插入搜索都需要解码以判断操作的有效性。多次解码操作会导致算法进行不必要的重复计算，从而增加算法的运行时间。为了提高算法的速度并避免算法运行过长时间，MBBO 在工件插入阶段参考了泰勒加速方法[54]。变量 $\Omega(u)$ 表示当前工厂第 u 个已处理的工件。$p_{i,\Omega(u)}$ 表示工件 $\Omega(u)$ 在机器 i 上加工所需的时间。假设当前工厂中有 b 个工件。第 u 个工件在机器 i 上的正向完成时间和反向完成时间分别表示为 E_{iu} 和 Q_{iu}。E_{iu} 和 Q_{iu} 的定义如下：

$$E_{1,1} = p_{1,\Omega(1)} \tag{3-16}$$

$$E_{i,1} = p_{1,\Omega(1)} + E_{i-1,1}, i = 2,...,m \tag{3-17}$$

$$E_{1,u} = p_{1,\Omega(u)} + E_{1,u-1}, u = 2,...,b \tag{3-18}$$

$$E_{i,u} = p_{i,\Omega(u)} + \max\{E_{i,u-1}, E_{i-1,u}\}, i = 2,...,m, u = 2,...,b \tag{3-19}$$

$$C_{\max} = E_{b,m} \tag{3-20}$$

$$Q_{m,b} = p_{m,\Omega(b)} \tag{3-21}$$

$$Q_{i,b} = p_{i,\Omega(b)} + Q_{i+1,b}, i = 1,...,m-1 \tag{3-22}$$

$$Q_{m,u} = p_{m,\Omega(u)} + Q_{m,u+1}, u = 1,...,b-1 \tag{3-23}$$

$$Q_{i,u} = p_{i,\Omega(u)} + \max\{Q_{i,u+1}, Q_{i+1,u}\}, i = 1,...,m-1, u = 1,...,b-1 \tag{3-24}$$

$$C_{\max} = Q_{1,1} \tag{3-25}$$

方程(3-16)-(3-20)是正向计算方法。方程(3-16)计算机器 1 上首个加工工件的完成时间。方程(3-17)计算机器 i 上首个加工工件的完成时间。方程(3-18)计算第 u 个工件在机器 1 上的完成时间。方程(3-19)计算第 u 个工件在机器 i 上的完成时间。方程(3-20)表示通过正向加工得到的最大完工时间。

方程(3-21)-(3-25)是反向计算方法。方程(3-21)计算机器 m 上最后加工工件的完成时间。方程(3-22)计算机器 i 上最后加工工件的完成时间。方程(3-23)计算第 u 个工件在机器 m 上的完成时间。方程(3-24)计算第 u 个工件在机器 i 上的完成时间。方程(3-25)表示通过反向加工得到的最大完工时间。

$$C_{\max} = \begin{cases} \max\limits_{i=1,...,m}\{Q_{i,a} + A_{i,a}\}, & a = 1,...,b \\ A_{a,m}, & a = b+1 \end{cases} \tag{3-26}$$

假设有一个包含 b 个工件的加工序列，该序列中有 $b+1$ 个可能的插入位置。插入位置用 a ($a = 1,...,b,b+1$)表示。 $A_{i,a}$ 表示在工件 a 插入位置后，通过正向计算得到机器 i 上的完成时间。公式(3-26)给出了每个插入位置的最大完工时间。通过比较 $C_{\max}$ ，得到最佳的插入位置。图 3-7 展示了使用泰勒加速方法选择最优插入位置的示例。

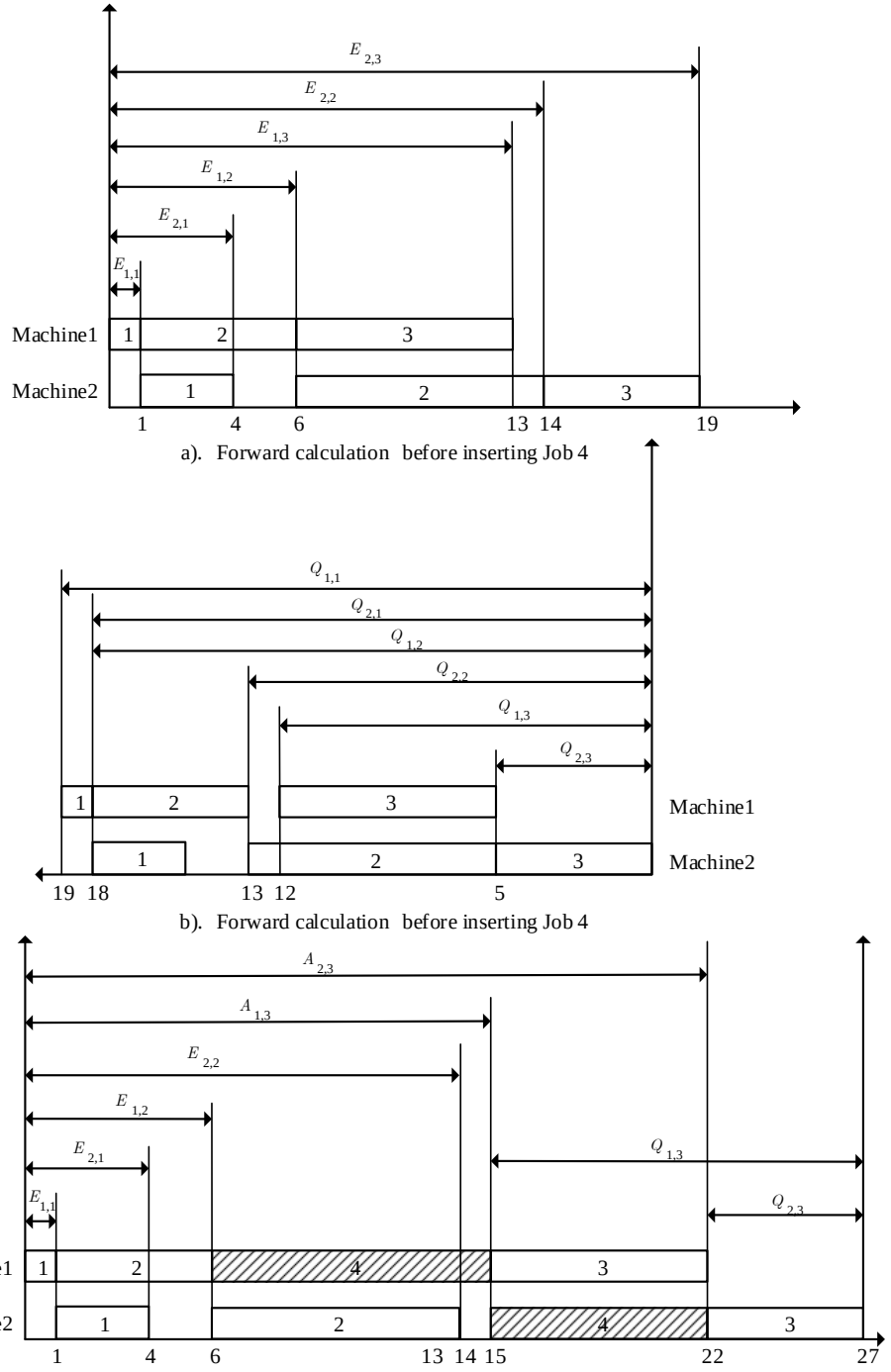

图 3-7 泰勒加速度方法示意图

3.2.6 算法流程

MBBO 通过加入有效的搜索方法和优化的突变策略来提升性能和搜索速度。在初始化阶段，DHI 生成了一批质量较好的栖地。通过改进的路径重联方法，整体栖地的 HSI 得到了提升。MBBO 通过采用改进的突变策略能够跳出局部最优，

从而提高搜索能力。邻域搜索可以有效地进行深度搜索。泰勒加速方法的加入加速了 MBBO 的搜索过程。图 3-8 展示了 MBBO 算法的流程图。

MBBO 的详细步骤如下：

(1) 使用 DHI 方法获得一组栖息地作为 MBBO 的初始栖息地。

(2) 根据迁入率和迁出率来选择迁入栖息地和迁出栖息地。重复此过程，直到所有栖息地都被考虑。

(3) 依次计算每个栖息地的突变概率。如果栖地发生变异，则随机扰动产品的加工顺序。如果没有突变发生，则进行局部搜索。

(4) 如果 MBBO 达到了最大迭代次数，则输出当前最优栖地；否则，重复步骤(2)和(3)。

(5) 对最优栖地进行 NSM 操作。

(6) 如果程序运行时间超过 CPU 终止时间，则输出 NSM 阶段搜索到的最优栖息地；否则，重复步骤(5)。

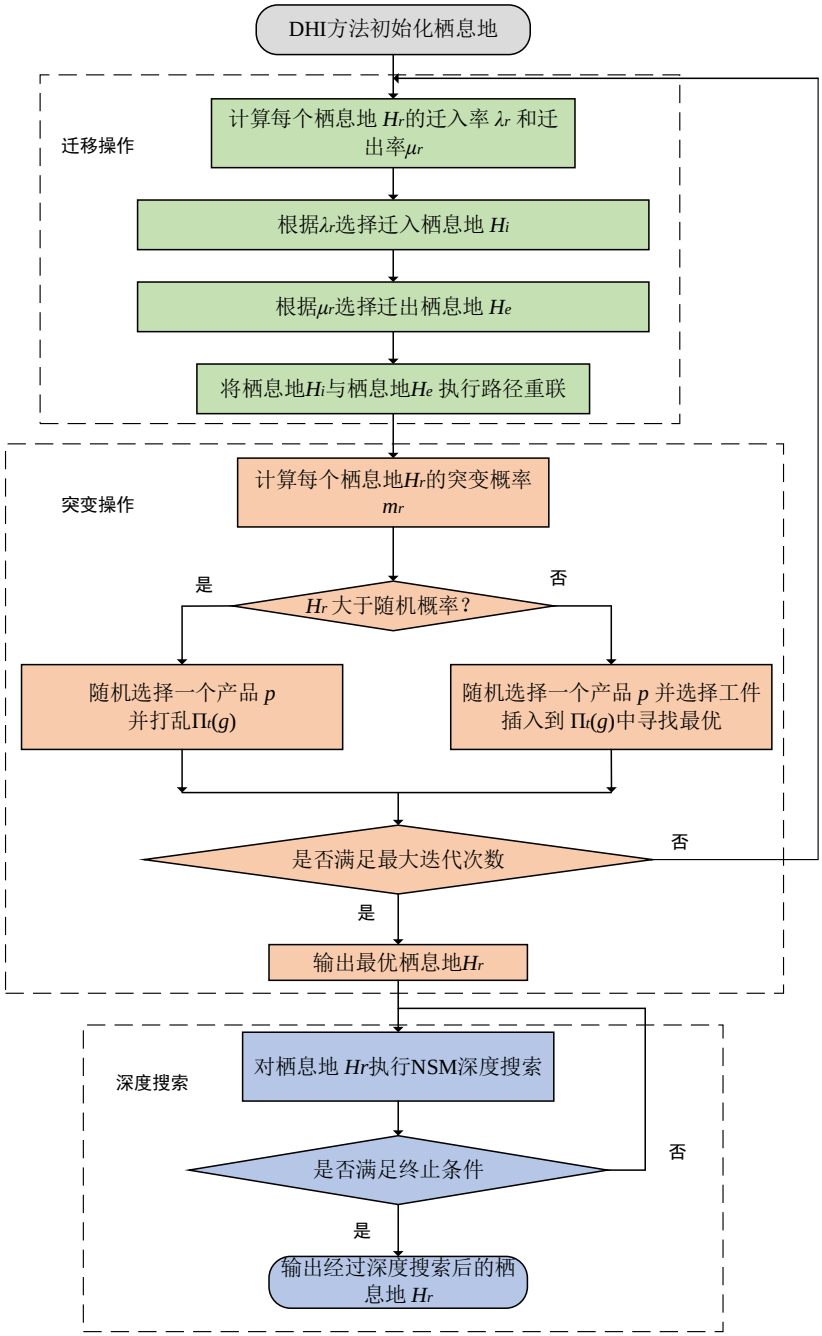

图 3-8 MBBO 算法流程图

3.3 仿真实验

3.3.1 测试案例与对比准则

仿真实验中所使用的 DAPFSP 实例的处理数据和最优解可以在 soa.iti.es 找到。选择了小规模的 DAPFSP 实例来进行算法性能测试。在小规模实例中，工件

数量$n\in\{8,12,16,20,24\}$、机器数量$m\in\{2,3,4,5\}$、工厂数量$f\in\{2,3,4\}$、产品数量$p\in\{2,3,4\}$，每种组合都有 5 个不同的实例，总共有 900 个实例。每个实例可以用$n_m_f_p$表示。MBBO 的终止条件是 CPU 运行时间，终止时间为$n\times f\times 0.1$秒。

平均相对百分比偏差(Average Relative Percentage Deviation, $ARPD$)用于表示 MBBO 解决 DAPFSP 的结果。最优相对偏差百分比(Percentage of Best Relative Deviation, RPD_{best})用于表示 MBBO 计算过程中最优运行结果。$ARPD$ 和 RPD_{best} 的定义如下：

$$ARPD=\sum_{T=1}^{R}(\frac{C_T-C_{best}}{C_{best}}\times 100)\times\frac{1}{R}(\%) \tag{3-27}$$

$$RPD_{best}=\min_{T=1,\ldots R}(\frac{C_T-C_{best}}{C_{best}}\times 100)(\%) \tag{3-28}$$

R 表示算法需要运行的次数，T 表示算法当前已运行的次数。C_T 表示算法运行第 T 次的结果，C_{best} 表示已知的最优结果，C_T 表示算法第 T 次运行时的最优结果。如果 MBBO 更新了已知的最优解，RPD_{best} 则为负值。

为了测试 MBBO 算法性能，增加了九种对比算法。变量邻域下降法(Variable Neighborhood Decent, VND)[9]采用了六种启发式方法作为初始解，两个工厂分配方法作为解码模式，并设计了两种邻域结构，LSJ 和 LSP。VND 在 DAPFSP 中具有更好的搜索能力。基于混合生物地理学优化算法(Hybrid Biogeography-Based Optimization, HBBO)[10]采用了优化的局部搜索方法。实验结果证明，HBBO 在解决 DAPFSP 问题中具有较大的优势。偏随机迭代局部搜索(Biased-Randomized Iterated Local Search, BR-ILS)[13]通过初始化-扰动-局部搜索的方式寻找最优解。BR-ILS 算法参数较少，复杂度较低。基于模因算法的分布估计(Estimation of Distribution Algorithm-based Memetic Algorithm , EDAMA)[14]提出了局部关键路径搜索，在 DAPFSP 仿真测试中具有良好表现。

3.3.2 参数实验

MBBO 有 3 个参数，分别是栖息地数量 N、最大迭代次数 $Iter_{\max}$ 和最大突变

率 $m_{\max}$。每个参数有四种不同的精度，共有 16 种参数组合。选取 20_3_2_2_1 实例进行测试；每种参数组合独立运行 50 次，使用平均完工时间(Average Makespan，AM)作为实验结果。表 3-3 显示了 MBBO 的三个参数在四种精度下的取值。表 3-4 展示了通过田口参数实验在不同参数组合下获得的正交数组实验结果。

根据图 3-9，可以观察到当栖息地数量 $N=150$、最大迭代次数 $Iter_{\max}=100$ 和最大变异率 $m_{\max}=0.5$ 时，MBBO 的性能优于其他情况。因此，这一组合被选择作为 MBBO 算法的参数。

表 3-3 MBBO 算法参数与精度等级

参数	参数等级			
	1	2	3	4
N	50	100	150	200
$Iter_{\max}$	30	50	80	100
$M_{\max}$	0.01	0.1	0.3	0.5

表 3-4 MBBO 正交实验结果

MBBO 参数组合	参数等级			
	N	$Iter_{\max}$	$m_{\max}$	AM
1	1	1	1	998.05
2	1	2	2	994.33
3	1	3	3	989.50
4	1	4	4	987.60
5	2	1	2	989.31
6	2	2	1	992.26
7	2	3	4	985.56
8	2	4	3	982.97
9	3	1	3	985.91
10	3	2	4	982.58
11	3	3	1	990.65
12	3	4	2	984.63
13	4	1	4	987.13
14	4	2	3	987.05
15	4	3	2	987.28
16	4	4	1	987.98

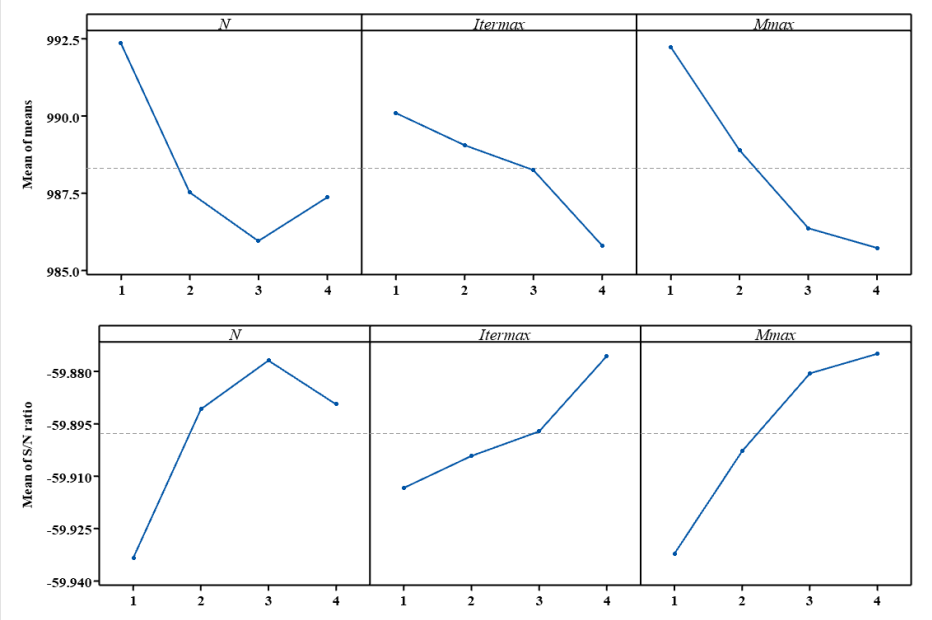

图 3-9 正交数组 $L_{16}(4^3)$折线图

3.3.3 对比实验

900 个 DAPFSP 案例根据 $f \times n$ 被分成了 15 组，每组大小不同。所有算法分别运行 5 次。运行结果如表 3-5 所示。图 3-10 是 9 种算法在 DAPFSP 实例中运行 *ARPD* 的 95%置信区间图，图 3-11 是 3 种算法在 DAPFSP 实例中运行 RPD_{best} 的 95%置信区间图。根据图 3-10 可知，MBBO 的表现远远优于大多数其他算法。同时，MBBO 的置信区间比 HBBO 的更小，这表明 MBBO 具有更强的稳定性。尽管 MBBO 的表现稍微高于 BR-ILS，但从图 3-11 可以看出，MBBO 的平均值低于 BR-ILS，这表明 MBBO 具有更好的搜索能力，更容易跳出局部最优。同时，MBBO 的稳定性优于 EDAMA 和 BR-ILS。

MBBO 在多个指标上表现优异。MBBO 的置信区间较小，且在不同规模的 DAPFSP 问题上具有较高的稳定性。同时，MBBO 的性能稳定，即使面对复杂问题的规模变化，MBBO 的表现也不会显著下降，性能依然稳定。表 3-6 展示了 MBBO 在 5 次实验中刷新已知最优解的情况。

表 3-5 不同算法在 DAPFSP 实例上的 *ARPD* 和 RPD_{best} 值

	ARPD									RPD_{best}		
$f \times n$	$VNDH_{11}$	$VNDH_{12}$	$VNDH_{21}$	$VNDH_{22}$	$VNDH_{31}$	$VNDH_{32}$	BR-ILS	HBBO	MBBO	EDAMA	BR-ILS	MBBO
2×8	1.00	0.76	1.00	0.76	1.02	0.78	0.26	0.38	0.21	0.00	0.26	0.00

2×12	0.93	0.87	0.93	0.87	0.93	0.87	0.07	0.41	0.31	0.00	0.07	0.02
2×16	0.73	0.55	0.72	0.53	1.09	0.53	0.02	0.16	0.24	-0.05	0.02	0.02
2×20	0.53	0.36	0.51	0.37	0.57	0.37	0.03	-0.15	0.11	-0.33	0.03	-0.06
2×24	0.54	0.21	0.54	0.21	0.54	0.21	0.04	-0.32	0.05	-0.48	0.04	-0.19
3×8	1.09	0.70	1.15	0.76	1.15	0.76	0.05	0.40	-0.19	0.00	0.02	0.00
3×12	0.44	0.28	0.44	0.28	0.44	0.28	-0.22	0.12	0.11	0.01	-0.25	0.01
3×16	0.86	0.56	0.91	0.56	0.91	0.56	0.08	0.08	0.15	-0.01	0.06	0.02
3×20	0.43	0.43	0.43	0.43	0.43	0.43	0.07	0.09	0.19	-0.01	0.03	-0.01
3×24	0.64	0.33	0.64	0.33	0.64	0.33	-0.33	-0.10	0.17	-0.16	-0.38	-0.05
3×8	1.08	0.63	0.99	0.63	0.99	0.63	-0.07	0.25	-0.19	0.00	-0.14	0.00
3×12	0.74	0.47	0.74	0.47	0.74	0.56	0.02	0.38	0.05	0.00	0.00	0.03
3×16	0.59	0.28	0.59	0.28	0.59	0.28	0.20	0.13	0.14	0.03	0.20	0.04
3×20	1.10	0.63	1.10	0.63	1.10	0.63	0.05	0.13	0.24	-0.01	0.03	0.03
3×24	0.57	0.26	0.57	0.26	0.57	0.26	0.01	-0.01	0.19	-0.05	0.00	0.05
Average	0.75	0.49	0.75	0.49	0.78	0.50	0.02	0.13	0.12	-0.07	0.00	-0.01

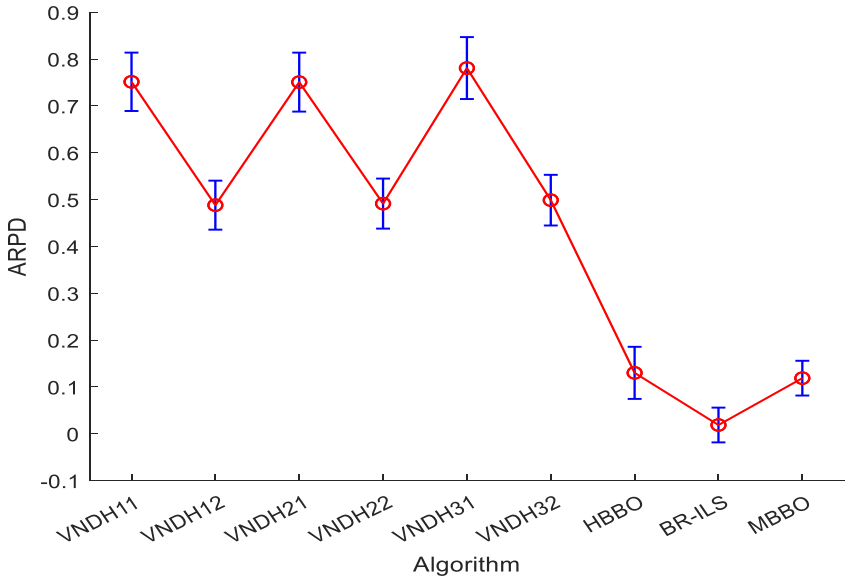

图 3-10 9 种算法 *ARPD* 结果的 95%置信区间图

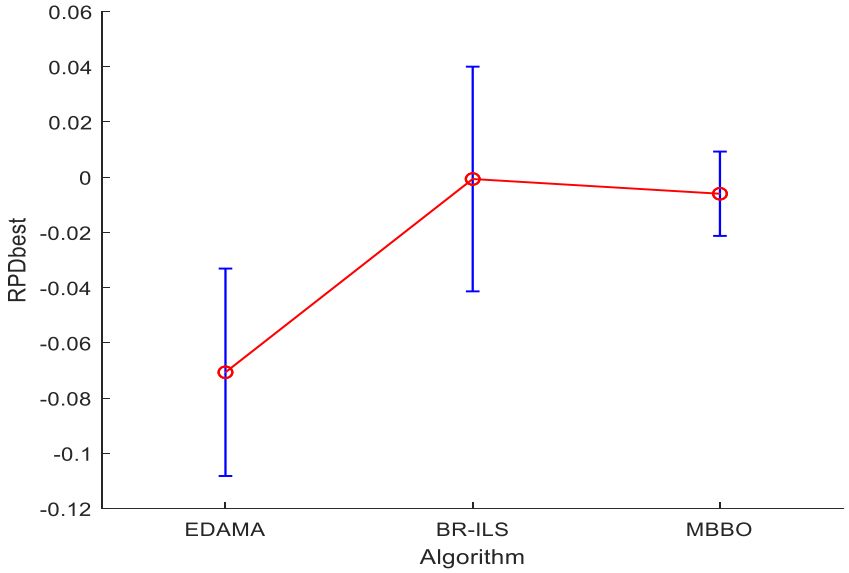

图 3-11 3 种算法 RPD_{best} 结果的 95%置信区间图

表 3-6 DAPFSP 基准实例部分刷新解

Instance	Best known	MBBO	RPD_{best}	Instance	Best known	MBBO	RPD_{best}
16_4_2_2_3	1146	1143	-0.26	24_3_3_3_2	1078	1073	-0.46
16_4_3_2_3	858	854	-0.47	24_3_3_4_1	918	917	-0.11
20_2_2_2_3	1462	1461	-0.07	24_3_3_4_2	1042	1037	-0.48
20_2_4_2_2	369	367	-0.54	24_3_3_4_3	1738	1736	-0.12
20_3_2_2_3	997	995	-0.20	24_4_2_2_1	1350	1346	-0.30
20_3_2_2_4	1722	1721	-0.06	24_4_2_2_5	999	982	-1.70
20_3_2_3_1	1756	1752	-0.23	24_4_2_3_2	1225	1214	-0.90
20_3_2_3_2	1121	1115	-0.54	24_4_2_3_3	1711	1684	-1.58
20_4_2_2_5	1394	1386	-0.57	24_4_2_3_5	2046	2045	-0.05
20_4_2_3_2	858	825	-3.85	24_4_2_4_2	1205	1198	-0.58
20_4_2_3_4	1725	1716	-0.52	24_4_3_2_2	1694	1684	-0.59
20_4_3_3_5	1137	1129	-0.70	24_4_3_2_4	1844	1843	-0.05
20_5_2_2_1	1478	1474	-0.27	24_4_3_4_5	1480	1473	-0.47
20_5_2_2_2	1258	1257	-0.08	24_4_4_2_5	1356	1355	-0.07
20_5_2_2_3	1500	1480	-1.33	24_5_2_2_1	1625	1616	-0.55
20_5_2_3_2	1181	1176	-0.42	24_5_2_2_2	2136	2118	-0.84
20_5_2_3_3	1499	1494	-0.33	24_5_2_2_3	2049	2046	-0.15
24_2_2_3_2	1514	1511	-0.20	24_5_2_3_1	2333	2328	-0.21

24_2_2_3_3	1495	1492	-0.20	24_5_2_3_3	1340	1333	-0.52
24_2_2_3_5	1861	1856	-0.27	24_5_2_3_5	914	910	-0.44
24_2_4_2_5	563	559	-0.71	24_5_2_4_2	1054	1029	-2.37
24_3_2_2_1	1705	1692	-0.76	24_5_2_4_3	1691	1687	-0.24
24_3_2_3_1	1077	1075	-0.19	24_5_3_2_2	1518	1500	-1.19
24_3_2_3_2	1417	1398	-1.34	24_5_3_2_3	1890	1887	-0.16
24_3_2_4_1	1323	1320	-0.23	24_5_3_3_3	1761	1759	-0.11
24_3_2_4_2	755	752	-0.40	24_5_3_3_4	2318	2308	-0.43
24_3_3_2_4	584	579	-0.86				

3.3.4 消融实验

为了评估在 3.2.4 节中提出的 NSM 方法的有效性，本节进行了针对 NSM 的消融实验。消融实验是算法研究领域常用的一种实验方法，通过移除或禁用某些特性来评估这些特性对算法整体性能的影响。表 3-7 列出了八种不同机制的算法标签。每个算法独立测试 900 个 DAPFSP 实例 5 次，并将 RPD_{best} 结果进行比较。

表 3-7 包括/不包括不同邻域结构的 MBBO 算法

Algorithm	
None	Not included $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3$
NSM ($\omega 1$)	Include $\mathcal{N}_1$
NSM ($\omega 2$)	Include $\mathcal{N}_2$
NSM ($\omega 3$)	Include $\mathcal{N}_3$
NSM ($\omega 4$)	Include $\mathcal{N}_1, \mathcal{N}_2$
NSM ($\omega 5$)	Include $\mathcal{N}_1, \mathcal{N}_3$
NSM ($\omega 6$)	Include $\mathcal{N}_2, \mathcal{N}_3$
MBBO	Include $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3$

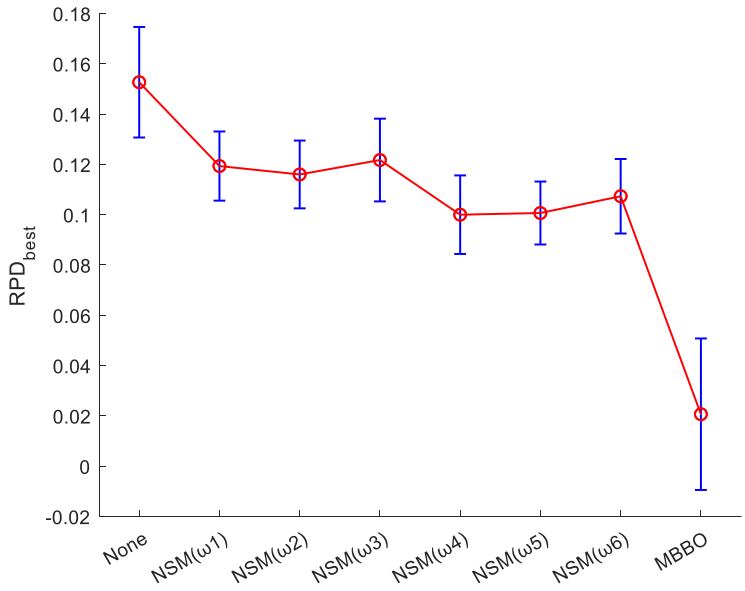

图 3-12 8 组不同算法标签下的 95%置信区间结果

图 3-12 记录了每组算法的运行结果。可以观察到，无论是哪种邻域结构，都能有效地提升算法的搜索能力。而 NSM($\omega 1$)，NSM($\omega 2$)和 NSM($\omega 3$)的均值为 0.118，相比于 None， *ARPD* 提升了 21.7%；同时，区间也有所缩小，这表明算法的稳定性也得到了提高。NSM($\omega 4$)，NSM($\omega 5$)和 NSM($\omega 6$)包含了两个邻域搜索结构，它们的 RPD_{best} 为 0.102。与单一邻域结构相比，两个邻域结构可以使算法的搜索能力提高 13.5%。MBBO 包含了所有邻域结构，算法通过加入三种邻域结构可以进一步提升搜索能力 80%。

3.4 本章小结

本章提出了一种改进的生物地理优化算法，用于求解分布式装配置换流水车间调度问题，以最小化 makespan 为目标函数。首先，MBBO 采用双栖地初始化方法生成初始解，DHI 通过两种不同的工厂分配方法和交换关键编码段生成一批具有高适应度的解。在迁移阶段，采用了改进的路径重联技术，可以有效地将编码段从优秀栖地转移到较差栖地，从而提高整个栖地的质量。在变异阶段，设计了一种改进的变异策略，以增强解的多样性并提高部分解的质量。在深度搜索阶段，打破了同一产品下工件连续加工的原规则，改为不同产品下工件的连续加工，以寻找更好的解。通过参数实验选择了适用于 MBBO 的参数，通过消融实验验

证了邻域结构的有效性。MBBO 与其他算法进行了比较，结果表明 MBBO 在 900 个 DAPFSP 实例中高效且刷新了 53 个已知的最优解。

第 4 章 模糊分布式装配置换流水车间调度问题研究

在现实生活中，工件的加工时间通常会受到不确定性因素的影响，这种不确定性可能由各种误差引起，例如加工延迟或机器故障。因此，研究具有模糊加工时间约束的分布式装配置换流水车间调度问题将更好地解决实际生产中的加工问题。为了解决以最小化最大模糊完成时间为目标的模糊分布式装配流水车间调度问题，提出了一种区域生物地理学优化算法(regional biogeography-based optimization algorithm, RBBO)。给出了三角模糊数及其操作方法。在 RBBO 中，所有栖息地根据栖息地适宜度指数划分为多个区域，每个区域的栖息地都经历跨区域迁移和替换过程。此外，四种迁移率模型作为常用模型的替代方案进行了全面测试。在 FDAPFSP 实例中，比较了 RBBO 算法与多种启发式算法和群体智能算法的测试结果。计算结果表明，在 10 个 FDAPFSP 实例中，所提出的 RBBO 算法在 9 个案例中优于其他算法。

4.1 三角模糊数及其操作

在模糊加工时间约束下解决分布式装配置换流水车间调度问题时，处理模糊加工时间是至关重要的。由于在现实生产环境中，作业的加工时间和装配时间常常受多种不确定因素的影响，使用传统的确定性数值可能无法准确反映实际情况。因此，在本研究中，工作加工时间和产品装配时间采用三角模糊数(Triangular Fuzzy Numbers, TFN)进行模糊处理[55]，以便更好地模拟实际操作中的不确定性。三角模糊数由三元组(a,b,c)表示，其中a和c分别为最小和最大可能值，而b则是最可能发生的值。通过这种方式，可以更有效地描述加工时间和装配时间的变化范围及其概率分布。

当 TFN 被用来表示调度问题中的加工数据时，通常需要借助一些运算符来处理模糊数据并构建调度计划。在此过程中，常用的运算符包括加法运算符、最大值运算符和排序运算符[56]。加法运算符用于计算多个工件或任务之间的模糊

加工时间的合成值，能够反映在调度过程中各个任务组合后的整体不确定性。最大值运算符则用于选择多个作业中的最大加工时间，确保调度计划中能够覆盖所有可能的最差情况。排序运算符则用于对作业进行优先级排序，从而帮助调度系统根据任务的紧急程度或资源需求进行合理安排。通过综合运用这些运算符，可以在模糊环境下更好地优化车间的调度过程，降低不确定性对调度结果的影响。

此外，采用模糊处理的调度方法可以更准确地应对实际生产中经常出现的时间波动和资源不确定性，能够提高生产调度的灵活性和可靠性，从而提升车间整体的生产效率和响应能力。有关 TFN 的加法运算符、最大运算符和排序运算符的描述如下：

对于两个 TFN, $\tilde{s}=(s_1,s_2,s_3)$ 和 $\tilde{t}=(t_1,t_2,t_3)$，加法运算符的计算过程如下：

$$\tilde{s}+\tilde{t}=(s_1+t_1,s_2+t_2,s_3+t_3) \tag{4-1}$$

为了比较 $\tilde{s}$ 和 $\tilde{t}$ 的大小，3 个比较公式 $c_1(\tilde{s})$，$c_2(\tilde{s})$ 和 $c_3(\tilde{s})$ 被用来进行 TFN 之间数值比较。比较公式的定义如下：

$$c_1(\tilde{s})=(s_1+2s_2+s_3)/4 \tag{4-2}$$

$$c_2(\tilde{s})=s_2 \tag{4-3}$$

$$c_1(\tilde{s})=s_3-s_1 \tag{4-4}$$

如果 $c_1(\tilde{s})>(<)c_1(\tilde{t})$，则 $\tilde{s}>(<)\tilde{t}$。如果 $c_1(\tilde{s})=c_1(\tilde{t})$ 并且满足 $c_2(\tilde{s})>(<)c_2(\tilde{t})$，则 $\tilde{s}>(<)\tilde{t}$。如果 $c_1(\tilde{s})=c_1(\tilde{t}),c_2(\tilde{s})=c_2(\tilde{t})$ 并且满足 $c_3(\tilde{s})>(<)c_3(\tilde{t})$，则 $\tilde{s}>(<)\tilde{t}$。

TFN 的最大运算符计算过程如下：

如果 $\tilde{s}>\tilde{t}$，则 $\tilde{s}\vee\tilde{t}=\tilde{s}$，否则 $\tilde{s}\vee\tilde{t}=\tilde{t}$。

4.2 区域生物地理学优化算法

RBBO 通过对整个解空间进行有效的区域划分，将不同的栖息地分配到各自的特定区域，每个区域的栖息地执行与其特点相匹配的任务。这样的任务分配不仅可以提高局部搜索的效率，还能够促进不同区域间的协同作用，从而达到整体优化的目标。通过这种区域化的策略，RBBO 能够有效避免过早的收敛，并在复杂的搜索空间中保持较好的全局搜索能力，最终提高优化过程的整体性能。在算法初期需要进行区域划分，划分的区域表示为：区域 A (R_A),区域(R_B), 和区域

C (R_C). 所有栖息地必须分配到 R_A、R_B 或 R_C 中，在此之前需根据 HSI 先对所有栖息地进行排序。 每一代迭代过程包括两轮区域的迁移和替换，这是相邻区域之间进行的活动。在图 4-1 中展示了 RBBO 模型。公式(4-5)-(4-7)为 RBBO 的迭代模型。

$$R_A^{t+1} = R_A^t + (R_B^{t+1}(h_{best}) - R_A^t(h_{worst})) \tag{4-5}$$

$$R_B^{t+1} = R_B^t + (R_B^t(h_{A.M}) - R_B^t(h_m)) + (R_C^{t+1}(h_{best}) - R_B^t(h_{worst})) \tag{4-6}$$

$$R_C^{t+1} = R_C^t + (R_C^t(h_{B,M}) - R_C^t(h_m)) \tag{4-7}$$

图 4-1 RBBO 算法模型原理图

t 表示当前的迭代次数，$t+1$ 表示下一代即将迭代的次数。R_k 表示区域 k 的栖息地聚集。h_{best} 表示该区域的最优栖息地，而 h_{worst} 表示该区域的最差栖息地。$h_{k,M}$ 是区域 k 中选定的迁出栖息地。h_m 是选定的移入栖息地。公式(4-5)表示，在 $t+1$ 迭代中，R_A 的栖息地集等于该区域的原始栖息地集，加上 $t+1$ 迭代中 R_B 的最优栖息地，减去 t 迭代中 R_A 的最差栖息地。公式(4-6)表示，在 $t+1$ 迭代中，R_B 的栖息地集等于原始栖息地集，加上 $t+1$ 迭代中 R_C 的最优栖息地，减去 t 迭代中 R_B 的最差栖息地，再加上 t 迭代中 R_B 通过 R_A 迁移后的栖息地，减去 t 迭代中 R_B 选定的移入栖息地。公式(4-7)表示，在 $t+1$ 迭代中，表示 R_C 的栖息地集等于原始栖息地集，加上 t 迭代中 RC 通过 RB 迁移后的栖息地，减去 t 迭代中 R_C 选定的移入栖息地。

4.3 迁移率模型

BBO 中的迁移率模型对于算法的收敛速度和全局搜索能力具有重要影响。传统上，线性迁移率模型是最为常见的一种方式，它通过逐步促使种群达到动态平衡来探索和开发。然而，除了线性迁移率模型外，还有其他几种迁移率模型，如二次迁移率模型、指数迁移率模型和余弦迁移率模型，如公式(4-8)-公式(4-13)

所示。这些模型各自具有不同的特性，可以在不同的搜索空间和优化任务中展现出不同的优势。

$$\lambda_r = Ie^{-\frac{N(r)}{N_{\max}}} \tag{4-8}$$

$$\mu_r = Ee^{\frac{N(r)}{N_{\max}}-1} \tag{4-9}$$

$$\lambda_r = I(1-\frac{N(r)}{N_{\max}})^2 \tag{4-10}$$

$$\mu_r = E(\frac{N(r)}{N_{\max}})^2 \tag{4-11}$$

$$\lambda_r = \frac{I}{2}(1+\cos(\frac{N(r)\pi}{N_{\max}})) \tag{4-12}$$

$$\lambda_r = \frac{E}{2}(1-\cos(\frac{N(r)\pi}{N_{\max}})) \tag{4-13}$$

在本文中，采用了多种迁移率模型进行比较与测试，在选择最适合解决 FDAPFSP 的迁移率模型。通过对比不同模型的性能，分析其在处理复杂调度问题时的优势与不足，最终选定了最具优势的迁移率模型。实验结果表明，选择合适的迁移率模型对于提升算法的优化效果和搜索效率至关重要，尤其是在面对具有模糊加工时间和资源约束的调度问题时，能够显著提高调度解的质量和收敛速度。具体的实验结果和分析将在 4.5.2 节中详细描述。

4.4 算法流程

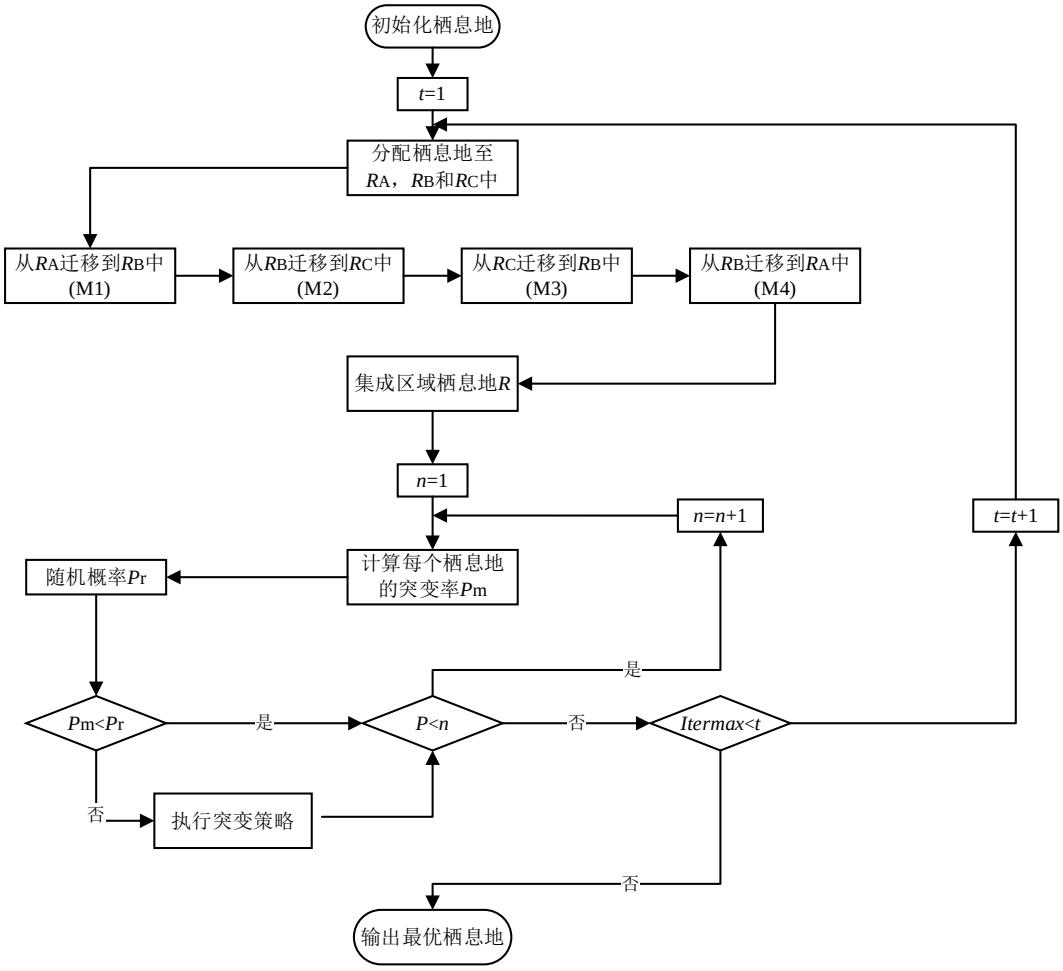

图 4-2 RBBO 算法流程图

图 4-2 为 RBBO 的算法流程图。在算法初期会执行初始化栖息地操作以保证提供一批完整的初始解。根据 4.2 节提供的方法 RBBO 会在迁移操作进行之前对每个栖息地进行区域划分，采用四种迁移率模型执行迁移操作。在保证所有栖息地完成迁移后 RBBO 将计算每个栖息地的突变概率，以随机概率相比较，判断栖息地是否执行突变。当满足最大迭代次数时，RBBO 执行排序操作并输出具有最大适应度的栖息地作为最优栖息地。

4.5 仿真实验

所有实验均在一台配备 16.0GB RAM 和 2.8GHz CPU 的笔记本电脑上进行，

使用 Microsoft Visual C++2022 进行程序编程。

4.5.1 参数实验

本文提供的十个案例都是通过随机数基于 DAPFSP 基本案例随机生成的。RBBO 的参数包括栖息地数量 P、最大变异率 $M_{\max}$ 和最大迭代次数 $Iter_{\max}$。本文将采用四组不同等级的参数进行田口参数实验，并进行测试。三个参数组合将生成 16 个测试案例。每个案例测试 10 次，并记录每次测试的数据。测试完成后，进行 *TFN* 比较，并生成一组从小到大的排序表。每次测试的 *TFN* 都有对应的排序序号，排序序号用于替代 *TFN*，以生成参数运行图。计算每个测试案例的平均排名(*AR*)。表 4-1 展示了本文提供的四组参数。表 4-2 展示了 16 个测试案例对应的 *TFN* 排名。表 4-3 展示了 16 组参数组合生成的 *AR*。图 4-3 为参数趋势图。

表 4-1 RBBO 参数精度设置表

参数	参数等级			
	1	2	3	4
栖息地数量 P	10	30	50	100
最大突变概率 $M_{\max}$	0.05	0.10	0.20	0.40
最大迭代次数 $Iter_{\max}$	30	50	100	150

表 4-2 参数实验 *TFN* 排名表

Rank	*TFN*
1	(776 1036 1266)
2	(777 1029 1278)
3	(778 1037 1252)
4	(781 1036 1252)
5	(786 1034 1274)
6	(795 1033 1257)
7	(798 1062 1271)
8	(808 1060 1279)

9	(813 1054 1280)
10	(815 1051 1285)

表 4-3 RBBO 正交实验结果

参数组合	参数等级			*AR*
	P	$M_{\max}$	$Iter_{\max}$	
1	1	1	1	5.3
2	1	2	2	6.7
3	1	3	3	4.3
4	1	4	4	4.7
5	2	1	2	5.1
6	2	2	1	3.4
7	2	3	4	4.1
8	2	4	3	5
9	3	1	3	3.9
10	3	2	4	3.4
11	3	3	1	4
12	3	4	2	4
13	4	1	4	3.3
14	4	2	3	3.3
15	4	3	2	3.5
16	4	4	1	3.2

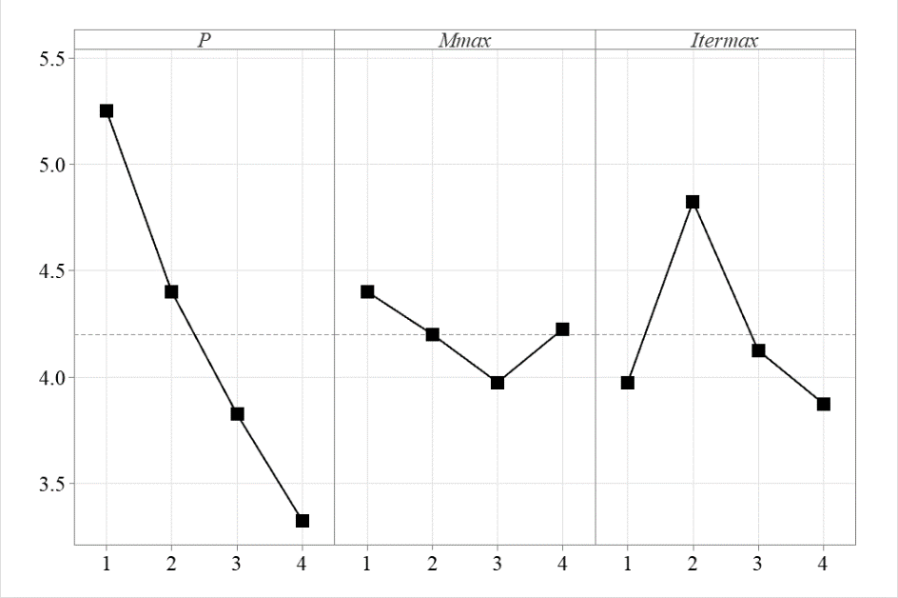

图 4-3 RBBO 各参数平均 AR 曲线图

从图 4-3 的仿真结果和表 4-3 中参数组合的仿真结果可以看出，当 P 增加时，算法的搜索性能有所提高；而 $M_{\max}$ 过大或过小都会导致算法性能的较差变化，$Iter_{\max}$ 也有类似的影响。因此，选择 P =100，$M_{\max}$ =0.20 和 $Iter_{\max}$ =150 来解决问题时，RBBO 算法表现最佳。因此，选择该参数组合作为 RBBO 算法的参数。

4.5.2 收敛性分析

在第 4.3 节中，提出了四种新的迁移率模型：线性迁移率模型、指数迁移率模型、二次迁移率模型和余弦迁移率模型。这些模型用于替代 RBBO 中的原始迁移率模型，以进行对比实验。这些模型的设计旨在提高算法在解决 FDAPFSP 问题中的性能，特别是在优化不同规模情况下的资源调度。为了评估这四种迁移率模型的有效性，选取了 10 个 FDAPFSP 测试案例进行实验。采取了去模糊化的方法(只保留 c_2)。

图 4-4 展示了不同迁移率模型优化算法在 FDAPFSP 小规模案例中的 95%置信区间。置信区间是基于多次实验结果计算的，直观地反映了每个模型在解决问题时的稳定性和可靠性。从图中可以观察到，各个模型的解质量存在显著差异。余弦迁移率模型在确保解质量的同时保持了稳定性，而其他模型则表现出更大的波动性和较差的解质量，可能需要更多的迭代才能得到更好的结果。

此外，为了进一步分析算法的收敛性，选择了 24_5_4_4 案例进行实验。图

4-5 展示了四种算法在相同测试案例上的收敛性分析结果。通过展示每种算法在迭代过程中目标函数值的变化，图中清晰地反映了不同迁移率模型的收敛特性。根据图中的趋势，余弦迁移率模型在早期迭代阶段迅速接近最优解，而其他模型在收敛过程中可能会出现较大的波动，并需要更长的时间才能稳定下来。通过收敛性分析，可以评估每个模型的搜索效率和解的稳定性，为优化调度问题提供理论依据。

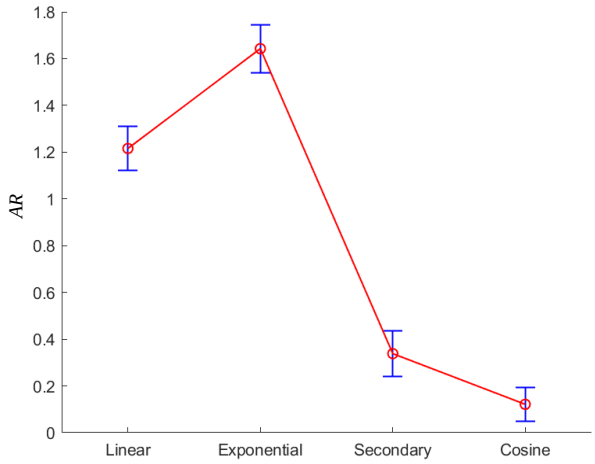

图 4-4 4 种迁移率模型的 95%置信区间图

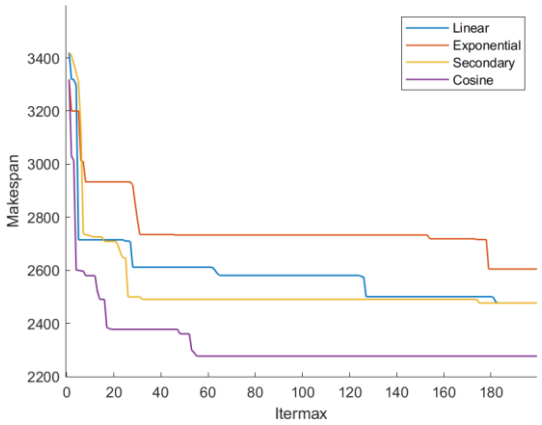

图 4-5 4 种迁移率模型的在案 24_5_4_4 的收敛曲线

最终，通过实验结果可知，本文为 FDAPFSP 提供一个更具竞争力的解决方案，特别是在复杂的生产调度环境中。它展示了迁移率模型的改进如何增强算法的全局搜索能力和局部优化能力，从而更有效地平衡优化问题中的各种需求。

4.5.3 对比实验

为了评估 RBBO 在 FDAPFSP 中的有效性，选择了 BBO 和 GA 进行对比测

试。除了算法模型不同外，实验采用相同的搜索方法进行公平比较。表 4-4 为测试结果的对比表。使用c_2作为对比方法，因为目前没有计算*TFN*平均值的方式。每个算法提供了三个度量指标：算法五次运行中的最差解（WST）、最优解（BST）和五次运行的平均输出结果（AVG），这些都用相应的缩写表示。本文设计的案例均基于 DAPFSP 案例，共提供了 10 个案例，案例的规模为$J=\{8,12,16,20,24\}$，$M=\{2,5\}$，$F=\{2,4\}$，$P=\{2,4\}$。

对 RBBO、BBO、GA 和 VND 在 10 个独立案例中的测试结果进行方差分析，可以更深入地考察这三种算法之间的差异。VND 的终止条件为 CPU 运行时间=$F\times C$，其中$C=0.1$。GA、BBO 和 RBBO 使用相同的迭代次数和种群规模，并将终止条件设置为最大迭代次数。图 4-7 展示了 95%置信区间水平下四种算法的均值变化曲线。

表 4-4 算法测试结果对比表(智能优化算法)

Problem scale	VND			GA			BBO			RBBO		
	WST	BST	AVG	WST	BST	AVG	WST	BST	AVG	WST	BST	AVG
8×2×2×2	501	501	501	501	501	501	501	501	501	**501**	**501**	**501**
8×5×4×4	876	876	876	876	876	876	876	876	876	**876**	**876**	**876**
12×2×2×2	969	955	961	956	949	952	**949**	945	**942**	962	**940**	950
12×5×4×4	763	763	763	763	763	763	763	763	763	**763**	**763**	**763**
16×2×2×2	926	917	920	921	910	911	915	901	909	**906**	**894**	**899**
16×5×4×4	1077	1062	1071	1062	1029	1051	1062	1020	1045	**1025**	**1020**	**1024**
20×2×2×2	1444	1439	1442	1436	1430	1432	1438	1435	1436	**1434**	**1430**	**1432**
20×5×4×4	1685	1685	1685	1685	1685	1685	1685	1685	1685	**1685**	**1685**	**1685**
24×2×2×2	1903	1886	1894	1880	1861	1867	1878	1856	1869	**1875**	**1849**	**1863**
24×5×4×4	2451	2451	2451	2451	2451	2451	2451	2451	2451	**2451**	**2451**	**2451**
Average	1259	1253	1256	1253	1246	1249	1251	1243	1247	**1247**	**1240**	**1244**

表 4-4 和图 4-6 显示，RBBO 显著优于 GA 和 VND，RBBO 的 WST、BST 和 AVG 值远大于 GA 的。这证明了 RBBO 在处理分布式问题时具有优势。由于 RBBO 的迁移规模比 BBO 更大，它提供了比 BBO 更多的新解。RBBO 通过在整体栖息地中迅速达到局部最优，可能加速突变过程。

提供了 RBBO 与 BBO、GA、VND 及六种启发式算法的 CPU 运行时间折线图。其中，三种启发式算法 SPT（最短加工时间）、LPT（最长加工时间）和 NEH 均取自参考文献[9]，六种启发式算法则通过[5]中的两种不同工厂分配方法得到。虽然启发式算法无法提供最优解，但解的质量远好于随机解，且启发式算法通常具有更短的运行时间。通过启发式算法，可以筛选出算法中的冗余解。如图 4-7 和表 4-6 所示，RBBO 在运行时间上相比 GA 具有优势，但比 BBO 要长，然而它通过牺牲一些 CPU 运行时间换取了更好的解，这是一种合理的平衡策略。

通过实验发现，在这十个案例中，RBBO 的栖息地演化速度远快于 BBO。由于栖息地进化速度的提高，预计该算法将有足够的时间通过局部最优来搜索更好的解。这是因为 RBBO 的栖息地发展速度比 BBO 更快，这个机制是在迁移阶段发展起来的。

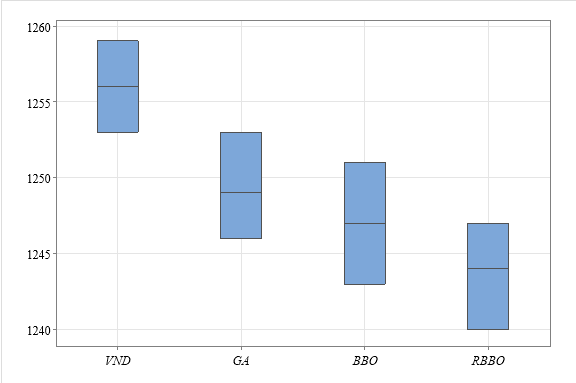

图 4-6 4 种算法在 FDAPFSP 案例下的 95%置信区间图

表 4-5 算法测试结果对比表(规则启发式算法)

Problem scale	SPT1	SPT2	LPT1	LPT2	NEH1	NEH2	RBBO
							BST
8×2×2×2	651	651	543	543	503	503	**501**
8×5×4×4	876	876	876	876	876	876	**876**
12×2×2×2	1150	1200	1294	1200	1275	1281	**940**
12×5×4×4	797	863	927	927	842	842	**763**
16×2×2×2	1667	1691	1850	1846	1869	1869	**894**
16×5×4×4	1447	1356	1485	1365	1489	1320	**1020**
20×2×2×2	2977	2977	2977	2977	2992	2991	**1430**

20×5×4×4	2291	2432	2291	2427	2485	2456	**1685**
24×2×2×2	1928	1928	2189	2226	1846	1850	**1849**
24×5×4×4	3719	3471	4059	4028	3828	3812	**2451**
Average	1750	1744	1849	1841	1800	1780	**1240**

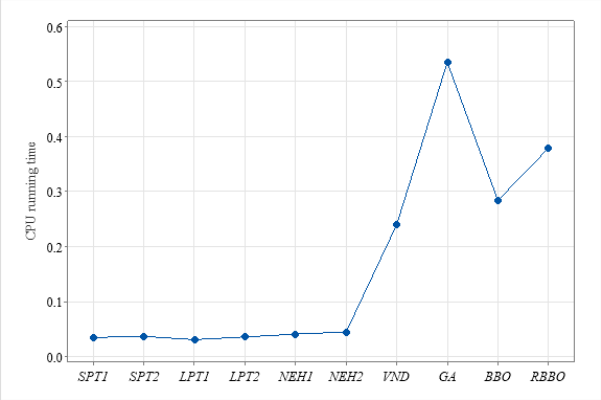

图 4-7 启发式算法与智能优化算法在 FDAPFSP 案例下的 95%置信区间图

4.6 本章小结

本章研究了在模糊加工时间约束下的 DAPFSP，设计了问题数学模型。本文提出了 RBBO 算法，设计了四组参数，并通过田口实验获得了最佳参数组合。设计并求解了 10 组实例。通过对比 SPT、LPT、NEH、VND、GA、BBO 和 RBBO 算法，得出了这三种算法在 FDAPFSP 案例中的应用区间图。为了验证迁移率模型的假设，进行了收敛性分析，并得到了 RBBO 解决 FDAPFSP 最佳的迁移率模型。

第 5 章 集成制造系统下的分布式柔性装配置换流水车间调度问题研究

分布式装配置换流水车间调度的研究启发了分布式制造系统的创新发展，为实现更符合实际的生产模式已经做出了许多努力。这些努力包括研究具有协同装配工厂与柔性装配机器的集成调度，这些问题的模型通常依赖于元启发式算法进行近似求解，但求解精度容易因算法的随机性导致解的质量不稳定，而不是像通过求解器那样能够在复杂度较高的情况下提供理论上的全局最优解。对此，本章构建了两种基于集成制造系统的分布式柔性装配置换流水车间调度问题(Distributed Flexible Assembly Flexible Permutation Flowshop Scheduling Problem based on Integrated Manufacturing System, IMS-DFAPFSP)的混合整数线性规划模型，并且受旅行商问题的启发，在 IMS-DFAPFSP 模型中融入了 Miller-Tucker-Zemlin(MTZ)方法消除冗余约束，促使模型能够通过求解器进行有效搜索。同时，在求解质量方面，提出了结合强化学习的改进策略，以提高对环境的感知能力。最后，通过设计 IMS-DFAPFSP 实例测试了模型与算法在不同问题规模下的求解能力，基于测试结果对 IMS-DFAPFSP 中的重要参数进行了敏感性分析，根据结果得出了一些管理方面的启示。实验结果表明，提出的模型和方法在处理此类复杂调度问题上具有有效性，为集成制造系统的优化提供了新的思路和方法。

5.1 基于集成制造系统的分布式装配置换流水车间调度问题

5.1.1 问题描述

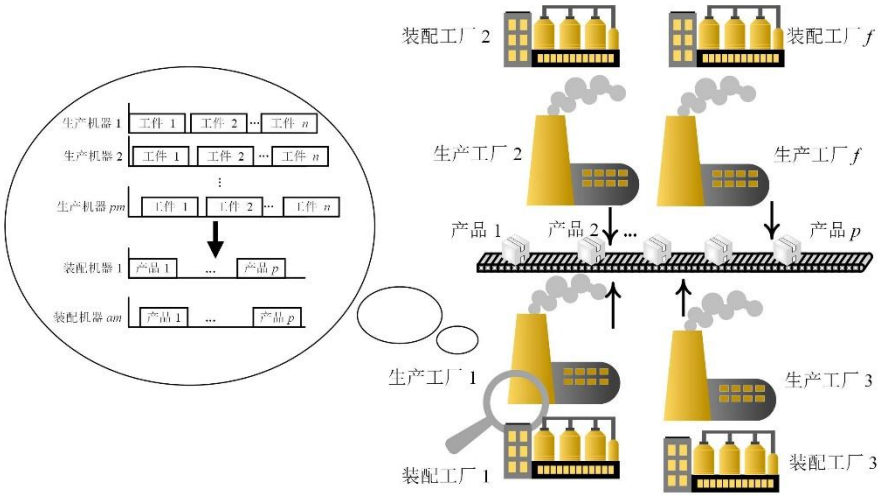

图 5-1　IMS-DFAPFSP 示意图

图 5-1 为问题的示意图，IMS-DFAPFSP 包含两个主要阶段，将这两个阶段看成一个集成制造系统：工件的生产阶段和产品的装配阶段。在生产阶段，工件 j ($j=1,...,n$)被分配到生产工厂 g ($g=1,...,f$)中进行加工，每个工件仅能被分配到一个生产工厂。每个生产工厂可以被视为一个流水车间调度问题，即包含相同数量的生产机器 i ($i=1,...,pm$)，每个工件需要在生产机器上按照相同的路径进行加工，例如：首先在机器 1 上加工，再到机器 2 上加工直到在机器 pm 上加工后才为一个完整的加工路径，这种加工方式为流水加工，工件每道工序的加工时间为 $pp_{i,j}$ ($pp_{i,j}>0$)，工件一旦开始加工就不能中断。在装配阶段中，每个工件只属于一个特定的产品 h ($h=1,...,p$)。当属于产品 h 的工件全部完成加工后，其中具有最大完工时间的工件所在的工厂为产品 h 进行装配所在的装配工厂，产品可以在当前装配工厂的任意一台装配机器 w ($w=1,...,am$)上进行装配。属于产品 h 的工件数量为 Ψ_h ， $n=\Sigma_{h=1}^{p}\Psi_h$ 。产品 h 的装配时间为 pa_h ($pa_h>0$)，产品一旦开始装配就不能中断

5.1.2 问题假设

在构建模型之前，提出了有关 IMS-DFAPFSP 的一般性假设。

(1)装配工厂环境假设：在 IMS-DFAPFSP 中，假设装配工厂包含多台装配机器，每个生产工厂配备一个独自的装配工厂以此提高生产的灵活性和效率。

(2)生产过程假设：假设生产过程是有序的，每个工件在加工过程中需要按照特定的顺序经过同一台机器。工件的每道工序一旦开始加工就不能中断。

假设的提出为模型的建立和优化提供了理论基础，有助于后续研究中调度策略的制定和性能的提升。

5.1.3 符号、参数和变量的定义与说明

在提出 IMS-DFAPFSP 的 MILP 模型之前，一些符号、参数和变量需要被提前定义。符号指的是用于表示问题中各个要素的记号或字母，用来简化和规范数学表达。参数指的是已知的常数值，描述了系统的固定特性和约束条件。变量分为连续变量(Continuous Variables, CVs)和二元变量(Binary Variable, BVs)，其中连续变量是可以取任意实数值的变量，表示系统中可以变化的量。二元变量是仅能取 0 或 1 的变量，通常用来表示决策问题中的选择或状态。

IMS-DFAPFSP 的 MILP 模型中符号、参数和变量定义如表 5-1 所示。

表 5-1 符号、参数和变量定义表

参数	描述
n	工件数量
pm	生产机器数量
am	装配机器数量
f	工厂数量
p	产品数量
$pp_{i,j}$	工件 j 在生产机器 i 上加工所需时间
pa_h	产品 h 装配所需时间
$G_{j,h}$	如果工件 j 属于产品 h 则为 1 否则为 0
M	一个足够大的正整数
符号	

j,k	表示工件，其中 $j,k=0,1,...,n$，0 为虚拟工件
i	表示生产机器，其中 $i=1,2,...,pm$
w	表示装配机器，其中 $w=1,2,...,am$
h,l	表示产品，其中 $h,l=0,1,...,p$，0 为虚拟产品
g	表示工厂，其中 $g=1,2,...,f$
二元变量	
$X_{j,k}$	如果工件 k 为工件 j 的前置加工工件则为 1，否则为 0，$\forall j,k$
$Z_{j,k}$	如果工件 k 为工件 j 的前置加工工件则为 1，否则为 0，$\forall j,k, j\neq 0, k\neq 0$
$Y_{h,l}$	如果产品 h 为产品 l 的前置装配产品则为 1，否则为 0，$\forall h,l$
$W_{h,l}$	如果产品 h 为产品 l 的前置装配产品则为 1，否则为 0，$\forall h,l,h\neq 0,l\neq 0$
Q_j	如果工件 j 为某个工厂的首个加工工件则为 1，否则为 0
E_h	如果产品 h 为某个装配机器的首个装配产品则为 1，否则为 0
连续变量	
$C_{i,j}$	工件 j 在生产机器 i 上的完工时间
CA_h	产品 h 的完工时间
H_j	辅助连续变量，工件 j 的加工顺序
L_h	辅助连续变量，产品 h 的装配顺序
$C_{\max}$	最大完工时间(Makespan)

5.1.4 基于多旅行商问题的 MILP 模型

Naderi[5]设计了六种分布式置换流水车间调度(Distributed Permutation FlowShop Scheduling Problem, DPFSP)的 MILP 模型，并通过大量实验验证了基于最小顺序序列(Minimum Sequence-Based Model)的 MILP 模型在性能上的优越性。在此基础上，Hatami[9]首次提出了基于 MSBM 的 DAPFSP 模型。在这之后，Hamzaday[11]对 Hatami 的 MSBM 模型进行了优化，将其与多旅行商问题(Multiple Traveling Salesman Problem, mTSP)相结合，提出了多旅行商的 DAPFSP 模型。本文进一步改进了 Hamzaday 提出的 mTSP 模型，提出了基于多旅行商的 IMS-DFAPFSP 模型。与 Hamzaday 的研究相比，IMS-DFAPFSP 模型不仅继承了先前模型在调度优化中的优点，还通过更合理和更少的约束来提升了调度效率和解决方案的质量。

以最小化 makespan 为目标的 mTSP 模型描述如下：

目标函数：

$$Minimise C_{\max} = \max_{h=1,2,\ldots,p} \{CA_h\} \tag{5-1}$$

约束：

$$\sum_{k=0,k\neq j}^{n} X_{k,j} = 1, \forall j, j \neq 0 \tag{5-2}$$

$$\sum_{j=0,j\neq k}^{n} X_{k,j} = 1, \forall k, k \neq 0 \tag{5-3}$$

$$\sum_{j=1}^{n} X_{0,j} = f \tag{5-4}$$

$$\sum_{k=1}^{n} X_{k,0} = f \tag{5-5}$$

$$X_{j,k} + X_{k,j} \leq 1, \forall j, k, j \neq k \neq 0 \tag{5-6}$$

$$\sum_{h=0,h\neq l}^{p} Y_{h,l} = 1, \forall l, l \neq 0 \tag{5-7}$$

$$\sum_{l=0,l\neq h}^{p} Y_{h,l} = 1, \forall h, h \neq 0 \tag{5-8}$$

$$\sum_{h=0}^{p} Y_{h,0} = am \cdot f \tag{5-9}$$

$$\sum_{l=0}^{p} Y_{0,l} = am \cdot f \tag{5-10}$$

$$Y_{h,l} + Y_{l,h} \le 1, \forall h, l, h \ne l \ne 0 \tag{5-11}$$

$$C_{1,j} \ge pp_{1,j}, \forall j, j \ne 0 \tag{5-12}$$

$$C_{i,j} \ge C_{i-1,j} + pp_{i,j}, \forall i, j, j \ne 0, i \ne 1 \tag{5-13}$$

$$C_{i,j} \ge C_{i,k} + pp_{i,j} + (X_{k,j} - 1) \cdot M, \forall i, j, k, j \ne k \ne 0 \tag{5-14}$$

$$CA_h \ge C_{pm,j} + (G_{j,h} - 1) \cdot M + pa_h, \forall h, j, h \ne 0, j \ne 0 \tag{5-15}$$

$$CA_h \ge CA_l + (Y_{l,h} - 1) \cdot M + pa_h, \forall h, l, h \ne l \ne 0 \tag{5-16}$$

公式(5-1)为 mTSP 模型的目标函数。约束(5-2)和约束(5-3)限制了每一个工件只能有一个前置加工工件和后置加工工件。约束(5-4)和约束(5-5)限制了每个工厂至少加工一个工件(不包括虚拟工件)。约束(5-6)表示工件 j 不能同时为工件 k 的前置加工工件和后置加工工件。约束(5-7)和约束(5-8)限制了每一个产品只能有一个前置装配产品和后置装配产品。约束(5-9)和约束(5-10)限制了每个装配工厂中的装配机器上至少装配一个产品(包括虚拟产品)。约束(5-11)表示产品 h 不能同时为产品 l 的前置装配产品和后置装配产品。约束(5-2)-约束(5-11)定义了 mTSP 中的 BV，mTSP 包含 $X_{j,k}$ 和 $Y_{h,l}$ 两个 BV。$X_{j,k}$ 的 BV 数量为 $(n+1)^2$，$Y_{h,l}$ 的 BV 数量为 $(p+1)^2$。因此，mTSP 的 BVs 数量总和为 $(n+1)^2 + (p+1)^2$。

公式(5-12)定义了每个工件在第一台生产机器上的完工时间要大于该工序所需时间。公式(5-13)定义了每个工件在第 i 台生产机器上的完工时间要大于该工件在前一台生产机器 $i-1$ 上的完工时间。公式(5-14)定义了工件 j 在生产机器 i 上的完工时间要大于前置加工工件 k 在生产机器 i 上的完工时间。公式(5-15)定义了每个产品 h 的完工时间要大于属于产品 h 的任意一个工件 j 在生产机器 pm 上的完工时间。公式(5-16)定义了产品 h 的完工时间要大于前置产品 l 在该装配机器上的完工时间。公式(5-12)-公式(5-16)定义了 mTSP 中的 CV，mTSP 包含 $C_{i,j}$ 和 CA_h 两个 CV。$C_{i,j}$ 的 CV 数量为 $n \cdot pm$，CA_h 的 CV 数量为 p。因此，mTSP 的 CVs 数量总和为 $n \cdot pm + p$。

为了更好的理解 mTSP 如何表示 IMS-DFAPFSP 的调度方案，列举了一个 IMS-DFAPFSP 调度案例。工件与产品的加工信息如表 2 所示。案例包含 9 个工件，2 台生产机器，2 台装配机器，2 个工厂，3 个产品。工件 4 和工件 9 属于产品 1。工件 2、工件 6 和工件 8 属于产品 2。工件 1、工件 3、工件 5 和工件 7 属于产品 3。假设 $X_{j,k}$ 和 $Y_{h,l}$ 定义如下：$X_{0,5}=X_{5,3}=X_{3,4}=X_{4,9}=X_{9,0}=X_{0,1}=X_{1,7}=X_{7,6}=X_{6,8}=X_{8,2}=X_{2,0}=1$。$Y_{0,3}=Y_{3,0}=Y_{0,1}=Y_{1,0}=Y_{0,2}=Y_{2,0}=1$，其余为 0。图 2 为采用 mTSP 模型方法定义的解决方案。图 3 为解决方案的甘特图。

表 5-2 调度案例的工件和产品的加工信息

工件 j									
i	1	2	3	4	5	6	7	8	9
$pp_{1,j}$	2	5	6	2	4	9	4	3	6
$pp_{2,j}$	6	1	2	3	2	3	4	4	4
产品 h									
		1			2			3	
pa_h		3			2			11	

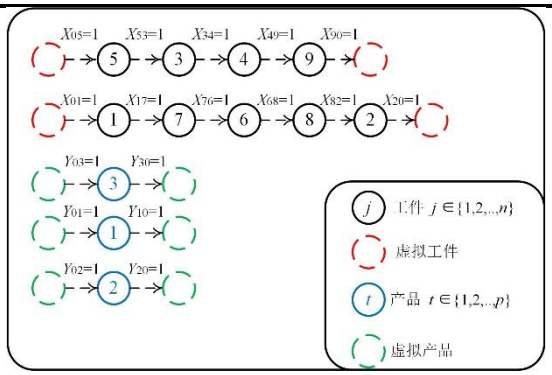

图 5-2　采用 mTSP 模型方法定义的解决方案

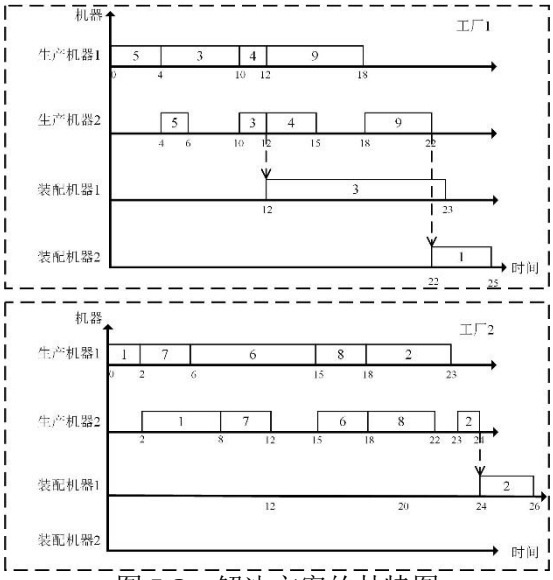

图 5-3　解决方案的甘特图

5.1.5 基于特殊位置的 MILP 模型

本节提出了一种基于特殊位置模型(Special Position-Based Model, SPBM)。与 mTSP 相比，SPBM 不再采用虚拟工件约束，而是通过创建两个决策变量来代替虚拟工件。通过标定特殊工件所在的特殊位置，SPBM 能够有效减少约束变量的数量，从而提高求解器的运行速度。此外，SPBM 还融入了 Miller-Tucker-Zemlin（MTZ）方法。MTZ 方法最初用于解决旅行商问题中的子环消除问题，通过引入辅助变量和约束条件，确保解的路径中不形成子环，从而保证模型的正确运行。在 SPBM 模型中，MTZ 方法的应用能够有效消除子环现象，确保调度过程的连续性和合理性。SPBM 模型通过优化决策变量和引入 MTZ 方法，不仅提升了算法的效率，还确保了模型的准确性和可靠性。这为 IMS-DFAPFSP 提供了一种新的有效解决方案。

以最小化 makespan 为目标的 SPBM 模型描述如下：

目标函数：

$$Minimise C_{\max} = \max_{h=1,2,\ldots,p} \{CA_h\} \tag{5-17}$$

约束：

$$\sum_{j=1}^{n} Q_j = f \tag{5-18}$$

$$\sum_{k=0,k\neq j}^{n} Z_{k,j} = 1, \forall j, j \neq 0 \tag{5-19}$$

$$\sum_{j=0,j\neq k}^{n} Z_{k,j} = 1, \forall k, k \neq 0 \tag{5-20}$$

$$H_j \leq n-1, \forall j, j \geq 2 \tag{5-21}$$

$$H_j - H_k + n \cdot Z_{j,k} \leq n-1, \forall j,k, j \geq 2, k \geq 2, j \neq k \tag{5-22}$$

$$H_1 = 1 \tag{5-23}$$

$$\sum_{h=1,h\neq l}^{p} W_{h,l} = 1, \forall l, l \neq 0 \tag{5-24}$$

$$\sum_{l=1,l\neq h}^{p} W_{h,l} = 1, \forall h, h \neq 0 \tag{5-25}$$

$$L_h \leq p-1, \forall p, p \geq 2 \tag{5-26}$$

$$L_h - L_l + p \cdot W_{h,l} \leq p-1, \forall h,l, h \geq 2, l \geq 2, h \neq l \tag{5-27}$$

$$L_1 = 1 \tag{5-28}$$

$$\sum_{h=1}^{p} E_h \leq f \cdot am \tag{5-29}$$

$$C_{1,j} \geq pp_{1,j}, \forall j, j \neq 0 \tag{5-30}$$

$$C_{i,j} \geq C_{i-1,j} + pp_{i,j}, \forall i, j, j \neq 0, i \neq 1 \tag{5-31}$$

$$C_{i,j} \geq C_{i,k} + (Z_{k,j} - 1) \cdot M - Q_j \cdot M + pp_{i,j}, \forall i, j, k, j \neq k \neq 0 \tag{5-32}$$

$$CA_h \geq C_{pm,j} + (G_{j,h} - 1) \cdot M + pa_h, \forall h, j, h \neq 0, j \neq 0 \tag{5-33}$$

$$CA_h \geq CA_l + (W_{l.h} - 1) \cdot M - E_h \cdot M + pa_h, \forall h, l, h \neq l \neq 0 \tag{5-34}$$

与 mTSP 的 MILP 约束模型相比，目标函数(5-17)以及约束(5-19)-约束(5-20)，约束(5-24)-约束(5-25)没有较大的变化，需要注意的是由于不再采用虚拟工件和虚拟产品，工件 j,k 和产品 h,l 的取值范围发生了变化，因此 CV 的数量也会发生变化。在 SPBM 中 CV 包含 $Z_{j,k}$，$W_{h,l}$，Q_j 和 E_h。$Z_{j,k}$ 的 BV 数量为 n^2，$W_{h,l}$ 的 BV 数量为 p^2，Q_j 和 E_h 的 BV 数量为 $n+p$。因此 SPBM 的 BV 数量为 $(n+1)n+(p+1)p$ 要远小于 mTSP 的 BV 数量。约束(18)定义了每个工厂都存在一个特殊工件即首个加工工件。约束(5-21)-约束(5-23)和约束(5-26)-约束(5-28)为 MTZ 方法消除工件和产品中可能出现的子环，以确保工件和产品能够按照正确的顺序加工和装配。

约束(5-29)表示可能存在装配机器空载的情况，造成这种情况的原因是部分 IMS-DFAPFSP 实例存在 $p < f \cdot am$ 。

约束(5-30)-约束(5-34)为 SPBM 定义 CV 约束。与 mTSP 相比没有较大的区别，需要注意的是约束(5-32)和约束(5-34)在判断前一个工件或产品的完工时间时需要保证当前工件为非特殊工件。由于 MTZ 方法的特殊性，SPBM 额外增加了两个 CV 即 H_j 和 L_h，因此与 mTSP 相比 SPBM 的 CV 数量要多 $n+p$ 。CV 数量的增加可能会导致 SPBM 在处理较大规模问题时运行速度变慢。

根据 5.1.4 所给出的案例，图 5-4 为采用 SPBM 模型方法定义的解决方案。其中决策变量定义如下：$Z_{5,3}=Z_{3,4}=Z_{4,9}=Z_{9,1}=Z_{1,7}=Z_{7,6}=Z_{6,8}=Z_{8,2}=Z_{2,5}=1$，$Q_5=Q_1=1$，$W_{3,1}=W_{1,2}=W_{2,3}=1$，$E_1=E_2=E_3=1$，其余为 0。由于此案例存在 $p < f \cdot am$ 的情况，此时装配机器存在冗余，产品 1、产品 2 和产品 3 都为特殊产品即装配机器上首个装配的产品。

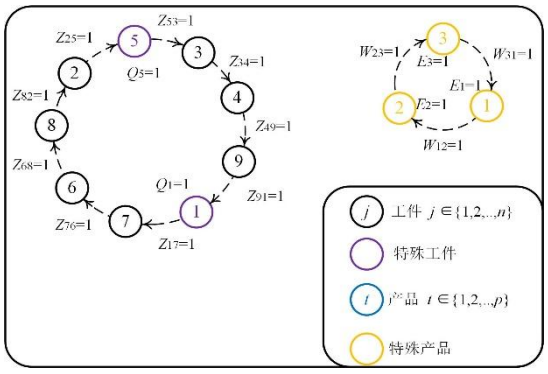

图 5-4　采用 SPBM 模型方法定义的解决方案

5.2 基于 SARSA-强化学习的生物地理学优化算法

5.2.1 SARSA 强化学习

强化学习（Reinforcement Learning, RL）算法通过与环境的交互，使智能体能够迭代学习，旨在最大化与其动作相关的奖励或最小化风险。RL 在多个问题领域中引起了广泛关注，并取得了显著成功。

马尔可夫决策过程（Markov Decision Process, MDP）是 RL 的基本模型，可

以用四元组$<S,A,Re,Pr>$来定义。其中，S表示状态空间，包含智能体可感知的所有可能环境状态；A表示动作空间，包含智能体在每个状态下可执行的所有可能动作；Re为奖励函数；Pr为状态转移概率。图 5-5 展示了智能体与环境之间的动态交互关系。

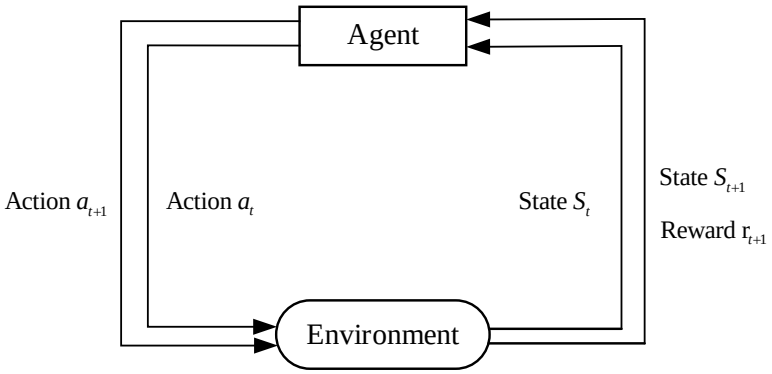

图 5-5　强化学习原理图

SARSA 学习（State-Action-Reward-State-Action）是最常用的无模型强化学习算法之一。它是一种基于策略的方法，通过学习策略生成训练数据，定义如下：

$$Q(s_t,a_t)\leftarrow Q(s_t,a_t)+\alpha(r_{t+1}+\gamma Q(s_{t+1},a_{t+1})-Q(s_t,a_t)) \tag{5-35}$$

其中，α为学习率，γ为折扣因子，r_{t+1}为智能体在状态s_t执行动作a_t时从环境中获得的奖励。

在强化学习的初始学习阶段，动作选择通常基于 Q 值。Q 值表示在每个状态下执行特定动作所关联的期望效用或长期奖励。最初，当智能体缺乏学习经验时，所有 Q 值均设为零，表示智能体没有任何先验知识或经验来指导其决策过程。随着时间推移，智能体通过最大化累积奖励来学习更优的决策，基于更新后的 Q 值不断优化动作选择。ε-贪婪策略被广泛采用，其表达式如下：如果随机数$rand<\varepsilon$，则随机选择一个动作a（探索）；否则，选择使 Q 值最大的动作a（利用），表示为：$a_t=\arg\max_{a_t}Q(s_{t+1},a_{t+1})$其中$\varepsilon$表示探索的概率，促使智能体尝试新的动作；$1-\varepsilon$表示利用当前知识选择最优动作的概率，以期获得最大奖励。

5.2.2 编码方式与接受准则

本章采用基于产品序列表示的编码方法。对于 IMS-DFAPFSP 问题，首先确定产品的装配序列Π_a（$\Pi_a=[\Pi_1,\Pi_2,...,\Pi_p]$），需要注意的是，假设产品的装配顺序是按产品 1、产品 2、……、产品 p 的顺序排列的，但实际的装配顺序可能并不一定遵循此顺序。接着，确定构成每个产品 p 的作业在生产工厂 pf 上的产品序列Π_p（$\Pi_p=[\Pi_p(1),\Pi_p(2),...,\Pi_p(pf)]$）。以第 5.1.4 节中的案例为例，生产工厂 $pf=2$，作业数量 $j=12$，产品数量 $p=3$。产品 1 的加工序列为$\Pi_1=[\Pi_1(1),\Pi_1(2)]$，其中$\Pi_1(1)=[1,3]$，$\Pi_1(2)=[10]$；产品 2 的加工序列为$\Pi_2=[\Pi_2(1),\Pi_2(2)]$，其中$\Pi_2(1)=[4,5]$，$\Pi_2(2)=[8,11]$；产品 3 的加工序列为$\Pi_3=[\Pi_3(1),\Pi_3(2)]$，其中$\Pi_3(1)=[6,12]$，$\Pi_3(2)=[2,7,9]$。因此，该案例中的产品装配顺序为$\Pi_a=[\Pi_1,\Pi_3,\Pi_2]$。图 5-6 是该案例编码方法的示意图。

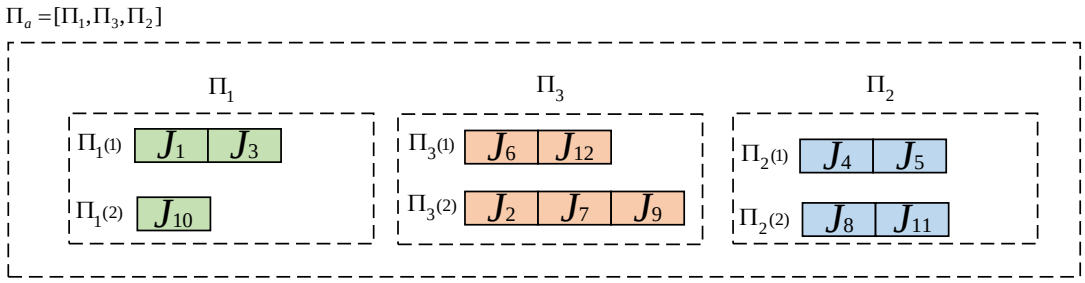

图 5-6 编码方式示意图

解码过程如下所述。按照产品装配顺序将Π_a中的作业分配到每个工厂，并为作业添加模糊加工时间，以获得模糊加工计划。选择产品所属作业的模糊加工时间最大的生产工厂 pf 作为该产品的装配工厂 af（即 $pf=af$）。同时，该作业的最大模糊加工时间即为该产品的模糊装配开始时间。通过加上该产品的模糊装配时间，可以得到模糊装配完成时间。选择装配工厂中产品的最大模糊装配时间即为解码得到的结果。为了更好地理解解码方式，以第 5.1.4 节中的案例为例。首先，根据Π_a将所有作业分配到生产工厂，并计算每个产品的装配时间。如图 5-7 所示，计算每个产品在每个工厂中的装配开始时间。最后，选择每个产品在

每个工厂中的最大装配开始时间作为该产品的装配开始时间，用参数 $\mathcal{U}_k$ 表示。在确定每个产品的 $\mathcal{U}_k$ 后，需要对产品进行装配。产品开始装配的工厂必须是 $\mathcal{U}_k$ 中选择作业完成时间最大的工厂。以图 5-7 为例，产品 1 的装配工厂应为 $\mathcal{U}_1=9$ 中选择作业 3 的工厂，即 $pf=1$，工厂 1。产品 3 的装配工厂为工厂 2，产品 2 的装配工厂为工厂 2。

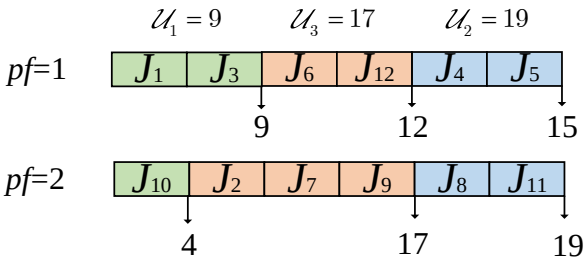

图 5-7 解码过程示意图

在确定产品的装配工厂后，需要为产品分配装配机器。由于装配工厂是相同的且装配机器也是相同的，因此产品可以在工厂中的任意装配机器上进行装配。分配规则如下：优先将产品分配到空闲状态的装配机器上进行装配；如果所有机器均被占用，则将产品分配到所有装配机器完成装配后装配时间最短的机器上。在确定算法的编码和解码方法后，需要为算法提供一些初始解以供搜索。初始化方法采用启发式方法，具体操作将在第 5.3.2 节中展示。

通常，算法优化的成功与否通过观察目标函数是否优化来判断。本文提出了一种接受准则来替代传统方法。该接受准则不仅判断优化是否成功，还增加了解的多样性，以提高算法的搜索能力。此外，还提出了两种接受准则，以进一步增强种群多样性并提高算法的搜索能力。首先，设 $MS_{\max}=\max\{FC_{\lambda^A_{af},p}\}$，$MS_{\max}$ 表示当前所有产品中的最大装配时间，$MS'_{\max}$ 表示优化后所有产品中的最大装配时间。接着，设 $MS_{rand}=rand\{\max\{FC_{\lambda^A_{af},p}\}\}$，$MS_{rand}$ 表示随机选择一个除最大装配时间产品外的产品的最大装配时间，MS'_{rand} 表示优化后随机选择一个除最大装配时间产品外的产品的最大装配时间。

需要注意的是，MS_{rand} 和 MS'_{rand} 与 $MS_{\max}$ 和 $MS'_{\max}$ 不同。MS_{rand} 和 MS'_{rand} 仅考

虑随机产品序列的最大装配时间。换句话说，假设所有作业都被清除，仅保存构成随机产品的分配部分。在这种情况下，最大装配时间为 MS_{rand} 和 MS'_{rand}。最后，提出了两种接受准则。

(1)如果装配时间有所改善，则接受优化，即当 $MS'_{\max} - MS_{\max} < 0$ 时接受优化结果。

(2)如果优化后的个体中存在良好的作业序列，则接受优化，即当 $(MS'_{\max} + MS'_{rand}) - (MS_{\max} + MS_{rand}) < 0$ 时接受优化结果。

为了更好地理解该接受准则的运作方式，图 5-8 提供了一个原始解和优化解作为示例。

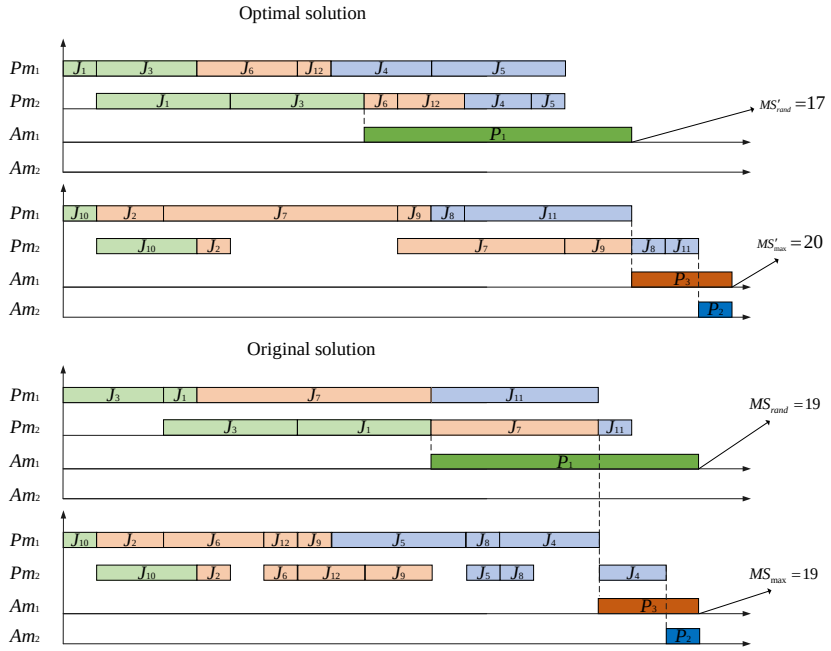

图 5-8 算法接受准则示意图

图 5-8 展示了原始解的编码甘特图。可以看出，工厂 1 的完成时间为 19，工厂 2 的完成时间为 19。由于时间相同，随机选择一个工厂作为关键工厂（即完成时间最长的工厂）。假设选择工厂 2 作为关键工厂，其完成时间为 $MS_{\max}$。在关键工厂之外随机选择另一个工厂。由于图 5-8 所示的案例中只有两个工厂，因此选择工厂 1 作为随机选择的工厂，其完成时间为 MS_{rand}。图 5-8 还展示了优化解的甘特图。优化解的关键工厂完成时间为 20，工厂 1 的完成时间为 17。因此，

$MS'_{\max}=20$，$MS'_{rand}=17$。可以发现，优化解的最大完成时间为20。如果使用接受准则（1）作为判断条件，则该解未被优化。然而，如果采用接受准则（2）作为判断条件，$(MS'_{\max}+MS'_{rand})-(MS_{\max}+MS_{rand})=-1$满足接受准则（2）的条件。因此，优化成功，优化解应被保留。通过这种接受准则，算法可以在保持种群多样性的同时，避免盲目追求最优解而牺牲潜在优秀解的情况。

5.2.3 启发式方法

启发式方法是一种通过使用某些规则快速有效地形成解的技术。启发式方法可以有效减少生成冗余解的概率，并提供高效的搜索方向。常见的启发式方法包括最短加工时间（Short Processing Time, SPT）。SPT 指的是将作业的加工时间从小到大排序。

本文提供的启发式方法结合了 SPT 和 *NR* 工厂分配方法，具体描述如下：

(1)将所有的作业保存到一维编码Π中。根据 *NR*1/ *NR*2 将Π中的作业分配到 *pf* 个工厂中。得到 *pf* 维编码$\Pi_p=[\Pi_p(1),\Pi_p(2),...,\Pi_p(pf)]$。

(2) 重复操作（1），直到考虑所有产品。得到Π_a（$\Pi_a=[\Pi_1,\Pi_2,...,\Pi_p]$）。

(3) 计算Π_a中每个Π_p的$C_{\max}$。使用 SPT 对Π_a中的Π_p进行产品顺序排序。将Π_a中Π_p的所有维度的作业组合起来，得到 *pf* 维编码$\lambda^P_{pf,p}$ $\lambda^P_{pf,p}=[\lambda^P_{pf,p}(n_1),\lambda^P_{pf,p}(n_2),...,\lambda^P_{pf,p}(n_{pf})]$。

(4) 根据 *NR*1/ *NR*2 将产品 *p* 分配到其最后一个作业完全处理的装配工厂中的装配机器 *Am* 上。

(5) 重复(4)，直到考虑所有产品。得到产品装配序列$\lambda^A_{af,p}$，$\lambda^A_{af,p}=[\lambda^A_{af,p}(n_1),\lambda^A_{af,p}(n_2),...,\lambda^A_{af,p}(n_{af})]$。

通过 SN 方法得到一个单一解。如果要在群体智能算法中使用 SN 方法，应删除步骤(3)中通过 SPT 对$C_{\max}$进行排序的步骤。然后，通过随机打乱产品顺序Π_a，可以得到一批质量较好的多个个体。

5.2.4 全局与局部搜索方式

全局搜索（global search, GS）是对当前解的广度搜索。使用全局搜索可以进一步提高解的质量。如图 5-9 所示，提出了三种 GS 方法（GS_1、GS_2、GS_3）。GS 的总体思想是通过将优秀个体的部分作业处理序列或产品生产序列替换为较差个体的相应部分，从而优化个体解的质量。以第 5.3.1 节中的案例为例，GS 操作的详细步骤如下：

(1) GS_1：GS_1 改变较差个体中的作业生产序列和产品装配序列。以第 5.3.1 节中的案例为例，假设 $\Pi_a=[\Pi_1,\Pi_3,\Pi_2]$ 是一个优秀个体。优秀个体的产品装配序列为 $[1,3,2]$，作业生产序列为 $\Pi_1(1)=[1,3]$，$\Pi_1(2)=[10]$，$\Pi_2(1)=[4,5]$，$\Pi_2(2)=[8,11]$，$\Pi_3(1)=[6,12]$，$\Pi_3(2)=[2,7,9]$。假设另一个较差个体为 $\Pi_a=[\Pi_2,\Pi_1,\Pi_3]$，其产品装配序列为 $[2,1,3]$，作业生产序列为 $\Pi_1(1)=[1]$，$\Pi_1(2)=[10,3]$，$\Pi_2(1)=[8,5,4]$，$\Pi_2(2)=[11]$，$\Pi_3(1)=[9,7]$，$\Pi_3(2)=[2,6,12]$。随机选择优秀个体产品装配序列中的连续部分，假设选择 [3,2]。GS_1 将产品装配序列保存为 $\Pi_a=[0,\Pi_3,\Pi_2]$ 并将其余部分替换为 0。在较差个体的 Π_a 中移除选中的装配序列 $[3,2]$ 并替换为 0，得到 $\Pi_a=[0,\Pi_1,0]$。将较差个体中非零的产品装配序列 Π_a 从左到右依次插入优秀个体的 $\Pi_a=[0,\Pi_3,\Pi_2]$ 中。更新后的较差个体为 $\Pi_a=[\Pi_1,\Pi_3,\Pi_2]$。

(2) GS_2：GS_2 与 GS_1 类似，但 GS_2 仅改变产品装配序列中的作业处理序列，而不保存产品装配序列。以 GS_1 中的个体为例，优秀个体为 $\Pi_a=[\Pi_1,\Pi_3,\Pi_2]$。随机选择若干产品的作业处理序列。假设 GS_2 选择产品 2 Π_2 和产品 3 Π_3。GS_2 将较差个体中的 Π_2 和 Π_3 替换为优秀个体中的 Π_2 和 Π_3。更新后的较差个体为 $\Pi_a=[\Pi_1,\Pi_3,\Pi_2]$。较差个体的产品装配序列不变。GS_2 改变了较差个体中部分产品的作业处理序列。

(3) GS_3：GS_3 与 GS_1 类似。然而，GS_3 仅改变产品装配序列。在 GS_1

中，随机选择优秀个体的产品装配序列[3,2]，然后将优秀个体中产品装配序列[3,2]的作业处理序列移动到较差个体中。然而，在 GS_3 中，优秀个体的产品装配序列是较差个体的学习目标，较差个体不再学习优秀个体中的作业处理序列。较差个体中所有产品的作业处理序列保持不变。GS_3 仅改变较差个体中的产品装配序列。

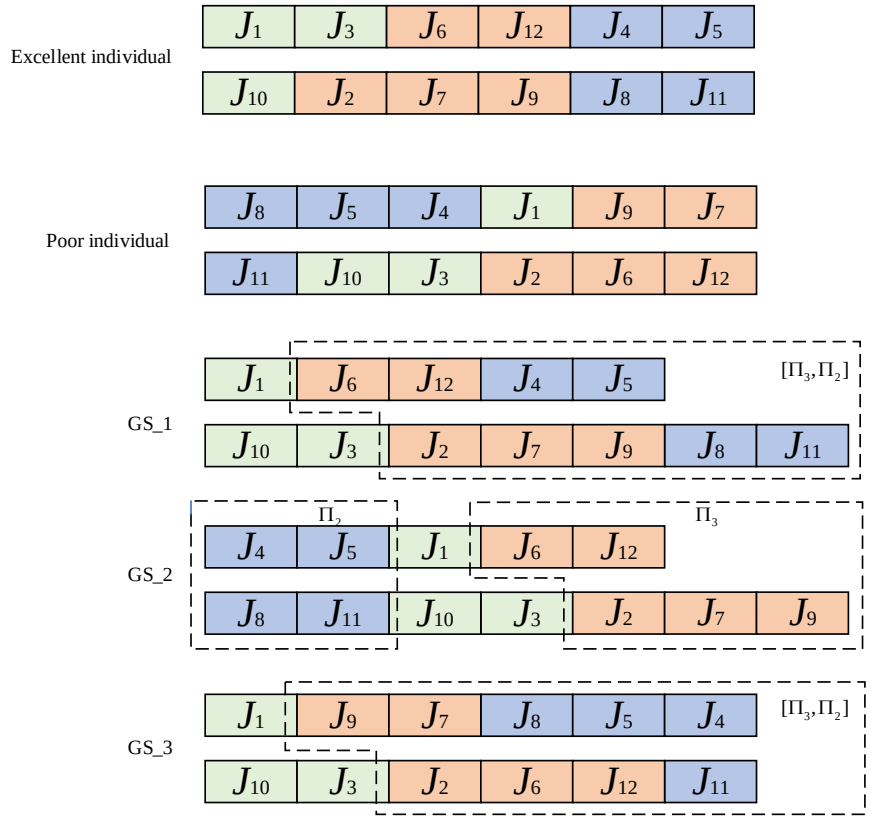

图 5-9 GS 搜索示意图

为了挖掘到更好的解，可以使用局部搜索（local search, LS）对当前解进行更深层次的搜索。如图5-10所示，提出了三种局部搜索方法（LS_1、LS_2、LS_3）。

以第 5.3.1 节中的案例为例，LS 操作如下：

LS_1：LS_1 通过交换不同产品之间的生产序列来寻找更好的解。案例 $\Pi_a=[\Pi_1,\Pi_3,\Pi_2]$，从 Π_a 中的第一个 Π_1 开始，依次与 Π_3 和 Π_2 交换，直到找到更好的解。如果第一个 Π_1 没有找到更好的解，则选择第二个 Π_3，并依次与后面的 Π_2 交换，直到找到更好的解。直到考虑完装配序列 Π_a 中的所有产品序列 Π_p。

LS_2：LS_2 考虑 Π_a 中 Π_p 的作业处理序列。案例 $\Pi_a=[\Pi_1,\Pi_3,\Pi_2]$，随机选择一个 Π_p。假设选择的是 Π_1，其中 $\Pi_1=[\Pi_1(1),\Pi_1(2)]$，$\Pi_1(1)=[1,3]$，$\Pi_1(2)=[10]$。选择 Π_1 中第一个生产工厂的作业集 $\Pi_1(1)$，并选择 $\Pi_1(1)$ 中的第一个作业 J_1。将 J_1 与同一生产工厂中的其他作业交换。如果找到更好的解，则退出 LS_2 搜索，否则选择另一个作业。考虑完第一个生产工厂中的所有作业。如果在第一个生产工厂中没有找到更好的解，则选择第二个生产工厂。直到考虑完所有生产工厂。

LS_3：LS_3 与 LS_2 类似，但 LS_3 仅交换不同工厂之间的作业处理序列。再次假设随机选择 Π_1。$\Pi_1=[\Pi_1(1),\Pi_1(2)]$，其中 $\Pi_1(1)=[1,3]$，$\Pi_1(2)=[10]$。将 Π_1 中第一个工厂的作业处理序列 $\Pi_1(1)$ 与第二个工厂的作业处理序列 $\Pi_1(2)$ 交换。得到 $\Pi_1(1)=[10]$，$\Pi_1(2)=[1,3]$。如果找到更好的解，则退出 LS_3，否则选择另一个生产工厂，直到考虑完所有生产工厂。

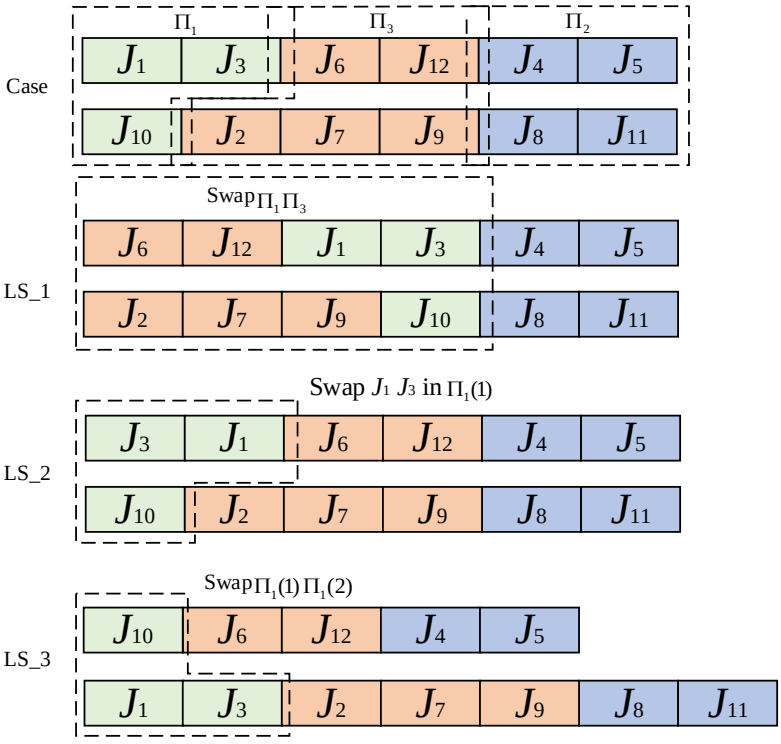

图 5-10 LS 搜索示意图

5.2.5 SARSA 强化学习过程

在 SARSA 学习算法中，包含状态 s_t 、动作 a_t 、奖励 r_t 以及动作选择策略。在本研究中，环境状态由栖息地评估结果表示，而动作则被描述为全局搜索解接受准则的组合。此外，还引入了一种新定义的奖励函数。

设计了一个栖息地优化成功率 S_{Iter} 来判断栖息地 H 的状态。栖息地 H 在第 $Iter$ 代的最优成功率的公式定义如下：

$$S_{Iter}=\sum_{H=1}^{N}(MAT_H^{Iter}-MAT_H^{Iter-1})/N/\overline{MAT_H^{Iter}}\times 100\% \tag{5-36}$$

MAT_H^{Iter} 表示 $Iter$ 代中栖息地 H 的最大完工时间。$\overline{MAT_H^{Iter}}$ 表示 $Iter$ 代的平均最大完工时间。

Bh 用于描述栖息地 H 的改进情况。如果 H 在第 $Iter$ 代有所改进，则 $Bh=1$；否则， $Bh=0$。S_{Iter} 有三种情况： $S_{Iter}>0$ 、 $S_{Iter}=0$ 和 $S_{Iter}<0$ 。根据 Bh 和 S_{Iter} ，定义了 SARSA 学习的六种状态，如表 5-3 所示。

表 5-3 状态表

状态	状态描述	状态	状态描述
1	$S_{Iter}<0$ and $Bh=1$	4	$S_{Iter}<0$ and $Bh=0$
2	$S_{Iter}=0$ and $Bh=1$	5	$S_{Iter}=0$ and $Bh=0$
3	$S_{Iter}>0$ and $Bh=1$	6	$S_{Iter}>0$ and $Bh=0$

接受准则并未专门设计为保证当前代的最优解一定优于前一代的最优解，因此 Bh 有两种可能的结果。 S_{Iter} 有三种可能的结果：种群优化成功率大于 0、等于 0 或小于 0。将 S_{Iter} 与 Bh 结合，可以得到六种状态。状态 1：$S_{Iter}<0$ 且 $Bh=1$；状态 2：$S_{Iter}=0$ 且 $Bh=1$；状态 3：$S_{Iter}>0$ 且 $Bh=1$；状态 4：$S_{Iter}<0$ 且 $Bh=0$；状态 5： $S_{Iter}=0$ 且 $Bh=0$；状态 6： $S_{Iter}>0$ 且 $Bh=0$。显然，状态 1 是最优的，状态 6 是最差的。使用状态来反馈算法执行动作时的搜索效果。

在本文中，将三种 GS 方法作为算法的三个动作 a_1、a_2 和 a_3。将 SARSA 学习集成到 BBO 的迁移过程中。如果 BBO 在执行迁移过程后从 $State_1$ 转移到 $State_2$（$State_1 > State_2 | State_g \in [1,6]$），则该动作将受到奖励，否则将受到惩罚。奖励函数定义为 $r_{t+1} = State_1 - State_2$。直接使用 ε−greedy 来选择动作。当算法首次搜索时，Q 表的初始值为 0，动作为 a_1。

以下示例展示了 Q 表的更新过程。假设上一次的动作为 a_2，状态为 $State_2$。在执行动作 a_2 后，状态变为状态 3。通过 $\varepsilon - greedy$ 选择动作 a_1。假设 $\alpha = 0.1$ 且 $\gamma = 0.9$，根据公式 (5-36)，更新后的 $Q(2,2) = 8.729$。表 5-4 描述了 Q 表更新前后的过程。

表 5-4 Q 表更新过程

	Q 表更新前				Q 表更新后		
	a_1	a_2	a_3		a_1	a_2	a_3
$State_1$	9.835	9.062	9.438	$State_1$	9.835	9.062	9.438
$State_2$	7.374	**9.075**	7.460	$State_2$	7.374	**8.729**	7.460
$State_3$	7.355	8.362	7.483	$State_3$	7.355	8.362	7.483
$State_4$	8.931	7.635	8.125	$State_4$	8.931	7.635	8.125
$State_5$	9.616	7.233	7.941	$State_5$	9.616	7.233	7.941
$State_6$	8.252	7.735	9.296	$State_6$	8.252	7.735	9.296

5.2.6 算法流程

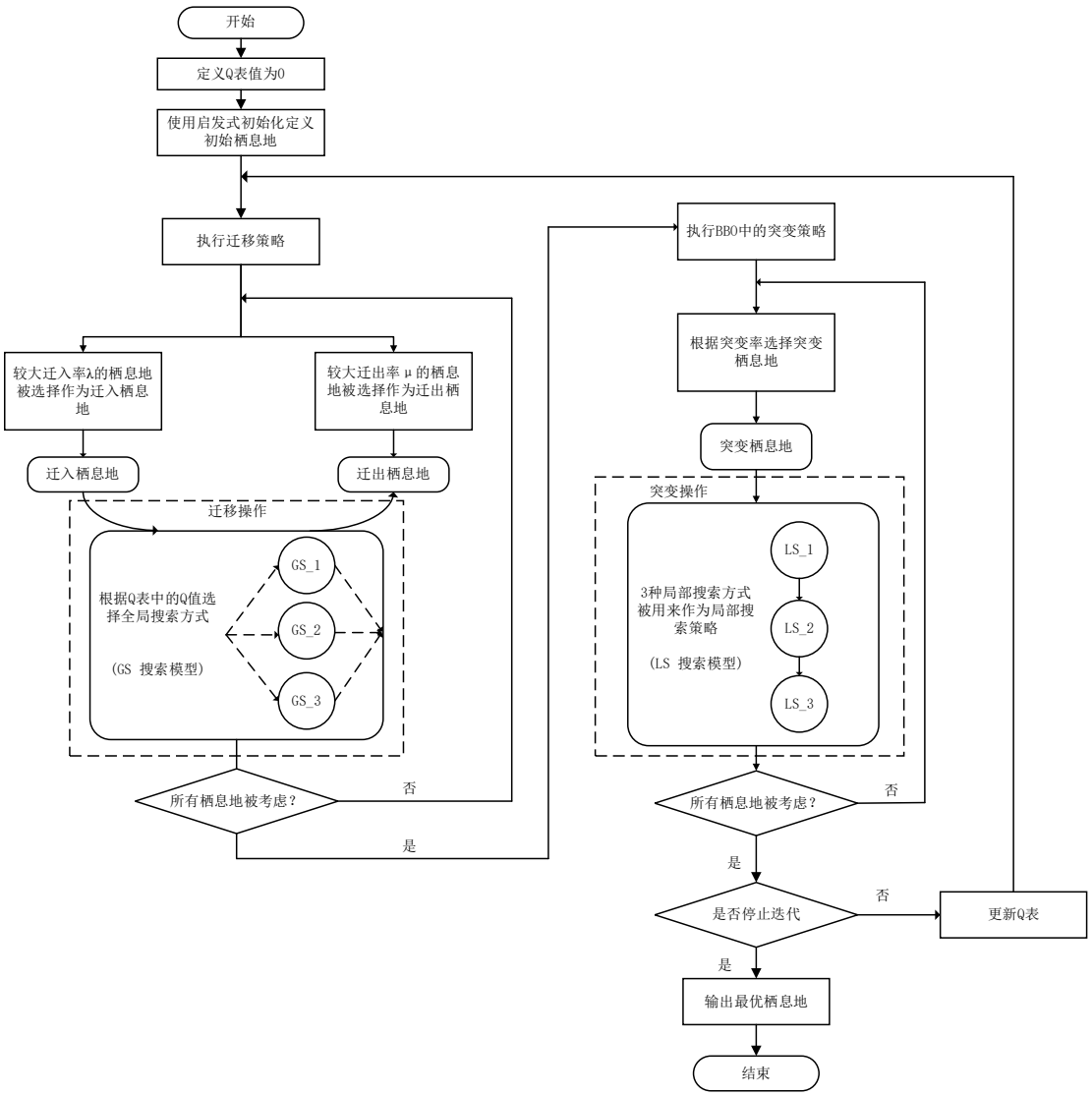

图 5-11 SARSA-BBO 算法流程图

图 5-11 为 SARSA-BBO 的算法流程图。在执行算法操作之前，需要定义 Q 表的值为空。同时根据 5.2.3 节的启发式方法为算法在搜索初期提供一批冗余解较少的初始栖息地作为算法的初始解。当执行迁移操作时，SARSA-BBO 会根据当前的状态判断采用哪一种动作作为算法的全局搜索方式，重复执行直到所有栖息地被考虑。本文设计了 3 种突变算子作为算法的局部搜索方式以此来使得算法跳出局部最优。每一次迭代完成后都需要即时更新 Q 表，当满足最大迭代次数时停止更新 Q 表，输出最优栖息地。

5.3 仿真实验

本节为实验仿真部分，第 5.4.1 节通过田口实验方法为 SARSA-BBO 找到最佳参数组合。第 5.4.2 节则通过测试两种 MILP 模型的表现，评估其在不同 IMS-DFAPFSP 问题规模下的效果。第 5.4.3 节将对本文提出的 SARSA-BBO 与两种工厂分配方式相结合进行算法性能的对比实验，以验证所提算法的有效性与合理性。第 5.4.4 节则针对敏感性分析实验展开，旨在探讨 IMS-DFAPFSP 中关键参数的影响，并基于实验结果提出管理方面的见解和建议。

IMS-DFAPFSP 将生产工厂与装配工厂视为一个集成制造系统，并结合柔性装配机器进行研究。由于目前尚无文献专门研究 IMS-DFAPFSP 问题，因此本研究参考了 Hatami[9]所提出的大规模 DAPFSP 基准实例进行测试实验，相关案例数据可通过访问 http://soa.iti.es.查询。共有 2430 个 IMS-DFAPFSP 测试实例，其中 $n \in [100,200,500]$，$pm \in [5,10,20]$，$am \in [2,3,4]$，$f \in [4,6,8]$，$p \in [30,40,50]$，每种组合有 10 个($d \in [1,...,10]$)不同的案例。工件的处理时间在$[1,99]$范围之间取值，产品的装配时间在$[1\times\Psi_h, 99\times\Psi_h]$范围之间取值，这个范围是生产调度研究中常见的处理时间取值范围。每个 IMS-DFAPFSP 实例可以通过 $n\_pm\_am\_f\_p\_d$ 表示。

对于每个实例，经过 6 种算法运行后得到的最优 makespan 记为$\mathbb{C}^*$，当前算法运行的结果记为$\mathbb{C}$。实验结果通过相对增幅百分比（Relative Percentage Increase, RPI）进行对比，RPI 的定义如公式(5-37)所示。

$$\mathrm{RPI} = \frac{\mathbb{C} - \mathbb{C}^*}{\mathbb{C}^*} \times 100\% \tag{5-37}$$

本节所有的仿真实验均在一台配备 16.0 GB 内存和 2.8 GHz CPU 处理器的笔记本电脑上进行。算法的对比实验在 Microsoft Visual C++ 2022 软件环境中运行，而 MILP 模型的计算则通过 CPLEX 12.7.1 求解器进行。CPLEX 的运行时间依据问题的工件数量和装配机器的规模进行调整，求解器的最大运行时间被限制为$n\times am$秒，其中n代表工件数量，am代表装配机器的数量。

5.3.1 参数实验

为了确保实验的准确性，设计了田口实验以获得 SARSA-BBO 的最佳参数。SARSA-BBO 的参数包括栖息地数量 N、最大变异率 $M_{\max}$、最大迭代次数 $Iter_{\max}$、学习率 α、折扣因子 γ 和贪婪系数 ε。对于 SARSA-BBO，提供了五组不同精度的参数组合。测试了 IMS-DFAPFSP 的案例 24_4_4_4。每个案例测试 5 次，并记录$\mathbb{C}$。表 5-5 展示了参数的水平。表 5-6 展示了正交表 $L_{25}(5^6)$。

表 5-5 SARSA-BBO 参数等级

参数	参数等级				
	1	2	3	4	5
N	50	70	90	110	130
$M_{\max}$	0.25	0.30	0.35	0.4	0.45
$Iter_{\max}$	50	80	110	140	170
α	0.10	0.20	0.30	0.40	0.50
γ	0.20	0.30	0.40	0.50	0.60
ε	0.10	0.20	0.30	0.40	0.50

表 5-6 $L_{25}(5^6)$ 正交表

SARSA-BBO 参数组合	参数等级						$\mathbb{C}$
	N	$M_{\max}$	$Iter_{\max}$	α	γ	ε	
1	1	1	1	1	1	1	2256
2	1	2	2	2	2	2	2227
3	1	3	3	3	3	3	2215
4	1	4	4	4	4	4	2242
5	1	5	5	5	5	5	2286
6	2	1	2	3	4	5	2249
7	2	2	3	4	5	1	2217
8	2	3	4	5	1	2	2224
9	2	4	5	1	2	3	2216

10	2	5	1	2	3	4	2183
11	3	1	3	5	2	4	2220
12	3	2	4	1	3	5	2215
13	3	3	5	2	4	1	2193
14	3	4	1	3	5	2	2187
15	3	5	2	4	1	3	2237
16	4	1	4	2	5	3	2206
17	4	2	5	3	1	4	2234
18	4	3	1	4	2	5	2180
19	4	4	2	5	3	1	2172
20	4	5	3	1	4	2	2167
21	5	1	5	4	3	2	2155
22	5	2	1	5	4	3	2179
23	5	3	2	1	5	4	2223
24	5	4	3	2	1	5	2182
25	5	5	4	3	2	1	2143

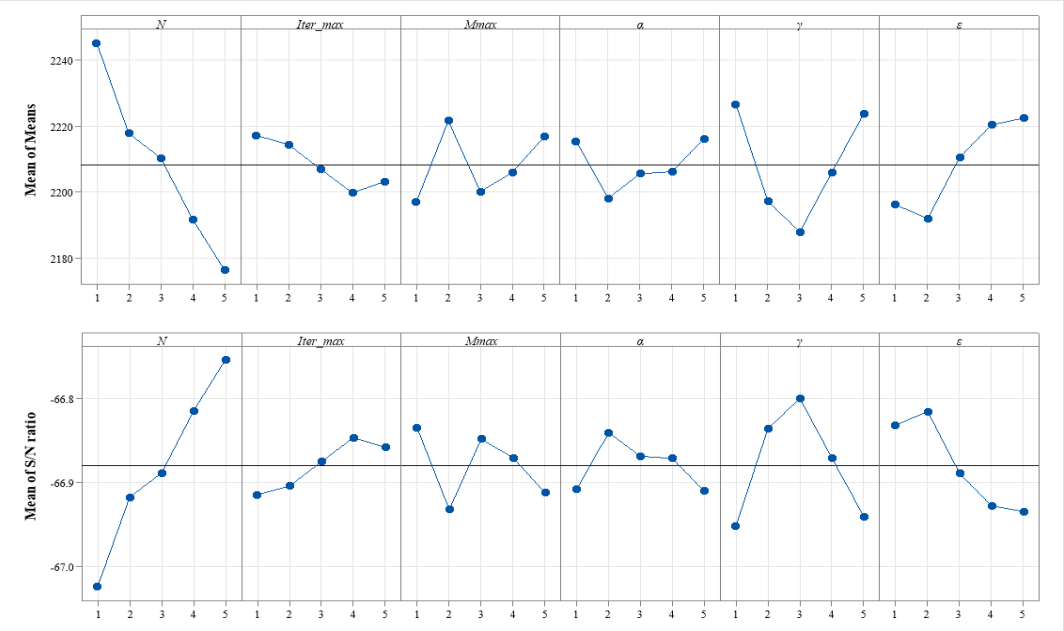

图 5-12 均值和 S/N 主效应图

如图 5-12 所示，参数设置为 $N=130$、$M_{\max}=0.25$、$Iter_{\max}=140$、$\alpha=0.2$、

$\gamma = 0.4$ 和 $\varepsilon = 0.2$ 的 SARSA-BBO 表现优于其他参数组合的 SARSA-BBO，因此采用上述设置。

5.3.2MILP 模型评估

部分 IMS-DFAPFSP 实例被选取在规定的实际按下运行，两种问题模型分别进行五次独立测试，记录每次运行过程中获得的最优 makespan 和平均 CPU 运行时间。表 4 和表 5 记录了 mTSP 和 SPBM 求解部分 IMS-DFAPFSP 实例的运行结果。图 5-13 为模型在不同装配机器下具有 95%置信区间的均值图。

表 5-7 mTSP 模型的最优值，RPI 值和 CPU 运行时间

IMS-DAPFSP 实例	$\mathbb{C}^*$	RPI (%)	CPU 运行时间 (秒)
100_5_2_4_30_1	2463	25.41	200.60
100_5_2_4_40_1	2354	25.21	200.58
100_5_2_4_50_1	2571	37.27	200.54
100_5_3_4_30_1	2101	6.98	300.92
100_5_3_4_40_1	1974	5.00	300.67
100_5_3_4_50_1	2142	14.55	301.07
100_5_4_4_30_1	1954	-0.51	401.06
100_5_4_4_40_1	1989	5.80	401.08
100_5_4_4_50_1	2107	12.67	400.96
100_10_2_4_30_1	3013	32.61	200.48
100_10_2_4_40_1	3107	33.52	200.74
100_10_2_4_50_1	2808	16.95	200.53
100_10_3_4_30_1	2669	17.89	300.56
100_10_3_4_40_1	2746	18.01	300.57
100_10_3_4_50_1	2928	21.95	300.62
100_10_4_4_30_1	2446	8.04	400.69
100_10_4_4_40_1	2698	15.94	400.61
100_10_4_4_50_1	2583	7.58	400.59

表 5-8 SPBM 模型的最优值，RPI 值和 CPU 运行时间

IMS-DAPFSP 实例	$\mathbb{C}^*$	RPI (%)	CPU 运行时间 (秒)
100_5_2_4_30_1	2379	21.13	200.52
100_5_2_4_40_1	2371	26.12	200.62
100_5_2_4_50_1	2494	33.16	200.51
100_5_3_4_30_1	2522	28.41	300.59
100_5_3_4_40_1	1910	1.60	300.62
100_5_3_4_50_1	2240	19.79	300.54
100_5_4_4_30_1	2111	7.48	400.62
100_5_4_4_40_1	1954	3.94	400.76
100_5_4_4_50_1	2092	11.87	400.89
100_10_2_4_30_1	2741	20.64	201.15
100_10_2_4_40_1	2950	26.77	200.50
100_10_2_4_50_1	2849	18.66	200.31
100_10_3_4_30_1	2601	14.89	300.56
100_10_3_4_40_1	2574	10.61	302.00
100_10_3_4_50_1	2800	16.62	300.70
100_10_4_4_30_1	2623	15.86	400.67
100_10_4_4_40_1	2590	11.30	400.58
100_10_4_4_50_1	2525	5.16	400.61

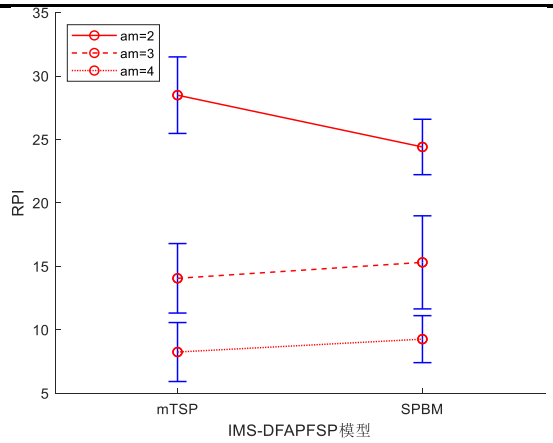

图 5-13 2 种 MILP 模型在不同装配机器下具有 95%置信区间的均值图

结合表 5-7、表 5-8 和图 5-13 可以观察到，在求解装配机器规模较小的 IMS-

DFAPFSP 问题时，SPBM 表现较为优异。然而，随着问题复杂度的增加，mTSP 的效果逐渐优于 SPBM。这一趋势主要归因于 SPBM 中的连续变量数量较多，随着问题规模的扩大，模型的复杂度也随之增加，导致运行时间显著延长。此外，SPBM 在装配机器数量为 2 和 4 时，其置信区间小于 mTSP，表明 SPBM 在这些问题规模下具有较高的稳定性。

5.3.3 对比实验

在本节中，除了测试 SARSA-BBO 与 VNS 和 IG 的对比外，还测试了使用三种不同全局搜索方法的 BBO。总共 6 种算法测试了 108 个案例，每个案例测试 10 次。同时，所有算法的解码模式均采用 $NR1/NR2$ 解码模式，并且所有算法均在三种不同的时间尺度下进行测试。总共需要进行 $10\times6\times2\times2\times108=25920$ 次实验测试。表 5-9、表 5-10 和表 5-11 展示了不同算法在不同时间参数下求解 IMS-DFAPFSP。

表 5-9 算法性能对比($C=0.2$)

算法 $NR1/NR2$	VNS	IG	BBO_GS1	BBO_GS2	BBO_GS3	SARSA-BBO
n						
100	**3374**/3379	3564/3493	7365/8376	7456/8367	7899/8760	3445/**3361**
200	4272/4262	4754/4615	9293/10742	9446/10665	10055/11184	**4213/4153**
500	8743/8742	9741/9535	12365/13174	12114/12905	12644/13662	**7901/7754**
f						
4	5941/5723	6637/6358	10462/11392	10470/11303	10903/11724	**5632/5507**
6	4985/4809	5402/5403	8887/10136	8874/9989	9496/10680	**4741/4672**
Pm						
5	5889/5905	6554/6175	10177/11489	10244/11320	10823/11951	**5664/5540**
10	5037/5017	5486/5586	9172/10040	9100/9971	9576/10453	**4709/4739**
Am						
3	5456/5480	6561/6434	7670/8708	7847/8761	8304/9036	**5339/5321**

4	5448/5462	5938/5792	9498/10439	9500/10199	9885/10961	**5240/5024**
5	5485/5441	5560/5416	11855/13147	11669/12977	12410/13609	**4980/4923**
平均值	5463/5422	6020/5881	9674/10764	9672/10646	10200/11202	**5186/5089**

表 5-10 算法性能对比($C=0.4$)

算法 *NR1*/ *NR2*	VNS	IG	BBO_GS1	BBO_GS2	BBO_GS3	SARSA-BBO
n						
100	3424/3337	3564/3493	3746/4038	3727/4025	3780/4222	**3316/3313**
200	**4163**/4177	4754/4615	4665/4957	4643/4908	4848/5062	4208/**4135**
500	8586/8666	9741/9535	9355/9650	9191/9451	9404/9482	**7830/7775**
f						
4	5849/5685	6637/6538	6534/6976	6460/6876	6673/7057	**5567/5551**
6	4861/4701	5402/5403	5309/5454	5248/5380	5349/5436	**4741/4614**
Pm						
5	5763/5761	6554/6175	6356/6845	6351/6748	6508/6849	**5510/5459**
10	4947/5009	5486/5586	5488/5584	5356/5508	5513/5661	**4798/4705**
Am						
3	**5337**/5375	6561/6434	5551/5781	5553/5739	5644/5790	5344/**5354**
4	5378/5375	5938/5792	5905/6122	5878/6101	5966/6197	**5118/5017**
5	5350/5406	5560/5416	6310/6742	6130/6544	6423/6779	**4999/4875**
平均值	5366/5349	6020/5881	5922/6215	5854/6128	6011/6255	**5153/5080**

表 5-11 算法性能对比($C=0.6$)

算法 *NR1*/ *NR2*	VNS	IG	BBO_GS1	BBO_GS2	BBO_GS3	SARSA-BBO
n						
100	3441/3376	3563/3492	3281/3342	3263/3262	3301/3321	**3237/3247**
200	4245/4107	4754/4614	4174/4250	4114/4172	4190/4270	**4005/4007**
500	8265/8242	9741/9534	8590/8835	8501/8602	8766/8763	**8023/7863**

f						
4	5705/5574	6637/6358	5849/5951	5785/5768	5923/5919	**5667/5477**
6	4768/4656	5401/5403	4847/5001	4801/4923	4916/4984	**4671/4485**
Pm						
5	5635/**5476**	6553/6175	5704/5810	5702/5666	5768/5759	**5502**/5500
10	4838/**4754**	5485/5586	4993/5142	4884/5025	5070/5145	**4836**/4831
Am						
3	5487/5278	6560/6433	5491/5615	5363/5412	5513/5519	**5181/5163**
4	5204/**5132**	5938/5792	5455/5503	5305/5319	5494/5457	**5097**/5153
5	5229/5181	5559/5416	5100/5309	5209/5305	5250/5379	**5018/4936**
平均值	5282/5178	6019/5880	5348/5476	5293/5345	5419/5452	**5124/5066**

表 5-9 展示了算法在时间参数 $c=0.2$ 下的性能表现。可以看出，SARSA-BBO 在大多数情况下取得了最佳结果，其性能明显优于其他算法。这表明 SARSA-BBO 通过强化学习增强了其搜索能力，能够在较短的搜索时间内找到更好的解。表 5-10 还比较了各算法在时间参数 $c=0.4$ 下的表现。SARSA-BBO 再次在大多数情况下表现优异，尤其是在作业数量 $n=500$ 时，其结果显著优于其他算法。这进一步证明了 SARSA-BBO 在处理复杂问题时的优越性。表 5-11 继续比较了算法在时间参数 $c=0.6$ 下的性能。可以看出，SARSA-BBO 在所有时间参数下均表现良好，其结果再次优于其他算法。这表明 SARSA-BBO 不仅能在短时间内找到更好的解，在长时间下也能表现更优，具有较强的稳定性和可靠性。

为了进一步可视化算法的平均性能值，图 5-14、5-15 和 5-16 展示了不同算法在 IMS-DFAPFSP 问题上的 95% 置信水平的平均区间图。

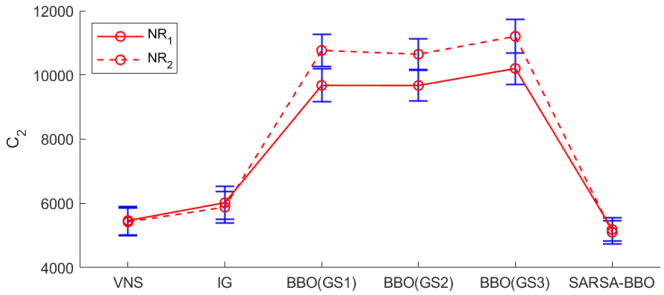

图 5-14 算法在 $C=0.2$ 中的 95%置信区间对比图

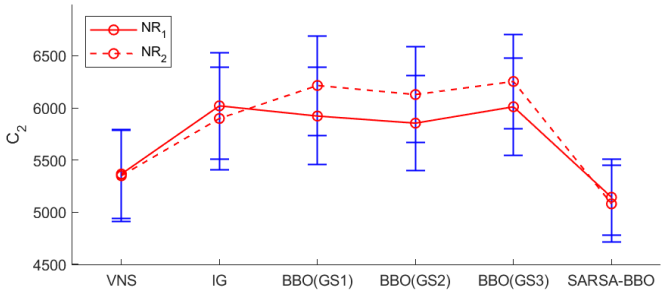

图 5-15 算法在 $C=0.4$ 中的 95%置信区间对比图

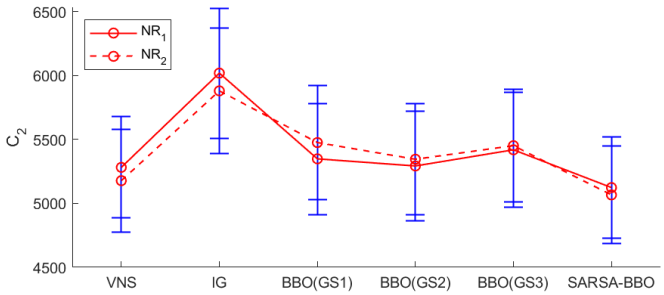

图 5-16 算法在 $C=0.6$ 中的 95%置信区间对比图

图 5-14、图 5-15 和图 5-16 展示了各算法在不同时间参数下的性能表现。可以看出，无论采用哪种解码方法，SARSA-BBO 始终优于其他算法。这表明 SARSA-BBO 能够更高效地搜索解空间并找到更好的解。从图中还可以明显看出，SARSA-BBO 在不同时间参数下的平均性能均优于其他算法，尤其是在长时间参数下，其优势更加显著。这表明 SARSA-BBO 通过引入强化学习，极大地提高了搜索效率和解的质量。

表 5-12、5-13 和 5-14 展示了算法在不同时间参数下求解 IMS-DFAPFSP 的能力，最优解以粗体标出。可以观察到，随着 n 的增加，IMS-DFAPFSP 的搜索能力得到了显著提升。随着 Am 的增加，多个不相关的装配机器相比单一装配机器问题能够有效缓解装配机器的负载。与 BBO_GS 相比，采用 SARSA 强化学习的 BBO 能够大幅提升原算法的搜索能力，这证明了 SARSA 强化学习的有效性。随着时间参数 C 的不断增加，算法有更多时间进行搜索，IMS-DFAPFSP 的解也逐渐变好。还可以观察到，无论是 $C=0.2$、$C=0.4$ 还是 $C=0.6$，VNS、IG 和 SARSA-BBO 通过 $NR1$ 工厂分配方法的搜索效果优于 $NR2$ 工厂分配方法，而 BBO_GS 通过 $NR2$ 的搜索效果优于 $NR1$。因此，针对不同算法应采用不同的工厂分配方法以获得最优解，而不是盲目选择固定方法。

表 5-12 算法 CPU 运行时间(秒)

CPU 运行时间	VNS	IG	BBO_GS1	BBO_GS2	BBO_GS3	SARSA-BBO

$C=0.2$	$NR1$	53	53	123	137	145	93
	$NR2$	53	53	146	155	163	106
$C=0.4$	$NR1$	107	107	222	220	222	169
	$NR2$	107	107	226	223	238	181
$C=0.6$	$NR1$	160	160	258	253	257	247
	$NR2$	160	160	257	250	265	256
平均值		106	106	205	206	215	175

表 5-12 展示了算法求解 C-DPFSP-FAFT 的平均耗时。除了 VNS 和 IG 外，BBO_GS 和 SARSA-BBO 采用 $NR2$ 工厂分配方法比 $NR1$ 工厂分配方法耗时更长，因为 $NR2$ 需要额外一步将作业添加到所有工厂进行比较。同时可以观察到，加入 SARSA 强化学习的 BBO 搜索速度比未加入 SARSA 强化学习的 BBO 更快。因此可以得出一个重要结论：加入 SARSA 强化学习的 BBO 能够在搜索过程中节省时间。与 VNS 和 IG 相比，尽管 SARSA-BBO 算法耗时更多，但牺牲时间可以显著增强算法的搜索能力。

5.3.4 敏感性分析

本节对 SARSA-BBO 的实验结果进行方差分析并讨论了重要问题参数在 IMS-DAPFSP 中的影响。敏感性分析主要是分析问题的内在特性，具体而言，采用了单因素方差分析了装配机器数量 pm 对 SARSA-BBO 求解问题的显著影响。最后，本节还讨论了有关 IMS-DFAPFSP 管理方面的一些启示。

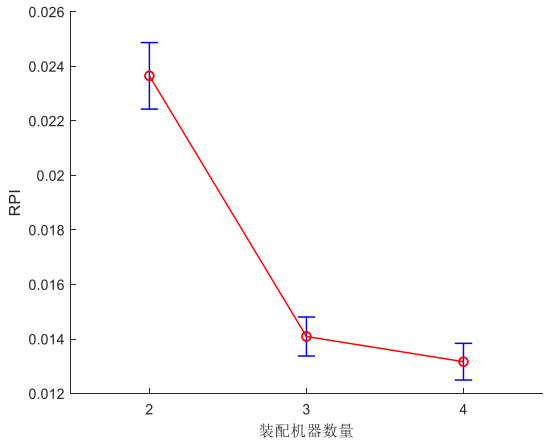

图 5-17 SARSA-BBO 在不同装配机器下具有 95%置信区间的均值图

图 5-17 为 SARSA-BBO 在不同装配机器下具有 95%置信区间的均值图。$am=2$时 RPI 均值为 0.023，置信区间为$[-1.21\times10^{-3},1.21\times10^{-3}]$，$am=3$时 RPI 均值为 0.014，置信区间为$[-7.14\times10^{-4},7.14\times10^{-4}]$，$am=4$时 RPI 均值为 0.013，置信区间为$[-6.72\times10^{-4},6.72\times10^{-4}]$。根据图 5-17 可以观察到随着装配机器数量 am 的增加，SARSA-BBO 的平均 RPI 有明显的下降趋势，其中 $am=2$ 到 $am=3$ 下降趋势最为明显，下降幅度为 69.46%。HM_2 性能的稳定性也随着 am 的增加不断提高。

基于敏感性分析实验结果，随着装配机器数量的增加，RPI 显著下降，尤其在装配机器数量从 2 增加到 3 时，RPI 下降最快，而从 3 增加到 4 时下降趋缓。由此可以得出以下管理启示：在生产管理中，合理配置装配机器数量对于提高有关 IMS-DAPFSP 生产模式的生产效率至关重要。特别是在资源有限的情况下，增加装配机器数量至 3 台可以带来显著的效益提升。然而，超过这一数量后，效率提升的幅度开始减小，因此在成本和资源分配上应更加谨慎，避免盲目增加设备而导致边际效益递减。同时，这一结果也提示管理者应考虑其他优化手段，如改进工序流程或技术升级，以进一步缩短完工时间。

5.4 本章小结

本文在分布式装配置换流水车间调度问题的基础上，进一步考虑了协同装配工厂和柔性装配机器的生产环境。IMS-DFAPFSP 将生产和装配视为一个集成制造系统，首次建立了两种以最小完工时间为目标的集成制造系统下的分布式柔性装配置换流水车间混合整数线性规划调度问题模型。基于 SARSA 强化学习的 BBO 被设计用来解决 IMS-DFAPFSP。为了提高算法的搜索速度，设计了三种局部和全局搜索方法，并且为了增强强化学习的状态判定，设计了种群优化成功率公式作为 SARSA 状态判断的条件之一。提供了 108 个 IMS-DFAPFSP 案例，并从文献中复现了几种对比算法。最终通过实验证明了 SARSA-BBO 的高效性。模型仿真实验验证了 mTSP 和 SPBM 在求解 IMS-DFAPFSP 问题中的有效性。此外，本文基于 IMS-DFAPFSP 的实验结果对其中的重要参数进行了敏感性分析，

得出了一些管理方面的启示。未来的研究将继续探讨分布式制造中生产工厂的特性，并扩展到实际生产模型的应用。

第 6 章 考虑绿色能耗下的多目标分布式装配置换流水车间调度问题研究

分布式装配置换流水车间调度问题在推动分布式制造系统发展中起着至关重要的作用。尽管许多研究关注于优化生产调度以提高效率，但能耗问题往往被忽视。结合可持续发展战略，本研究聚焦于 DAPFSP 的多目标绿色调度问题，提出了一个混合整数线性规划模型，旨在最小化最大完工时间和总机器能耗。为解决 MO-DAPFSP 问题，本文提出了一种有效的生物地理学的优化算法(Effective biogeography-based optimization, EBBO)。EBBO 采用了一种双种群启发式初始化方法，专门设计用于根据问题特征生成高质量的初始解。在迁移过程中添加了一种新颖的扰动算子，提升了部分解的 Pareto 质量，并加速了向真实 Pareto 前沿的收敛。为评估 EBBO 的性能，设计了 810 个不同规模的测试实例。实验结果表明，EBBO 算法在解决复杂调度问题方面表现出色，为分布式制造环境中的多目标绿色调度优化提供了一种有前景的方法。

6.1 考虑绿色调度下的分布式装配置换流水车间调度问题

如图 6-1 所示，MO-DAPFSP 包含两个阶段，即作业生产阶段和产品装配阶段。在生产阶段，多个作业在工厂中进行加工。每个工厂可以被视为一个置换流水车间调度问题。当所有作业在工厂中加工完成后，产品可以被送至装配线进行装配。无论作业被分配到哪个工厂，所有作业的加工路线都是相同的。装配线仅包含一台装配机器。

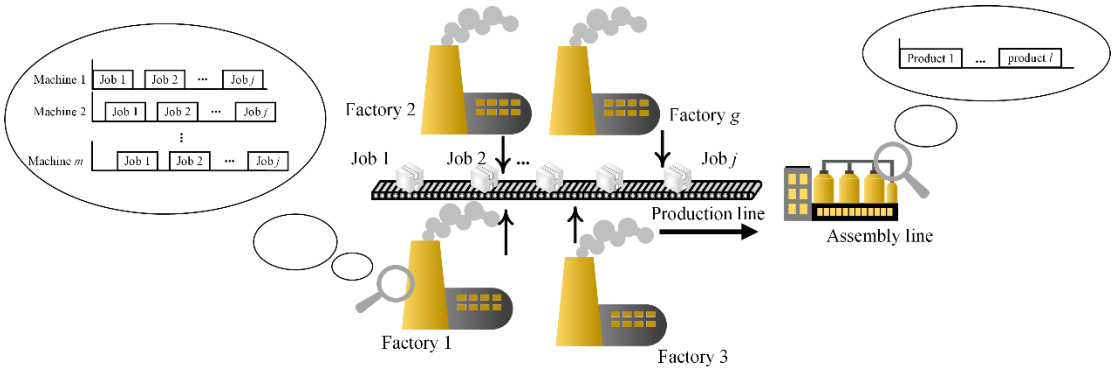

图 6-1 MO-DAPFSP 示意图。

DAPFSP 可以描述如下。在生产阶段，作业 j（$j=1,...,n$）需要被分配到工厂 g（$g=1,...,f$）进行加工。每个作业只能被分配到一个工厂。所有工厂拥有相同数量和类型的机器 i（$i=1,...,m$）。每个作业包含 m 道工序。每道工序必须在机器 i（$i=1,...,m$）上加工。所有作业按照相同的路线进行加工，具体为：首先在机器 1 上加工，然后在机器 2 上加工，依此类推，直到机器 i。这种加工方式称为流水车间加工。作业 j 在机器 i 上所需的加工时间为 $p_{ij}(p_{ij}>0)$。作业一旦开始加工，便不能被中断。在装配阶段，完成所有工序的作业需要被装配成一个产品。每个作业仅属于一个产品 l（$l=1,...,t$），每个产品至少包含一个作业。属于产品 l 的作业数量为 Ψ_l，且 $n=\sum_{l=1}^{t}\Psi_l$。产品的装配必须在其所有作业完成后才能开始。产品 t 的装配时间为 $pp_l(pp_l>0)$。产品在装配线上开始装配后，不能被中断。

6.1.1 混合整数线性规划模型

混合整数线性规划模型用于表示 MO-DAPFSP。通过引入整数和变量，该模型建立了线性约束和目标函数，以便在已知可行区域内优化目标值。它能够有效处理同时包含离散和连续决策变量的复杂问题，例如资源分配和生产调度。在建立 MO-DAPFSP 的 MILP 模型之前，需要定义一些参数、符号和变量。变量分为连续变量（continuous variables, CVs）和二进制变量（binary variables, BVs）。

参数部分中，n 表示工件数量，m 表示机器数量，f 表示工厂数量，t 表示产品数量，p_{ij} 表示工件 j 在机器 i 上的加工时间，pp_l 表示产品 l 的装配时间，G_{lj} 表示工件 j 是否属于产品 l，E_i^{po}、E_i^{sd}、E_i^{w} 和 E_i^{nl} 分别表

示机器 i 的启动、关闭、工作和空载能耗系数，M 为一个足够大的正数。符号部分中，j,k 表示工件，i 表示机器，l,s 表示产品，g 表示工厂，T_{ij}^{end} 表示工厂中工件 j 在机器 i 上的最大完成时间，T_i^{free} 表示机器 i 在所有工厂中的空载时间总和。变量部分中，X_{jkg} 表示工厂 g 中工件 j 是否在工件 k 之前加工，Y_{jg} 表示工件 j 是否在工厂 g 中加工，Z_{ls} 表示产品 l 是否在产品 s 之前装配，EB_1 和 EB_2 表示机器能耗分支，C_{ij} 表示工件在机器上的完成时间，CA_l 表示产品在装配线上的完成时间，$C_{\max}$ 和 $E_{\max}$ 分别表示最大完成时间和最大能耗的目标函数。

基于序列的模型（Sequence-based model, SBM）是一种常用于分布式生产调度的 MILP 模型。SBM 通过限制作业在工厂中的加工位置来获取所有作业的加工顺序，并限制产品的加工顺序。SBM 在二进制变量（BV）中引入了虚拟作业，通过虚拟作业可以限制第一个和最后一个作业的位置。SBM 的 MILP 模型如下：

$$\sum_{k=0,k\neq j}^{n}\sum_{g=1}^{f}X_{kjg}=1,\forall j,j\neq 0 \tag{6-1}$$

$$\sum_{j=0,j\neq k}^{n}\sum_{g=1}^{f}X_{kjg}=1,\forall k,k\neq 0 \tag{6-2}$$

$$\sum_{j=1}^{n}X_{0jg}=1,\forall g \tag{6-3}$$

$$\sum_{k=1}^{n}X_{k0g}=1,\forall g \tag{6-4}$$

$$\sum_{g=1}^{f}(X_{kjg}+X_{jkg})\leq 1,\forall k,j,k\neq j \tag{6-5}$$

$$\sum_{g=1}^{f}Y_{jg}=1,\forall j,j\neq 0 \tag{6-6}$$

$$\sum_{k=1,k\neq j}^{n}(X_{kjg}+X_{jkg})\leq 2\cdot Y_{jg},\forall j,g \tag{6-7}$$

$$\sum_{l=0,l\neq s}^{t}Z_{ls}=1,\forall s,s\neq 0 \tag{6-8}$$

$$\sum_{s=1,s\neq l}^{t}Z_{ls}\leq 1,\forall l \tag{6-9}$$

$$Z_{ls}+Z_{sl}\leq 1,\forall l,s,l\neq s \tag{6-10}$$

$$C_{1j} \geq p_{1j}, \forall j, j \neq 0 \tag{6-11}$$

$$C_{1j} \geq p_{1j} + C_{1k} + (\sum_{g=1}^{f} X_{kjg} - 1) \cdot M, \forall j,k, j \neq 0, k \neq 0, j \neq k \tag{6-12}$$

$$C_{ij} \geq C_{i-1j} + p_{ij}, \forall i, j, i \neq 1, j \neq 0 \tag{6-13}$$

$$C_{ij} \geq p_{ij} + C_{ik} + (\sum_{g=1}^{f} X_{kjg} - 1) \cdot M, \forall i, j, k, i \neq 1, j \neq 0, k \neq 0, j \neq k \tag{6-14}$$

$$CA_s \geq C_{mj} \cdot G_{lj} + pp_s, \forall j, s, j \neq 0, s \neq 0 \tag{6-15}$$

$$CA_s \geq CA_l + pp_s + (Z_{ls} - 1) \cdot M, \forall l, s, l \neq 0, s \neq 0, l \neq s \tag{6-16}$$

$$C_{\max} \geq CA_s, \forall s, s \neq 0 \tag{6-17}$$

$$\min C_{\max} \tag{6-18}$$

$$T_{ij}^{end} \geq C_{ij} + (X_{j0g} - 1) \cdot M, \forall i, j, g, j \neq 0 \tag{6-19}$$

$$T_{ij}^{end} \geq X_{j0g}, \forall i, j, g, j \neq 0 \tag{6-20}$$

$$EB^1 \geq \sum_{j=1}^{n} \sum_{i=1}^{m} (T_{ij}^{end} \cdot E_i^w + E_i^{po} + E_i^{sd}) \tag{6-21}$$

$$T_i^{free} \geq \sum_{j=1}^{n} T_{ij}^{end} - \sum_{j=1}^{n} p_{ij}, \forall i \tag{6-22}$$

$$EB^2 \geq \sum_{i=1}^{m} (T_i^{free} \cdot E_i^{nl}) \tag{6-23}$$

$$E_{\max} \geq EB^1 + EB^2 \tag{6-24}$$

$$\min E_{\max} \tag{6-25}$$

在基于 SBM 的 MILP 模型中，首先对三个二进制变量（BVs）进行了约束。约束(6-1)表明，对于任意作业 j（$j = 1,...,n$），在工厂 g 中存在且仅存在一个作业 k（$k = 0,...,n$）作为作业 j 的前置加工作业。约束(6-2)表明，对于任意作业 k（$k = 1,...,n$），在工厂 g 中存在且仅存在一个作业 j（$j = 0,...,n$）作为作业 k 的后置加工作业。约束(6-3)-(6-4)限制了每个工厂中至少存在一个作业。约束(6-5)限制了任意作业 j 和 k（$j,k = 0,...,n$，$j \neq k$）不能同时作为彼此的前置加工作业和后置加工作业。约束(6-6)表示每个作业 j（$j = 1,...,n$）只能被分配到一个工厂进行加工。约束(6-7)说明每个作业在其分配的工厂中只能是某个作业的后继或前驱。约束(6-8, 6-9, 6-10)表明，对于任意产品 s（$s = 1,...,t$），在装配线上存在且仅存在一个产品 l（$l = 0,...,t$）作为产品 s 的前置装配产品。

在定义了二进制变量（BVs）的约束之后，BSM 对四个连续变量（CVs）进行了约束。在这四个 CVs 中，C_{ij} 和 CA_s 定义了作业的完成加工时间和产品的完成装配时间，以最小化目标函数 $C_{\max}$。T_{ij}^{end} 和 T_i^{free} 定义了每个工厂中每台机器的结束时间和空闲时间，以获取最小化的目标函数 $E_{\max}$。约束(6-11)限制了每个工厂中第一个作业的第一道工序的完成时间必须大于该工序所需的加工时间。约束(6-12)限制了每个工厂中非第一个作业的第一道工序的完成时间必须大于其前置加工作业的第一道工序的完成时间加上该工序所需的加工时间。约束(6-13)限制了每个工厂中作业的非首道工序的完成时间必须大于该作业的前一道工序的完成时间加上该工序所需的加工时间。约束(6-14)限制了每个工厂中作业的完成时间必须大于其前置加工作业的完成时间加上该工序所需的加工时间。约束(6-15)限制了每个产品 s 的装配完成时间必须大于属于产品 s 的所有作业的最大完成时间加上产品 s 所需的装配时间。约束(6-16)限制了每个产品 s 的装配完成时间必须大于其前置装配产品的装配完成时间加上产品 s 所需的装配时间。约束(6-17)表明目标函数 $C_{\max}$ 必须大于每个产品的装配时间。约束(6-18)的目的是最小化 MO-DAPFSP 的目标函数 $C_{\max}$。约束(6-19)-(6-20)用于获取每个工厂中每台机器的完工时间。首先，约束(6-19)用于保留虚拟作业在机器 i 上的前置作业 j 的完成时间，然后其他作业的 T_{ij}^{end} 小于 0。接着，约束(6-20)将 T_{ij}^{end} 设置为 0（如果其小于 0）。通过此操作，T_{ij}^{end} 仅保留了虚拟作业在每台机器上的前置作业 j 的完成时间，其余值为 0。然后，为了计算所有工厂中机器 i 的完成时间总和，只需对虚拟作业在机器 i 上的前置作业 j 的完成时间求和，约束(6-21)可以通过这种方式求和，并乘以相应的能耗因子，得到机器能耗分支 1。约束(6-22)表明，所有工厂中机器 i 的空闲时间 T_i^{free} 是通过从处理所有作业所需的时间总和中减去 T_{ij}^{end} 的时间总和得到的。约束(6-23)表明，T_i^{free} 的总和乘以相应机器的空载能耗系数，得到机器能耗分支 2。约束(6-24)表明，目标函数 $E_{\max}$ 必须大于能耗分支的总和。约束(6-25)的目的是最小化 MO-DAPFSP 的目标函数 $E_{\max}$。

计算表明，SBM 中三个二元变量（BVs）的空间数量为 $f(n^2+3n+2)-(n+1)+t(t+1)$。第一项是 X_{kjg}，其中 k, j 的规模为 $n+1$，g 的规模

为 f，因此约束空间为 $(n+1)^2 f$。由于 X_{kjg} 限制了 $k \neq j$，因此需要从 $(n+1)$ 中减去相应的空间大小。第二项 BVs 定义了 Y_{jg} 的约束空间大小，第三项定义了 Z_{ls} 的约束空间大小。SBM 中四个连续变量（CVs）的总和为 $m(2n+1)+t$。

6.1.2 机器能耗描述

在绿色调度问题中，需要设置生产工厂中机器的能耗参数。机器的能耗通常由多种状态组成，例如空载状态、工作状态、开机状态和关机状态，且机器在不同状态下的功率不同。对机器的运行状态进行一般性假设如下：

(1) 开机状态：当机器从关闭状态切换到活动状态时，能耗会急剧上升。机器的各个部件在启动时消耗大量电力。每个工厂中同一台机器启动所需的功耗是相同的。

(2) 空载状态：当机器处于开启状态但未执行加工任务时，能耗较低但仍然存在，机器始终处于待机状态。

(3) 工作状态：当机器执行加工任务时，能耗达到峰值，机器功率保持稳定。

(4) 关机状态：当机器从活动状态切换到关闭状态时，能耗会逐渐减少，机器部件依次停止工作。每个工厂中同一台机器关闭所需的功耗是相同的。

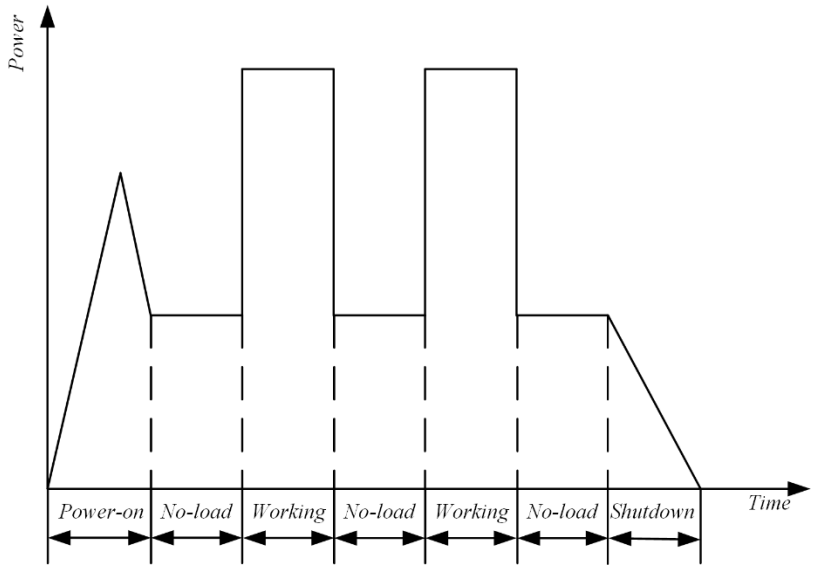

图 6-2 机器功耗分段函数示意图

图 6-2 展示了某台机器加工过程中的功率变化趋势。通过建立能耗模型，可以深入理解机器在不同状态下的能耗特性及能耗分布规律，这为优化生产调度和

提高能源利用效率提供了理论依据。

6.1.3 多目标问题优化

多目标优化问题是指在可行域内确定一个由决策变量组成的向量，该向量满足所有约束条件，并优化由多个目标函数组成的向量。多目标问题的定义如下：

(1) 支配解/非支配解：假设有两个解 S_1 和 S_2。对于所有目标函数（ $f_1(x), f_2(x),..., f_\eta(x)\ f_1(x), f_2(x),..., f_\eta(x)$ ），如果 S_1 优于 S_2（ $f_\kappa(S_1) < f_\kappa(S_2), \kappa = 1,..., \eta, f\kappa(S_1) < f\kappa(S_2), \kappa = 1,..., \eta$ ），则定义 S_1 支配 S_2。如果 S_1 的解不被其他解支配，则 S_1 称为非支配解。如果解 S_2 的所有目标函数均劣于 S_1，则称 S_1 优于 S_2，也称 S_1 支配 S_2，S_2 为被支配解。

(2) Pareto 最优解：在多目标算法搜索过程中，未被支配的解的集合称为 Pareto 最优解。

(3) Pareto 前沿：由多个 Pareto 最优解映射的所有点形成的边界称为 Pareto 前沿（ PF ）。

(4) 欧几里得距离：欧几里得距离（ ED ）通常用于计算空间中两点之间的直线距离。在本文中，ED 用于描述两个解之间的距离。公式(6-26)和(6-27)分别为 ED 的计算公式和目标函数的归一化方法：

$$ED(S_1, S_2) = \sqrt{\sum_{\kappa=1}^{\eta} (f_\kappa^{norm}(S_1) - f_\kappa^{norm}(S_2))^2} \tag{6-26}$$

$$f_\kappa^{norm}(S_1) = \frac{f_\kappa(S_1) - \min f_\kappa(S)}{\max f_\kappa(S) - \min f_\kappa(S)}, \kappa = 1,..., \eta, S = S_1, S_2,... \tag{6-27}$$

(5) 接受准则：优化后，接受准则决定新解 S_1 是否可以替换旧解 S_2。优化结果中存在三种情况：S_1 支配 S_2，S_2 支配 S_1，S_1 和 S_2 处于同一 Pareto 层级。S_2 被接受的概率定义为 ξ。ξ 的定义如下:

$$\xi = \exp(-(\frac{\sum_{\kappa=1}^{\eta} (f_\kappa^{norm}(S_2) - f_\kappa^{norm}(S_1))}{I})), S_1 \sim S_2 \tag{6-28}$$

$$\xi = 1, S_1 \prec S_2 \tag{6-29}$$

$$\xi = 0, S_1 \succ S_2 \tag{6-30}$$

如果S_1和S_2处于同一 Pareto 层级，则使用公式 6-28 计算选择S_2的概率。如果S_2支配S_1或S_1支配S_2，则被选择的概率分别为 1 或 0。

6.2 多目标生物地理学优化算法

6.2.1 编码方式

基于 MO-DAPFSP 的特征，合理的编码方式能够有效且完整地描述该问题。根据第 6.1 节对 MO-DAPFSP 的描述，可以明确地将其分为两个子问题：生产和装配。如果装配车间中不存在阻塞情况，则每个产品l的开始装配时间等于工件j的最大完工时间（即$G_{lj}=1,\forall j$）。否则，每个产品l的装配开始时间可能大于或等于工件j的最大完工时间。因此，只要已知各工厂中工件的分配情况，就可以确定产品的装配顺序。首先，$\theta_{g,l}$表示工厂g内所有工件j（满足$G_{lj}=1$）的加工顺序。如果工厂g中不存在产品l的工件，则其值为空集（即$\theta_{g,l}=\varnothing$）。其次，Φ_l（$\Phi_l=[\theta_{1,l},\theta_{2,l},...,\theta_{f,l}]$）用于表达所有$\theta_{g,l}$。最后，通过$\Pi$（$\Pi=[\Phi_1,\Phi_2,...,\Phi_t]$）表示所有产品的加工优先级，即解的表达形式。当 $\Psi_l / f<1$ 时，在$\theta_{g,l}$中必然存在空集$\varnothing$。

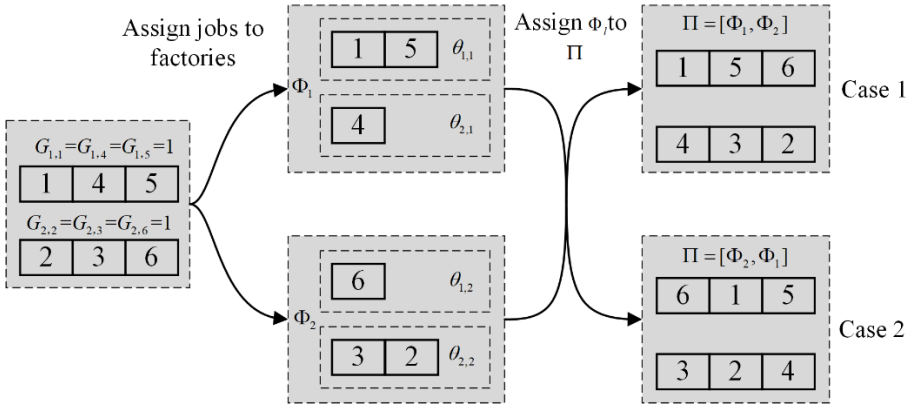

图 6-3 MO-DAPFSP 编码方法示意图

图 6-3 为 MO-DAPFSP 的编码示意图。假设$G_{11}=G_{14}=G_{15}=1$，$G_{22}=G_{23}=G_{26}=1$，即工件 1、4 和 5 属于产品 1，工件 2、3 和 6 属于产品 2。Φ_l表示各工厂中工件的加工顺序，并基于Φ_l生成了两种不同的解，分别为$\Pi=[\Phi_1,\Phi_2]$ 和$\Pi=[\Phi_2,\Phi_1]$。

6.2.2 初始化方式

在 MO-DAPFSP 中，初始解的多样性和有效性直接影响多目标算法的搜索速度和求解质量。因此，本节提出了一种双种群启发式初始化（Dual-population Heuristic Initialization, DHI）方法。DHI 能够生成具有不同目标函数优势的初始种群，并保持种群的多样性。具体而言，该方法通过两种不同的分配规则生成具有单侧主导性的两个初始种群。DHI 的核心思想是通过结合由不同策略生成的个体，有效增强种群的多样性和探索能力。在此基础上，两个种群的部分编码片段会进行交换，以生成新的种群，从而提高初始解的质量，并减少冗余解的出现频率。DHI 不仅提升了算法的初始性能，还为后续的优化过程奠定了坚实基础。其详细过程如下：

首先，依次选取产品l（确保无重复），然后随机打乱产品l的工件顺序，并存储在一维数组v中。接着，按照最小完工时间或最小总能耗的原则，依次将v中的工件分配到相应的工厂。该过程重复进行，直到所有产品的所有工序都被分配完毕。最终，两个生成的解分别存储在解集合S^{MS}和S^{EC}中。该过程持续进行，直至S^{MS}和S^{EC}均包含N个解。随后，从S^{MS}和S^{EC}中随机选择一个解（且不重复），同时随机选择一个产品l，交换两个解中的Φ_l，并将生成的两个新解存入集合S。该过程重复执行，直到S包含$2N$个解。

当种群S包含$2N$个初始解后，双种群终止生成。然后使用快速非支配排序和拥挤度计算方法对初始解进行排序，并选取排名前N的解作为算法的初始解。快速非支配排序根据解的非支配等级，将种群划分为不同层级，从而有助于在多目标优化中选择和保留高质量解。快速非支配排序的详细过程如表 6-1 所示。

对于相同 Pareto 级别的解，通过计算其与所有其他解的欧几里德距离的平均值来确定每个解的拥挤度。然后，按照拥挤度从高到低进行排序。较高的拥挤度表示其周围的解较少，说明该解在目标空间中较为稀疏或独特。保留这些解有助于维持解集的多样性，并确保 Pareto 前沿的均匀覆盖。

表 6-1 快速非支配排序伪代码

1: **for** each ss **in** Π //依次选择 Π 中的每个解 ss
2: set $\chi(\Pi(ss)) = \varnothing$ //初始化解 ss 被支配的集合 $\chi(\Pi(ss))$ 为空集

3: set $\vartheta(\Pi(ss))=0$ //初始化解 ss 被支配的数量 $\vartheta(\Pi(ss))$ 为 0

4: **for** each sc in Π //依次选择 Π 中的每个解 sc

5: **if** ss dominates sc//如果解 ss 支配 sc

6: then $\chi(\Pi(ss))=\chi(\Pi(ss))\cup sc$ //则添加 sc 到集合 $\chi(\Pi(ss))$ 中

7: **elseif** sc dominates ss //如果解 sc 支配 ss

8: then $\vartheta(\Pi(ss))=\vartheta(\Pi(ss))+1$ //则令解 ss 的支配数量 $\vartheta(\Pi(ss))$ 加 1

9: **endif**

10: **endfor**

11: **if** $\vartheta(\Pi(ss))=0$://如果 $\vartheta(\Pi(ss))$ 支配数为 0

12: then $ss_{rank}=1$ //则将解 ss 的等级设置为 1

13: and $PL_1=PL_1\cup\{ss\}$ //将解 ss 放入第一层非支配解前沿

14: **endif**

15: **endfor**

16: set $pl=1$// pl 为当前非支配前沿索引

17: **while** $PL_{pl}\neq\varnothing$ //终止条件为非支配前沿为空

18: set $\sigma\neq\varnothing$ // σ 被用来储存第 $pl+1$ 层的非支配解

19: **for** each ss in PL_{pl} //依次选择第 pl 层非支配前沿 PL_{pl} 的解 ss

20: **for** each sc in $\chi(\Pi(ss))$ //依次选择被 ss 支配的集合 $\chi(\Pi(ss))$ 的每个解 sc

21: **then** $\vartheta(\Pi(sc))=\vartheta(\Pi(sc))-1$ //将 $\vartheta(\Pi(sc))$ 数量减 1

22: **if** $\vartheta(\Pi(sc))=0$ //如果 sc 的支配数为 0

23: $sc_{rank}=pl+1$ //则将 sc 的等级设置为 $pl+1$

24: and $\sigma=\sigma\cup\{sc\}$ //将解 sc 添加至集合 σ 中

25: **endif**

26: **endfor**

27: **endfor**

28: set $pl=pl+1$ //开始循环 $pl+1$ 层非支配前沿

29: set $PL_{pl}=\sigma$ //第 $pl+1$ 层非支配前沿为集合 σ

30: **endwhile**

6.2.3 扰动算子

在 BBO 中，迁移算子主要通过从迁出栖息地迁移优秀的编码片段来改进迁

入栖息地的编码。高质量的解充当信息输出者，而低质量的解则充当信息接收者。在此进化过程中，低质量解的性能得到提升，而高质量解则受到变异算子的最小扰动，基本保持不变。因此，种群内高质量解可能会出现严重的同质化，从而影响算法的收敛速度。为了解决这一问题，在执行迁移算子时引入了扰动算子，在从迁出栖息地迁移后施加一定程度的扰动，以提高迁出栖息地的 。栖息地扰动算子（ PO ）公式定义如下。

$$\theta_{g,l}^{'} = \theta_{g,l} + \varepsilon \cdot \Delta\theta_{g,l} \tag{6-31}$$

$\theta_{g,l}^{'}$ 是扰动后的变量， $\theta_{g,l}$ 是扰动前的变量， ε （ $\varepsilon \in (0,1]$ 是控制扰动大小的扰动因子。 $\Delta\theta_{g,l}$ 表示扰动的变化量。

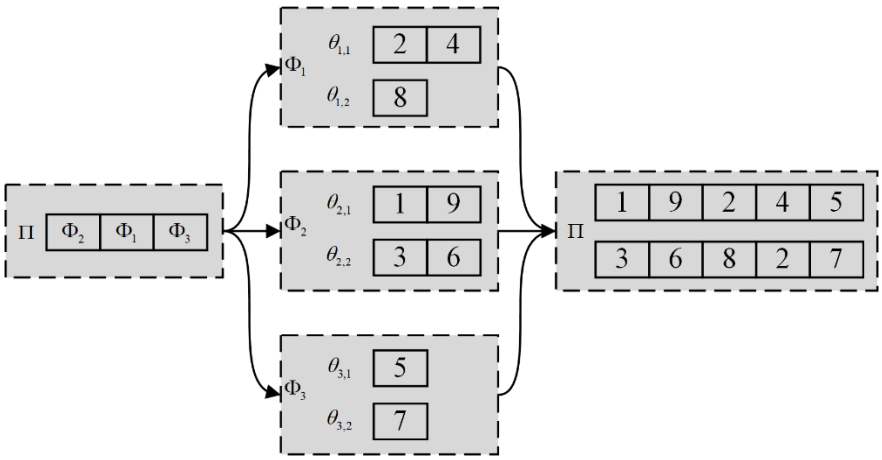

图 6-4 工件分布示例

为了更好地描述扰动算子的执行过程，这里给出一个示例。作业的分布如图 6-4 所示。假设选择解 Π 执行扰动操作，其中 $\Pi = [\Phi_2, \Phi_1, \Phi_3]$ 。选择产品 1 进行扰动， $\varepsilon = 0.4$ ，产品 1 的完成时间为 $CA_1 = 33$ ，即作业 $[3,6,8]$ 的完成时间加上产品 1 的装配时间 $C_{2,8} + pp_1$ 等于 33。因此，选择工厂 1 进行扰动。由于 $|\theta_{1,1}| = 2$ ，则 $|\theta_{1,1}| - \lfloor \varepsilon \cdot |\theta_{1,1}| \rfloor = 2$ 。从 $\theta_{1,1}$ 中随机选择长度为 2 的编码段进行扰动。依次选择该编码上的每个作业，并将其插入到机器能耗最小的位置。由于 $\theta_{1,1}$ 的长度为 2， $\theta_{1,1} = \bar{\theta}_{1,1}$ ，因此只有两种可能的插入结果： $[2,4]$ 和 $[4,2]$ 。两种 $\bar{\theta}_{1,1}$ 结果都将替换 Π 中的原始 $\theta_{1,1}$ ，然后计算能耗以选择最佳编码配置。该案例的详细扰动过程如图 6-5 所示。

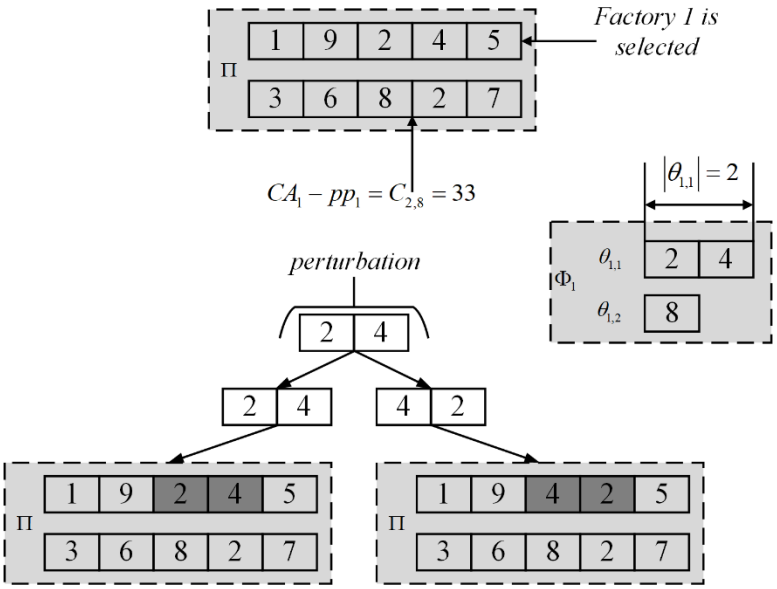

图 6-5 详细描述的扰动过程示例

6.3 仿真实验

6.3.1 案例以及评价指标

为了评估所提出方法在解决 MO-DAPFSP 问题上的性能，随机生成了 810 个实例。作业数量为 $n \in \{100, 200, 500\}$，机器数量为 $m \in \{5, 10, 20\}$，工厂数量为 $f \in \{4, 6, 8\}$，产品数量为 $t \in \{30, 40, 50\}$，通过组合共得到 81 种组合。每种组合包含 10 个实例。每个实例可以表示为 $n/m/f/t$。根据公式 6-32-公式 6-35 生成四个能耗系数。作业的加工时间在 [1,99] 范围内，产品的装配时间在 $[\Psi_l, 99 \times \Psi_l]$ 范围内。加工时间范围在作业车间调度问题领域中更为常用。

$$E_i^w = \left\lfloor (1 + \varphi \cdot \eta) \sum_{j=1}^{n} \overline{p_{ij}} \right\rfloor, \forall i \tag{6-32}$$

$$E_i^{nl} = \left\lfloor \sum_{j=1}^{n} \overline{p_{ij}} \right\rfloor, \forall i \tag{6-33}$$

$$E_i^{sd} = \left\lfloor (1 - \varphi \cdot \eta) \sum_{j=1}^{n} \overline{p_{ij}} \right\rfloor, \forall i \tag{6-34}$$

$$E_i^{po} = \left\lfloor (1 - \varphi \cdot \eta) \sum_{j=1}^{n} \overline{p_{ij}} \right\rfloor, \forall i \tag{6-35}$$

变量 φ 和 η 是在区间[0,1]上均匀分布的随机数。$\lfloor \mathbb{Z} \rfloor$ 是不大于 $\mathbb{Z}$ 的最大整数值。所有算法均在 Microsoft Visual C++ 中编码，并在配备 i9-13980HX CPU @ 2.20 GHz 和 16.0 GB RAM 的笔记本电脑上运行。MILP 模型使用 CPLEX 12.6.3 软件进行测试。

由于每个实例的真实 Pareto 前沿未知，因此将所有测试算法获得的非支配解作为一个集合，并从该集合中筛选出支配解。筛选后的集合被定义为近似最优 Pareto 前沿（PF^*），并用 PF^* 替代真实 Pareto 前沿。使用三个指标来评估算法的性能。

(1) 扩散程度（Δ）

Δ 用于衡量算法获得的 PF 的拥挤程度，如公式 6-36 所示。Δ 值越大，PF 的非支配解越分散，非支配解的多样性越高，决策者可以根据实际需求选择的方案越多。如果 Δ 值较小，决策者的选择范围将受到限制，无法充分展示多目标之间的权衡关系。PF 中非支配解多样性的增加有助于算法更好地逼近真实 Pareto 前沿，从而提高整体优化质量，获得更全面的解集。

$$\Delta = \frac{ED + ED^* + \sum_{h=1}^{D-1} \left| ED_h - \overline{ED} \right|}{ED + ED^* + (D-1) \times \overline{ED}} \tag{6-36}$$

ED 和 ED^* 分别表示 PF 和 PF^* 中边界点之间的距离。D 表示 PF 中非支配解的数量。ED_h 表示 PF 中第 h 个非支配解与相邻非支配解之间的欧几里得距离。$\overline{ED}$ 表示 ED_h 的算术平均值。

(2) 世代距离（GD）

公式 6-37 中，GD 用于衡量 PF^* 和 PF 之间的相似性。GD 值越小，PF 越接近 PF^*。

$$GD = \frac{\sqrt{\sum_{h=1}^{D} EDF_h^2}}{D} \tag{6-37}$$

EDF_h 表示 PF 中第 h 个解到 PF^* 中非支配解的最小欧几里得距离。

(3) 反向世代距离（IGD）公式 6-38 中，IGD 用于衡量 PF^* 中第 h 个非支配解到 PF 中解的最小欧几里得距离。作为一种双向距离评估方法，GD 和 IGD

可以更全面地比较 PF 和 PF^* 之间的关系。通过计算 IGD，可以清楚地了解当前 Pareto 前沿中的解集是否充分覆盖了最优 Pareto 前沿中的解集。IGD 越小，PF 的解集覆盖越好。

$$IGD = \frac{\sum_{h=1}^{D} ED(h, PF)}{D} \tag{6-38}$$

ED 表示 PF^* 中第 h 个非支配解到 PF 的最小欧几里得距离。

6.3.2 多目标算法比较

为了验证 EBBO 在面对不同规模的 MO-DAPFSP 时的整体搜索性能，将 EBBO 与非支配排序遗传算法 II[57]、多目标模拟退火（MOSA）算法[58]以及提出的多目标粒子群优化算法[59]进行了对比测试。实验记录了 Δ、GD 和 IGD。每个实例测试 5 次，记录 5 次试验中的最优解。终止条件设置为最大 CPU 运行时间不超过 $n \times f \times m \times 0.1s$。记录每个案例的相对百分比差异。$RPD$ 的定义如下：

$$RPD = \frac{\mathbb{C}^* - \mathbb{C}}{\mathbb{C}} \times 100\% \tag{6-39}$$

对于每个 MO-DAPFSP 实例，运行四种算法后获得的最优目标函数值记为 $\mathbb{C}^*$，算法的当前结果记为 $\mathbb{C}$。表 6-2 展示了四种算法的结果。图 6-6 至图 6-8 记录了四种算法的 95% 置信区间图。图 6-9 和图 6-10 展示了四种算法在 $200/10/6/40$ 和 $100/10/6/40$ 的 MO-DAPFSP 实例时获得的非支配解的分布情况。

根据表 6-2 和图 6-6 至图 6-8 的结果，可以观察到所有算法的性能通过在 MO-DAPFSP 中基于三个指标和不同规模实验的综合分析得到了充分验证。与其他算法相比，EBBO 展示了最佳的解质量和最广泛的非支配解覆盖范围。在大多数情况下，EBBO 优于其他三种具有代表性的竞争算法。

图 6-9 和图 6-10 进一步展示了 EBBO 的非支配解分布，其不仅覆盖范围广，而且保持了解的高质量。相比之下，NSGA-II 表现最差，而 MOSA 和 MOPSO 在平衡多个目标函数方面表现不佳。这表明 EBBO 不仅在探索解空间

方面表现出色，而且在多个目标上开发高质量解的能力也很强。

表 6-2 四种算法在不同规模 MO-DAPFSP 的实验结果

		Δ				*GD*				*IGD*			
		EBBO	NSGA-II	MOSA	MOPSO	EBBO	NSGA-II	MOSA	MOPSO	EBBO	NSGA-II	MOSA	MOPSO
n	100	**0.328**	0.132	0.247	0.257	**0.009**	0.026	0.017	0.019	**0.010**	0.036	0.018	0.019
	200	**0.353**	0.187	0.262	0.242	**0.014**	0.029	0.024	0.027	**0.009**	0.041	0.017	0.018
	500	**0.474**	0.264	0.388	0.296	**0.023**	0.034	0.029	0.033	**0.014**	0.036	0.024	0.018
m	5	**0.292**	0.145	0.135	0.123	**0.005**	0.020	0.011	0.012	**0.008**	0.027	0.017	0.024
	10	**0.315**	0.229	0.214	0.281	**0.019**	0.027	0.023	0.028	**0.015**	0.024	0.022	0.019
	20	**0.361**	0.271	0.279	0.237	**0.024**	0.034	0.031	0.036	**0.018**	0.033	0.023	0.026
f	4	**0.335**	0.158	0.256	0.265	**0.011**	0.021	0.015	0.018	**0.017**	0.027	0.024	0.025
	6	**0.349**	0.193	0.291	0.210	**0.016**	0.030	0.026	0.027	**0.019**	0.026	0.031	0.031
	8	**0.384**	0.216	0.324	0.299	**0.020**	0.037	0.034	0.035	**0.011**	0.038	0.021	0.025
t	30	**0.319**	0.121	0.167	0.331	**0.007**	0.025	0.021	0.016	**0.014**	0.034	0.026	0.022
	40	**0.376**	0.154	0.238	0.286	**0.018**	0.031	0.028	0.027	**0.016**	0.036	0.023	0.017
	50	**0.472**	0.180	0.283	0.372	**0.026**	0.045	0.038	0.031	**0.015**	0.036	0.030	0.027
均值		**0.363**	0.187	0.257	0.266	**0.016**	0.030	0.025	0.026	**0.014**	0.033	0.023	0.023

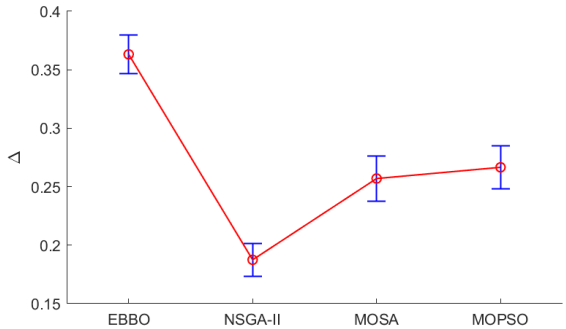

图 6-6 四种算法在 MO-DAPFSP 案例下 Δ 的 95%置信区间图

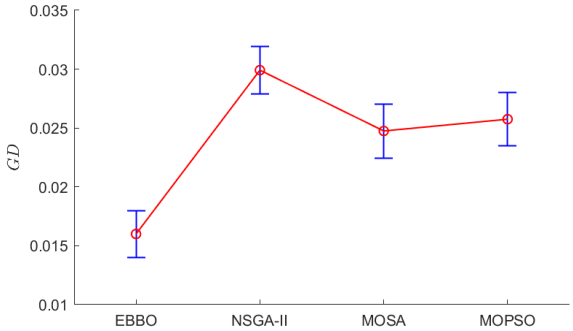

图 6-7 四种算法在 MO-DAPFSP 案例下 *GD* 的 95%置信区间图

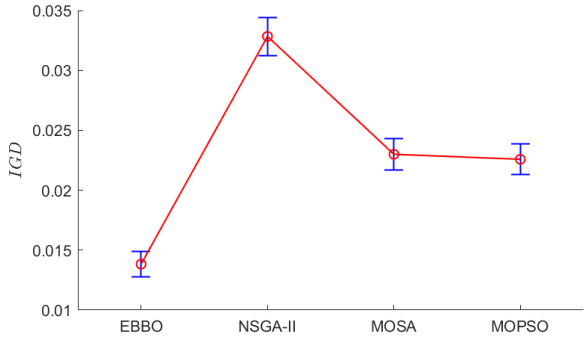

图 6-8 四种算法在 MO-DAPFSP 案例下 *IGD* 的 95%置信区间图

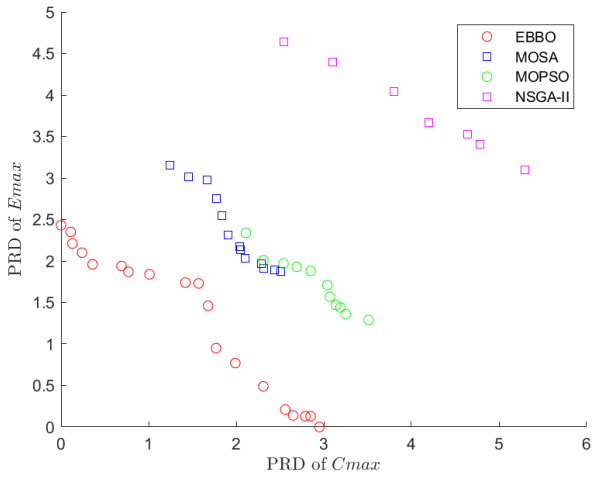

图 6-9 四种算法在 200 / 10 / 6 / 40 案例下的非支配解分布图

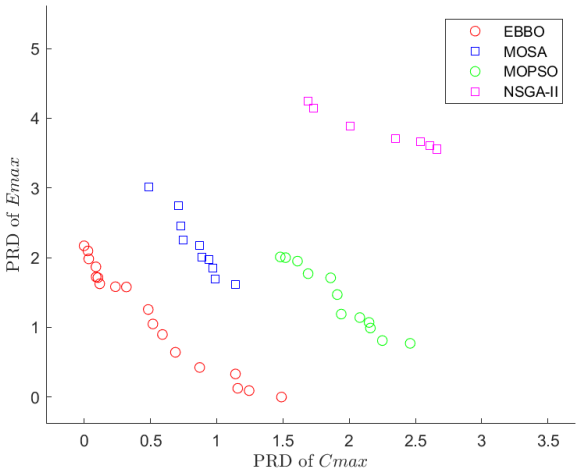

图 6-10 四种算法在100 / 10 / 6 / 40 案例下的非支配解分布图

6.4 本章小结

本文提出了一种有效的基于生物地理学的优化算法，用于解决分布式装配置换流水车间问题中的多目标绿色调度问题。基于 MO-DAPFSP 的特点，设计了扰动因子以改进部分解的 Pareto 等级。随机生成了 810 个测试案例，并通过实际加工案例验证了所提出方法的有效性。尽管 MO-DAPFSP 考虑了机器能耗的可能约束，但在实际制造环境中，可能会出现机器故障和作业到达延迟等不确定因素。因此，研究鲁棒启发式算法对于解决这些问题至关重要。

第 7 章 总结与展望

7.1 全文总结

随着全球化生产的普及，制造任务往往分布在多个工厂或车间中，如何高效协调这些分布式资源成为提升整体生产效率的关键。DPFSP 研究如何在多个车间之间合理分配任务并优化调度顺序，有助于降低生产周期、减少等待时间，从而提高资源利用率和企业竞争力。DPFSP 的研究对实际生产具有直接指导意义。许多行业（如汽车制造、电子产品组装等）都涉及多车间协作生产，优化调度方案可以显著降低生产成本、缩短交货时间并提高客户满意度。研究 DPFSP 不仅有助于解决实际生产中的复杂调度问题，还能推动相关领域的理论和技术进步，具有重要的学术价值和工业应用前景。本文研究的工作成果如下所示：

(1) 本研究针对分布式置换流水车间调度问题（DPFSP）展开深入探讨，构建了相应的数学模型。为提高求解效率，创新性地设计了一种高效初始化方法，有效提升了初始解的质量并降低了冗余解的产生概率。在此基础上，提出了专门适用于 DPFSP 的编码方案、迁移算子以及四种突变算子。通过田口实验对 BBO 算法参数进行优化调校，使算法性能达到最优。改进后的 BBO 算法在包含 420 个小型案例和 720 个大型案例的测试集上进行了广泛验证，实验结果充分证实了该算法的有效性和优越性。

(2) 在 DPFSP 研究的基础上，本研究进一步扩展了生产模式，将产品装配环节纳入考量，构建了分布式装配置换流水车间调度问题（DAPFSP）的数学模型。针对问题特征，设计了一种新型初始化方法以避免冗余解的产生。创新性地提出了一种改进的路径重联方法作为全局搜索策略，并采用优化的突变策略来增强解的多样性，有效避免了局部最优问题。此外，设计了三种关键路径的深度搜索机制以强化算法后期的搜索能力，同时引入泰勒加速度方法降低了算法的计算复杂度。通过系统的消融实验，验证了所提出搜索策略的有效性。

(3) 考虑到实际生产环境中加工时间的不确定性，本研究引入三角模糊数来

表示模糊加工时间，建立了模糊加工时间下的分布式装配置换流水车间调度问题模型，以最小化模糊完工时间为优化目标。为高效求解该问题，提出了一种基于区域地理学优化算法的智能优化方法，通过迁移率模型模拟解空间中的种群迁移行为，显著增强了算法的全局搜索能力。

(4) 在集成制造系统背景下，本研究探讨了分布式柔性装配置换流水车间调度问题，以应对多工厂协同生产中的复杂动态特性。首先采用混合整数线性规划方法建立了问题的数学模型，以最小化最大完工时间和优化资源配置为目标。针对问题的高维性和非线性特征，创新性地提出了一种融合 SARSA 强化学习的生物地理学优化算法，有效提升了算法的求解性能。

(5) 在绿色制造理念指导下，本研究重点研究了考虑机器能耗的绿色分布式装配置换流水车间调度问题。建立了包含能耗指标的多目标优化模型，旨在同时最小化最大完工时间和总能耗。为解决这一复杂问题，提出了一种高效的多目标生物地理学优化算法，该算法结合了混合迁移算子和改进的突变算子，显著增强了算法的全局搜索能力和解的多样性。实验结果表明，所建立的能耗模型能够准确反映实际生产中的能源消耗情况。改进后的多目标算法在 Pareto 解集的分布性和收敛性方面表现出色。通过显著性分析，揭示了关键参数对调度结果的显著影响，为绿色制造实践提供了重要的理论支撑。

7.2 工作展望

尽管本文在 DPFSP 问题上取得了一定的研究成果，但仍存在诸多不足之处。未来的研究工作可以从以下几个方面进一步展开：

7.2.1 问题层面

(1) 多约束环境下的调度问题：本文研究的模型主要针对基本的加工时间与能耗优化问题，未来可进一步考虑资源共享、批处理和设备维护等约束条件，构建更加贴近实际生产环境的调度模型。

(2) 动态调度问题：实际生产环境中，任务到达时间具有不确定性，机器故障等动态因素对调度方案影响较大。后续研究可探索在线调度方法，开发实时调

度优化算法。

(3) 多目标优化问题：未来可扩展研究多目标调度问题，综合考虑完工时间、能耗、碳排放、设备利用率等多个优化目标，设计多目标协同优化方法。

7.2.1 算法层面

(1) 自适应参数调节机制：目前算法参数主要通过实验预设，未来可研究基于反馈学习的自适应参数调节策略，进一步提升算法的鲁棒性。

(2) 深度学习与智能优化的融合：探索深度强化学习与元启发式算法相结合的方法，通过学习历史搜索经验来指导搜索过程。

(3) 混合优化算法：进一步研究遗传算法、粒子群优化算法等智能优化算法的融合机制，发挥不同算法的互补优势，提升求解性能。

7.2.1 应用层面

(1) 智能制造系统集成：将 HBBO 算法嵌入智能制造执行系统（MES）中，实现生产调度的自动化与智能化。

(2) 绿色低碳制造调度：考虑碳排放约束，研究面向绿色制造的调度优化技术，助力低碳生产。

(3) 工业物联网与边缘计算应用：结合工业物联网技术，开发基于实时数据感知的动态调度系统。

(4) 跨行业应用：将 HBBO 算法推广至物流配送、能源调度等领域，验证算法在多领域的适用性和泛化能力。

参考文献

[1] Lohmer J, Lasch R Production planning and scheduling in multi-factory production networks: a systematic literature review[J]. International Journal of Production Research, 2020, 59(7): 2028-2054.

[2] Han X, Han Y, Chen Q, et al. Distributed flow shop scheduling with sequence-dependent setup times using an improved iterated greedy algorithm[J]. Complex System Modeling and Simulation, 2021, 1(3): 198-217.

[3] Cai J C, Lu S, Cheng J, et al. Collaborative variable neighborhood search for multi-objective distributed scheduling in two-stage hybrid flow shop with sequence-dependent setup times[J]. Scientific Reports, 2022, 12(1).

[4] Cai J C, Zhou R, Lei D M. Fuzzy distributed two-stage hybrid flow shop scheduling problem with setup time: collaborative variable search[J]. Journal of Intelligent & Fuzzy Systems, 2020, 38(3): 3189-3199.

[5] Naderi B, Ruiz R. The distributed permutation flowshop scheduling problem[J]. Computers & Operations Research, 2010, 37(4): 754-768.

[6] Jing L, Kegao Q. Local search-based metaheuristics for the robust distributed permutation flowshop problem[J]. Applied Soft Computing, 2021, 105(1).

[7] Pan Q K. Han, B, Gao L. A cooperative iterated greedy algorithm for the serial distributed permutation flowshop scheduling problem[J]. International Journal of Production Research, 2024, 62(12): 4245-4272.

[8] Taillard E, Some efficient heuristic methods for the flow shop sequencing problem[J]. European Journal of Operational Research, 1990, 47(1): 65-74.

[9] Hatami S, Andres-Romano C. The distributed assembly permutation flowshop scheduling problem[J]. International Journal of Production Research, 2013, 51(17): 5292-5308.

[10] Lin J, Zhang S. An effective hybrid biogeography-based optimization algorithm

for the distributed assembly permutation flow-shop scheduling problem [J]. Computers & Industrial Engineering, 2016, 97.

[11] Hamzadayı A, Arvas MA, Elmi A. Distributed assembly permutation flow shop problem; single seekers society algorithm[J]. Journal of Manufacturing Systems, 2021, 61: 613-631.

[12] Zhang Z, Hu R, Qian B, et al. A matrix cube-based estimation of distribution algorithm for the energy-efficient distributed assembly permutation flow-shop scheduling problem[J]. Expert Systems with Applications, 2022, 194: 116484.

[13] Ferone D, Hatami S, González-Neira EM, et al. A biased-randomized iterated local search for the distributed assembly permutation flow-shop problem[J]. International Transactions in Operational Research, 2020, 27: 1368-1391.

[14] Wang S, Wang L. An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem[J]. IEEE Transactions on Systems Man & Cybernetics Systems, 2015, 46(1): 139-149.

[15] Hatami S, Ruiz R, Andres-Romano C. Heuristics and metaheuristics for the distributed assembly permutation flow shop scheduling problem with sequence dependent setup times[J]. International Journal of Production Economics, 2015, 169: 76-88.

[16] Yang S, Xu Z. The distributed assembly permutation flowshop scheduling problem with flexible assembly and batch delivery[J]. International Journal of Production Research, 2020, 59(13): 1-19.

[17] Niu W, Li J Q, Jin H, et al. Bi-objective optimization using an improved NSGA-II for energy-efficient scheduling of a distributed assembly blocking flowshop[J]. Engineering Optimization, 2023, 55(5): 719-740.

[18] Huang Y Y, Pan Q K, Gao L. An effective memetic algorithm for the distributed flowshop scheduling problem with an assemble machine[J]. International Journal of Production Research, 2023, 61(6): 1755-1770.

[19] Zhang Z, Qian B, Hu R. A matrix-cube-based estimation of distribution algorithm for the distributed assembly permutation flow-shop scheduling problem[J]. Swarm

and Evolutionary Computation, 2021, 60: 100785.

[20] Song H, Lin J. A genetic programming hyper-heuristic for the distributed assembly permutation flow-shop scheduling problem with sequence dependent setup times[J]. Swarm and Evolutionary Computation, 2021, 60: 100807.

[21] Cai J C, Lei D M, Wang J, et al. A novel shuffled frog-leaping algorithm with reinforcement learning for distributed assembly hybrid flow shop scheduling[J]. International Journal of Production Research, 2023, 61(4): 1233-1251.

[22] Huang, Y, Pan Q K, Gao L, et al. A two-phase evolutionary algorithm for multi-objective distributed assembly permutation flowshop scheduling problem[J]. Swarm and Evolutionary Computation, 2022, 74.

[23] Wang M, Zhang J, Zhang P, et al. Independent double DQN-based multi-agent reinforcement learning approach for online two-stage hybrid flow shop scheduling with batch machines[J]. Journal of Manufacturing Systems, 2022, 65: 694-708.

[24]吕佑龙, 张洁, 汪俊亮, 等. 混流装配线生产计划智能优化方法[M]. 北京:电子工业出版社, 2021.

[25]张洁, 吕佑龙, 汪俊亮, 等. 智能车间的大数据应用[M]. 北京:清华大学出版社, 2020.

[26] Cheng L, Wang L, Cai J C. A regional biogeography-based optimization algorithm for the distributed assembly permutation flow-shop scheduling problem with fuzzy processing time[J]. Journal of Intelligent and Fuzzy Systems, 2024, 46(2): 3827-3841.

[27] Zhang G, Xing K, Cao F. Scheduling distributed flowshops with flexible assembly and set-up time to minimize makespan[J]. International Journal of Production Research, 2018, 56(9): 3226-3244.

[28] Yang S, Xu Z. The distributed assembly permutation flowshop scheduling problem with flexible assembly and batch delivery[J]. International Journal of Production Research, 2020, 59(13): 1-19.

[29] Wang J, Wang L. A cooperative memetic algorithm with feedback for the energy-aware distributed flow-shops with flexible assembly scheduling[J]. Computers and Industrial Engineering, 2022, 168.

[30] Pan Q K, Gao L, Li X, et al. Effective constructive heuristics and meta-heuristics for the distributed assembly permutation flowshop scheduling problem[J]. Applied Soft Computing, 2019, 81.

[31] Pourhejazy P, Cheng C, Ying K, et al. Meta-Lamarckian-based iterated greedy for optimizing distributed two-stage assembly flowshops with mixed setups[J]. Annals of Operations Research, 2023, 322:125-146.

[32] Li M, Wang G. A review of green shop scheduling problem[J]. Information Sciences, 2022, 589: 478-496.

[33] Zhang R, Chiong R. Solving the energy-efficient job shop scheduling problem: a multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption[J]. Journal of Cleaner Production, 2016, 112: 3361-3375.

[34] Din S, Khan Q, Rehman F, et al. A comparative experimental study of robust sliding mode control strategies for underactuated systems[J]. IEEE Access, 2017, 5: 10068-10080.

[35] Yannibelli V, Amandi A. Hybridizing a multi-objective simulated annealing algorithm with a multi-objective evolutionary algorithm to solve a multi-objective project scheduling problem[J]. Expert Systems with Applications, 2013, 40(7): 2421-2434.

[36] An Y, Chen X, Li Y, et al. An improved nondominated sorting biogeography-based optimization algorithm for the (hybrid) multi-objective flexible job-shop scheduling problem[J]. Applied Soft Computing, 2021, 99: 106869.

[37] Petrovic M, Jokic A, Miljkovic Z, et al. Multi-objective scheduling of a single mobile robot based on the grey wolf optimization algorithm[J]. Applied Soft Computing, 2022, 131: 109784.

[38] Yagmahan B, Yenisey M. Ant colony optimization for multi-objective flow shop scheduling problem[J]. Computers & Industrial Engineering, 2008, 54(3): 411-420.

[39] Cai J C, Lei D M, Li M. A shuffled frog-leaping algorithm with memeplex quality for bi-objective distributed scheduling in hybrid flow shop[J]. International Journal

of Production Research, 2021, 59(18): 5404-5421.

[40] FarajiAmiri M, Behnamian J. Multi-objective green flowshop scheduling problem under uncertainty: Estimation of distribution algorithm[J]. Journal of Cleaner Production, 2020, 251: 119734.

[41] Huang Y Y, Pan Q K, Gao L et al. A two-phase evolutionary algorithm for multi-objective distributed assembly permutation flowshop scheduling problem[J]. Swarm and Evolutionary Computation, 2022, 74: 101128.

[42] Meng L, Zhang C, Zhang B et al. Milp modeling and optimization of multi-objective flexible job shop scheduling problem with controllable processing times[J]. Swarm and Evolutionary Computation, 2023, 82: 101374.

[43] Wei L, He J, Guo Z, et al. A multi-objective migrating birds optimization algorithm based on game theory for dynamic flexible job shop scheduling problem[J]. Expert Systems with Applications, 2023, 227: 120268.

[44] He L, Li W, Chiong R, et al. Optimising the job shop scheduling problem using a multi-objective jaya algorithm[J]. Applied Soft Computing, 2021, 111: 107654.

[45] Dong J, Ye C. Green scheduling of distributed two-stage reentrant hybrid flow shop considering distributed energy resources and energy storage system[J]. Computers & Industrial Engineering, 2022, 169: 108146.

[46] Hou Y, Wang H, Huang X. A q-learning-based multi-objective evolutionary algorithm for integrated green production and distribution scheduling problems[J]. Engineering Applications of Artificial Intelligence, 2024, 127: 107434.

[47] Xue Y, Rui Z, Yu W, et al. Estimation of distribution evolution memetic algorithm for the unrelated parallel-machine green scheduling problem[J]. Memetic computing, 2019, 11(4): 423-437.

[48] Li Y, Pan Q, Gao K, et al. A green scheduling algorithm for the distributed flowshop problem[J]. Applied Soft Computing, 2021, 109: 107526.

[49] Simon D. Biogeography-based optimization[J]. IEEE Transactions on Evolutionary Computation, 2008, 12(6): 702-713.

[50] Beaver R, Montgomery D. Design and analysis of experiments[J]. Biometrics,

1977, 33(1).

[51] Pan Q K, Gao L, Wang L, et al. Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flow shop problem[J]. Expert Systems with Applications, 2019, 124: 309-324.

[52] Naderi B, Ruiz R. A scatter search algorithm for the distributed permutation flow shop scheduling problem[J]. European Journal of Operational Research, 2014, 239(2): 323-334.

[53] Glover F. Tabu search and adaptive memory programming-advances, applications and challenges[J]. Interfaces in computer science and operations research, 1997, 7: 1-75.

[54] Taillard E. Some Efficient Heuristic Methods for the Flow Shop Sequencing Problem[J]. European Journal of Operational Research, 1990, 47(1): 65-74.

[55] Lei D M. Fuzzy job shop scheduling problem with availability constraints[J]. Computers & Industrial Engineering, 2010, 58(4): 610-617.

[56] Lei D M. Pareto archive particle swarm optimization for multi-objective fuzzy job shop scheduling problems[J]. International Journal of Advanced Manufacturing Technology, 2008, 37(1-2): 157-165.

[57] Niu W, Li J, Jin H, et al. Bi-objective optimization using an improved NSGA-II for energy-efficient scheduling of a distributed assembly blocking flowshop[J]. Engineering Optimization, 2022, 55(5): 719-740.

[58] Khan B, Govindan K. A multi-objective simulated annealing algorithm for permutation flow shop scheduling problem[J]. International Journal of Advanced Operations Management, 2011, 3(1): 88.

[59] Coello C, Pulido G, Lechuga M. Handling multiple objectives with particle swarm optimization[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 256-279.

[60] 张静, 宋洪波, 林剑. 分布式装配置换流水车间调度问题研究综述[J]. 计算机工程与应用, 2024, 60(6): 1-9.

[61] 王雷, 蔡劲草, 石鑫. 基于改进遗传算法的多目标柔性作业车间节能调度问

题[J]. 南京理工大学学报, 2017, 41(4): 494-502.

[62] 王雷, 蔡劲草, 唐敦兵, 等. 基于改进遗传算法的柔性作业车间调度[J]. 南京航空航天大学学报, 2017, 49(6): 779-785.

[63] 张梓琪, 钱斌, 胡蓉, 等. 基于多维 EDA 算法的低碳分布式装配流水车间调度[J]. 控制与决策, 2022, 37(5): 1367-1377.

攻读硕士学位期间科研成果及获奖情况

一、 发表学术论文及申请专利情况

[1] **第一完成人.** A Regional Biogeography-based Optimization Algorithm for the Distributed Assembly Permutation Flow-shop Scheduling Problem with Fuzzy Processing Time, 2024, Journal of intelligent & fuzzy system, 3827-3841. (中科院四区)

[2] **第一完成人.** Collaborative assembly factory in the distributed permutation flow-shop scheduling problem considering flexible assembly and fuzzy processing time, 2025, Engineering Optimization, 132. (中科院三区)

[3] **第一完成人.** A genetic biogeography-based optimization algorithm for distributed assembly permutation flow-shop scheduling problem, 2025, International Journal of Modeling, Simulation, and Scientific Computing. (EI 收录，小修)

[4] **第一完成人.** An effective biogeography-based optimization algorithm for multi-objective green scheduling of distributed assembly permutation flowshop scheduling problem, 2025, Computers & Operation Research. (中科院二区，小修)

[5] **第一完成人.** A hybrid biogeography-based optimization algorithm for distributed assembly permutation flowshop scheduling problem, 2025, AI Communication. (中科院四区，小修)

二、 参与科研项目

[1] 安徽省高校重点项目《基于改进群智能优化算法的柔性作业车间绿色调度问题研究》(项目编号：2022AH050978)

[2] 检测技术与节能装置安徽省重点实验室项目《基于动态群智能优化的分布式混合流水车间节能调度问题研究》(项目编号：JCK2022B01)

[3] 安徽省机器视觉检测与感知重点实验室《基于深度强化学习的移动机器人路径规划研究》(项目编号：KLMVI-2024-HIT-15)

[4] 安徽省高校重点项目《基于动态群智能的多约束分布式混合流水车间调

度研究》(项目编号：2023AH050935)

[5] 芜湖市科技项目《面向节能的多 AGV 动态路径规划研究》(项目编号：2022jc26)

[6] 鸠江区产业协同项目《焊接机器人路径规划智能优化方法及应用研究》(项目编号：2022cyxtb6)

三、 获奖情况

2023 年，三等学业奖学金

2024 年，二等学业奖学金

2024 年，安徽工程大学数学建模竞赛二等奖

致 谢

时光荏苒，转眼间我的研究生生涯即将结束。在此，我想向一路走来给予我支持和帮助的亲人、导师、同门、朋友们表达最诚挚的感谢。

首先，衷心感谢我的父母。从小到大，你们用无私的爱陪伴我成长，在我迷茫时给予指导，在我疲惫时给予鼓励，让我有勇气不断追求更高的目标。你们的支持是我前行的最大动力。

其次，感谢我的导师王雷老师和蔡劲草老师。在科研的道路上，你们不仅给予我悉心的指导，还用严谨的治学态度和宽广的学术视野影响着我，使我受益匪浅。感谢你们在课题研究过程中给予的耐心指导和鼓励，让我得以不断进步。

感谢课题组的同门胡孔夫、王天成、王艺璇，以及所有的师弟师妹们。在科研的日子里，我们相互讨论、共同进步，这段并肩奋斗的时光将成为我人生中宝贵的回忆。感谢我的寝室室友许宇翔、樊金生、吴文超。感谢你们在生活中给予的陪伴与支持，尤其怀念我们一起跑步的闲暇时光，那些挥洒汗水的瞬间。

特别感谢我的发小范逸帆。从中考到高考，再到研究生考试，每一个重要的考试阶段都有你的陪伴，你的鼓励让我在关键时刻更加坚定。希望你未来一切顺利，越来越好！感谢我的朋友高谦，回想起那些一起学习的日子，仍然倍感珍惜。你让我在求学路上不再孤单，你的陪伴和帮助让我收获颇多。

衷心感谢所有在我成长道路上给予帮助和支持的人，愿我们都能在各自的领域里不断前行，收获更好的自己。

尚德敏学　唯实惟新