学校代码： 10491 研究生学号： 1202011179

中国地质大学

硕士学位论文

模型驱动的学习型模因算法求解绿色分布式柔性作业车间调度问题

姓 名： 李瑞

学 科 专 业： 计算机科学与技术

指 导 教 师： 龚文引 教授

培 养 单 位： 计算机学院

二〇二三年六月

A Dissertation Submitted to China University of Geosciences

For the Master Degree of Atmospheric Science

Model-driven learning-based memetic algorithms for green distributed flexible job shop scheduling problem

Master Candidate： Li Rui

Major： Computer Science and Technology

Supervisor： Prof. Wenyin Gong

China University of Geosciences

Wuhan 430074 P.R. China

中国地质大学（武汉）研究生学位论文原创性声明

本人郑重声明：本人所呈交的硕士学位论文《模型驱动的学习型模因算法求解绿色分布式柔性作业车间调度问题》，是本人在导师的指导下，在中国地质大学（武汉）攻读硕士学位期间独立进行研究工作所取得的成果。论文中除已注明部分外不包含他人已发表或撰写过的研究成果，对论文的完成提供过帮助的有关人员已在文中说明并致以谢意。

本人所呈交的硕士学位论文没有违反学术道德和学术规范，没有侵权行为，并愿意承担由此而产生的法律责任和法律后果。

学位论文作者签名：_____

日　期：　　年　　月　　日

中国地质大学（武汉）研究生学位论文导师承诺书

本人郑重承诺：本人所指导的硕士学位论文《模型驱动的学习型模因算法求解绿色分布式柔性作业车间调度问题》，是在本人的指导下，研究生在中国地质大学（武汉）攻读硕士学位期间独立进行研究工作所取得的成果，论文由研究生独立完成。

研究生所呈交的硕士学位论文没有违反学术道德和学术规范，没有侵权行为，并愿意承担由此而产生的与导师相关的责任和后果。

指导教师（签字）：_____

日　期：　　年　　月　　日

中国地质大学（武汉）学位论文使用授权书

本人授权中国地质大学（武汉）可采用影印、缩印、数字化或其它复制手段保存本学位论文；学校可向国家有关部门或机构送交本学位论文的电子版全文，编入有关数据库进行检索、下载及文献传递服务；同意在校园网内提供全文浏览和下载服务。

涉密论文解密后适用于本授权书。

学位论文作者签名：_____

日　期：　　年　　月　　日

作者简历

一、基本情况

姓名：李瑞，性别：男，民族：汉族，出生年月：1998-10-01

籍贯：湖北省荆州市洪湖市

学习经历：

2016.09—2020.06 中国地质大学（武汉） 计算机学院，信息安全工学学士

2020.09—2023.06 中国地质大学（武汉） 计算机学院，计算机科学与技术，攻读工学硕士学位(推免)

二、学术论文

1. **R. Li,** W. Gong*, L. Wang, C. Lu, A learning-based memetic algorithm for energy-efficient flexible job shop sc heduling with type-2 fuzzy processing time, IEEE Transactions on Evolutionary Computation [J]. 2022, Online first. (CUG-T1, IF=16.459) (第二章中基于 Q 学习和指标反馈的自适应参数选择模型，贪心节能策略)
2. **R. Li,** W. Gong*, L. Wang, C. Lu, Surprisingly Popular-based Adaptive Memetic Algorithm for Energy-efficient Distributed Flexible Job Shop Scheduling, IEEE Transactions on Cybernetics[J]. 2022, Online first. (CUG-T1, IF=19.118)(第三章中基于意外流行算法的算子选择模型)
3. **R. Li**, W. Gong*, L. Wang, C. Lu, S Jiang, Two-stage knowledge-driven evolutionary algorithm for distributed flexible job shop scheduling problem with type-2 fuzzy processing time, Swarm and Evolutionary Computation [J]. 2022,74,101139. (CUG-T1, IF= 10.267)(第三章中基于全主动调度解码的节能策略)
4. **R**. Li, W. Gong*, C. Lu, A reinforcement learning based RMOEA/D for bi-objective fuzzy flexible job shop scheduling [J], Expert System with Application. 2022, 203, 117380. (CUG-T2, IF=8.665).(第二章中增强策略)
5. **R. Li**, W. Gong*, C. Lu, Self-adaptive multi-objective evolutionary algorithm for flexible job shop scheduling with fuzzy processing time, Computers & Industrial Engineering [J]. 2022, 168, 108099. (CUG-T2, IF=7.81)(第三章中基于置信度的选择模型)

6. **李瑞**, 王凌, 龚文引*, 知识驱动的模因算法求解分布式绿色柔性调度, 华中科技大学学报（自然科学版）. 2022，50(06): 55-60. (EI) (第三章中基于关键路径的局部搜索)
7. **李瑞**, 龚文引*, 改进的基于分解的多目标进化算法求解双目标模糊柔性作业车间调度, 控制理论与应用. 2022, 39(1): 31-40. (EI) (第二章中MOEA/D 算法)
8. **R. Li,** W. Gong*, L. Wang, C. Lu, Co-evolution with Deep Reinforcement Learning for Energy-Aware Distributed Heterogeneous Flexible Job Shop Scheduling, IEEE Transactions on Systems, Man and Cybernetics: Systems[J]. 2022. (R2 Under Review, CUG-T1, IF=11.471) (第四章算法，第五章实验结果)

三、获奖、专利情况

1. 2021 年，中国地质大学（武汉），研究生国家奖学金
2. 2022 年，中国地质大学（武汉），研究生国家奖学金
3. 2023 年，第六届智能优化与调度学术会议，优秀论文汇报二等奖
4. 2022 年，中国地质大学（武汉），第三十三届校级科技论文报告会，校级特等奖
5. 2022 年，中国地质大学（武汉），校级优秀共青团员
6. 2020 年，第十七届“华为杯”全国研究生数学建模竞赛，三等奖（国家级）
7. 2021 年，第十八届“华为杯”全国研究生数学建模竞赛，三等奖（国家级）
8. 龚文引，卢超，**李瑞**，一种多园区模糊下料车间系统和方法，位次 3，专利号：202211548118.9，申请日期 2022 年 12 月 6 号

四、研究项目

1. 复杂施工环境下大型工程装备设计/制造/运维一体化平台研发与应用，课题二，不确定环境下工程装备供应链动态协同与智能柔性排产方法，科技部国家重点研发计划项目课题，项目编号：2021YFB3301602，参加人员，2022.1-2024.12
2. 智能互联装备群组的分层制造协同技术，科技部国家重点研发计划项目课题，项目编号：2020YFB1712102，参加人员，2023.1-2023.12
3. 面向车间能效调度的增强智能优化理论与方法，国家自然科学基金项目，项目编号：62273193，参加人员，2023.1-2026.12

五、参加会议

1. 2022 年 8 月 10 日，第五届“智能优化与调度”学术年会，山西太原
2. 2023 年 5 月 18 日，第六届“智能优化与调度”学术年会，湖南张家界

摘　要

制造业对工业国家的经济、就业、民生有着至关重要的影响。随着我国智能制造战略的推进，要求加快企业数字化转型，通过智能技术提升企业产能，增强企业国际竞争力。智能制造的核心为智能车间调度，通过建模和智能算法辅助企业制定排产方案。本文将实际生产场景建模为绿色分布式异构柔性作业车间调度问题(Green distributed heterogeneous flexible job shop scheduling problem, GDHFJSP)，同时优化最大完工时间和总能耗。为了更好地理解 GDHFJSP 的性质，本文展开从绿色柔性作业车间调度问题(Green flexible job shop scheduling problem, GFJSP)到同构绿色分布式柔性作业车间调度问题(Distributed GFJSP, GDFJSP)再到 GDHFJSP 的研究，构建了多种 MILP 模型，并提出模型驱动的学习型模因算法求解，最后在实际工程问题上进行验证。

本文具体研究内容和创新点如下：

(1) 构建了最小化最大完工时间和总能耗的 GFJSP 模型，提出了基于 Q 学习模型驱动的模因算法(LRVMA)

LRVMA 主要特点是：通过对模型性质的分析，提炼知识，依据知识设计了多种增强策略例如混合启发式初始化策略，模型驱动的局部搜索策略和启发式贪心节能策略。结合 Q 学习设计了指标反馈的参数自学习模型，引导算法向分布性更优的方向进化。在公开的 T2FJSP 测试集上测试，验证了 Q 学习参数自适应策略能增强种群分布性，贪心节能策略可以有效降低生产车间的能耗。

(2) 构建了最小化最大完工时间和总能耗的同构 GDFJSP，提出了基于意外流行学习的模型驱动的模因算法(SPAMA)

SPAMA 的主要特点是：基于析取图理论，设计了四种基于关键路径和关键块的局部搜索算子。受全主动调度解码理论启发，设计了全主动调度节能策略。受到意外流行算法启发，设计了自适应概率修正的算子选择模型。把 SPAMA 在经典柔性作业车间调度测试集 Mk 和 DP 上进行测试，验证了基于关键路径设计算子可以有效提高局部搜索成功率，全主动调度解码可以有效降低车间能耗，意外流行算法可以增强小权重高效算子的选择概率，加速算法收敛。

(3) 构建了研究最小化最大完工时间和总能耗的 GDHFJSP，提出了基于深度 Q 学习的模型驱动的模因算法(DQPEA)

DQPEA 的主要特点是：通过对工厂异构和机器柔性特性的分析，设计了 9 种基于知识的局部搜索算子。设计了改进的模因进化框架，通过多种群协同平衡全局和局部搜索的计算资源。提出了基于深度 Q 学习的算子选择模型，通过神经网络

建立解和算子的映射实现为所有解定制化选择最优算子的目的。把 DQPEA 在生成 20 个不同规模的测试问题上测试，通过对比最新相关算法验证了提出算法的有效性和鲁棒性，在 GDHFJSP 有良好性能。

(4) 将提出的理论和方法应用于某国有大型工程装备生产企业的下料车间场景。基于实际工程数据，将问题抽象为 DHFJSP 问题，并利用提出的算法和改进策略进行求解，结果验证了提出算法在实际问题上的有效性。

本文从 GFJSP 到同构 GDFJSP 再到 GDHFJSP 递进地研究了分布式柔性作业车间调度问题模型的特性，在模型上丰富了柔性作业车间调度问题的研究。其次，设计了模型驱动的学习型 MA 求解不同类型的问题。学习策略从 Q 学习到意外流行学习再到深度 Q 学习，通过大量实验验证了提出算法的有效性，在方法上进一步丰富模因算法的研究成果。最后，把改进算法应用到大型工程装备分布式下料车间优化问题的求解中，展示了改进算法的高效性能，在应用上可为大型工程装备分布式下料车间提供可靠优化工具。因此本文的研究具有良好的理论和现实意义。

关键词：分布式异构柔性作业车间调度问题；模型驱动的模因算法；强化学习；意外流行算法；绿色调度

ABSTRACT

Manufacturing industry has a vital impact on the economy, employment and people's livelihood of industrial countries. Along with the advancement of intelligent manufacturing strategy, enterprises are required to make digital transformation, enhance their production capacity and enhance their international competitiveness through intelligent technology. The core of intelligent manufacturing is intelligent shop scheduling, which assists enterprises to make production scheduling schemes through modeling and intelligent algorithms. This thesis abstracted the actual production scene as a green distributed heterogeneous flexible job shop scheduling problem (GDHFJSP) with minimizing makespan and total energy consumption (TEC). To better understand the features of DHFJSP, this work studies from green flexible job shop scheduling problem (GFJSP) to homogeneous distributed GFJSP (GDFJSP) and then to GDHFJSP. Several MILP models are built, and the model-driven and learning-based memetic algorithms are proposed to solve them. Finally, a practical engineering problem is used to verify the effectiveness of proposed algorithms. Specific research contents are as follows:

(1) A GFJSP model was constructed with minimizing the makespan and TEC, and a Q learning and model-based memetic algorithm (LRVMA) was proposed. The main characteristics of LRVMA are as follows: based on problem features, various enhancement strategies are designed according to the knowledge, such as hybrid heuristic initialization strategy, model-driven local search strategy and heuristic greedy energy saving strategy. Combined with Q learning, a parameter self-learning model based on metric feedback is designed to guide the algorithm to evolve in the direction of better distribution. It was tested on the published T2FJSP benchmark, and it was verified that Q learning parameter adaptive strategy can enhance the population distribution, and greedy energy saving strategy can effectively reduce the TEC in the production workshop.

(2) An homegeneous GDFJSP was constructed with minimizing the makespan and TEC, and a surprisingly popular algorithm and model-based meme algorithm (SPAMA) was proposed. The main features of SPAMA are as follows: based on disjunctive graph theory, four kinds of local search operators based on critical path and critical block are designed. Inspired by the decoder theory of full active scheduling, the energy-saving strategy of full active scheduling is designed. Inspired by the emergent popular algorithm, an operator selection model with adaptive probability correction is designed. SPAMA was

tested on classical flexible job-shop scheduling benchmarks Mk and DP. It is verified that the success rate of local search can be effectively improved based on critical path design operator, the full active scheduling decoding can effectively reduce the shop energy consumption, and the unexpected popular algorithm can enhance the selection probability of low-weight efficient operator and accelerate the algorithm convergence.

(3) A GDHFJSP was constructed to study the minimization of makespan and TEC, and a deep Q-learning and model-based meme algorithm (DQPEA) was proposed. The main characteristics of DQPEA are as follows: By analyzing the characteristics of plant heterogeneity and machine flexibility, nine kinds of knowledge-based local search operators are designed. An improved framework of meme evolution is designed to balance the computational resources of global and local search through multi-population collaboration. An operator selection model based on deep Q learning is proposed. The mapping between solutions and operators is established by neural network to customize the optimal operator for all solutions. DQPEA was tested on 20 different scale test questions, and the effectiveness and robustness of the proposed algorithm were verified by comparing with the latest correlation algorithms. It has good performance in GDHFJSP.

(4) The theory and method proposed in this thesis is applied to the blanking workshop scene of a state-owned large-scale engineering equipment production enterprise. Based on the actual engineering data, the problem is abstracted into DHFJSP problem, and the proposed algorithm and improved strategy are used to solve the problem. The results verify the effectiveness of the proposed algorithm on the actual problem.

This thesis studies the characteristics of distributed flexible job shop scheduling problem model from GFJSP to homegeneous GDFJSP and then to GDHFJSP, which enricfies the research of flexible job shop scheduling problem. Secondly, the model driven learning MAs are designed to solve different types of problems. The learning strategies range from Q learning to surprisingly popular learning and then to deep Q learning. A large number of experiments have verified the effectiveness of the proposed algorithms and further enriched the research results of meme algorithms in terms of methods. Finally, the improved algorithms are applied to solve the optimization problem of distributed blanking shop of large engineering equipment, and the high efficiency of the improved algorithm is demonstrated. In application, it can provide a reliable optimization tool for distributed blanking shop of large engineering equipment. So the research of this thesis has good theoretical and practical significance.

Key Words: Distributed heterogeneous flexible job-shop scheduling problem; Memetic algorithm; Reinforcement learning; Surprisingly popular algorithm; Green scheduling

目 录

第一章 绪论

1.1 课题来源、背景、目的和意义

1.1.1 课题来源

本研究获得了以下科研项目的资助：

国家重点研发计划课题“复杂施工环境下大型工程装备设计/制造/运维一体化平台研发与应用”，课题二“不确定环境下工程装备供应链动态协同与智能柔性排产方法”，项目编号 2021YFB3301602。

1.1.2 研究背景

二十大报告提出“建设现代化产业体系。坚持把发展经济的着力点放在实体经济上，推进新型工业化，加快建设制造强国”。智能制造是我国自“十四五”计划以来重点发展方向，以国家发改委发布的《“十四五”智能制造发展规划》作为指导性文件，加快产业转型升级，实现数字化网络化。

国有大型生产企业为了增加国际市场竞争力，选择从批量制造模式转型为高端定制化模式[1]。大型工程装备的工艺路线主要由下料，机加工，铆焊和涂色装配四步。由于产品高度定制化，每一批订单的生产工时没有历史数据参加。这导致了零件设计图经常修改，加工工时无法估计的问题，进而造成生产计划变更频繁，订单交付不及时的问题。其次，企业为了加强国内市场的竞争力，采用快速生产的模式，将原本需要六个月周期的装备的生产周期缩短至三个月，使得排产难度极具增加。如何有效组织不同园区的资源，合理安排园区内的产能，成为了企业最迫切解决的问题。更多地，在各个生产车间中都有不同的痛点急需解决。在机加工车间中，部分机床具有兼容性可以加工不同种类的零件，这给排产带来了巨大的挑战。其次，不同的园区功能相同，相同订单可以在每个园区生产。因此，同时面对园区柔性和机床柔性等条件。合理的安排生产计划使得经济指标最小是关键问题。机加工车间可以转换为分布式异构混合柔性作业车间调度问题（Distributed Heterogeneous Flexible Job Shop Scheduling Problem，DHFJSP）。因此，对 DHFJSP 的模型和算法的研究可以帮助国有企业实际生产中减少成本、增加企业竞争力和提升企业效

率。理论研究抽象于实际问题又落地回实际问题，实现理论与实际的紧密结合。

“十三五”以来可持续发展一直是我国发展的一大主题，其中包含《“十三五”工业绿色发展规划》等指导性文件要求企业加快绿色化转型，加强绿色制造[2]。因此，在从实际生产问题中提炼排产模型时必须考虑绿色指标同时要设计节能技术，以保证设计的算法可输出多种解，实现经济指标与绿色指标协同优化。

自 2015 年以来，分布式车间调度问题[3]已经发展成组合优化中一个热门和前沿的方向。相较于传统车间调度问题，分布式车间调度除了求解车间中工序顺序选择问题和零件的机床选择问题，还需求解订单的工厂分配问题。面对三个耦合的子问题，目标搜索空间更大，问题更加复杂和难以求解。近年来，分布式车间调度问题成为前沿热点，各类模型不断地被提出，例如并行机分布式调度[4,5]，流水车间分布式调度[6,7]，混合流水车间分布式调度[8]，作业车间分布式调度[9]和柔性作业车间分布式调度[10,11]。由此可见，分布式车间调度问题具有较强的理论研究价值和实际应用价值。目前求解分布式车间调度问题的主流算法框架为模因算法[12]（Memetic algorithm，MA）和贪婪迭代算法[13]。MA 的主要由全局搜索和局部搜索两部分组成，且同时作用于一个单一种群。该设计模式可以在特定的问题上求解良好，但面对分布式车间调度问题的复杂解空间，种群过早地收敛到局部最优会降低算法全局搜索的性能。其次，针对所研究的实际问题模型，一般的局部搜索算子具有较强的随机性，无法在短时间内加速算法收敛。因此，分析模型提取知识并根据知识设计局部搜索算子十分关键。更多地，在定制化制造中，算法面对的都是不确定的问题规模。传统的 MA 无法很好地求解各个规模的问题。因此需要为 MA 引入学习机制，增强 MA 的智能性、生命力和自学习能力，使得 MA 能够根据不同问题自我学习和调节。因此，研究模型驱动的学习型 MA 十分重要。

1.1.3 课题的目的与意义

本课题致力于研究学习型 MA 求解绿色 DHFJSP 问题，将设计的改进算法应用于国有大型装备生产商的平台中，为实际生产案例生产甘特图和帮助企业优化排产方案，为实际车间的排产提供理论指导，指导那些订单自产，那些对外协助生产，实现降本增效，提高企业市场竞争力。

（1）问题模型研究：首先，单工厂柔性作业车间调度问题（Flexible job shop scheduling problem，FJSP）是 DHFJSP 的基础。因此，研究 FJSP，分析单工厂内的知识设计合理局部搜索算子加速收敛减少完工时间。其次，同构分布式 FJSP（distributed FJSP，DFJSP）问题是 DHFJSP 问题的基础。研究同构 DFJSP 问题可分析在没有异构资源的干扰下，提炼工厂选择知识设计工厂选择局部搜索算子，减

少所有工厂的总完工时间。最后，研究 DHFJSP 模型，可以模拟大型工程装备生产中机加工车间的实际场景。帮助企业解决实际生产中的痛点。

（2）绿色调度研究：首先，在优化经济指标的基础上加入绿色指标，实现经济指标和绿色指标的双目标优化。其次，针对绿色指标，分析车间生产中与之相关的知识，设计节能技术降低生产车间中的能耗。

（3）学习技术研究：为了增强 MA 的自主能力，为 MA 增加不同的学习机制包括强化学习算法、深度强化学习算法、基于意外流行算法的置信度反馈学习，实现参数自学习和算子自学习，为分布式车间调度问题中学习与反馈驱动增强技术提供理论参考。

综上所述，本文针对模型驱动的学习型 MA 的分布式柔性作业车间调度方法的研究，既可以丰富分布式车间调度理论与方法，又能解决国有大型装备生产企业的生产痛点。因此，本研究具有重要的理论研究意义和实际应用价值。

1.2 国内外研究现状综述

本文针对 FJSP、DFJSP 和 DHFJSP 等三类调度问题展开研究。每类调度问题的优化目标为车间总能量消耗和最大完工时间。因此对于三类调度问题的研究现状重点展开综述。其次，本文设计学习型 MA 求解上述三类问题，在 MA 的框架上加入学习型机制增强算法的自主性。因此，与 MA 相关的研究也将重点综述。本文综述总共分为三个部分：第一部分是对 FJSP 的历史发展进行系统性回顾，并分析阐述目前 FJSP 的研究现状；第二部分是对目前新兴的 DFJSP 的研究进行全面综述；第三部分是对 MA 在车间调度问题方向的研究进行详细的分析阐述。

1.2.1 柔性作业车间调度问题的研究现状

（1）FJSP 问题的发展概述

FJSP 问题是经典作业车间问题的拓展，其最大的特点是工序选择机器的柔性。FJSP 在 1990 年被 Brucker 和 Schlie[14]首次提出。而此前对生产调度问题的研究源于更早的 1950 年后，由于生产调度问题具有现实意义，随之在运筹优化学等学科中逐渐分离出来，形成一门新的学科。1950 年随着工业化的发展，工厂的订单量极具增加，如何高效调度车间内的生产变成继续解决的问题。1954 年 Johnson 首次针对流水车间调度问题提出了优化算法[15]。1975 年首篇车间调度方向的论文在

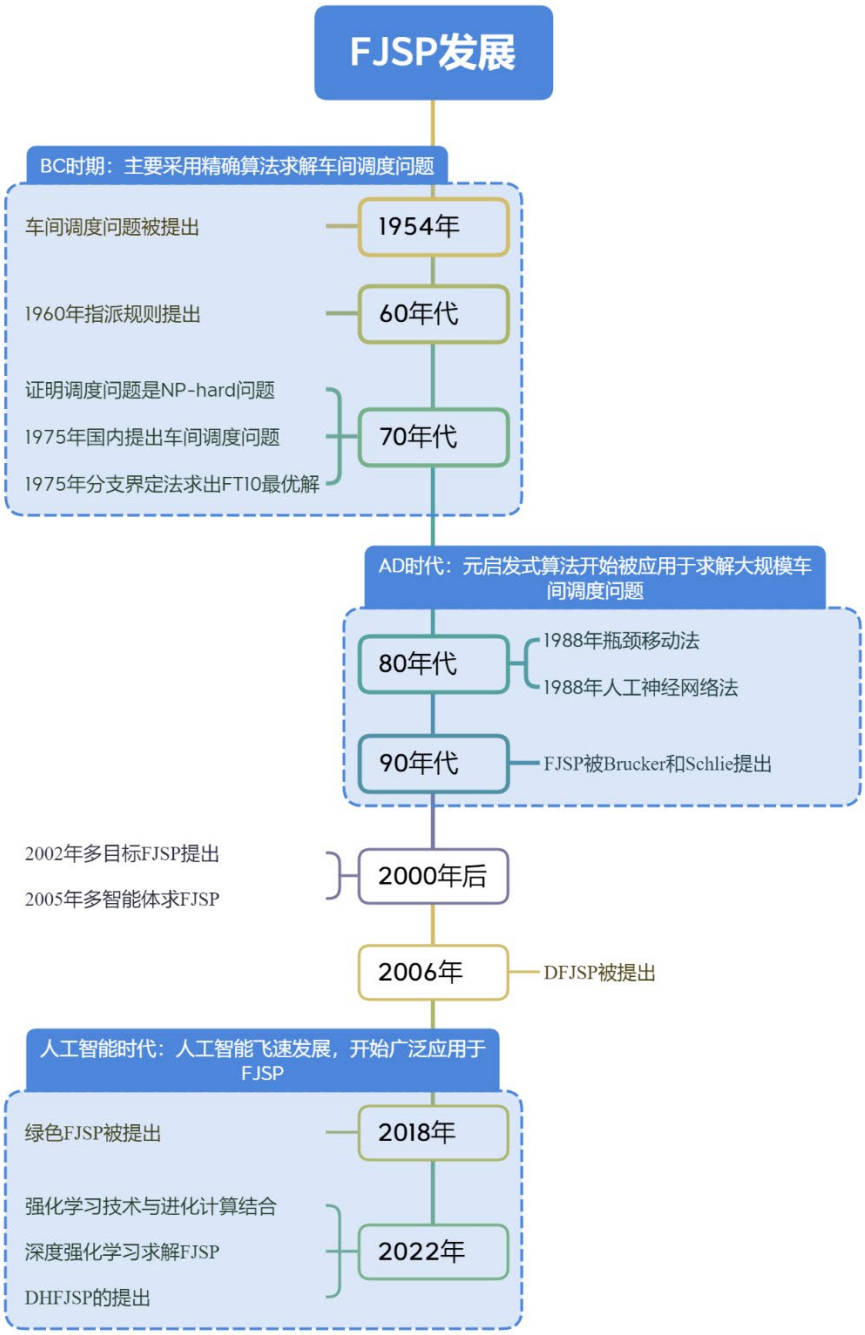

图 1.1 FJSP 的发展历程

《中国科学》发表 [16]，奠定了我国车间调度问题的研究基础。早期人们对于车间调度问题的研究主要集中于精确解法例如动态规划、分支界定和混合整数线性规划等数学方法。该类方法通过严格的数学推导，通过数学运算求得解空间的全局最优解。60 年代，传统的数学推导无法解决大规模的调度问题，复杂而庞大的计算量远超出了人类的能力范围，于是启发式规则被提出，通过专家知识设计简单规则运用计算机辅助计算解决大规模调度问题。随后，车间调度问题的主题框架已经初

步成型。时间过渡到 70 和 80 年代，Cook[17]等计算理论方向的专家通过数学推导证明了车间调度问题属于 NP-hard 问题。从此人们便开始研究有效的优化方法来解决大规模的车间调度问题，经典车间调度理论已具备完整框架。因此，Parker[18]根据车间调度问题的特征，将 70 年代以前的车间调度问题的研究称为 BC 时代（Before Complexity），70 年代以后的研究其后称为 AD 时代（Advanced Difficulty）。1980 年后，随着计算机科学的发展人工智能技术开始渗透到工程领域，学者通过对自然界生物和物理现象的思考抽象的提出了元启发算法并开始应用于求解车间调度问题。该类方法具有快速全局搜索的优点例如，早期的遗传算法、模拟退火和禁忌搜索算法，在大规模车间调度问题上可以快速搜索到全局最优解；90 年代后元启发式算法逐渐占据主导地位。许多元启发式算法如粒子群优化算法、蚁群算法和蜂群算法等被用于车间调度问题。时至今日，由于人工智能技术的兴起，深度强化学习、深度进化学习等方法已经开始被用于求解车间调度问题。图 1.1 展示了 FJSP 的详细发展过程。下一节将对 FJSP 问题的求解方法的发展做详细的阐述。

（2）FJSP 的求解方法的发展

精确方法和近似方法是求解 FJSP 问题的两类主要方法。精确方法有拉格朗日松弛法、混合整数规划、分解方法和分支定界法等，求解精度高但求解规模有限。于是近似方法被提出解决更大规模的车间调度问题，近似方法求解速度快，在精度要求不高的场景下应用广泛。近似方法主要分为四类：瓶颈移动算法、优先分配规则法、人工智能方法等。图 1.2 详细划分了多种 FJSP 求解方法的类别。

精确方法：如图 1.2 所示，一种是数学规划法，另一种是分支界定法。该方法用于精确求解车间调度问题的全局最优解，但由于计算复杂度高，求解时间随着问题规模指数爆炸。因此，在实际问题的应用中受限。精确方法又分为如下几类：

（i）**数学规划法**：主要思想是将调度问题分解成多组等式和不等式约束，通过规划的方法求解[19]。混合整数线性规划（Mixed integer linear programming，MILP）法是主流数学方法，其中部分决策变量为整数，通过约束条件限制求解范围。但无法求解大规模问题，Blazewicz 等[20]提出 MILP 并不能对车间调度问题的求解有实质上的提升。因此，MILP 常嵌入于 CPLEX 和 Gurobi 等商业求解器用于车间调度问题的模型验证和求解问题下界。拉格朗日松弛法[21,22]和分解法[23] 是目前面对求解车间调度问题最有效的数学方法。分解法的中心思想是“分而治之”，将一个复杂的问题分解为许多相同类型的易求解的子问题，再求出每个子问题的最优解，最后将子问题的最优解视为全局最优解。拉格朗日松弛法用非负拉格朗日因子松弛约束条，随后在目标函数中加入惩罚因子，能对车间调度问题求得近似解。该方法在单机调度和多并行机调度问题上表现优异[24]。

（ii）**分支界定法**：是最通用且有效的求解小规模车间调度问题的方法。分支界定法通过树状结构划分解空间以缩小搜索范围，最后精确定位到全局最优解。Balas[25]在 1969 首次提出基于析取图的分支界定法求解车间调度问题。McMahon 和 Florian[26]在 1975 年提出分支界定法求解著名的 FT10 问题的并取得了当时最好解。随后许多学者对分支界定法进行了大量研究和改进，Carlier 和 Pinson[27]在 1989 年通过改进的分支定界法求得 FT10 测试集的最优解是 930。但是分支界定法严重依赖于初始上界的设定，且无法求解工序数大于 250 的车间调度问题，因此在后续的发展中逐渐被近似方法取代。

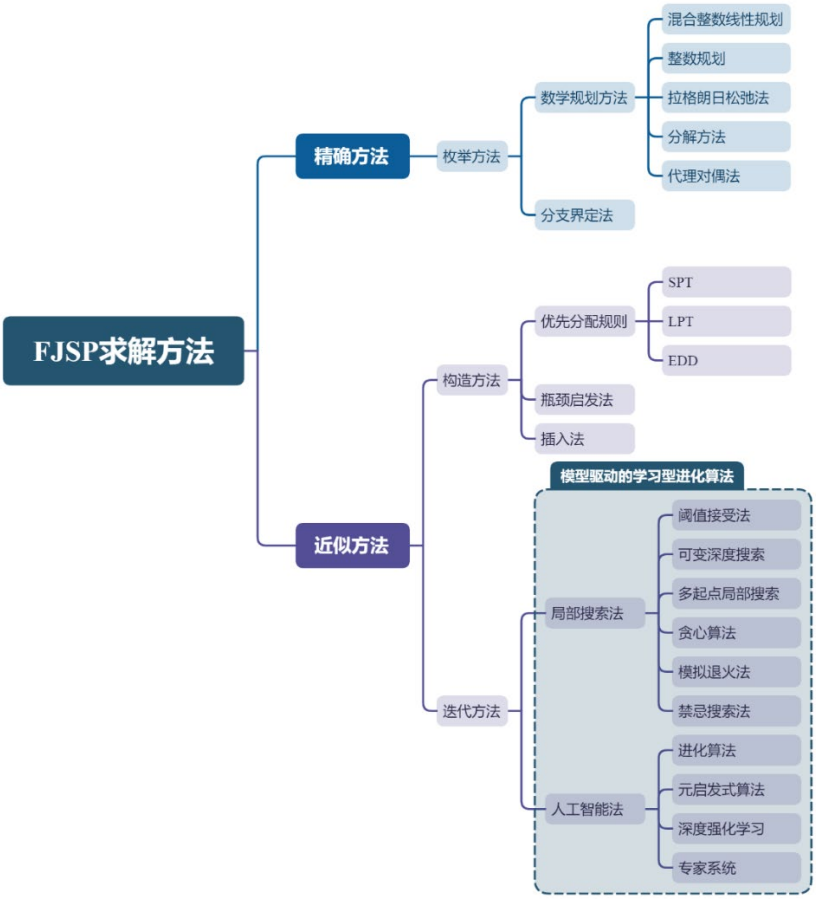

图 1.2 FJSP 的求解方法分类

近似方法：如图 1.2 所示，FJSP 的求解方法中有一大类为近似方法，其在求解大规模问题时能够快速全局搜索得到近似最优解，在实际问题中应用广泛。近似方法主要分为以下几类：

（1）**构造法**：构造法是根据专家经验设计的，能够快速构造解逼近全局最优解。

（i）**优先指派规则**（Priority dispatch rules，PDR）：最先被学者们提出的近似方法，

通过给不同的机床或工序分配优先级，并根据优先级对工序排序和机床选择实现解的构造。常用的 PDR 包括最早空闲机床规则、最短加工时间规则、最早交货期规则、先入先出规则、最长加工时间规则等[28]。Panwalkar 和 Iskander[29]对 113 中 PDR 进行了总结，虽然求解速度快，但由于贪心的特性使得 PDR 具有短视性，当求解的车间调度问题的规模增大时，基于优先指派规则的算法通过反复的迭代最终会陷入局部最优。

（ii）**瓶颈移动算法**（Shifting Bottleneck Procedure，SBP）：1988 年由 Adams[30]等提出。该方法首次求解出著名的 FT10 基准测试问题的下界，推动了车间调度问题中近似方法的研究。首先，瓶颈机器被定义为具有最大下界的机器。在 SBP 中，找到瓶颈机器并将该机器上的工件加工顺序重排，同时不改变其他机器上工件的顺序。找到瓶颈机器后，其工序重排都是单并行机排序问题。Carlier[31]提出贪心迭代算法解决排序问题，而瓶颈移动算法的核心思想是松弛瓶颈机器来优化加工顺序，其缺点在于会产生不可行解。Dauzere-Peres[32] Demirkol[33]，Huang[34]等学者对上述问题提出了改进。SBP 虽然能比 PDR 求得更好的解，但计算过于复杂，时间复杂度很高，随着问题规模增大变得非常耗时。

（2）人工智能法：主要分为元启发式算法，多智能体方法和学习型进化算法三类。

（i）**元启发式算法**：2002 年 Kacem[35]提出遗传算法（Genetic Algorithm，GA）、求解多目标 FJSP 问题，并在结果上超过了分解法和局部搜索法，随后元启发式算法便在 FJSP 的求解上大放异彩。2003 年 Jensen[36]采用 GA 求解动态 FJSP 并取得了良好的效果。2008 年 Nhu[37]等首次采用 GA 加变邻域局部搜索的框架求解 FJSP，并在标准测试上取得了最好结果。Gao[38]采用多种启发式规则混合初始化，得到高质量初始化种群，加速了 GA 的收敛。李尚函[39]混合了 6 种局部搜索算子，增强了 GA 的局部搜索能力。Zhang [40]和 Defersha[41]采用 GA 求解多时间约束的 FJSP。GA 的通用性、稳定性和易扩展性使得其广泛应用于 FJSP 中，GA 中各个算法组件如交叉变异、启发式初始化和局部搜索都可根据具体问题替换。随着人工智能和进化计算理论的发展，各元启发式算法被广泛应用于 FJSP。Gao[42] 和 Li[43]对此元启发式算法求解 FJSP 做了系统性回顾。粒子群优化算法（Particle swarm optimization，PSO）则是另一类应用最多的算法。2005 年 Xia[44]首次提出了 PSO 加模拟退火算法求解最小化最大完工时间（Cmax），总机器负载（Total workload, TWL）和最大机器负载（Max workload，MWL）的 FJSP。为 PSO 在 FJSP 的应用打下了基础。近三年来，PSO 在 FJSP 各类拓展问题中得到了广泛的应用。顾幸生[45]提出博弈 PSO 求解绿色 FJSP（Green FJSP，GFJSP）最小化 Cmax 和总能耗（Total energy consumption，TEC）。Ebrahimi[46]采用 PSO 求解 GFJSP 最小化总延期时间（Total

tardiness，TTD）和 TEC。李黎[47]和李俊萱[48]对 PSO 进行改进提出量子 PSO 算法求解 FJSP。吴晓雯[49]则结合 GA 和 PSO 解决 PSO 易陷入局部最优解的缺陷。陈魁[50]则应用 PSO 求解带 AGV 运输小车的 FJSP。虽然 PSO 具有易陷入局部最优的缺点，但在各类 FJSP 问题上仍应用广泛。人工蜂群算法（Artificial bee colony algorithm，ABC）自带侦查蜂的优点，使得算法在陷入局部最优时可以借助计数器重启。因此 ABC 是第三类在 FJSP 中应用广泛的算法。Li[51]结合 ABC 和禁忌搜索（Tabu search，TS）的算法求解带工序插入的 FJSP。郑小操[52]提出多局部搜索的 ABC 求解模糊 FJSP 问题。Li[53]提出改进 ABC 求解 GFJSP。唐浩[54]和李俊青[55]改进 ABC 求解带运输资源约束的 FJSP。近几年，另一种无参数的算法 Jaya 得到了广泛关注，因为其无需调参的稳定性，Jaya 算法在 FJSP 问题得到了广泛应用。Gao[56] 提出改进 Jaya 求解带工件插入的 FJSP。Caldeira[57]提出 Jaya 算法结合基于关键块局部搜索求解 FJSP 问题， Li[58]采用 Jaya 算法求解考虑运输时间的 FJSP。Caldeira [59]又采用 Jaya 算法求解最小化 Cmax,TWL 和 MWL 的 FJSP。结果表明 Jaya 算法的性能好于 GA 和 PSO。近年来，各类元启发式算法被提出，主要启发来源于物理原理，社会关系，人类群体，生物种群关系。但算法内容大致相同，只是概念不同。因此，下面将简单举例其他元启发式算法在 FJSP 的应用。Lei[60]和 Li[61]提出了帝国竞争算法求解带能耗峰值约束和运输时间的 GFJSP。Gnanavelbabu[62]和 Caldeira[63]提出回溯搜索算法求解带工人上班柔性和带工件插入的 GFJSP。Sun[64]提出 GA 全局搜索 PSO 的局部搜索求解模糊加工时间的 FJSP。Lin[65]提出多宇宙优化算法求解模糊 FJSP。Wu[66]提出鸽子优化算法求解带机器老化影响的 GFJSP。更多地还有萤火虫算法[67]，引力搜索算法[68]，蚁群优化算法[69]，人工免疫算法[70]，猫群算法[71]，状态转移算法[72]，布谷鸟算法[73]，迭代贪心算法[74]，模拟退火算法[75]，差分进化算法[76]，灰狼优化算法[77]，生物迁徙算法[78]，禁忌搜索算法[79] ，模因算法[80]和多种元启发式算法混合[81]求解 FJSP。此外还有基于析取图编码的 GA 求解 FJSP[82] [83]。针对多目标优化 FJSP，主流框架分为 NSGA 系列和 MOEA/D 框架。Yuan[84]使用 NSGA-II 算法替换 MA 框架中的全局搜索模块，使得算法可以处理多目标 FJSP 问题。GFJSP 由 Piroozfard[85]于 2016 年首次提出，并用 NSGA-II 算法求解。相关 NSGA-II 求解 FJSP 的工作还有[86][87] [88] [89]。针对超多目标 FJSP，NSGA-III[90][91]成为主要求解框架。此外，MOEA/D[92] 也应用广泛。孟磊磊[93]，PARK[94]和 Rakovitis[95]也对 FJSP 和 GFJSP 的混合整数线性规划模型和约束规划模型给出了定义和证明。最后 Gao[96]对近 10 年来 GFJSP 的研究进行了系统性综述。表 1.1 给出了元启发式算法求解 FJSP 的总结和概括。

表 1.1 元启发式算法求解 FJSP 的研究现状汇总

作者	求解问题	优化目标	时间	元启发式算法
Kacem[35]	FJSP	Cmax,TWL	2002	GA
Jensen[36]	Robust FJSP	Weighted Cmax	2003	
Nhu[37]	FJSP	Cmax,TAT,TMT	2008	
Gao[38]	FJSP with new job insert	Cmax,E/T	2015	
李尚函[39]	Fuzzy FJSP	Cmax	2019	
ZHANG[40]	FJSP with multiple times constrains	Cmax,TAT,TMT	2020	
DEFERSHA[41]	FJSP with multiple times constrains	Cmax	2022	
Xia[44]	FJSP	Cmax,TWL,MWL	2005	PSO
顾幸生[45]	FJSP	Cmax,TEC	2020	
EBRAHIMI[46]	GFJSP	TTD,TEC	2020	
李黎[47]	FJSP	Cmax	2022	
李俊萱[48]	Fuzzy FJSP	Cmax	2022	
吴晓雯[49]	FJSP	Cmax	2022	
陈魁[50]	FJSP with AGV	Cmax	2022	
Li[51]	FJSP with new job insert	Cmax	2017	ABC
郑小操[52]	Fuzzy FJSP	Cmax	2019	
LI[53]	GFJSP	Cmax,TEC	2020	
唐浩[54]	FJSP with transportation time	Cmax	2021	
李俊青[55]	FJSP	Cmax	2021	
GAO[56]	FJSP with job insert	Cmax,TWL,MWL	2019	Jaya
CALDEIRA[57]	FJSP	Cmax	2019	
LI[58]	FJSP with transportation and setup times	Cmax,TEC	2020	
CALDEIRA[59]	FJSP	Cmax,TWL,MWL	2021	
LEI[60]	GFJSP with peak power	Cmax,TEC	2019	ICA
LI[61]	GFJSP with transportation time	Cmax,TEC	2022	
GNANAVELBABU[62]	FJSP with worker flexibility	Cmax,std Cmax	2021	BSA
CALDEIRA[63]	GFJSP with job inserting	Cmax,TEC	2022	

表 1.1（续） 元启发式算法求解 FJSP 的研究现状汇总

作者	求解问题	优化目标	时间	元启发式算法
Sun[64]	Fuzzy FJSP	Cmax	2019	GA+PSO
Lin[65]	Fuzzy FJSP	Cmax	2019	MVO
WU[66]	GFJSP with deterioration effect	Cmax,TEC	2019	PIO
夏俊红[67]	FJSP	Cmax	2019	萤火虫算法
詹欣隆[68]	GFJSP	Cmax,TEC	2020	引力搜索算法
ZHANG[69]	Assembly FJSP	Cmax,Duedate,TWL	2020	ACO
LI[70]	Fuzzy FJSP	Cmax,TEC	2020	AIS
姜天华[71]	FJSP	E/T	2020	猫群优化算法
吴贝贝[72]	FJSP	Cmax,TWL	2020	状态转移算法
Cao[73]	FJSP with sequence-dependent setup time	Cmax,TEC	2022	布谷鸟算法
LI[74]	Crane Transportation FJSP	Cmax,TEC	2022	IG
DEFERSHA[75]	FJSP	Cmax	2022	SA
LI[76]	FJSP with with outsourcing operations and job priority	Cmax	2022	DE
ZHU[77]	FJSP with job precedence constraints	Cmax	2022	GWO
刘 璐[78]	FJSP with resource constrain	Cmax,TEC	2022	生物迁徙算法
DING[79]	GFJSP with time-of-use electricity prices	Cmax,TEC	2022	TS
GONG[80]	GFJSP	Cmax,TEC	2022	MA
DING[81]	FJSP	Cmax	2022	HLO+PSO
BOYER[82]	Generalized FJSP	Cmax	2021	LS
GARCÍA[83]	FJSP	Cmax	2022	析取图
Yuan[84]	FJSP	Cmax,TWL,MWL	2015	NSGAII
AN[91]	FJSP with job insert and machine maintaince	Cmax,TWL,Stability	2022	
JIANG[92]	GFJSP with time-of-use electricity prices	Cmax,TEC	2020	MOEA/D
孟磊磊[93]	GFJSP	Cmax,TEC	2020	MILP,CP

（ii）**多智能体直接求解**：2005 年 Wu[97]首次提出多智能体概念，将工序排序和机

器选择分由两个智能体构造，为多智能体的直接求解打下了基础。王嘉喆[98] 从系统的宏观的角度设计了多智能体系统，通过合同控制网络整合负责 FJSP 生产车间中不同模块的智能体。近年来端到端直接求解 FJSP 受到了广泛关注，Lei[99]采用多指针图神经网络描述 FJSP 状态，并用近端优化算法离线学习，最终直接求解 FJSP。张凯[100]集成 5 种深度强化学习算法对 FJSP 进行训练，在 MK 测试集上求解结果超过 GA。Song[101]提出异构图神经网络结合改进的注意力机制对 FJSP 的工序选择和机床选择进行描述和学习，在 MK 测试集取得了最好的结果。目前端到端直接求解 FJSP 的最有效方法是转换为图神经网络结合强化学习进行训练。此外，还有结合多智能体系统和强化学习的研究。胡一凡[102]采用价值分解网络协同四个智能体分别对工序选择，机床选择，待装零部件和装配车间进行控制，并用深度强化学习对 PDR 的决策进行训练。相似工作还有 Zhang[103]提出的多智能体表征析取图的不同特征，并用深度 Q 网络训练 PDR 的决策。表 1.2 对多智能体直接求解 FJSP 给出了总结和概括。虽然智能体训练得到的端到端网络具有泛化性，可以求解不同规模 FJSP，但求解问题规模仍有限，总工序数无法超过 100 以上。因此，目前求解 FJSP 最快最有效的模式仍是 MA 结合学习机制。下面将对学习型进化算法进行详细介绍。

表 1.2 多智能体直接求解 FJSP 的研究现状汇总

作者	特征	求解问题	优化目标	时间	核心技术
Wu[97]	多智能体	FJSP	E/T	2005	工件智能体和机器智能体协同构造解
王嘉喆[98]		FJSP	Cmax	2022	合同控制网络
Lei[99]	端到端	FJSP	Cmax	2022	multi-pointer GNN+PPO
张凯[100]		Fuzzy FJSP	Cmax	2022	集成 5 种 DQN
Song[101]		FJSP	Cmax	2022	同构 GNN+PPO+PDR
胡一凡[102]	多智能体	Assembly FJSP	Cmax	2022	价值分解网络
Zhang[103]	+DRL	FJSP	Cmax	2023	GNN+DQN+PDR

（iii）**学习型进化算法：**2020 年强化学习开始进入学者的事业，研究人员开始在进化算法中加入强化学习机制，使得进化算法表现出自主性以适应不同的求解环境。Su[104]采用长短期记忆网络对 FJSP 进行编码，并用近端策略优化学习最优突变算子选择，实现自学习局部搜索的目的。Chen[105]提出基于 Q 学习的 GA 求解 FJSP，文中使用 Q 学习对启发式局部搜索算子进行选择，在 MK 标准测试集上取得了最好的结果。Du[106,107]采用深度强化学习对启发式局部搜索算子进行学习和选择，极大提升了分布式估计算法的性能。Pan[108]加入反馈机制依据进化效果调整双种群

大小，增强优势种群的收敛。Li[109,110]提出 Q 学习算法依赖多目标优化指标的反馈对 MOEA/D 算法的参数进行自适应调整，增强非支配解集的分布性。Pan[111]采用 Q 学习对种群的交配个体进行自学习选择，增强算法的收敛性。Lin[112]采用 Deep Q-Network 算法对灰狼优化算法的参数进行自学习调整，加速了算法的收敛。Li[113]提出基于历史置信度的参数选择模型，自学习地选择 MOEA/D 的参数，增强了算法的收敛性。Zhang[114]采用卷积神经网络对 FJSP 中 GA 调度的鲁棒性进行预测指导 GA 进行算子选择。叶孙飞[115]采用 Q 学习对老鹰优化算法的参数进行自学习调整，加强算法的自主性。Lou[116]提出结合基于聚类的自学习局部搜索的 MA 求解带工人影响的 FJSP，学习型 MA 取得了良好结果。表格 1-3 对学习型进化算法求解 FJSP 的研究现状给出了详细的总结。

表 1.3 学习型进化算法求解 FJSP 的研究现状汇总

作者	特征	求解问题	优化目标	时间	核心技术
Chen[105]	参数选择	FJSP	Cmax	2020	GA+Qlearning+SARSA
Pan[108]		Fuzzy GFJSP	Cmax,TEC	2022	GA+Feedback
Li[109,110]		Fuzzy GFJSP	Cmax,TEC	2022	MOEA/D+Qlearning
Lin[112]		SFJSP	Cmax	2022	GWO+DQN
Li[113]		Fuzzy FJSP	Cmax,TWL	2022	MOEA/D+memory matrix
叶孙飞[115]		FJSP with resource constrain	Cmax	2022	HHO+Qlearning
Su[104]	算子选择	FJSP-PMMO	TTD	2022	GA+LSTM+PPO
Du[106,107]		GFJSP	Cmax,TEC	2022	EDA+DQN
Pan[111]		FJSP-T	Cmax	2022	MA+Qlearning
Zhang[114]	参数回归	dynamic FJSP	robustness, expectation of delay	2022	CNN
Lou[116]	解的聚类	FJSP with human factors	Cmax, max WL, TWL	2022	K-means

1.2.2 分布式柔性作业车间调度问题的研究现状

由于全球贸易的发展，工厂的订单量极具增加，企业开设更多园区并行分布式制造产品。同时，如果协调多园区的订单生产成为一大难题，分布式柔性作业车间调度问题（Distributed FJSP，DFJSP）随即收到关注。2006 年 Chan[117]首次提出 DFJSP 问题，采用 GA 进行求解，并为工厂选择问题设计了新的交叉算子，使得 GA 可以有效求解 DFJSP 问题。Chung[118]在前人研究的基础上，考虑了机器维护因素，并在编码中加速机器老化因子，更加复合实际生产。De Giovanni[119]提出了改进的 GA，为每个子问题设计了不同的变异算子，并针对完工时间设计了新的局

部搜索策略，增强了 GA 的性能。周蓉[120]对针对 DFJSP 的特点，设置多种变异算子改进 GA 性能。黄英杰[121]则改进 PSO 使其能够求解 DFJSP。Mohsen[122]则使用启发式构造法加快收敛速度。Liu[123]对编码方式进行改进，提高了历史解的利用率，提升了 GA 的性能。随着，王凌[124]在 2016 年发表关于分布式车间调度问题的综述，DFJSP 迅速受到了广泛的关注。何怡[125]提出改进的蚁群优化算法求解 DFJSP。Bilel[126]则提出禁忌搜索算法求解 DFJSP。Li[127]在禁忌搜索算法加入了基于关键路径局部搜索算子，提升了局部搜索的成功率。Lu[128]对编码方式进行改进，引入影子染色体降低算法搜索空间。Wu[129,130]提出改进差分进化算法求解 DFJSP。吴锐[131]使用 ABC 全局搜索结合模拟退火局部搜索增强算法的收敛性。Lin[132]从析取图的角度编码 DFJSP，同时用启发式方法构造解。Luo[133]提出了 MA 求解 DFJSP，全局搜索加局部搜索的模式让 MA 取得了最好结果。Guo[134]在 DFJSP 的基础上考虑能耗指标，提出了 GDFJSP。Meng[135,136]进一步对 DFJSP 和 GDFJSP 的 MILP 模型进行了补充和证明。Du[137]对 MA 中全局搜索算子进行了替换，采用分布估计算法替代 GA 使得算法在工厂选择上学习历史分布，做出更合理选择。Wang[138]将 DFJSP 落地，实现线上调度线下排产的系统，拓展了 DFJSP 的实际应用。Xu[139]将 MA 中的局部搜索替换为禁忌搜索算法，提升了局部搜索效率。但总体算法设计不脱离 MA 的计算框架。李佳磊[140]研究带机器维护的 DFJSP，提出双种群协同进化，为每个种群设计单独的交叉变异算子，取得了良好的结果。Bilei[141]在 DFJSP 的基础上考虑机器之间的运输时间。并提出化学反应算法求解。Li[142]首次提出异构工厂的 DFJSP（heterogeneous DFJSP，DHFSP，并提出改进化学反应算法求解。相似的工作还有灰狼优化算法[143]，交叉熵算法[144]，蜂群算法[145]。目前求解 DFJSP 及其扩展问题最有效的框架仍为 MA。Sang[11]求解超多目标 DFJSP 将 MA 的全局搜索替换为 NSGA-III 算法，取得了良好效果。Luo[146]提出 MA 求解考虑工人排班的 DFJSP。唐红涛[147]将 MA 的全局搜索替换为灰狼优化算法求解带模糊加工时间的 DFJSP。Li [148]提出基于 N6 邻域搜索结构的局部搜索算子，并针对能耗目标采用全主动调度解码节省能耗，提出的 MA 能够有效求解 GFJSP。Li[149]对 MA 进行了改进，将全局搜索和局部搜索分别由两个种群执行，主种群执行全局搜索，精英存档执行局部搜索，并将进化过程划分为两个阶段，在全局搜索充分探索目标空间后，局部搜索在精英解附件快速搜索更优解。即保证了全局搜索的分布性不受过分局部搜索的干扰，也保证了少量计算资源便可以执行高质量的局部搜索。所提的双阶段知识驱动的 MA 在带二型模糊加工时间的 GDFJSP 上取得了最好结果。因此，目前求解 GDFJSP 的最有效框架仍为 MA，全局搜索加精英局部搜索的模式可以有效探索目标空间以保持分布性，同时，仅花费少量计算资源进行高质量局部搜索。

表格 1-4 详细总结了 DFJSP 的研究现状。

表 1.4 DFJSP 研究现状汇总

作者	问题模型	优化目标	时间	核心技术
Chan[117]	DFJSP	Cmax	2006	GA 交叉变异都执行
Chung[118]	DFJSP with maintenance	Cmax	2009	GA 交叉变异二选一
De Giovanni[119]	DFJSP	Cmax	2010	GA+LS
周蓉[120]	DFJSP	Cmax	2012	多变异算子 GA
黄英杰[121]	DFJSP	Cmax	2012	PSO
Mohsen[122]	DFJSP	Weighted Cmax	2013	启发式构造法
Liu[123]	DFJSP	Cmax	2015	启发式构造编码 GA
何怡[125]	DFJSP	Cmax	2016	ACO
Bilel[126]	DFJSP	Cmax	2017	TS+多智能体
Li[127]	DFJSP	Cmax,TWL, E/T	2018	TS+LS
Lu[128]	DFJSP	Cmax	2017	基于影子编码 GA
Wu[129,130]	DFJSP	Cmax	2018	DE+SA
吴锐[131]	DFJSP	Cmax	2019	ABC+SA
Lin[132]	DFJSP	Cmax	2020	启发式构造
Luo[133]	GDFJSP with transfer	Cmax, max WL, TEC	2020	NSGAII+LS
Guo[134]	GDFJSP	Cmax, TEC	2020	GA+TS
Meng[135,136]	DFJSP,GFJSP	Cmax,TEC	2020	MILP
Du[137]	GDFJSP with crane transportations	Cmax,TEC	2021	EDA+VNS
Wang[138]	GDFJSP with real-time scheduling	Cmax,workload balance index,TEC	2021	GA+PDR
Xu[139]	GDFJSP	Cmax,TEC, Cost,Quality	2021	GA+TS
李佳磊[140]	DFJSP with preventive maintenance	Cmax	2021	NSGAII+LS
Bilei[141]	DFJSP with transfer	Cmax	2022	CRO+LS
Li[142]	DHFJSP	Cmax	2022	CRO+LS+SA
Li[143]	DFJSP	Cmax	2022	GWO+LS
罗文冲[144]	DAFJSP	Cmax	2022	交叉熵算法+VNS
李佳磊[145]	DFJSP	Cmax	2022	ABC+LS
Sang[11]	GDFJSP	Cmax,TEC, TDD,TWL,Quality	2022	NSGAIII+LS
Luo[146]	DFJSP considering worker arrangement	Cmax, max WL, WL of workers	2022	GA+VNS
唐红涛[147]	Fuzzy DFJSP	Cmax	2022	GWO+LS
Li[148]	Type-2 Fuzzy GDFJSP	Cmax,TEC	2022	双阶段 MA+节能技术

1.2.3 模因算法的研究现状

算法1 - 1: 模因算法框架 (MA)

输入: 最大函数评价次数($MaxNFEs$)，种群大小(ps)，交叉率(p_c)，邻域个数(T) 和突变率(p_m)

输出: 非支配解集$\mathbb{S}$

```
P ← 初始化种群(ps).
F ← 计算适应值(P).
t = 1.
while NFEs ⩽ MaxNFEs do
    P_{t+1} ← 交叉变异(P_t, p_c, p_m).
    while the stop critera are met do
        应用局部搜索算子N_i, i = {1,...,T}更新种群P_{t+1}.
    end
    t = t + 1.
end
PF ← Get Pareto Front (P_t).
S ← 找到非支配解集(P)
```

模因学[150]和遗传学一样是激发人类灵感的一门重要学科，模因（meme）的概念表示有效解决问题而在计算表示中编码的循环现实模式或领域特定知识。该概念最早由 Dawkins[151]提出。而模因算法 MA 最早由 Moscato[152]提出，表示局部重复搜索，不断强化结果的框架，算法 1-1 详细展示了 MA 的框架。MA 主要由全局搜索和不断强化的局部搜索组成。MA 最早被 IEEE Fellow，Ishibuchi[153]应用于求解多目标置换流水车间调度问题，为 MA 在车间调度领域的研究打下了基础。随后在 Liu 和 Wang[154]的推广下逐渐进入学者的视野。此外，IEEE Fellow，One-yew-soon[12]对 MA 进行了详细的综述，可见 MA 在各个领域都有广泛的应用。Mohammad[155]提出基于关键路径的局部搜索强化 MA 在作业车间中的搜索效率。Pan[156]提出 MA 求解带阻塞的流水车间调度问题，取得了最好的结果。Mencía[157]采用 MA 求解带工人操作约束的作业车间调度问题，在性能上超过传统 GA。Yuan[84]采用 MA 求解多目标 FJSP，MA 在性能上超过其他算法的原因本质上是求解车间调度问题的关键在于设计高效的局部搜索算法，而全局搜索加局部搜索的模式使得算法可以针对性的增强求解对应问题的性能。更多地，MA 在其他调度问题中得到了广泛的应用，Wang[158]提出求解分布式装配置换流水车间调度问题。Deng[159]在 MA 中引入竞争机制求解分布式流水车间调度问题。Mohamed[160]提出基于多岛屿模型的 MA，Gong[161]设计了基于关键路径搜索的 MA 求解多目标作业车间调度问题。Shao[162]提出 MA 求解无等待的流水作业车间调度问题。Batuhan[163]提出 MA 求解多处理器混合流水车间调度问题。Ding[164]采用 MA 求解带工件老化效益的并

行机调度问题。Lei[165]提出 MA 求解带动态运输时间的混合流水车间调度问题。Mehdi[166]采用 MA 求解带机器老化影响的绿色作业车间调度问题。Mohamed[167]在 MA 中融入模拟退火算法求解流水车间调度。Lu[168]则求解绿色分布式异构流水车间调度。Mariappan[169]应用于不确定的绿色流水车间。Wang[170]提出双种群 MA 求解分布式混合流水车间调度，Wang[171]还提出深度强化学习网络学习 PDR 的选择辅助 MA 局部搜索。更多模型还包括组调度问题[172]，绿色作业车间调度[173]，考虑机器维修的分布式流水车间调度问题[174]，不确定绿色作业车间[175]，绿色分布式混合流水车间[176]，绿色分布式异构混合流水车间调度问题[177]，绿色分布式装配流水车间[178]，分布式差异化流水车间调度问题[179]，混合差异化流水车间调度[180]，带有限缓冲区的混合差异化流水车间调度[181]。如表 1.5 所示虽然模型变复杂，但 MA 框架可以有效的解决各类车间调度问题。

表 1.5 MA 在车间调度领域的应用

作者	求解问题	优化目标	时间	核心技术
Ishibuchi[153]	PFSP	Cmax 和 duedate	2003	GA+LS
Liu 和 Wang[154]	PFSP	Cmax	2007	PSO+SA
Mohammad[155]	JSP	Cmax	2012	GA+LS
Pan[156]	blockPFSP	Cmax	2013	GA+LS
Mencía[157]	JSPO	Cmax	2015	GA+TS
Yuan[40]	FJSP	Cmax 和 TWL	2015	NSGAII+LS
Wang[158]	DAPFSP	Cmax	2016	EDA+LS
Deng[159]	DPFSP	Cmax 和 TTD	2017	竞争 GA+LS
Mohamed[160]	JSP	Cmax, TWT, TWE	2017	多岛屿 GA+LS
Gong[161]	JSP	Cmax 和 duedate	2019	GA+LS
Shao[162]	No-Idle PFSP	Cmax	2017	GA+VNS+SA
Batuhan[163]	HFSP	Cmax	2020	GA+LS
Ding[164]	PMSP with Job Deteriorating	Cmax	2020	分治法+LS
Lei[165]	HFSP with dynamic transport waiting times	Cmax	2020	IG+NEH+ GA+LS
Mehdi[166]	energy-efficient JSP with deteriorating machines	TWT 和 TEC	2020	协同 GA+LS
Mohamed[167]	PFSP	Cmax	2020	GA+SA
Lu[168]	DPFSP	Cmax,TEC, NSI	2021	NSGAII+LS
Mariappan[169]	Uncertain EEPFSP	TEC	2021	GA+VNS

表 1.5（续） MA 在车间调度领域的应用

作者	求解问题	优化目标	时间	核心技术

Wang[170]	DHFSP	Cmax,TEC	2021	双种群 EDA +LS
Wang[171]	EADHHFSP	Cmax,TEC	2022	协同搜索 +DRL-based LS
Zhao[172]	Group Scheduling	number of late jobs and total setup time	2021	NSGAII+LS
Lu[173]	GJSP with Variable Machining Speeds	Cmax,TEC	2022	MOIG+LS
Mao[174]	DPFSP with preventive maintenance	TFT	2022	GA+hash map LS
Afsar[175]	Fuzzy GJSP	Cmax,TEC	2022	NSGAII+TS
Shao[176]	EEDHFSP	TWT,TEC	2022	NSGA+VNS
Shao[177]	EADHHFSP with worker change	TTD, TEC, Total production cost	2022	Neural Network+VNS
Wang[178]	EADPFSP with flexible assembly scheduling	TTD,TEC	2022	协同搜索 +LS with feedback
Zhang[179]	Distributed Hybrid Differentiation FSP	Cmax	2022	协同进化+VNS
Zhang[180]	Hybrid Differentiation FSP	Total completion time	2022	GA+VNS
Zhang[181]	Hybrid Differentiation FSP with limited buffer	Total completion time	2022	GA+VNS

1.3 研究现状分析

综上所述，结合绿色制造战略，GFJSP 和 GDFJSP 的研究是智能制造的一个热点问题。随着深度强化学习型的崛起，学习型 MA 框架成的求解能力最好。因此，研究学习型 MA 求解 GFJSP 和 GDFJSP 的调度方法是十分必要的，目前对于 GFJSP 和 GDFJSP 的研究在算法，模型，学习型策略，还存在以下几点不足：

算法增强策略设计方面：

（1）目前 GFJSP 的研究仅只优化能耗目标，缺乏有效的节能策略降低生产车间中的能耗。为了解决上述问题。本研究通过分析能耗目标函数特性提炼问题知识。首先，根据知识设计启发式规则，通过启发式规则构造能耗低的解。其次，通过算子设计，在局部搜索过程中对解进行扰动降低能耗。

（2）在同构 GDFJSP 的研究中，缺乏对邻域结构的设计。面对工序选择，机床选择和工厂选择三个耦合的子问题。目前同构 DFJSP 的研究中缺乏对问题知识的提取，没有根据问题特性设计有效的邻域结构辅助算法快速选择合适的工厂，设

计的算子具有盲目性和随机性，无法产生高效的扰动减少最大完工时间，造成计算资源的浪费。因此，为了解决上述问题。本研究从关键路径以及关键块的角度，针对关键工厂中关键路径上的关键工序进行扰动，同时，设计工厂间的扰动算子，平衡工厂负载减少总最大完工时间。

（3）目前求解 GFJSP 最有效的算法框架是 MA。但面对更复杂的目标空间大一个量级的 GDFJSP，传统的 MA 框架中，直接作用局部搜索在全体种群的方式会使得种群快速收敛到小范围内，进而使得种群分布性极具下降，不利于对目标空间潜在非支配解的探索。因此，为了使得种群能够在目标空间充分探索而不受局部搜索的影响，本研究对 MA 进行改进，将局部搜索分离到精英存档中执行，使 80%的计算资源分配给全局搜索，20%的计算资源保证在精英解附近执行高质量局部搜索。

问题模型方面：

（4）现有的 GFJSP 模型过于理想化，工厂的设置相同。而实际生产中，考虑到成本，人工等因素，工厂之间的产能是有差异的。相同的零件在不同的工厂中加工，加工时间是不同的。目前仅有一篇文献研究的 GDHFJSP。其次，结合实际生产调研，发现 GDHFJSP 普遍存在。因此，GDHFJSP 模型来源于实际生产场景，研究 GDHFJSP 可以落地到实际生产车间。使学术问题和实际生产相互促进。

学习策略方面：

（5）GFJSP 属于多目标优化问题，主流多目标优化算法 MOEA/D 对参数邻域更新大小 T 十分敏感。在不同规模的测试问题上，在种群进化的不同阶段，最优 T 的选择是不同的。这使得固定参数的 MOEA/D 对不同规模的问题反复调整最优参数，算法也不具有通用性。因此，为了解决上述问题，研究参数自学习机制使得 MOEA/D 框架具备自主学习和选择最优参数的能力，让算法具有更强自主性和生命力是十分重要的。

（6）目前有许多局部搜索算子被提出，之前的研究采用随机，轮询，或者基于置信度的模式选择多算子。前两种模式具有随机性和短视性，后一种模式依赖于算子历史表现决定不同算子的选择概率，基于置信度的方法受历史累计影响严重，而忽视了最近几代进化中高效但选择概率小的算子，这使得算法在最优选择选择上有误导性。因此，为了解决上述问题，本研究改进传统的基于置信度的算子选择模型，重点关注每轮迭代中权重小但高效的算子，并在发现后修正该算子的选择概率使得高效的算子分配更大的选择概率。

（7）目前的算子选择模型主要分为两类：强化学习算法如 Q 学习和 SARSA，基于置信度的模型。前者的选择效率高度依赖于状态集的设定，后者粒度较粗无法

准确为不同类型的解定制最好的选择算子。因此，为了解决上述问题，本研究采用深度强化学习算法进行算子选择，用神经网络代替 Q 表的设定解决状态集设计困难的问题。其次，通过神经网络的训练和推理，建立解的数据分布与算子选择概率的函数，为不同解定制最好的局部搜索算子选择，加速算法的收敛，增强算法自主性。

综上所述，本文拟研究以下内容：（1）GFJSP 和 GDFJSP 中的节能策略；（2）设计基于 GDFJSP 知识的局部搜索算子，提升局部搜索效率；（3）面对复杂的分布式调度问题，改进 MA 框架平衡全局和局部搜索的计算资源；（4）结合实际生产场景，研究 GDHFJSP，结合新问题的特征构建 MILP 模型；（5）为 MOEA/D 设计参数自学习机制；（6）改进基于置信度的算子选择模型；（7）采用深度强化学习学习和定制化算子选择。

1.4 本文主要研究内容

针对 DHFSP 的相关调度问题，本文将从问题模型和学习型算法设计两个方面展开深入研究，主要研究工作如下：

（1）针对 GFJSP，设计节能策略降低能耗，采用 MOEA/D 作为 MA 框架中全局搜索算子，并设计参数自学习机制，增强 MOEA/D 在进化过程中对种群分布性的调节能力。

（2）针对同构 GDFJSP，分析问题知识，根据知识设计 MA 框架中的局部搜索算子解决三个耦合的子问题。设计局部搜索策略降低能耗。改进基于置信度的算子选择模型，增强其发现高效但小权重算子的调节能力，加速算法的收敛。

（3）结合实际生产场景，研究 GDHFJSP，结合新问题的特征构建 MILP 模型。面对更复杂的分布式异构调度问题，采用改进的 MA 框架分离全局和局部搜索。其次，采用深度强化学习算法进行算子选择，为每个解定制化最优算子选择，加速算法收敛。

本文的主要研究工作如图 1.3 所示，各章的主要内容如下：

第一章，介绍了课题来源、课题背景、研究的目的和意义；针对 FJSP、DFJSP 以及 MA 算法的研究现状进行了详细综述和分析，总结前人研究中在模型、算法和学习策略上存在的不足，基于存在的问题提出了本文的研究内容。

第二章，提出基于学习的模因算法求解 GFJSP，以最小化 Cmax 和 TEC 为优化目标。对 MA 算法进行改进。首先，以 MOEA/D 为全局搜索算子替换 GA 求解。其次，设计增强策略加速算法收敛例如混合启发式规则的初始化策略，基于问题知

识的局部搜索算子。再次，设计通过分析目标函数设计节能策略。随后，采用 Q 学习算法设计基于指标的反馈机制，实现参数自学习和自调整，使算法依据环境调整参数增强分布性。最后，公开测试集的测试结果验证了提出改进点的有效性，并说明了本章节提出的算法在求解性能上优于相关算法。

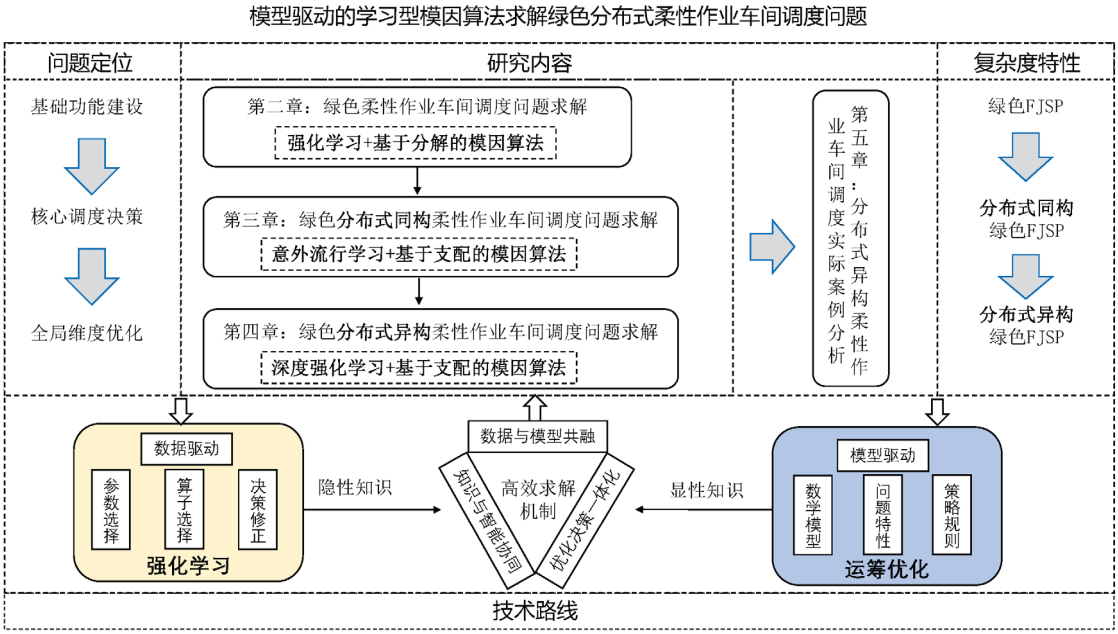

图 1.3 论文总体结构图

第三章，提出基于意外流行算法的模因算法求解 GDFJSP，以最小化 Cmax 和 TEC 为优化目标。首先，以 NSGA-II 为全局搜索算子替换 GA 求解多目标优化问题。其次，设计基于问题知识和关键块的局部搜索算子。再次，设计通过全调度解码设计节能策略。随后，提出基于意外流行算法的算子选择模型以发现小权重但高效的局部搜索算子，修正基于置信度的模型的历史累计误差。最后，两类公开测试集的测试结果验证了提出改进点的有效性，并说明了本章节提出的算法在求解性能上优于相关算法。

第四章，提出基于深度强化学习的寄生进化算法求解 GDHFJSP，以最小化 Cmax 和 TEC 为优化目标。首先，改进了 MA 提出寄生进化框架，分离全局和局部搜索到两个种群，保证主种群的分布性。其次，基于 GDHFJSP 问题的知识设计新的局部搜索算子。再次，设计通过全调度解码设计节能策略。随后，提出基于深度强化学习的算子选择模型以自学习和为每个解选择好的算子。最后，自制的测试集的测试结果验证了提出改进点的有效性，并说明了本章节提出的算法在求解性能上优于相关算法。

第五章，将本文提出的理论和方法应用于国内某大型装备生产商下料车间的

实际生产情况。该案例可归结为 DHFJSP，并利用本文所提算法和策略进行了求解，结果验证了所提算法和策略的实用性。

第六章，总结全文，并对未来的研究进行展望。

第二章　绿色柔性作业车间调度问题求解

2.1 引言

随着我国绿色制造和智能制造战略的发布，绿色经济和绿色调度受到广泛关注。针对柔性作业车间调度问题，之前的研究仅优化 TEC 但没有进一步的对 GFJSP 分析设计节能策略。其次，在之前的研究中，MOEA/D 作为主流的多目标优化框架对邻域更新范围 T 是分敏感。求解不同规模问题和不同进化阶段最优的 T 值选取不同。因此，为了解决上述问题，本章提出了基于学习的模因算法包含以下改进点：（1）设计了混合启发式初始化策略以获得高质量初始化种群；（2）分析问题知识，设计模型驱动的局部搜索策略；（3）提出基于 Q 学习的依赖指标反馈的参数自学习策略，使得算法能根据环境调整最优参数选择；（4）通过分析能耗目标，设计两条启发式节能规则，通过贪心执行两条规则降低解的能耗；将提出的算法在文献公开的包含 30 个测试问题的测试集上测试，通过对实验结果的分析和讨论验证了本章提出算法的有效性。通过对比其他相关算法，说明了本章节提出的算法在 GFJSP 的求解上有更好的性能。

2.2 基于分解的多目标优化框架

2.2.1 多目标优化理论

多目标优化问题（Multi-objective optimization problems，MOPs）定义为：同时优化多个相互冲突目标，最终输出一组目标收敛良好和尽可能分散均匀的解集，为决策者提供多样的建议，使得决策者可以根据需求做出不同的选择。因此，如何平衡收敛性和分布性并保证两者都保持良好是 MOPs 中关键的科学问题。假设一个多目标优化问题有 k 维决策变量和 n 个目标函数，则该 MOP 的数学模型描述如下：

$$\begin{aligned} &\min f(x) = \{f_1(x), f_2(x), ..., f_n(x)\}, \\ &x = (x_1, x_2, ..., x_n) \in X, f(x) \in Y, \end{aligned} \tag{5-1}$$

其中，x 表示决策向量，$f(x)$表示目标向量，X 表示决策空间，Y 表示目标空间。

MOEA/D[182]是求解 MOPs 的主流框架之一，其中心思想是将 MOPs 分解成多个相同的子问题，同时通过聚合函数将多目标问题在子问题中转换为单目标问题求解。接下来对聚合函数和 MOEA/D 算法进行详细介绍。

2.2.2 聚合函数

(1)权重聚合函数：假设求解多目标凹函数，设定有 m 个参考向量 λ={λ1，λ2，…，λm}将目标空间划分为 m 个子区域。其中所有参考向量都大于等于 0 且和为 1。随后将公式(2-1)的多个优化目标转换为单目标，权重聚合函数的公式如下：

$$\min g^{ws}(x \mid \lambda)=\sum_{i=1}^{m}\lambda_i f_i(x), x \in X, \tag{5-2}$$

其中权重聚合函数值越小表示解越靠近 Pareto 前沿面，则收敛程度越好。

(2)切比雪夫(Tchebecheff)聚合函数：该方法则考虑每次更新时按照加权目标下降最快的方向更新解。切比雪夫聚合函数的公式如下：

$$\min g^{te}(x \mid \lambda, z^*)=\max(\lambda_i \mid f_i(x)-z_i^*), \forall i \in \{1,...,m\}, x \in X, \tag{5-3}$$

图 2.1 展示了切比雪夫聚合函数的原理。切比函数依照所有解中目标函数的最小值组成目标空间中的参考点 z^*，通过对比当前解的每一维度的目标值与参考点的差距，选择下降方向最大的维度进化。切比雪夫函数越小，表示该解距离参考点最近则收敛越好。

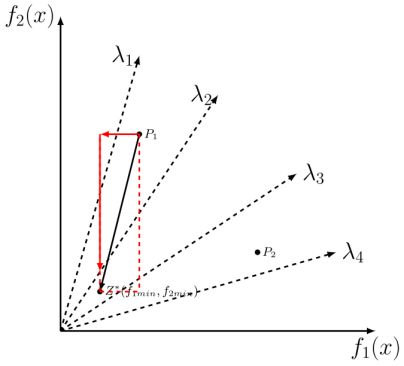

图 2.1 切比雪夫聚合函数举例

(3)基于惩罚的边界交点方法(Penalty-based boundary intersection approach, PBI)：该方法是 Zhang[182]与 2007 年提出，PBI 通过两个距离衡量当前解与 Pareto Front 的关系，分别是：和参考线的垂直距离，和参考线的垂足与全局参考点 z^*的欧式距离。PBI 聚合函数的公式如下：

$$\begin{aligned} &\min g^{pbi}(x \mid \lambda, z^*) = d_1 + \theta \times d_2, x \in X, \\ &d_1 = \frac{\| (z^* - f(x))\lambda \|}{\| \lambda \|}, d_2 = \| f(x) - (z^* - d_1\lambda) \|, \end{aligned} \tag{5-4}$$

其中 d_1 表示当前解到参考线 λ 的欧式距离，d_2 表示垂足到参考点的直线距离，θ 表示惩罚因子，用于平衡种群的分布性。以优化凹函数为例 PBI 聚合函数值越小，说明解沿着参考向量下降的距离越大，距离参考点越近则收敛越好。图 2.2 展示了 MOEA/D 原文中关于 PBI 聚合函数优化凸函数的例子。凸函数优化中目标函数值越大则收敛越好，则 PBI 聚合函数越小，说明距离上端前沿面越近，则收敛效果越好。

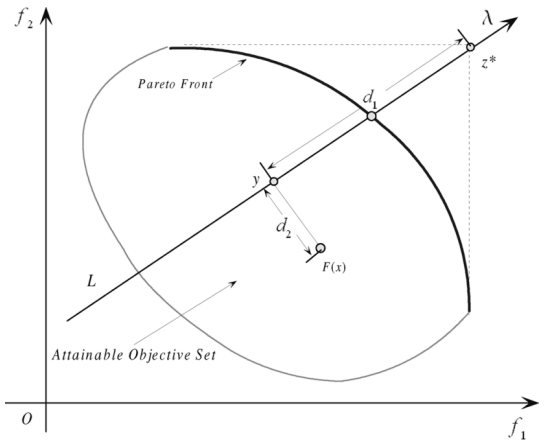

图 2.2 基于惩罚的边界交点方法举例

算法2 - 1: 基于分解的多目标优化算法(MOEA/D)

输入: 初始化种群$\mathcal{P}$，种群大小ps，邻域更新范围T
输出: 非支配解集$\mathbb{S}$

```
1  F ← 计算种群适应值(P);
2  [λ, N] ← 初始化参考向量(F, ps + 1, T);
3  z* ← min(F) //更新参考点;
4  while 停止条件未满足 do
5      foreach i=1 to ps do
6          X_new ← 遗传算法(P_i, T) //生成子代;
7          F_new ← 计算适应值(X_new);
8          z* ← min(z*, F_new) //更新参考点;
9          foreach k=1 to T do
10             h ← N_{i,k} //找到邻域解的索引;
11             更新邻域解(P_h, F_h, X_new, F_new, λ_i, z*) ;
12         end
13     end
14 end
15 S ← 找到非支配解集(P)
```

2.2.3 MOEA/D 算法

MOEA/D 算法的核心思想是将 MOPs 的目标空间依赖参考向量分解成 m 个相同的子空间。通过聚合函数控制所有目标空间中的解在进化的过程中逐渐聚集到参考向量附近以保证良好的分布性。同时，聚合函数可以保证所有解沿着参考线最快的方向下降保证了收敛性。

算法 2-1 详细说明了 MOEA/D 的算法流程，首先，计算初始化种群的适应值。其次，根据适应值给每个个体计算余弦相似度最高的 T 个邻域解构成邻域集合 N，同时输出个数为 *ps* 的参考线向量集合 λ。再次，更新参考点。随后，对每个解在邻域范围内随机选择一个解与其交叉变异以产生新的子代 X_{new}，计算子代的适应值同时更新参考点。然后，用 X_{new} 更新邻域范围内 T 的所有解，根据聚合函数公式计算聚合函数值，选择更小聚合函数的个体进入下一代进化。重复 4-13 行直到满足停止条件。最后输出种群的非支配解集。

2.3 GFJSP 描述与 MILP 模型

2.3.1 GFJSP 描述

在绿色柔性作业车间调度问题中有工件集合 $I=\{1,2,...,i,...,n\}$，每个工件有工序集合 $J_i=\{O_{i,1},O_{i,2},\ldots,O_{i,j},\ldots,O_{i,ni}\}$，在工厂中有机器集合 $M=\{1,2,...,k,...,m\}$。每个工序可以被 M 的子集 $K_{i,j}$ 加工。每个机器可以加工多种工件的工序。GFJSP 主要解决两个耦合的子问题：确定所有工序的加工顺序；为每个工序分配加工机床；使得总完工时间和总能耗最小。关于 GFJSP 的问题假设如下：（1）所有工序和机器在零时刻都是可用的；（2）所有机器在同一时刻只能加工一个工序且无法被抢占；（3）所有加工时间和能耗相关的数据都是确定的；（4）一个工件同一时间只能被一个机器加工；（5）运输时间，启动时间以及相关能耗不考虑。

2.3.2 GFJSP 的 MILP 模型

在介绍 GFJSP 的模型之前，首先要介绍 MILP 模型中的变量如下：

索引变量：

i,i': 工件索引号 $i\in\{1,2,...,n\}$；

j,j': 工序索引号 $j\in\{1,2,\ldots,n_i\}$；

k,k': 机器索引号 $k\in\{1,2,\ldots,m\}$；

t,t': 机器上加工位置索引号 $t\in\{1,2,\ldots,p_k\}$；

参数：

n: 总工件个数；

m: 总机器个数；

n_i: 工件 i 的工序个数；

p_k: 每个机器上加工位置数；

I: 工件集合 $I=\{1,2,...,n\}$；

J_i: 工件 i 的工序集合 $J_i=\{O_{i,1},O_{i,2},\ldots,O_{i,j},\ldots,O_{i,ni}\}$；

M: 机器集合 $M=\{1,2,\ldots,m\}$；

P_k: 机器 k 的加工位置集合 $P_k=\{1,2,\ldots,p_k\}$；

P_k': 机器 k 的前 p_k -1 个加工位置集合 $P_k=\{1,2,\ldots,p_k-1\}$；

$T_{i,j,k}$: 工序 $O_{i,j}$ 在机器 k 的加工时间；

W_O: 机器加工功率；

W_I: 机器空闲功率；

L: 充分大的整数保证不等式的一致性；

$x_{i,j,k}$: 0-1 变量，表明工序 $O_{i,j}$ 能否在机器 k 上加工，如果能值为 1，否则为 0；

决策变量：

$S_{i,j}$: 工序 $O_{i,j}$ 开始加工时间；

$C_{i,j}$: 工序 $O_{i,j}$ 完成加工时间；

$B_{k,t}$: 机器 k 第 t 个位置开始加工时间；

E_I: 机器加工总能耗；

$X_{i,j,k,t}$: 0-1 变量，表明工序 $O_{i,j}$ 在机器 k 的第 t 个位置上加工，如果是值为 1 否则为 0；

GFJSP 的两个目标函数描述如下：

(1)最大完工时间通常被认为是车间调度问题的经济指标，车间内最大完工时间越小带来的企业受益越大。最大完工时间的优化目标的定义如下：

$$F_1=C_{\max}=\max\{C_{i,n_i}\},1\le i\le n, \tag{5-5}$$

(2)总的能量消耗通常被认为是关键的环境指标，总能耗 TEC 反应企业在系统层面二氧化碳排放量。能耗目标主要由两个决策表达式组成分别是：机器的总加工能耗 E_W 和机器的总空闲能耗 E_I。总能耗的优化目标的定义如下：

$$F_2=TEC=E_W+E_I, \tag{5-6}$$

$$E_W=\sum_{i\in I}\sum_{j\in J_i}\sum_{k\in M}\sum_{t\in P_k}W_O\times T_{i,j,k}\times X_{i,j,k,t}, \tag{5-7}$$

$$E_I = \sum_{k\in M}\sum_{t\in P_k'} W_I \times (B_{k,t+1} - B_{k,t+1} - \sum_{i\in I}\sum_{j\in J_i} X_{i,j,k,t} \times T_{i,j,k}), \tag{5-8}$$

GFJSP 的约束条件如下：

$$\sum_{k\in M}\sum_{t\in P_k} X_{i,j,k,t} = 1, \forall i \in I, j \in J_i, \tag{5-9}$$

$$S_{i,n_i} + \sum_{k\in M}\sum_{t\in P_k} T_{i,n_i,k} \times X_{i,n_i,k,t} \le C_{\max}, \forall i \in I, \tag{5-10}$$

$$S_{i,j} + \sum_{k\in M}\sum_{t\in P_k} T_{i,j,k} \times X_{i,n_i,k,t} \le S_{i,j+1}, \forall i \in I, j \in J_i - 1, \tag{5-11}$$

$$\sum_{i\in I}\sum_{j\in J_i} X_{i,j,k,t} \ge \sum_{i\in I}\sum_{j\in J_i} X_{i,j,k,t+1}, \forall k \in M, t \in P_k', \tag{5-12}$$

$$B_{k,t+1} - B_{k,t} \ge \sum_{i\in I}\sum_{j\in J_i} X_{i,j,k,t} \times T_{i,j,k}, \forall k \in M, t \in P_k', \tag{5-13}$$

$$E_{Ik,t} \ge (B_{k,t+1} - B_{k,t} - \sum_{i\in I}\sum_{j\in J_i} X_{i,j,k,t} \times T_{i,j,k}) \times W_I, \forall k \in M, t \in P_k', \tag{5-14}$$

$$B_{k,t} \ge S_{i,j} - L \times (1 - X_{i,j,k,t+1}), \forall i \in I, j \in J_i, k \in M, t \in P_k', \tag{5-15}$$

$$B_{k,t} \le S_{i,j} - L \times (1 - X_{i,j,k,t+1}), \forall i \in I, j \in J_i, k \in M, t \in P_k', \tag{5-16}$$

$$0 \le B_{k,t}, S_{i,j} \le L, \forall i \in I, j \in J_i, k \in M, t \in P_k, \tag{5-17}$$

约束(2-9)确保同一时刻每个机器的每个加工位置只加工一个工序，约束(2-10)表示优化目标最大完工时间与工序完工时间的关系，约束(2-11)表示每个工序的开始加工前必须等上一个工序完工，约束(2-12)确保每个机器上必须等前一个位置的工序完工才能加工下个位置工序，约束(2-13)表示每个机器上开始加工时间和完工时间的关系，约束(2-14)表示机器空闲能耗的计算方式，约束(2-15)和约束(2-16)是为了便于 MILP 模型采用分支界定算法搜寻解而对等式约束进行松弛，表示机器开始加工时间和工序的开始加工时间是相同的，约束(2-17)为决策变量的取值范围。

2.4 基于 LRVMA 的 GFJSP 求解

本章提出了基于学习和参考向量的模因算法（Learning and reference vectors based memetic algorithm，LRVMA），以 MOEA/D 算法为全局搜索算子，配合混合启发式初始化规则和局部搜索算子等增强策略加速算法收敛。更多地，为了解决 MOEA/D 对参数敏感的问题，提出基于强化学习的参数选择模型自学习地调整参数以增强求得非支配解的分布性。最后，针对能耗目标，提出贪心节能策略降低能耗。

2.4.1 编解码机制

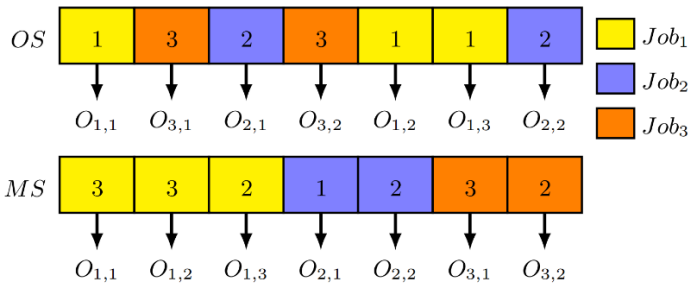

图 2.3 GFJSP 编码模式

解的编码方式是算法设计的核心之一，优良的编码方案有效地涵盖了求解空间，保证算法的搜索效率。为了覆盖整个解空间范围，本章采用双层编码模式描述 GFJSP 的解分别是：工序顺序（Operation sequencing，OS）和机器选择（Machine selection，MS）。两个向量的长度和总工序数相等。图 2.3 给出了编码模式的例子，图中包含 3 个工件和 3 台机器。每个工件有两到三个工序。图中的数字表示工件号而数字出现的次数表示该工件第几个工序。图中的加工顺序为 $O_{1,1}\rightarrow O_{3,1}\rightarrow O_{2,1}\rightarrow O_{3,2}\rightarrow O_{1,2}\rightarrow O_{1,3}\rightarrow O_{2,2}$。而 MS 向量中每个位置机器选择对应的工序是一直不变的，从 $O_{1,1}$ 到 $O_{3,2}$ 进而在后续的交叉变异中保证不会产生不可行解。

解码模式：首先，从左到右遍历向量 OS 获取加工顺序，随后从左到右遍历 MS 向量获取每个工序的机器选择，把每个工序安排在对应的机器上加工，计算每个工序的开始时间和完工时间，计算每个加工位置的等待时间，保证开工时间服从 MILP 模型中的约束条件。最后，当前解的总能耗和最大完工时间就可获得。

2.4.2 种群初始化

高质量的种群初始化可以极大程度减少搜索范围，节省大量计算资源。达到只花费少数评价次数即可获得收敛性和分布性良好的种群的目的。为了获得高质量的初始化种群，本节提出一种混合启发式初始化策略（Hybrid heuristic initialization strategy，HHIS），包含四种启发式初始化规则其描述如下：

(1) 最小加工时间规则：随机生产 OS 向量，针对每个工序选择加工时间最小的机器。选择最小的加工时间可以靠近 Cmax 的下界。

(2) 最小机器负载规则：随机生产 OS 向量，针对每个工序选择当前调度序列中机器负载最小的机器。平衡生产车间中的机器负载，使得每个机器的任务量相近有助于减少 Cmax。

(3) 最小完工时间规则：随机生产 OS 向量，针对每个工序选择当前调度序列中完工时间最小的机器。最大完工时间与完工时间直接相关，贪心地选择最小完工时间机器有助于减少 Cmax。

(4) 双贪心规则：将工序加工顺序按照可选机器数量从小到大排序，如果数量相同则随机排序。针对每个工序，选择当前机器负载最小的机器加工。将可选机器数量较少的工序先加工可以避免加工后期该工序可选机器有较大机器负载而造成完工时间过长的情况发生来减少 Cmax。

本章提出的初始化规则 HHIS 的具体步骤如下：

Step1：采用最小加工时间规则生产 ps/5 的子种群 P_1；

Step2：采用最小机器负载规则生产 ps/5 的子种群 P_2；

Step3：采用最小完工时间规则生产 ps/5 的子种群 P_3；

Step4：采用双贪心规则生产 ps/5 的子种群 P_4；

Step5：采用随机初始化规则生产 ps/5 的子种群 P_5；

Step6：合并种群 $P=P_1 \cup P_2 \cup P_3 \cup P_4 \cup P_5$ 得到最后的种群，即通过启发式规则快速构造收敛性好的子种群，又采用随机初始化保证种群的分布性，良好地平衡了收敛性与分布性，得到高质量的初始化种群。

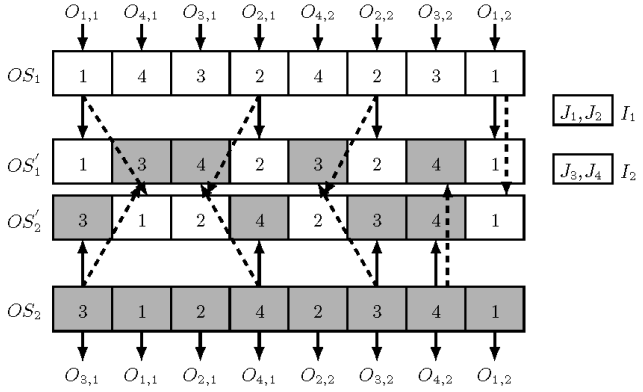

图 2.4 POX 交叉算子

2.4.3 进化算子

由于 GFJSP 的复杂性，为了保证算法能够产生较大的扰动来探索目标空间，本章采用偏于工序的交叉算子（Precedence operation crossover，POX）[113]对两个 OS 父代向量进行交叉交换信息。采用普遍交叉算子（Universial crossover，UX）[110]对两个父代 MS 向量进行交叉。此外，还为 OS 和 MS 向量设计了变异策略以保证种群的多样性。

POX 算子：图 2.4 展示了 POX 算子的原理，图中共有 4 个工件，每个工件 2

个工序共 8 个工序。(1)首先，将所有工件随机划分成两个集合 A={J_1，J_2}和 B={J_3，J_4}；（2）针对第一个父代 OS_1，找到 OS_1 中所有属于集合 A 的工序，将这些工序复制到子代向量 OS_1'中相同的位置，即 OS_1'中被箭头所指的白色部分。同样地，将 OS_2 中属于集合 B 的工序复制到子代向量 OS_2'中即 OS_2'中被箭头所指的灰色部分。（3）最后，将 OS_2 中属于 B 的工序按照从左到右的顺序填入 OS_1'中的无基因的空白位置，即 OS_1'中被虚线箭头所指的灰色部分。同样地，将 OS_1 中属于集合 A 的部分按顺序填入 OS_2'中。实现交换信息但不产生不可行解的目的。POX 可以产生较大扰动，搜索效率极高。

UX 算子：图 2.5 展示了 UX 算子的原理，图中共有 3 个工件共 7 个工序和 3 台机器。1) 首先，随机生产和总工序长度相等的 0-1 向量 R。2) 交换父代 MS_1 和 MS_2 中 R_i==0 全部的基因，即图中红色双向箭头所指部分。UX 可以保证 MS 向量中机器交换的部分仅在各个工序可选机器范围内进行而不会产生不可行解。

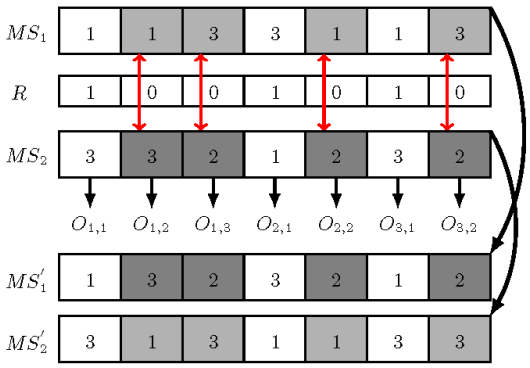

图 2.5 UX 交叉算子

OS 突变算子：为了保证基因的多样性，本章采用随机双点交换算子（Two point swap，TPS）对 OS 向量执行突变操作。首先，随机选择 OS 向量上的两个位置。随后交换两个基因。最后用突变的解替换原解。突变操作发生的概率为 P_m。

MS 突变算子：同样地，对 MS 向量采用随机替换机器策略。随机选择一个工序，在其可选的机器范围内，选择和当前机器不同的另一台机器加工。突变概率同样为 P_m。

2.4.4 模型驱动的局部搜索算子

通过分析问题模型，解构影响优化目标的因素，通过专家知识辅助设计局部搜索算子可提高局部搜索算子的成功率，避免局部搜索算子随机性太强而浪费大量计算资源。通过对 Cmax 的分析，提炼出 3 条知识：（1）最大完工时间与最后一个完成的工序之间相关，减少最后一个完工工序的完工时间可以直接减少最大完

工时间；（2）最大完工时间一定等于机器负载最大机器的完工时间，减少最大负载机器的完工时间可以直接减少完工时间。因此，于 Cmax 相关的关键操作主要在于替换机床。（3）此外，Cmax 还与机器上的空闲时间相关，如果减少了空闲时间则最大完工时间也能减少。根据上述知识，设计了四种模型驱动的局部搜索算子：

最后工序突变(Last operation mutation)：根据知识 1，找到最后完工的工序，在其可选机器范围内尝试放入到其他机器上，选择插入后完工时间最小的机器，更新机器选择。

最大负载机器突变（Max workload machine mutation）：根据知识 2，找到负载最大的机器，随机选择一个工序，在可选机器范围内选择加工时间更小的机床加工。

随机交换突变（Random swap mutation）：根据知识 3，随机在 OS 向量上选择两个工序，交换其位置，尝试通过改变工序加工顺序减少空闲时间。

随机插入突变（Random insertion mutation）：根据知识 3，随机在 OS 向量上选择两个工序，将后者插入到前者前面的位置，通过提取加工工序减少空闲时间。

最后，在算法执行完全局搜索更新种群后，针对新种群中的每个解，执行变邻域搜索操作（Variable neighborhood search，VNS）。在提出的四种局部搜索算子中随机选择一个执行，生产新的邻域解，并计算其目标函数。如果新解的各维度目标函数更优则替换原解，如果仅一个维度更好其他维度变差则加入经验存档 Ω 中，如果各维度目标函数都更差则放弃替换原解。

2.4.5 节能策略

节能策略是 GFJSP 中关键的一环，降低生产车间中的能耗不仅可以包含环境更可以实现企业可持续发展的道路。通过对 TEC 目标函数的分析，总能耗主要由加工能耗和空闲能耗决定。因此，提炼出两条知识来辅助设计节能策略，知识的定义和证明如下：

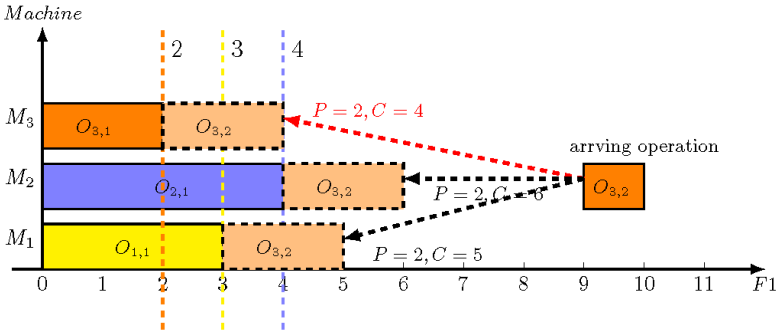

图 2.6 没有空闲时间出现时的节能策略

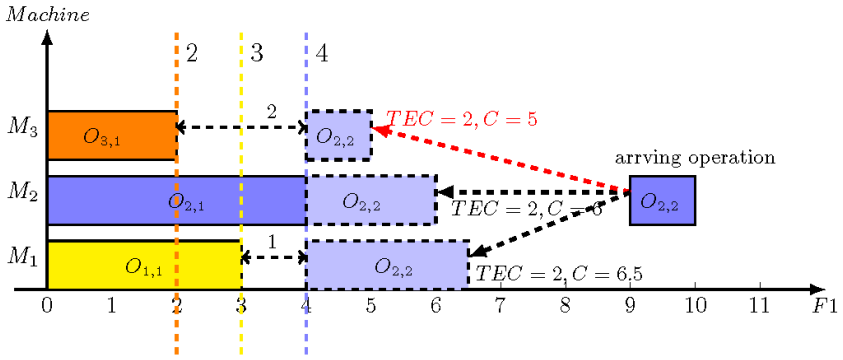

图 2.7 有空闲时间出现时的节能策略

算法2 - 2: 节能策略

输入: 一个解$\Pi(OS, MS)$，精英存档Ω

输出: 新解$\Pi'(OS', MS')$，精英存档Ω

$F \leftarrow \{\}, C \leftarrow \{\}, OpC \leftarrow zeros\,[1, N]$
for *i=1:length(OS)* **do**
  $OpC(OS(i)) \leftarrow OpC(OS(i)) + 1$
  $J(i) \leftarrow OpC(OS(i))$
end
for *i=1:length(OS)* **do**
  $I = OS(i)$
  if $J(i)==1$ **then**
    $MI \leftarrow$ 找到完工时间最小的机器(F).
    $MS'(i) \leftarrow$ 找到加工时间最小的机器$(I, J(i), MI)$
    $F_{MS'(i)} \leftarrow F\{MS'(i)\} + P^{O}_{I,J(i),MS'(i)}$
  else
    $Flag \leftarrow$是否存在空闲时间$(F, C_{I,J(i)-1})$.
    if $Flag == 1$ **then**
      $MS'(i) \leftarrow$ 找到能耗最小的机器$(F, C_{I,J(i)-1})$.
      $F_{MS'(i)} \leftarrow C_{I,J(i)-1} + P^{O}_{I,J(i),MS'(i)}$
    else
      $MS'(i) \leftarrow$ 找到加工时间最小的机器$(MFT, I, J(i))$.
      $F_{MS'(i)} \leftarrow F_{MS'(i)} + P^{O}_{I,J(i),MS'(i)}$
    end
  end
  $C_{I,J(i)} \leftarrow F_{MS'(i)}$
end
if $\Pi' \succ \Pi$ **then**
  $\Pi \leftarrow \Pi'$.
  $\Omega \leftarrow \Omega \cup \Pi'$
else
  Π' 和Π 互不支配
end
$\Omega \leftarrow \Omega \cup \Pi'$

定理 1：假设 Π 是当前的调度序列，下一个即将到来加工的工序是 $O_{i,j}$，通过与每个机器上的最后一个工序对比，以及和自身的前序工序 $O_{i,j-1}$ 对比，如果没有等待时间出现则选择 $O_{i,j}$ 加工时间最小的机器加工。

证明 1：假设 Π_1 是调度序列在工序 O_{ij} 选择最小加工时间机器 k 后的下一个状

态，假设 Π_2 是调度序列在工序 O_{ij} 选择其他机器 k’后的下一个状态。由于没有空闲时间则总能耗 TEC 的变化为 $TEC(\Pi_1)= TEC(\Pi)+P_W*P_{i,j,k}$，另外 $TEC(\Pi_2)= TEC(\Pi)+P_W*P_{i,j,k'}$。由于 $P_{i,j,k'}> P_{i,j,k}$ 则 $TEC(\Pi_1)< TEC(\Pi_2)$。因此选择最小加工时间的机器可以降低能耗。证明完毕。

定理 2：当 $O_{i,j}$，通过与每个机器上的最后一个工序对比，以及和自身的前序工序 $O_{i,j-1}$ 对比，如果有等待时间出现则选择 O_{ij} 总能耗最小的机器加工。

证明 2：假设 Π_1 是调度序列在工序 O_{ij} 选择最能耗机器 k 后的下一个状态，假设 Π_2 是调度序列在工序 O_{ij} 选择其他机器 k’后的下一个状态。则总能耗 TEC 的变化为 $TEC(\Pi_1)= TEC(\Pi)+TEC_1$，另外 $TEC(\Pi_2)= TEC(\Pi)+ TEC_2$。由于 $TEC_1< TEC_2$ 则 $TEC(\Pi_1)< TEC(\Pi_2)$。因此选择最小能耗的机器可以降低能耗。证明完毕。

图 2.6 列举了没有空闲时间出现时的节能策略，假设 O_{32} 是即将到来的工序，通过对比每个机器上的工序后，发现没有等待时间。通过计算后发现选择机器 M_3 使得能耗最小完工时间最小。说明了定理 1 的有效性。

图 2.7 列举了有空闲时间出现时的节能策略，假设 O_{22} 是即将到来的工序，通过对比每个机器上的工序后，因为 O_{22} 必须等待 O_{21} 完成，所以 M_3 和 M_1 出现等待时间。通过计算后发现选择机器 M_3 使得总能耗最小。说明定理 2 有效。

算法 2-2 详细说明了节能策略的过程。首先 1-4 行对 OS 向量进行解码，统计每个出现的工件是第几个工序。随后，从左到右扫描 OS 向量，获取当前工件 I，如果当前是工件 I 的第一个工序则不会出现等待时间，找到完工时间最小的机器进行加工。如果有多个完工时间最小的机器则选择当前工序 O_{ij} 加工时间最小的机器进行加工。否则，探索当前的调度序列的各个机器上是否存在等待时间。如果存在等待时间则选择能耗最小的机器进行加工，否则找到加工时间最小的机器加工，同时更新每个机器的完工时间。最后，计算新解的目标函数值，如果支配旧解则替换旧解，并将新解存入精英存档；否则如果新旧解相互之间不支配，则不替换旧解，并将新解存入精英存档。

2.4.6 基于 Q 学习的参数选择策略

(1)动机：本章所提出的 LRVMA 算法的主体框架是 MOEA/D 算法，所以 LRVMA 的分布性同样依赖于邻域更新范围 T 的取值。在进化的早期阶段，T 取值较大时收敛性更好。但前沿面的分布性与 T 之间没有明显的规律性。为了引导算法根据自身环境总是选择分布性更好的 T。强化学习具有强大的非监督自学能力，适合应用到算法中的自我调整模块。本节设计了一种基于 Q 学习带指标反馈的参数自学习（Q-learning and indicators feedback based parameter self-learning，QIFPSL）

模型来自动调整 T，以增强求得非支配解集的分布性。

(2) QIFPSL 模型：图 2.8 展示 QIFPSL 的模型框架。在该模型中每次算法更新后种群的 Pareto Front（PF）被看做一个智能体。模型中通过参考文献中两个多目标优化指标 GD 和 Delta[183]来衡量 PF 的收敛性和分布性。通过 GD 和 Spread 的变化量来衡量智能体 PF 状态的变化。在获取到智能体当前的状态后，根据 PF 分布性指标的反馈情况给予一定的奖励，同时智能体进入下一个状态，并根据 Q 表选择当前状态下回报最大的参数 T 输入到算法中，使得 LRVMA 朝着分布性增强的方向进化。

(3) 状态集定义：为了衡量 PF 的状态，借助 MOPs 问题中的两个指标 GD 和 Delta 来衡量收敛性和分布性，GD 越小收敛性和 Delta 越大分布性越好，两个指标的公式如下：

$$GD(P,P^*)=\frac{\sqrt{\sum_{y\in P}\min_{x\in P^*}d(x,y)^2}}{\|P\|} \tag{5-18}$$

$$Delta=\frac{\sum_{i=1}^{N-1}|d_i-\overline{d}|}{(N-1)\times\overline{d}} \tag{5-19}$$

在公式(2-18)中 P*表示真实 PF 前沿，P 表示算法求得的非支配解集，d 表示欧式距离。但 GFJSP 中真实前沿面无法提前获知，因此借助人工预设参考点来衡量收敛性指标。在公式(2-19)中 d_i 表示相邻两点之间的欧式距离。在本章中通过收敛性的变化量 $\Delta CV=GD(PF_t,P^*)-GD(PF_{t+1},P^*)$ 和分布性的变化 $\Delta DV=Delta(PF_t)-Delta(PF_{t+1})$。来划分智能体的状态，根据 ΔCV 和 ΔDV 两种变化可以组合出四种智能体的状态分别是：状态 1：ΔCV>0 和 ΔDV>0，状态 2：ΔCV>0 和 ΔDV≤0，状态 3：ΔCV≤0 和 ΔDV>0，状态 4：ΔCV≤0 和 ΔDV≤0。如此构成 4 个闭环的状态，经过环境的变化在 4 个状态之间转移。

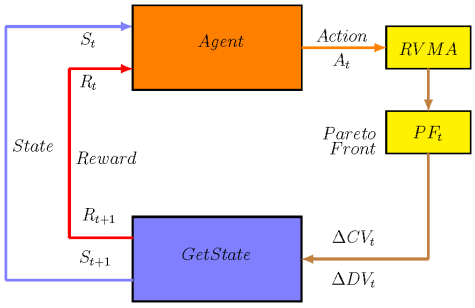

图 2.8 基于 Q 学习带指标反馈的参数自学习

(4) 动作集定义：在 QIFPSL 模型中，为智能体设定了 4 个动作即参数 A_t={5，10，15，20}。每个参数使得算法表现不同的性能。取值越大收敛性越强但分布性

急剧降低适合前期进化，取值较小分布性较好适合后期进化搜寻更多的非支配解。

(5) 奖励定义：为了引导算法选择参数增强前沿面的分布性，当 PF 更新后如果 ΔDV>0 说明 Delta 指标变大则相邻解的分布性更均匀，分布性更好则奖励 R=10。并通过以下公式更新 Q 表：

$$Q(S_t,A_t)=Q(S_t,A_t)+\alpha\times(R_t+\gamma\times\max Q(S_{t+1},:)-Q(S_t,A_t)) \tag{5-20}$$

其中 α 表示学习速率，γ 表示现实折扣率。

(6) 举例说明：图 2.9 举例了 QIFPSL 模型是如何进行的，假设当前智能体的状态是 S_1，选取当前 Q 表第一行 Q 值最大的动作 A_3，但有小概率随机选择动作。经过参数值为 A_3 的 MOEA/D 更新后，前沿面的收敛性变差，分布性变好，则给予奖励且智能体的状态变为 S_3。循环往复的执行以上步骤直到算法停止。

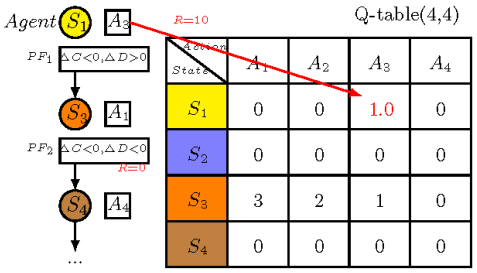

图 2.9 举例 QIFPSL 模型

算法2 - 3: LRVMA算法框架.

输入: 种群大小(ps), 最大函数评价次数($MaxNFEs$), 折扣因子(γ), 学习率(α), 贪心因子(ϵ)
输出: 非支配解PF

$\mathcal{P} \leftarrow$ 初始化种群(ps).
t$\leftarrow$ 0, $C_t = C_{t+1} = D_t = D_{t+1} = 0$.
$Q(4,4) \leftarrow 0$, $A \leftarrow \{5,10,15,20\}$, $\Omega \leftarrow []$.
$[\lambda, NM] \leftarrow$ 初始化参考向量($\mathcal{P}_t, ps+1, A(4)$).
while $t \leqslant MaxNFEs$ do
  $\Delta C = C_{t+1} - C_t, \Delta D = D_{t+1} - D_t$, $C_t = C_{t+1}, D_t = D_{t+1}$
  $S_t \leftarrow$ 获取当前状态($\Delta C, \Delta D$).
  $A_t \leftarrow$ 选择动作(S_t, A, ϵ, Q).
  $z^* \leftarrow$更新参考点.
  for i=1 to ps do
    $X_1 \leftarrow$ 进化算子($\mathcal{P}_t(i)$).
    $X_2 \leftarrow$ 变邻域搜索(X_1).
    $X_3 \leftarrow$ 节能策略(X_2).
    $z^* \leftarrow$更新参考点.
    for j=1 to A_t do
      $k \leftarrow NM(j)$.
      更新邻域($\lambda_k, \mathcal{P}_t(k), X_3, z^*$). //Tchebycheff聚合函数
    end
  end
  $PF_t \leftarrow$ 获取PF前沿($\mathcal{P}_t$).
  $\Omega \leftarrow \Omega \cup PF_t$.
  删除精英存档Ω重复解.
  $\Omega \leftarrow$ 快速非支配排序(Ω).
  $C_{t+1} \leftarrow GD(PF_t), D_{t+1} \leftarrow Diversity(PF_t)$.
  $\Delta C = C_t - C_{t+1}, \Delta D = D_t - D_{t+1}$.
  if $\Delta D_{t+1} > 0$ then
    $R_t = 10$
  end
  $S_{t+1} \leftarrow$ 获取下一个状态($\Delta C, \Delta D$).
  $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha * (R_t + \gamma * maxQ(S_{t+1}, :) - Q(S_t, A_t))$
end
$PF \leftarrow$ 快速非支配排序(Ω).

2.4.7 LRVMA 算法流程

算法 2-3 详细展示了本章提出的 LRVMA 算法的框架。首先，初始化种群和参考向量并初始化当前智能体的状态为 S_1。其次，进入循环体 5-8 行获取当前状态，并采用 ϵ-greedy 策略选择参数。随后，在 10-19 行执行 MOEA/D 算法部分进行全局搜索、局部搜索和节能策略，并用切比雪夫聚合函数更新邻域解，更新范围为选择的参数 A_t。然后，在更新完种群后，获取当前种群的前沿面，将前沿面归入精英存档并删除存档中重复解，对存档执行快速非支配排序选择第一层的前沿面保留。最后，对当前的最新前沿面根据公式(2-18)和(2-19)重新计算收敛性和分布性，依据指标的变化,反馈给智能体判断下一阶段状态，并更新 Q 表。

2.5 数值实验

2.5.1 实验设置

在上节中已经详细介绍了算法 LRVMA 的细节，本节设计了详细的实验来验证所提出算法的有效性。在本节中，首先在 10 个小测试问题上，通过 CPLEX 与 LRVMA 的对比验证了 GFJSP 的 MILP 模型的正确性。其次，通过田口正交实验对算法的最优参数选择进行了讨论。再次，经过成分分离试验验证 LRVMA 中每个改进点的有效性。最后，通过与最新的相关算法对比验证了 LRVMA 在 GFJSP 的求解上表现最好，并通过图表对比详细讨论了算法性能优越的原因。所有算法都用 MATLAB2016Rb 编程运行在 Intel Core i7-6700 CPU@ 3.4GHz 8G RAM Win10 上。为了公平起见，每个算法独立运行 20 次，通过收集收敛性，分布性和综合指标的均值和方差来衡量算法性能。

2.5.2 MILP 模型验证

CPLEX12.6.3 软件在本节被用于验证 MILP 模型的正确性。MILP 模型用 CPLEX 软件中特有的语言 optimization language 编程。由于 CPLEX 的只能求解 1000 维度的决策变量，所以生成了 10 个小测试问题来对比 CPLEX 和 LRVMA。表 2.1 展示了 LRVMA 和 CPLEX 的对比结果。可以看到 CPLEX 在两个目标上的下界都好于 LRVMA 这是因为 CPLEX 执行的是分支界定算法可以精确求解问题下界。侧面反映了所提的 MILP 模型的正确性。

表 2.1 LRVMA 和 CPLEX 在小规模测试问题上的对比结果

测试问题	Cmax		TEC	
	CPLEX	LRVMA	CPLEX	LRVMA
J3M3O3	144	170	177.56	178.84
J3M4O3	189	189	206.06	210.78
J3M5O3	181	254	197.56	207.36
J4M3O3	73	96	81.56	84.24
J4M4O3	106	109	119.56	121.76
J4M5O3	129	129	195.06	202.14
J5M3O3	138	168	192.56	194.92
J5M4O3	146	182	259.56	265.48
J5M5O3	147	147	242.06	243.86

2.5.3 测试问题和指标

为了验证 LRVMA 的有效性，本章选择了文献[70]中的公开测试集，总共包含 30 个测试问题。工件数范围 $n \in \{20,30,40,50,80,100\}$ 和机器数范围 $m \in \{6,7,8,9,10\}$，每个工件包含 5 个工序，每个工序可选机器范围有限。车间的加工能耗 P_W=0.5，机器的空闲能耗为 P_I=0.04。更多地，为了衡量多目标进化算法（Multi-objective evolutionary algorithms，MOEAs）的性能。本章采用三个 MOPs 的指标来衡量综合性能、收敛性和分布性，分别是超体积（Hypervolume，HV）[184]，世代距离（Generational distance，GD）[183]和分散性 Spread[185]。GD 在前面公式(2-18)有介绍，因此下面将介绍 HV 和 Spread 的公式如下：

$$HV(P)=\bigcup_{i=1}^{|P|}v_i, \tag{5-21}$$

其中 P 表示算法求得的 PF 前沿，v_i 表示所有解的目标函数值经过归一化后与参考点(1.1，1.1)围成的超立方体的体积，最后取所有超立方体围成的并集即为最后的指标。HV 指标可以综合判断算法求得非支配解集的收敛性和分布性，HV 值越大综合性能越好。

$$\begin{aligned} &Spread(P,P^*)=\frac{\sum_{i=1}^{|P^*|}d(P,P_i^*)+\sum_{X\in P}|d(X,P)-\overline{d}|}{\sum_{i=1}^{|P^*|}d(P,P_i^*)+(|P|-|P^*|)\overline{d}},\\ &d(X,P)=\min_{Y\in P,Y\neq X}||F(X)-F(Y)||,\\ &\overline{d}=\frac{1}{|P|}\sum_{X\in P}d(X,P), \end{aligned} \tag{5-22}$$

其中 P*表示真实的前沿面，由于在车间调度问题中真实的前沿面无法提前获得，因此 P*用所有对比算法求得的非支配解集并集中的非支配解集替代。d 表示欧式

距离。Spread 值越小，表明算法求得前沿面分布地更加均匀，说明分布性越好。

2.5.4 参数实验

算法参数是影响算法求解问题的性能最直接的因素。LRVMA 共有四个参数分别是种群大小 ps，折扣因子 γ，学习率 α 和贪心因子 ε。本章采用田口(Taguchi)正交实验（design-of-experiment，DOE）[186]来测试最优的参数组合。不同参数的级别设置如下：$ps \in \{40,60,80,100\}$，$\gamma \in \{0.9,0.8,0.7,0.6\}$，$\alpha \in \{0.1,0.2,0.3,0.4\}$，$\varepsilon \in \{0.95,0.9,0.85,0.8\}$。更多地，生成了正交实验表 $L_{16}(4^4)$来矫正最优参数选择。为了保证公平，每个参数组合在 30 个测试问题上独立运行 20 次。同时，收集三个指标的均值，并采用 Minitab18 进行田口实验分析，得出指标的主影响图。如图 2.10，2.11 和 2.12 所示，根据三个指标的综合评判，最后的最优参数组合为 ps=100，γ=0.8，α=0.1，ε=0.95。

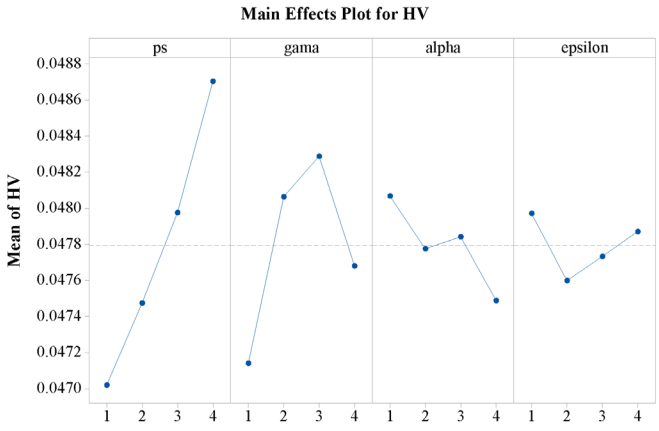

图 2.10 HV 指标主影响图

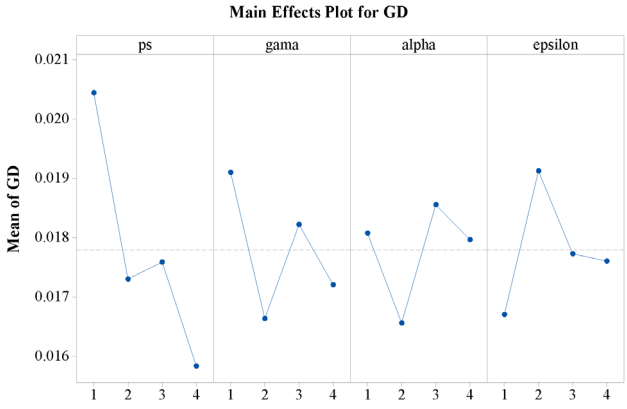

图 2.11 GD 指标主影响图

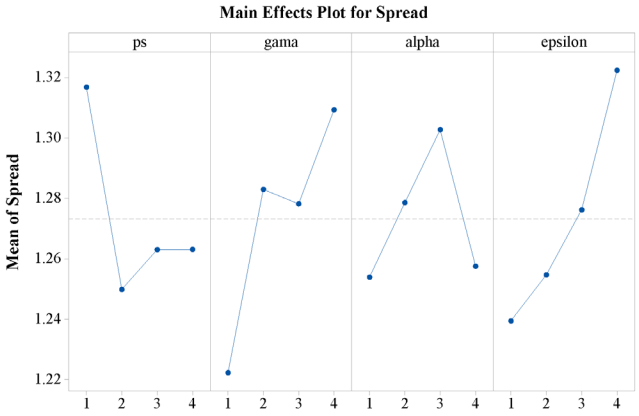

图 2.12 Spread 指标主影响图

2.5.5 成分分离实验

表 2.2 LRVMA 变体算法 Frideman 秩和检验统计结果(置信度 α=0.05)

MOEAs	HV		GD		Spread	
	rank	*p*-value	rank	*p*-value	rank	*p*-value
LRVMA1	4.967		4.967		3.033	
LRVMA2	4.000		4.000		**1.933**	
LRVMA3	1.967	3.36E-20	1.933	6.54E-20	3.400	1.15E-03
LRVMA4	2.400		2.333		3.233	
LRVMA	**1.667**		**1.767**		3.400	

为了验证 LRMVA 每个改进点的有效性，本节设计了四个 LRVMA 的变种算法分别是：LRVMA1 表示 LRVMA 没有混合初始化策略，LRVMA2 表示 LRVMA 没有变邻域搜索，LRVMA3 表示 LRVMA 没有贪心节能策略，LRVMA4 表示 LRVMA 没有提出的 QIFPSL 模型。为了公平起见，每个变种算法在 30 个测试问题上独立运行 20 次，停止条件为最大迭代次数 MaxG=200。为了验证改进点的有效性，本章采用 KEEL 软件对变体算法之间的显著性进行了统计和分析。表 2.2 展示了 LRVMA 变体算法的 Frideman 秩和检验统计结果。如表所示**加粗**表示排名最高的算法，其中 LRVMA 算法的排名最高，表明了 LRVMA 算法相较于去除改进点的变体算法表现出更优的性能，同时 p 值小于 0.05 说明算法之间具有显著性的差异，说明 LRVMA 的每个改进点显著性地有效。同时表 2.3 详细展示了每个指标的均值和标准差在独立运行 20 次后的统计结果。灰色和加粗表示最好。“-/=/+”表示各变体算法在该测试问题上显著性地“差于/无显著性/好于”LRVMA。在每个子表的最后统计了变体算法的显著性。可以得知 LRVMA 在所有测试上超过一半优于各变体算法，进一步证明了改进点的有效性。

表 2.3 LRVMA 变体算法 HV(max)，GD(min)和 Spread(min)指标对比结果

	HV									
测试问题	LRVMA1		LRVMA2		LRVMA3		LRVMA4		LRVMA	
	mean	std	mean	std	mean	std	mean	std	mean	std
J20M6	0.0801-	0.0020	0.0789-	0.0031	0.0827=	0.0014	0.082=	0.0012	**0.0830**	0.0014
J20M7	0.0704-	0.0055	0.0718-	0.0056	**0.0778=**	0.0023	0.0758-	0.0022	0.0778	0.0024
J20M8	0.0672-	0.0066	0.0728-	0.0045	0.0817=	0.0054	0.0809-	0.0036	**0.0832**	0.0032
J20M9	0.0674-	0.0103	0.0776-	0.0054	0.0843=	0.0024	0.0808-	0.0034	**0.0862**	0.0033
J20M10	0.0675-	0.0123	0.0933-	0.0059	0.1034=	0.0083	0.1013=	0.0057	**0.1066**	0.0068
J30M6	0.0592-	0.0023	0.0617-	0.0016	0.0642=	0.0011	0.0639=	0.0018	**0.0644**	0.0011
J30M7	0.0602-	0.0035	0.0619-	0.0020	**0.0667=**	0.0022	0.0658=	0.0015	0.0658	0.0030
J30M8	0.0597-	0.0050	0.0731-	0.0024	0.079=	0.0032	**0.0809=**	0.0016	0.0803	0.0034
J30M9	0.0611-	0.0049	0.0803-	0.0028	**0.089=**	0.0034	0.0885=	0.0048	0.0889	0.0033
J30M10	0.0668-	0.0068	0.0983-	0.0055	0.1074=	0.0035	0.1062=	0.0033	**0.1088**	0.0045
J40M6	0.0582-	0.0042	0.0636=	0.0016	0.0647=	0.0011	**0.065=**	0.0014	0.0648	0.0012
J40M7	0.0548-	0.0025	0.0612-	0.0013	0.063=	0.0017	**0.0633=**	0.0010	0.0631	0.0011
J40M8	0.0599-	0.0093	0.0738-	0.0021	0.081=	0.0034	0.0806=	0.0030	**0.0818**	0.0037
J40M9	0.065-	0.0072	0.0818-	0.0030	0.0871=	0.0027	0.0875=	0.0013	**0.0878**	0.0032
J40M10	0.0668-	0.0089	0.1121-	0.0034	0.1208=	0.0040	0.1195=	0.0046	**0.1224**	0.0039
J50M6	0.0611-	0.0036	0.0685-	0.0019	0.071=	0.0013	**0.0712=**	0.0017	0.0705	0.0017
J50M7	0.0583-	0.0042	0.0694-	0.0023	0.0725=	0.0010	0.0724=	0.0021	**0.0729**	0.0015
J50M8	0.0498-	0.0068	0.0677-	0.0017	0.0746=	0.0025	**0.075=**	0.0027	0.0749	0.0025
J50M9	0.0593-	0.0073	0.0688-	0.0048	0.0768=	0.0045	0.0775-	0.0029	**0.0811**	0.0025
J50M10	0.0675-	0.0074	0.1023-	0.0037	**0.1155=**	0.0040	0.1154=	0.0033	0.1154	0.0030
J80M6	0.049-	0.0033	0.0595=	0.0020	0.0593=	0.0008	**0.0602=**	0.0009	0.0601	0.0011
J80M7	0.0633-	0.0059	0.0746-	0.0014	0.0773=	0.0009	0.0771=	0.0011	**0.0775**	0.0012
J80M8	0.0532-	0.0040	0.0722-	0.0020	0.077=	0.0022	0.0757-	0.0012	**0.0776**	0.0025
J80M9	0.0723-	0.0074	0.0932=	0.0026	**0.0997=**	0.0020	0.0987=	0.0016	0.0973	0.0032
J80M10	0.0675-	0.0105	0.1034-	0.0043	**0.1105=**	0.0033	0.1083=	0.0031	0.1102	0.0038
J100M6	0.049-	0.0027	0.0596-	0.0011	0.0607=	0.0007	**0.0608=**	0.0006	0.0606	0.0006
J100M7	0.0506-	0.0038	0.0626-	0.0012	**0.0656=**	0.0011	0.0652=	0.0008	0.0655	0.0010
J100M8	0.0598-	0.0072	0.0857-	0.0021	**0.0911=**	0.0017	0.0906=	0.0022	0.0907	0.0022
J100M9	0.0565-	0.0047	0.0886-	0.0015	0.0921=	0.0011	0.0908=	0.0016	**0.0923**	0.0025
J100M10	0.0698-	0.0117	0.1167-	0.0031	**0.123=**	0.0027	0.1216=	0.0026	0.1229	0.0018
-/=/+	30/0/0		27/3/0		0/30/0		5/25/0		0/30/0	

表 2.3（续） LRVMA 变体算法 HV(max)，GD(min)和 Spread(min)指标对比结果

	GD									
测试问题	LRVMA1		LRVMA2		LRVMA3		LRVMA4		LRVMA	
	mean	std	mean	std	mean	std	mean	std	mean	std
J20M6	0.0043-	0.0013	0.0049-	0.0021	0.003=	0.0006	0.0027=	0.0011	**0.0027**	0.0009
J20M7	0.0089-	0.0038	0.0066-	0.0018	**0.0042=**	0.0021	0.005=	0.0017	0.0042	0.0017
J20M8	0.022-	0.0086	0.0114-	0.0057	0.0051=	0.0030	0.0069-	0.0040	**0.0038**	0.0029
J20M9	0.0209-	0.0099	0.011-	0.0072	**0.0041=**	0.0009	0.0074=	0.0039	0.0045	0.0026
J20M10	0.0418-	0.0107	0.0123-	0.0063	0.0067=	0.0045	0.0085=	0.0054	**0.0049**	0.0026
J30M6	0.0089-	0.0020	0.0085-	0.0044	0.0043=	0.0011	**0.0041=**	0.0016	0.0049	0.0021
J30M7	0.0092-	0.0028	0.0057=	0.0018	**0.0022=**	0.0010	0.0042=	0.0023	0.0035	0.0020
J30M8	0.0227-	0.0049	0.0059-	0.0016	**0.003=**	0.0018	0.0031=	0.0013	0.0030	0.0014
J30M9	0.0324-	0.0079	0.0147-	0.0052	0.0067=	0.0040	0.0077=	0.0055	**0.0064**	0.0031
J30M10	0.0504-	0.0088	0.0239-	0.0128	0.0086=	0.0049	0.0115=	0.0070	**0.0072**	0.0055
J40M6	0.0092-	0.0048	0.0042=	0.0020	0.0025=	0.0007	0.0023=	0.0011	**0.0021**	0.0008
J40M7	0.0128-	0.0037	0.0047=	0.0021	**0.0024=**	0.0018	0.0031=	0.0020	0.0029	0.0011
J40M8	0.0216-	0.0083	0.0065-	0.0017	0.0039=	0.0015	0.0031=	0.0013	**0.0030**	0.0012
J40M9	0.0268-	0.0120	0.0079-	0.0027	0.0034=	0.0021	0.0031=	0.0012	**0.0028**	0.0018
J40M10	0.064-	0.0155	0.0202-	0.0098	0.0079=	0.0046	0.007=	0.0055	**0.0067**	0.0049
J50M6	0.0112-	0.0029	0.003-	0.0005	0.0016=	0.0006	**0.0014=**	0.0007	0.0022	0.0015
J50M7	0.0181-	0.0060	0.0039-	0.0018	0.0025=	0.0007	**0.0024=**	0.0012	0.0027	0.0011
J50M8	0.0295-	0.0113	0.0071-	0.0027	**0.0023=**	0.0012	0.0028=	0.0014	0.0024	0.0008
J50M9	0.0241-	0.0096	0.0127-	0.0062	0.0068=	0.0044	0.0055-	0.0019	**0.0034**	0.0016
J50M10	0.0494-	0.0151	0.0111-	0.0063	**0.0017=**	0.0009	0.0024=	0.0008	0.0025	0.0015
J80M6	0.0159-	0.0040	0.0043=	0.0024	0.0054-	0.0019	0.0031=	0.0023	**0.0028**	0.0016
J80M7	0.0218-	0.0081	0.0066-	0.0021	**0.0028=**	0.0008	0.0041=	0.0024	0.0029	0.0015
J80M8	0.0292-	0.0084	0.0044=	0.0015	0.0038=	0.0008	0.0042=	0.0014	**0.0038**	0.0018
J80M9	0.0309-	0.0099	0.0106-	0.0033	**0.0029=**	0.0028	0.0043=	0.0018	0.0056	0.0035
J80M10	0.0379-	0.0099	0.0042-	0.0033	**0.0018=**	0.0010	0.0024=	0.0014	0.0020	0.0013
J100M6	0.0184-	0.0050	0.0046=	0.0028	0.0024=	0.0017	**0.002=**	0.0014	0.0020	0.0009
J100M7	0.0168-	0.0067	0.003-	0.0009	0.0011=	0.0006	0.0015-	0.0006	**0.0009**	0.0005
J100M8	0.0328-	0.0106	0.0048-	0.0036	**0.0015=**	0.0008	0.0017=	0.0014	0.0016	0.0007
J100M9	0.0342-	0.0082	0.0043-	0.0017	0.0024=	0.0017	0.0022=	0.0011	**0.0019**	0.0016
J100M10	0.0496-	0.0186	0.0043-	0.0011	**0.0017=**	0.0008	0.0019=	0.0012	0.0020	0.0014
-/=/+	30/0/0		24/6/0		1/29/0		3/27/0		0/30/0	

表 2.3（续） LRVMA 变体算法 HV(max)，GD(min)和 Spread(min)指标对比结果

	Spread									
测试问题	LRVMA1		LRVMA2		LRVMA3		LRVMA4		LRVMA	
	mean	std	mean	std	mean	std	mean	std	mean	std
J20M6	1.0493=	0.1942	**0.9122=**	0.1799	0.9491=	0.1632	0.9745=	0.1146	1.0220	0.1960
J20M7	0.9909=	0.1320	1.03=	0.1787	1.0205=	0.2427	0.9461=	0.1210	**0.9398**	0.1832
J20M8	1.0536=	0.0951	**0.9665=**	0.0590	1.1465=	0.3596	1.1017=	0.2639	1.0301	0.1583
J20M9	1.0385=	0.1108	1.0265=	0.1305	1.0521=	0.1465	**0.9262=**	0.1171	1.0540	0.3179
J20M10	**0.9751=**	0.0334	0.9997=	0.0613	1.0616=	0.1681	1.0913=	0.1852	1.2575	0.6223
J30M6	1.0051=	0.1969	0.9611=	0.1892	**0.9383+**	0.1749	1.0712=	0.1556	1.1669	0.2549
J30M7	0.9817=	0.0988	**0.9046=**	0.1421	1.0504=	0.3018	1.0111=	0.1668	1.1023	0.3820
J30M8	1.0009=	0.0586	0.9877=	0.0478	1.0119=	0.1097	**0.9494=**	0.1537	1.0231	0.2077
J30M9	1.012=	0.0656	**0.9633+**	0.0572	1.084=	0.2104	1.0005=	0.2293	1.0924	0.2195
J30M10	**0.9883=**	0.0428	1.0394=	0.1486	1.0642=	0.1651	1.0453=	0.1785	1.0851	0.2351
J40M6	1.0709=	0.1914	0.949=	0.2270	0.9682=	0.1653	1.1053=	0.2340	**0.9270**	0.1380
J40M7	1.1035=	0.1836	1.0313=	0.2827	1.0315=	0.1319	**0.9831=**	0.1776	1.0765	0.2216
J40M8	1.0262=	0.0842	0.9971=	0.1322	1.0418=	0.1771	**0.986=**	0.1290	0.9863	0.1276
J40M9	1.0001=	0.0383	**0.9136=**	0.1711	0.9476=	0.1554	0.9781=	0.2276	0.9563	0.1308
J40M10	1.0117=	0.0403	**0.9644=**	0.0443	1.1342=	0.4323	1.0282=	0.1505	1.0462	0.1968
J50M6	1.0057=	0.1732	0.9141=	0.1377	**0.8962=**	0.1475	0.9912=	0.1085	0.9488	0.0905
J50M7	1.0129=	0.0759	0.9909=	0.1243	0.9466=	0.1201	**0.8922=**	0.1938	1.0258	0.1537
J50M8	1.0401=	0.0881	**0.9768+**	0.1006	1.0617=	0.1966	1.2798=	0.3971	1.0864	0.1419
J50M9	0.9953=	0.0456	0.9635=	0.0589	1.039=	0.2902	**0.9473=**	0.0824	1.1474	0.3041
J50M10	0.9971=	0.0261	**0.9978=**	0.0664	1.0489=	0.1393	1.0248=	0.1256	1.0011	0.1352
J80M6	1.0517-	0.0731	0.9424=	0.2370	1.1268-	0.2714	1.0545-	0.2067	**0.8310**	0.1595
J80M7	1.041+	0.1201	1.0708=	0.1137	**1.0325=**	0.3191	1.2055=	0.3740	1.1801	0.2745
J80M8	1.0491=	0.1786	**0.9691+**	0.1445	1.0079=	0.1447	1.0092=	0.0574	1.0297	0.1092
J80M9	1.0558-	0.1102	0.9862=	0.0987	1.0624=	0.2734	1.0114=	0.2170	**0.9473**	0.1373
J80M10	0.9973=	0.0323	**0.9499=**	0.0817	0.9764=	0.0930	1.0072=	0.1577	1.0011	0.0997
J100M6	1.0336=	0.0809	**0.8914=**	0.2393	1.162=	0.4549	1.0418=	0.2233	1.0447	0.2302
J100M7	1.0016=	0.0485	0.9904=	0.1891	**0.9567=**	0.1102	1.0906=	0.2569	0.9691	0.1613
J100M8	0.9841=	0.0299	1.0048=	0.1194	1.073=	0.2998	1.0484=	0.1744	**0.9135**	0.1180
J100M9	0.9852=	0.0135	1.0041=	0.1236	1.0108=	0.1030	1.0076=	0.1551	**0.9761**	0.1997
J100M10	0.9969=	0.0120	0.9799=	0.0657	**0.9979=**	0.0896	1.0318=	0.0772	1.0179	0.1361
-/=/+	2/27/1		0/27/3		1/28/1		1/29/0		0/30/0	

2.5.6 对比实验

为了进一步验证 LRMVA 每个改进点的有效性，本节选取了几类 MOEAs 作为对比算法分别是：（1）主流 MOEAs：MOEA/D[182]和 NSGA-II[183]；（2）近期提出的 MOEAs：AR-MOEA[187]和 C-TSEA[188]；（3）GFJSP 相关算法：IAIS[70]和

FBEA[108]。每个算法的参数设置如下：所有算法的种群大小 ps=100，交叉概率 P_c=1，突变概率 P_m=0.2。MOEA/D 的邻域更新范围 T=10，IAIS 的参数模拟退火参数 Ts=0.5，克隆数 nc=10，种群拥挤阈值 CR=0.00001。为了对比公平，每个算法独立运行 20 次，停止条件为最大迭代次数 MaxG=200。表 2.4 展示了所有对比算法的 Frideman 秩和检验统计结果，结果显示 LRVMA 在 HV 和 GD 指标上排名第 1 显著地好于所有对比算法，在 Spread 指标上排名第 4。表 2.5，2.6，2.7 展示了所有指标的均值和标准差的统计结果，结果显示 LRVMA 在 HV 和 GD 指标上有超过一半以上的测试集的表现显著好于所有对比算法，验证了本章提出算法的有效性。

2.5.7 讨论与分析

LRVMA 算法在 GFJSP 问题上的成功在于它的设计。首先，混合启发式初始化策略可以保证算法获得收敛性和分布性良好的种群，这即节省了计算资源又使得在起点上比其他算法更好。其次，通过对模型的分析提炼出了多条知识辅助局部搜索算子的设计，提高了局部搜索的成功率节省了计算资源。再次，本章提出的贪心节能策略成功降低生产车间的能耗。最后，提出的基于 Q 学习和指标反馈的参数自学习模型帮助 MOEA/D 算法自动调整最优的参数选择，使得算法一直向分布性更好的方向进化。为了进一步说明 LRVMA 算法的有效性，本节还通过图来展示算法性能并分析其有效的原因。图 2.12 展示了所有算法在 J20M6 测试集上求得 HV 值最好的非支配解集，可以看到 LRVMA 的非支配解集更能探索前沿面，表明了算法的性能最好。图 2.13 展示了 LRVMA 在一轮进化过程中在不同阶段 CV 和 DV 的变化。由图可知，CV 随着进化一直减少而每一阶段 DV 随着进化分布性急剧降低，但可以明显发现 DV 在周期性的增大，这是因为 QIFPSL 模型使得算法朝着 DV 增大的方向调整参数，验证了提出 QIFPSL 模型的有效性。图 2.14 展示了 LRVMA 在 J20M6 测试集上找到的 Cmax 最小解的甘特图(Cmax=220.33，TEC=636.326)。图 2.15 展示了 LRVMA 在 J20M6 测试集上找到的 TEC 最小解的甘特图(Cmax=233.17，TEC=633.56)。综述所述，LRVMA 在求解 GFJSP 有很好的性能。

表 2.4 LRVMA 与对比算法的 Frideman 秩和检验统计结果(置信度 α=0.05)

MOEAs	HV		GD		Spread	
	rank	*p*-value	rank	*p*-value	rank	*p*-value
NSGA-II	6.933		6.833		4.200	
MOEA/D	5.800		5.733		**2.800**	
AR-MOEA	3.700		3.433		3.033	
IAIS	2.233	8.83E-31	2.800	5.60E-26	4.867	3.86E-09
C-TSEA	4.633		4.300		3.100	
FBEA	3.600		3.633		5.967	
LRVMA	**1.100**		**1.267**		4.033	

表 2.5 LRVMA 与对比算法 HV(max)指标对比结果

测试问题	NSGA-II		MOEA/D		AR-MOEA		IAIS		C-TSEA		FBEA		LRVMA	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
J20M6	0.0428-	0.0064	0.0498-	0.0024	0.0546-	0.0019	0.0513-	0.0025	0.0545-	0.0016	0.0301-	0.0042	**0.0566**	0.0010
J20M7	0.0629-	0.0064	0.074-	0.0039	0.0815-	0.0045	0.0843-	0.0030	0.081-	0.0034	0.077-	0.0018	**0.0877**	0.0012
J20M8	0.0207-	0.0039	0.0318-	0.0048	0.0532-	0.0395	0.0509-	0.0034	0.0405-	0.0032	0.0334-	0.0021	**0.0568**	0.0025
J20M9	0.0334-	0.0064	0.0506-	0.0062	0.0616-	0.0048	0.0757-	0.0051	0.0608-	0.0050	0.0611-	0.0012	**0.0827**	0.0035
J20M10	0.0298-	0.0064	0.0488-	0.0076	0.071-	0.0320	0.0882-	0.0081	0.0629-	0.0050	0.0836-	0.0249	**0.0960**	0.0066
J30M6	0.0419-	0.0052	0.0524-	0.0026	0.058-	0.0031	0.0583-	0.0049	0.0573-	0.0018	0.0391-	0.0035	**0.0624**	0.0011
J30M7	0.0375-	0.0057	0.0486-	0.0030	0.0545-	0.0048	0.0591-	0.0026	0.0534-	0.0023	0.0449-	0.0024	**0.0633**	0.0020
J30M8	0.0297-	0.0070	0.0527-	0.0027	0.0639-	0.0153	0.0787-	0.0036	0.0596-	0.0051	**0.0911+**	0.0047	0.0832	0.0033
J30M9	0.0307-	0.0070	0.0598-	0.0074	0.0678-	0.0051	0.0987=	0.0041	0.0677-	0.0054	0.0977=	0.0032	**0.1006**	0.0029
J30M10	0.0264-	0.0055	0.052-	0.0053	0.0698-	0.0200	0.1067-	0.0054	0.0642-	0.0053	**0.1335+**	0.0141	0.1152	0.0038

表 2.5（续） LRVMA 与对比算法 HV(max)指标对比结果

测试问题	NSGA-II		MOEA/D		AR-MOEA		IAIS		C-TSEA		FBEA		LRVMA	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
J40M6	0.0415-	0.0058	0.0539-	0.0032	0.0593-	0.0017	0.0648-	0.0029	0.0588-	0.0024	0.0514-	0.0368	**0.0695**	0.0011
J40M7	0.0443-	0.0053	0.0672-	0.0029	0.0716-	0.0041	0.0833-	0.0028	0.0697-	0.0048	0.0629-	0.0037	**0.0873**	0.0012
J40M8	0.0235-	0.0055	0.0475-	0.0048	0.0668-	0.0132	0.0837-	0.0037	0.0561-	0.0045	0.0673-	0.0160	**0.0906**	0.0030
J40M9	0.0424-	0.0094	0.0754-	0.0057	0.083-	0.0047	0.1221-	0.0030	0.0821-	0.0050	0.1038-	0.0021	**0.1250**	0.0029
J40M10	0.0282-	0.0148	0.0572-	0.0064	0.0718-	0.0190	0.1274-	0.0048	0.0646-	0.0063	0.1231-	0.0247	**0.1330**	0.0044
J50M6	0.0344-	0.0050	0.0549-	0.0036	0.0588-	0.0027	0.0666-	0.0029	0.0583-	0.0028	0.0423-	0.0081	**0.0705**	0.0013
J50M7	0.0503-	0.0068	0.0774-	0.0052	0.0811-	0.0033	0.1031-	0.0023	0.0813-	0.0038	0.0954-	0.0005	**0.1052**	0.0011
J50M8	0.028-	0.0057	0.0612-	0.0051	0.0696-	0.0197	0.106-	0.0036	0.0654-	0.0053	0.111=	0.0013	**0.1112**	0.0020
J50M9	0.0343-	0.0060	0.0709-	0.0062	0.0784-	0.0050	0.1269-	0.0035	0.077-	0.0072	0.105-	0.0035	**0.1342**	0.0024
J50M10	0.0276-	0.0059	0.0635-	0.0068	0.0766-	0.0154	0.1496-	0.0042	0.0712-	0.0046	0.1468=	0.0144	**0.1537**	0.0033
J80M6	0.023-	0.0066	0.0404-	0.0026	0.0458-	0.0020	0.0642-	0.0021	0.0459-	0.0032	0.0574-	0.0253	**0.0676**	0.0010
J80M7	0.0234-	0.0052	0.0521-	0.0046	0.0569-	0.0021	0.0967=	0.0013	0.0539-	0.0049	0.0875-	0.0030	**0.0976**	0.0010
J80M8	0.0194-	0.0131	0.045-	0.0062	0.0853-	0.0040	0.1084-	0.0027	0.0494-	0.0050	0.0881-	0.0152	**0.1131**	0.0023
J80M9	0.0184-	0.0042	0.047-	0.0048	0.0638-	0.0089	0.1186-	0.0027	0.0511-	0.0054	0.106-	0.0011	**0.1230**	0.0029
J80M10	0.0245-	0.0135	0.0578-	0.0065	0.0736-	0.0190	0.1574-	0.0037	0.0658-	0.0065	0.156-	0.0094	**0.1635**	0.0034
J100M6	0.0325-	0.0044	0.0601-	0.0039	0.0652-	0.0046	0.0998=	0.0016	0.0643-	0.0053	0.0972-	0.0182	**0.1008**	0.0006
J100M7	0.0479-	0.0073	0.0805-	0.0038	0.0854-	0.0038	0.1276-	0.0019	0.0846-	0.0050	0.1168-	0.0023	**0.1308**	0.0009
J100M8	0.0168-	0.0040	0.0477-	0.0039	0.0607-	0.0301	0.1264-	0.0025	0.0512-	0.0052	0.1129-	0.0254	**0.1332**	0.0018
J100M9	0.0211-	0.0059	0.0533-	0.0059	0.0643-	0.0028	0.1461-	0.0035	0.0606-	0.0056	0.1327-	0.0010	**0.1519**	0.0026
J100M10	0.0169-	0.0026	0.0427-	0.0076	0.0604-	0.0188	0.171-	0.0159	0.0522-	0.0057	**0.1862+**	0.0018	0.1731	0.0029
-/=/+	30/0/0		30/0/0		30/0/0		27/3/0		30/0/0		24/3/3		0/30/0	

表 2.6 LRVMA 与对比算法 GD(min)指标对比结果

测试问题	NSGA-II		MOEA/D		AR-MOEA		IAIS		C-TSEA		FBEA		LRVMA	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
J20M6	0.0194-	0.0096	0.0099-	0.0039	0.0039-	0.0016	0.0065-	0.0028	0.0046-	0.0015	0.0479-	0.0078	**0.0027**	0.0007
J20M7	0.0288-	0.0107	0.0158-	0.0055	0.0083-	0.0035	0.0102-	0.0150	0.0083-	0.0033	0.0684-	0.0059	**0.0035**	0.0017
J20M8	0.2209-	0.0519	0.1946-	0.0378	**0.1204=**	0.0117	0.1841-	0.0360	0.153=	0.0281	0.1678-	0.0319	0.1364	0.0311
J20M9	0.0686-	0.0151	0.039-	0.0120	0.0266-	0.0156	0.0067-	0.0037	0.0241-	0.0064	0.0274-	0.0033	**0.0039**	0.0023
J20M10	0.17-	0.0538	0.141-	0.0395	0.0819=	0.0107	0.102=	0.0217	0.107=	0.0185	**0.077=**	0.0111	0.0933	0.0293
J30M6	0.0262-	0.0095	0.0156-	0.0058	0.0065-	0.0012	0.0083-	0.0065	0.0084-	0.0029	0.0326-	0.0031	**0.0032**	0.0010
J30M7	0.0326-	0.0094	0.0201-	0.0066	0.0116-	0.0031	0.0069-	0.0035	0.0126-	0.0028	0.0136-	0.0016	**0.0031**	0.0017
J30M8	0.0872-	0.0201	0.0603-	0.0155	0.0356-	0.0046	0.0266-	0.0080	0.0385-	0.0079	0.0309-	0.0045	**0.0139**	0.0055
J30M9	0.0947-	0.0213	0.0537-	0.0150	0.0347-	0.0052	0.0051=	0.0032	0.0357-	0.0056	0.0336-	0.0066	**0.0043**	0.0019
J30M10	0.1254-	0.0204	0.0853-	0.0174	0.0545-	0.0114	0.0282-	0.0143	0.0585-	0.0085	0.0342-	0.0065	**0.0075**	0.0049
J40M6	0.1145-	0.0206	0.1034-	0.0175	0.0674=	0.0053	0.1142-	0.0231	0.0924-	0.0099	0.0956-	0.0087	**0.0638**	0.0073
J40M7	0.0488-	0.0109	0.0218-	0.0051	0.0356-	0.0112	0.0089-	0.0135	0.0168-	0.0040	0.0467-	0.0135	**0.0030**	0.0014
J40M8	0.0923-	0.0224	0.0494-	0.0105	0.0262-	0.0026	0.0056-	0.0028	0.037-	0.0091	0.0269-	0.0042	**0.0026**	0.0012
J40M9	0.089-	0.0175	0.0559-	0.0133	0.0533-	0.0185	0.0043-	0.0035	0.0411-	0.0090	0.0446-	0.0121	**0.0021**	0.0012
J40M10	0.1288-	0.0307	0.0976-	0.0253	0.0657-	0.0044	0.0215-	0.0117	0.0737-	0.0098	0.0397-	0.0030	**0.0088**	0.0064
J50M6	0.0463-	0.0140	0.0205-	0.0068	0.012-	0.0031	0.0072-	0.0054	0.0148-	0.0047	0.0187-	0.0043	**0.0028**	0.0026
J50M7	0.0571-	0.0122	0.0281-	0.0062	0.0212-	0.0052	0.0038=	0.0040	0.0228-	0.0067	0.0724-	0.0028	**0.0020**	0.0010
J50M8	0.0974-	0.0115	0.0654-	0.0173	0.0431-	0.0072	0.0099-	0.0063	0.0485-	0.0089	0.013-	0.0012	**0.0030**	0.0016
J50M9	0.1015-	0.0179	0.0686-	0.0157	0.0432-	0.0047	0.0082-	0.0043	0.0493-	0.0106	0.0377-	0.0016	**0.0029**	0.0012
J50M10	0.1238-	0.0226	0.0786-	0.0177	0.0467-	0.0051	0.002=	0.0013	0.0641-	0.0115	0.0035-	0.0007	**0.0018**	0.0008
J80M6	0.1803-	0.0345	0.1673-	0.0478	**0.0931+**	0.0083	0.1808-	0.0414	0.1318-	0.0263	0.109=	0.0037	0.1028	0.0128

表 2.6（续） LRVMA 与对比算法 GD(min)指标对比结果

测试问题	NSGA-II		MOEA/D		AR-MOEA		IAIS		C-TSEA		FBEA		LRVMA	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
J80M7	0.0839-	0.0135	0.0432-	0.0108	0.0316-	0.0012	0.0036=	0.0021	0.0372-	0.0105	0.0092=	0.0125	**0.0024**	0.0010
J80M8	0.1195-	0.0241	0.0836-	0.0201	0.047-	0.0039	0.011-	0.0049	0.0715-	0.0122	0.0364-	0.0024	**0.0030**	0.0021
J80M9	0.1197-	0.0235	0.0861-	0.0329	0.0489-	0.0038	0.0028-	0.0018	0.0691-	0.0095	0.0185-	0.0007	**0.0015**	0.0010
J80M10	0.1535-	0.0198	0.1126-	0.0312	0.0738-	0.0129	0.0068-	0.0045	0.0849-	0.0179	0.0316-	0.0011	**0.0024**	0.0029
J100M6	0.1208-	0.0296	0.0992-	0.0201	0.0635-	0.0045	0.0634-	0.0109	0.0808-	0.0171	**0.0399=**	0.0038	0.0425	0.0073
J100M7	0.0729-	0.0160	0.0412-	0.0084	0.0407-	0.0125	0.0029-	0.0019	0.0378-	0.0108	0.0255-	0.0096	**0.0008**	0.0006
J100M8	0.2116-	0.0429	0.1734-	0.0405	0.1089-	0.0091	0.1167-	0.0213	0.1508-	0.0241	**0.0583+**	0.0035	0.0728	0.0176
J100M9	0.1005-	0.0180	0.0805-	0.0249	0.0473-	0.0025	0.004=	0.0035	0.0706-	0.0144	**0.001+**	0.0002	0.0017	0.0008
J100M10	0.1911-	0.0638	0.1521-	0.0558	0.1027-	0.0069	0.0755-	0.0154	0.1311-	0.0240	**0.0195+**	0.0006	0.0464	0.0129
-/=/+	30/0/0		30/0/0		26/3/1		24/6/0		28/2/0		23/4/3		0/30/0	

表 2.7 LRVMA 与对比算法 Spread(min)指标对比结果

测试问题	NSGA-II		MOEA/D		AR-MOEA		IAIS		C-TSEA		FBEA		LRVMA	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
J20M6	1.0088=	0.1327	0.9658=	0.1103	1.0103=	0.1819	1.0862=	0.2524	1.0378=	0.1676	1.0647-	0.0843	**0.9557**	0.1546
J20M7	1.1011=	0.1774	0.9992=	0.1195	1.021=	0.1696	1.3405=	0.4352	1.0173=	0.1904	1.0618-	0.0821	**0.9289**	0.1906
J20M8	1.0027=	0.0146	1.0003=	0.0145	1.0246-	0.2083	1.0053=	0.0188	**0.9945=**	0.0132	1.0019=	0.0212	1.0034	0.0175
J20M9	1.0259=	0.0572	**0.9878=**	0.0612	1.0242=	0.1318	1.0606=	0.1370	1.0325=	0.0754	1.0353=	0.0678	1.0347	0.2279
J20M10	1.0136=	0.0389	0.9933=	0.0215	**0.9762=**	0.0557	1.0089=	0.0240	1.0046=	0.0208	1.0355=	0.2120	1.0171	0.0530
J30M6	1.0033=	0.0560	1.0147=	0.0382	1.0274+	0.1643	**1.0025=**	0.0754	1.031=	0.0754	1.0312=	0.0277	1.0284	0.0459
J30M7	1.0158=	0.0327	1.0065=	0.0405	**0.9805+**	0.0309	1.0077=	0.0217	1.0061=	0.0239	1.1537-	0.1082	1.0231	0.0454
J30M8	1.0035=	0.0343	1.0043=	0.0170	1.0088=	0.0805	**0.9966=**	0.0191	0.999=	0.0264	1.26-	0.2059	1.0006	0.0375
J30M9	1.0169=	0.0484	0.9928=	0.0350	**0.9791+**	0.0263	0.9994=	0.0192	0.9941=	0.0299	1.4338-	0.0890	1.0186	0.0380

表 2.7（续） LRVMA 与对比算法 Spread(min)指标对比结果

测试问题	NSGA-II		MOEA/D		AR-MOEA		IAIS		C-TSEA		FBEA		LRVMA	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
J30M10	1.0068=	0.0216	0.9959+	0.0161	**0.978+**	0.0515	1.0048=	0.0140	1.003=	0.0293	1.5726-	0.0898	1.0087	0.0210
J40M6	0.9995=	0.0148	0.9974=	0.0053	**0.9901+**	0.0163	1.0049=	0.0167	1.0101=	0.0249	1.0617-	0.1475	1.0017	0.0181
J40M7	1.0174=	0.0660	1.017=	0.0814	1.4657-	0.3223	**0.9964=**	0.1257	1.0041=	0.1079	1.0148=	0.0329	1.0542	0.2197
J40M8	1.0104=	0.0231	1.0009=	0.0193	1.037-	0.1665	1.0001=	0.0328	**0.9991=**	0.0202	1.0045=	0.0205	1.0021	0.0418
J40M9	1.0034=	0.0186	1.0067=	0.0316	1.0576=	0.1943	1.008=	0.1973	1.0132=	0.0314	1.5547-	0.2358	**0.9873**	0.1755
J40M10	1.0031=	0.0133	0.9938=	0.0184	1.0009=	0.0390	1.0155=	0.0366	**0.9931+**	0.0146	1.3641-	0.0766	1.0182	0.0516
J50M6	1.004=	0.0229	1.0046=	0.0358	**0.9853=**	0.0232	1.0175=	0.0278	1.0065=	0.0264	1.0009=	0.1073	0.9935	0.0190
J50M7	1.0095=	0.0435	0.9986=	0.0513	**0.9819=**	0.0681	1.0628-	0.2293	0.9952=	0.0450	1.082-	0.0781	0.9868	0.1514
J50M8	1.0024+	0.0273	1.0014=	0.0245	**0.9958=**	0.0265	1.0033=	0.0174	0.9991=	0.0121	1.0375=	0.0490	1.0133	0.0228
J50M9	1.0119=	0.0442	0.9978=	0.0167	**0.9764+**	0.0199	1.0033=	0.0108	0.9962+	0.0130	1.045-	0.0421	1.0060	0.0162
J50M10	1.0011=	0.0141	0.9946=	0.0129	**0.9852=**	0.0155	1.0094=	0.0233	0.9999=	0.0107	1.0437=	0.0569	1.0049	0.0155
J80M6	1.0175=	0.0677	0.9972=	0.0104	**0.9847+**	0.0093	1.0013=	0.0076	1.0007=	0.0095	1.0172=	0.0747	0.9980	0.0066
J80M7	0.9942+	0.0213	0.9976=	0.0248	**0.959+**	0.0281	1.0146=	0.0316	1.0055=	0.0340	1.0869-	0.0787	1.0126	0.0298
J80M8	1.0033=	0.0211	0.9972=	0.0112	1.044=	0.1271	1.0071=	0.0490	**0.9957=**	0.0133	0.9996=	0.0485	1.0070	0.0241
J80M9	0.9984=	0.0107	0.993=	0.0172	1.2277=	0.2527	1.0017=	0.0129	0.9941=	0.0080	**0.9598+**	0.0237	1.0007	0.0191
J80M10	0.9989=	0.0062	1.0014=	0.0150	**0.9872=**	0.0277	1.0122=	0.0469	0.9986=	0.0150	1.0358=	0.0914	0.9959	0.0367
J100M6	1.0051=	0.0138	**0.9981=**	0.0181	1.0012=	0.0325	1.0042=	0.0141	1.0003=	0.0087	1.0796-	0.1970	1.0044	0.0101
J100M7	1.0035=	0.0236	0.996=	0.0301	0.9934=	0.0382	1.0118=	0.0399	1.0012=	0.0234	1.4421-	0.3083	**0.9902**	0.0381
J100M8	0.9969=	0.0076	1.0027=	0.0135	**0.9894=**	0.0153	1.0013=	0.0090	0.9967=	0.0060	1.069-	0.1673	0.9976	0.0083
J100M9	**0.9981=**	0.0092	0.9984=	0.0102	1.006=	0.0453	1.0028=	0.0114	1.0012=	0.0140	1.002=	0.0158	0.9982	0.0205
J100M10	1.0023=	0.0093	0.9961=	0.0078	**0.9817+**	0.0108	1.0336=	0.1316	0.9961=	0.0065	1.0414-	0.0255	0.9973	0.0127
-/=/+	0/28/2		0/29/1		3/18/9		1/29/0		0/28/2		16/13/1		0/30/0	

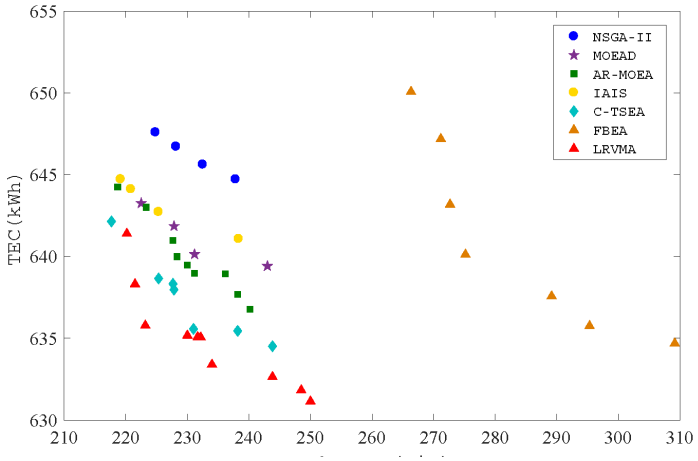

图 2.12 对比算法在 J20M6 上 PF 对比图

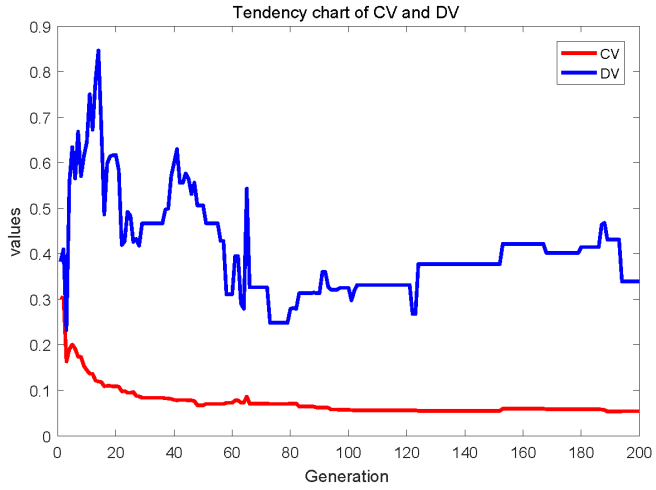

图 2.13 LRVMA 在 J20M6 上 CV 和 DV 变化图

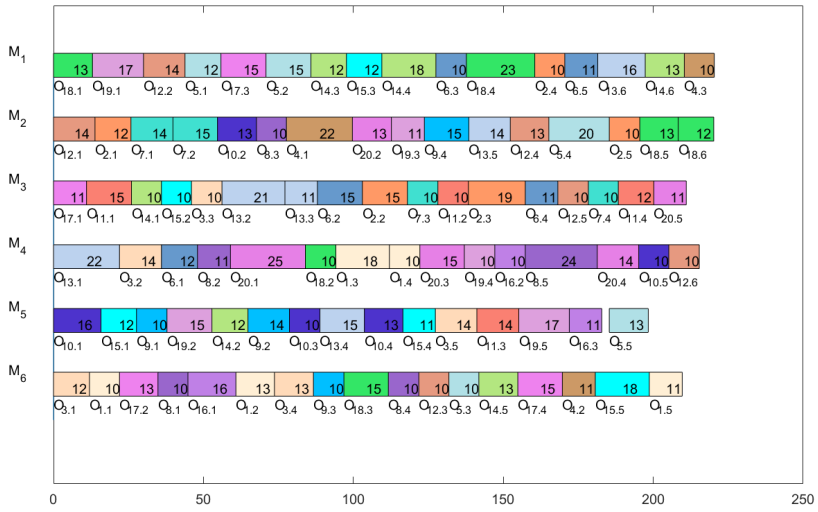

图 2.14 LRVMA 在 J20M6 求得 Cmax 最好解的甘特图

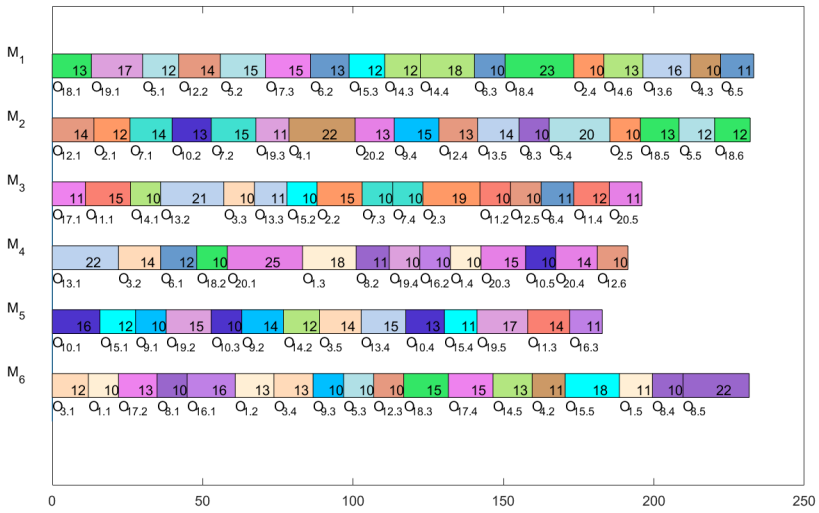

图 2.15 LRVMA 在 J20M6 求得 TEC 最好解的甘特图

2.6 本节小结

本章根据绿色柔性作业车间调度问题的特点，提出了基于学习和参考向量的模因算法，并在文献[70]公开的 30 个测试问题上进行测试，同时与一些最先进的算法比较。大量实验结果表明，本章所提的 LRVMA 算法在求解 GFJSP 上取得了最好的结果。本章的贡献主要有以下几点：(1) 提出了混合启发式初始化策略使得算法获得高质量的初始化种群加速收敛；(2) 根据模型特性提取知识，设计多种局部搜索算子，增强局部搜索的成功率；(3) 提出了节能策略，有效降低车间的能耗优化能耗目标；(4) 提出了基于学习和指标反馈的参数自学习模型，帮助算法依据环境调整参数，使得种群向分布性增加的方向进化。GFJSP 是同构 GDFJSP 研究的基础，本章提出的 LRVMA 算法为后续章节的算法设计提供算法支持。

第三章 绿色分布式同构柔性作业车间调度问题求解

3.1 引言

随着全球化贸易的推进，现代企业多采取多工厂并行制造的模式，使得多工厂之间协同排产变得更加困难。因此，在第二章的基础上，本章考虑同构分布式GFJSP即GDFJSP问题。同构GDFJSP在GFJSP的基础上增加了工件工厂选择的子问题。三个耦合的子问题使得同构 GDFJSP 更难求解。因此，本章提出了基于意外流行算法的模因算法（Surprisingly popular-based adaptive memetic algorithm，SPAMA）求解GDFJSP包含以下改进：（1）对模型进行分析，设计了四种基于关键路径的局部搜索算子；（2）提出基于意外流行算法的算子选择模型，发现小权重但高效的算子修正算子选择决策；（3）提出了基于全调度解码节能策略，通过提前和延迟加工部分工序减少空闲时间以减少能耗。最后，在著名的 Mk[189]和 DP[190]测试集共20个测试问题上验证了所提算法的有效性。

3.2 基于 Pareto 支配的多目标优化框架

本章提出的SPAMA以NSGA-II[183]算法MA框架中的全局搜索算子。NSGA-II作为MOPs中另一大主流框架，在介绍SPAMA之前先对基于支配序的多目标优化框架进行介绍。

3.2.1 支配序关系

假设一个多目标优化问题$f(x)=\min(f_1(x),f_2(x),\ldots,f_n(x))$，在决策空间$X$中存在两个解$x_1$和$x_2$。如果，$\forall i\in\{1,\ldots,n\}, f_i(x_1)\leq f_i(x_2)$且$\exists\, j\in\{1,\ldots,n\}, f_j(x_1)< f_j(x_2)$，则说明$x_1$支配$x_2$记为$x_1\prec x_2$。理解为所有目标值中$x_1$至少有一个比$x_2$好。如果，$\exists i\in\{1,\ldots,n\}$，$f_i(x_1)>f_i(x_2)$且$\exists\, j\in\{1,\ldots,n\}, f_j(x_1)< f_j(x_2)$，则说明$x_1$和$x_2$互不支配记为$x_2\sim x_1$。

3.2.2 快速非支配排序算法

快速非支配排序算法由K.Deb在2002年提出，是NSGA-II算法的核心。采用

该算法可以迅速将种群中的解根据目标函数的支配序关系分层。算法 3-1 详细描述了快速非支配排序的过程。首先，找到种群中第一层前沿面并统计每个个体支配其他个体的集合 S_p 和被支配数 n_p。其次，将第一层前沿面的个体从种群中去除，且所有剩余个体的被支配数减一，再找出被支配数为零的个体加入第二层的前沿面。循环上述过程直到没有个体可以加入下一层的前沿面。

算法3 - 1: 快速非支配排序算法.

输入: 种群P

输出: 非支配前沿面集合$\{F_1, F_2, ..., F_n\}$

```
foreach p ∈ P do
    S_p = ∅//被个体p支配个体的集合
    n_p = 0//个体p被其他个体支配的数量
    foreach q ∈ P do
        if p ≺ q then
            S_p = S_p ⋃ q
        else
            n_p = n_p + 1
        end
    end
    if n_p == 0 then
        p_rank = 1
        F_1 = F_1 ⋃ p //将个体p并入第一层前沿面
    end
end
while F_i ≠ ∅ do
    Q = ∅
    foreach p ∈ P do
        foreach q ∈ S_p do
            n_q = n_q + 1
            if n_q == 0 then
                q_rank = i + 1
                Q = Q ⋃ q //将个体p并入第一层前沿面
            end
        end
    end
    i=i+1
    F_i = Q
end
```

3.2.3　拥挤度距离

在经过快速非支配排序算法后，种群的收敛性得到了保证。但如何保证种群的分布性是另一个关键问题。因此，K.Deb 提出拥挤度距离在最后一层设计了拥挤度距离衡量每一层相邻解之间的拥挤度。并根据拥挤度排序，选择稀疏的个体保留。拥挤度距离公式如下：

$$d_i=\begin{cases}\sum_{j=1}^{n}|f_j(i+1)-f_i(i-1)|/(f_i^{\max}-f_i^{\min}), i\in\{2,...,|P|-1\}\\ \infty, i=1\text{或}\ |P|,\quad \text{即两端点解}\end{cases} \tag{6-1}$$

其中 $f_i^{\max}$ 和 $f_i^{\min}$ 表示当前层中目标函数的极值，P 表示当前层中所有个体。由于两端的点没有前后邻域解，因此拥挤度为无穷大。d_i 越大说明越稀疏则越应该保留。

3.2.4　NSGA-II 算法

图 3.1 详细地展示了 NSGA-II 的算法流程。首先，父代种群 P_t 通过双人竞标赛算法选择交配池的精英个体。其次，父代种群从交配池中随机选择个体进行杂交生成子种群 Q_t。随后合并 P_t 和 Q_t 成为 R_t。再次，对合并种群执行快速非支配排序算法，同时每一层中计算拥挤度并根据拥挤度从小到大再次排序。最后，选择前$|P_t|$个个体保留进入下一代种群，即完成了一轮进化。

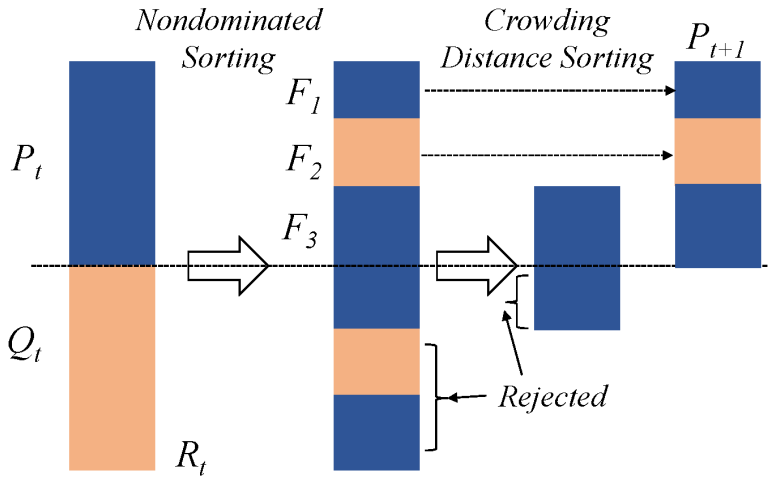

图 3.1 NSGA-II 算法环境选择流程

3.3 意外流行算法

在介绍 GDFJSP 之前，需要介绍意外流行算法（Surprisingly popular algorithm,

SPA）的原理为后面算法设计打下基础。SPA 由 Prelec[191]于 2017 年发表在 Nature 上，目的是修正群体决策中由于主观意识和经验主义造成的错误性知识和决策。如图 3.2 所示，Prelec 通过两个份在线问卷调查询问人们“费城是否为宾夕法尼亚洲的首府？并判断你的答案在人群中占比多少。”由于在现实世界中大多数人并未真正去过宾夕法尼亚洲，缺乏相应的地理知识(专家知识)，而凭借自己的意识和经验判断“费城是最著名的城市，那么肯定是宾夕法尼亚洲的首府。”而现实答案却是“否”。通过调查，有 80%的人认为答案是“是”，10%的人弃权，而 10%的人认为答案是“否”，并给出了对应答案的估值分别是预计有 90%的人答案为“是”，18%的人弃权和 20%的人可能知道答案是“否”。经过贝叶斯概率公式修正现实世界中每个答案真正的得票率。答案“是”的真实概率为：80%*90%+10%*(1-18%)+10%*(1-20%)=88.2%。答案“否”的真实概率为：80%*10%+10%*18%+10%*20%=11.8%。计算答案“是”的意外流行度为 80/88.2＝0.907。计算答案“否”的意外流行度为 20/11.8＝1.695。可见答案“否”给人带来了更多的意外，而答案“是”也名不副实。恰巧发现一个规律，群体中意外流行度大于 1 的答案更加接近真实答案。随后，Prelec 又在其他城市做了类似的问卷调查，结果表明该群体决策中真实答案复合上述规律。

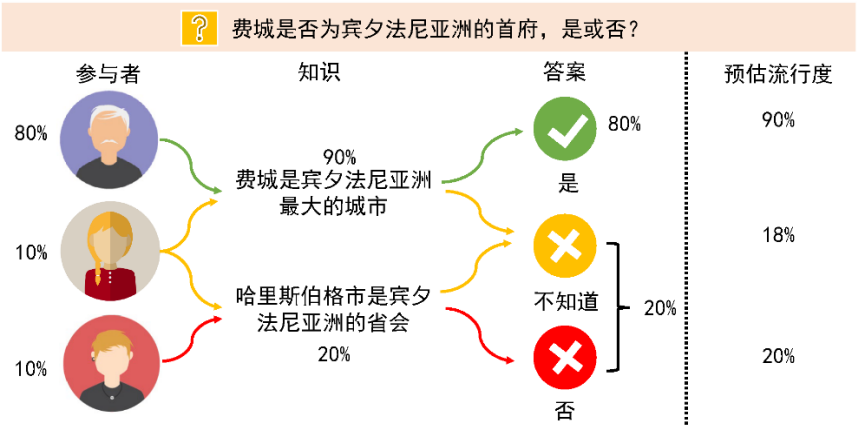

图 3.2 意外流行算法举例

动机：那么 SPA 给本节算法设计带来的思考是，类比进化算法中整个种群对着许多决策问题，如何在一个种群的决策中发现真实的最合适的答案是关键问题。如果将 SPA 于进化计算结合起来？在车间调度问题中，有许多局部搜索算子，传统的基于置信度的算子选择模型[110]会根据不同算子的成功率分配选择概率。而在该模型中就会存在如下问题：由于历史的记录存在记忆体矩阵中，历史的选择答案对于当前的进化代不一定适用，也许存在那些占比小的正确答案有待发现。因此，本章采用 SPA 改进基于置信度的算子选择模型，在每一轮进化过程中都会用 SPA

算法去询问每个局部搜索算子是有效或无效。去发现那些有效但占比小的局部搜索算子，并增强其分配概率，使得高效的算子被更多的使用，加速算法收敛。

3.4 同构 GDFJSP 描述与 MILP 模型

3.4.1 同构 GDFJSP 描述

在同构绿色分布式柔性作业车间调度问题中有工件集合 $I=\{1,2,\ldots,i,\ldots,n\}$，每个工件有工序集合 $J_i=\{O_{i,1},O_{i,2},\ldots,O_{i,j},\ldots,O_{i,ni}\}$，有工厂集合 $F=\{1,2,\ldots,f,\ldots,n_f\}$，在工厂中有机器集合 $M=\{1,2,\ldots,k,\ldots m\}$。每个工序可以被 M 的子集 $K_{i,j}$ 加工。每个机器可以加工多种工件的工序。每个工序在不同工厂的相同机器上的加工时间 $T_{i,j,k}$ 是相同的。在工件分配了工厂后，所有工件必须都在该工厂中加工。同构 GDFJSP 主要解决 3 个耦合的子问题：确定每个工件的工厂选择；确定每个工厂中所有工序的加工顺序；为每个工序分配加工机床；使得所有工厂的总完工时间和总能耗最小。关于同构 GDFJSP 的问题假设如下：（1）所有工序和机器在零时刻都是可用的；（2）所有机器在同一时刻只能加工一个工序且无法被抢占；（3）所有加工时间和能耗相关的数据都是确定的；（4）一个工件同一时间只能被一个机器加工；（5）运输时间，启动时间以及相关能耗不考虑。（6）一个工件只能分配到一个工厂，且所有工序都要在该工厂中加工。

3.4.2 同构 GDFJSP 的 MILP 模型

在介绍 GDFJSP 的模型之前，首先要介绍 MILP 模型中的变量如下：

索引变量：

i,i': 工件索引号 $i\in\{1,2,...,n\}$;

j,j': 工序索引号 $j\in\{1,2,...,n_i\}$;

k,k': 机器索引号 $k\in\{1,2,...,m\}$;

t,t': 机器上加工位置索引号 $t\in\{1,2,..., p_{f,k}\}$;

f: 机器上加工位置索引号 $f\in\{1,2,..., n_f\}$;

参数：

n: 总工件个数；

m: 总机器个数；

n_f: 总工厂数；

n_i: 工件 i 的工序个数；

$p_{f,k}$: 每个机器上加工位置数；

I: 工件集合 I={1,2,...,n}；

J_i: 工件 i 的工序集合 J_i={$O_{i,1}$,$O_{i,2}$,…,$O_{i,j}$,…,$O_{i,ni}$}；

M: 机器集合 M={1,2,...,m}；

$K_{i,j}$: 工序 $O_{i,j}$ 在可选机器集合在所有工厂相同;

$P_{f,k}$: 机器 k 在工厂 f 的加工位置集合 P_k ={1,2,..., $p_{f,k}$ }；

$P_{f,k}'$: 机器 k 在工厂 f 的前 p_k -1 个加工位置集合 P_k ={1,2,...,$p_{f,k}$-1}；

$T_{i,j,k}$: 工序 $O_{i,j}$ 在机器 k 的加工时间，在所有工厂相同;

W_O: 机器加工功率;

W_I: 机器空闲功率;

L: 充分大的整数保证不等式的一致性;

$x_{i,j,k}$: 0-1 变量，表明工序 $O_{i,j}$ 能否在机器 k 上加工，如果是值为 1 否则为 0；

决策变量：

$S_{f,i,j}$: 工序 $O_{i,j}$ 在工厂 f 开始加工时间；

$C_{f,i,j}$: 工序 $O_{i,j}$ 在工厂 f 完成加工时间；

$B_{f,k,t}$: 在工厂 f 机器 k 第 t 个位置开始加工时间；

E_I: 机器加工总能耗;

$X_{i,j,f,k,t}$: 0-1 变量，表明工序 $O_{i,j}$ 在在工厂 f 的机器 k 的第 t 个位置上加工，如果是值为 1 否则为 0；

$Y_{i,f}$: 0-1 变量，工件 i 被分配到工厂 f 加工，如果是值为 1 否则为 0；

同构 GDFJSP 的两个目标函数描述如下：

(1)最大完工时间通常被认为是车间调度问题的经济指标，车间内最大完工时间越小带来的企业受益越大。最大完工时间的优化目标的定义如下：

$$F_1 = C_{\max} = \max\{C_{f,i,n_i}\}, \forall i \in I, f \in F, \tag{6-2}$$

(2)总的能量消耗通常被认为是关键的环境指标，总能耗 TEC 反应企业在系统层面二氧化碳排放量。能耗目标主要由两个决策表达式组成分别是：机器的总加工能耗 E_W 和机器的总空闲能耗 E_I。总能耗的优化目标的定义如下：

$$F_2 = TEC = E_W + E_I, \tag{6-3}$$

$$E_W = \sum_{i\in I}\sum_{j\in J_i}\sum_{f\in F}\sum_{k\in M}\sum_{t\in P_k} W_O \times T_{i,j,k} \times X_{i,j,f,k,t}, \tag{6-4}$$

$$E_I = \sum_{f\in F}\sum_{k\in M}\sum_{t\in P_k'} W_I \times (B_{f,k,t+1} - B_{f,k,t} - \sum_{i\in I}\sum_{j\in J_i} X_{i,j,f,k,t} \times T_{i,j,k}), \tag{6-5}$$

GDFJSP 的约束条件如下：

$$\sum_{f\in F} Y_{i,f} = 1, \forall i \in I, \tag{6-6}$$

$$\sum_{k\in K_{i,j}} \sum_{t\in P_k} X_{i,j,f,k,t} = Y_{i,f}, \forall i \in I, j \in J_i, f \in F, \tag{6-7}$$

$$S_{f,i,n_i} + \sum_{k\in K_{i,j}} \sum_{t\in P_k} T_{i,n_i,k} \times X_{i,n_i,f,k,t} \le C_{\max}, \forall i \in I, f \in F, \tag{6-8}$$

$$C_{f,\max} \le C_{\max}, \forall f \in F, \tag{6-9}$$

$$S_{f,i,j} + \sum_{k\in M} \sum_{t\in P_k} T_{i,j,k} \times X_{i,n_i,f,k,t} \le S_{f,i,j+1}, i \in I, j \in J_i - 1, f \in F, \tag{6-10}$$

$$\sum_{i\in I} \sum_{j\in J_i} X_{i,j,f,k,t} \le 1, \forall f \in F, k \in M, t \in P_{f,k}, \tag{6-11}$$

$$\sum_{i\in I} \sum_{j\in J_i} X_{i,j,f,k,t} \ge \sum_{i\in I} \sum_{j\in J_i} X_{i,j,f,k,t+1}, \forall f \in F, k \in M, t \in P'_{f,k}, \tag{6-12}$$

$$B_{f,k,t+1} - B_{f,k,t} \ge \sum_{i\in I} \sum_{j\in J_i} X_{i,j,f,k,t} \times T_{i,j,k}, \forall f \in F, k \in M, t \in P'_{f,k}, \tag{6-13}$$

$$E_{I\,f,k,t} \ge (B_{f,k,t+1} - B_{f,k,t} - \sum_{i\in I} \sum_{j\in J_i} X_{i,j,f,k,t} \times T_{i,j,k}) \times W_I, \forall f \in F, k \in M, t \in P'_{f,k}, \tag{6-14}$$

$$B_{f,k,t} \ge S_{f,i,j} - L \times (1 - X_{i,j,f,k,t}), \forall i \in I, j \in J_i, f \in F, k \in M, t \in P_{f,k}, \tag{6-15}$$

$$B_{f,k,t} \le S_{f,i,j} - L \times (1 - X_{i,j,f,k,t}), \forall i \in I, j \in J_i, f \in F, k \in M, t \in P_{f,k}, \tag{6-16}$$

$$0 \le B_{f,k,t}, S_{f,i,j} \le L, \forall i \in I, j \in J_i, f \in F, k \in M, t \in P_{f,k}, \tag{6-17}$$

约束(3-6)确保一个工件只能分配到一个工厂加工，约束(3-7)确保工件分配到工厂 f 后所有工序都在该工厂加工，约束(3-8)表示优化目标最大完工时间与工序完工时间的关系，约束(3-9)表示每个工厂最大完工时间和总最大完工时间的关系，约束(3-10)表示每个工序的开始加工前必须等上一个工序完工，约束(3-11)确保同一时刻每个机器的每个加工位置只加工一个工序，约束(3-12)确保每个机器上必须等前一个位置的工序完工才能加工下个位置工序，约束(3-13)表示每个机器上开始加工时间和完工时间的关系，约束(3-14)表示机器空闲能耗的计算方式，约束(3-15)和约束(3-16)是为了便于 MILP 模型采用分支界定算法搜寻解而对等式约束进行松弛，表示机器开始加工时间和工序的开始加工时间是相同的，约束(3-17)表示决策变量的取值范围。

3.5 基于 SPAMA 的同构 GDFJSP 求解

本章提出了基于意外流行算法的模因算法，以 NSGA-II 算法为全局搜索算子，配合基于关键路径和关键块的局部搜索算子加速算法收敛。更多地，提出基于意外流行算分的算子选择模型来发现真正有效的但小权重的局部搜索算子，增强其选择概率，加速算法收敛。最后，针对能耗目标，提出基于全调度解码的节能策略，有效减少空间时间从而降低能耗。

3.5.1 编解码机制

由于同构 GDFJSP 有更复杂的目标空间，因此为了覆盖整个解空间范围，本章在第二章的基础上引入工厂码向量，采用三层编码模式描述 GFJSP 的解分别是：工序顺序（Operation sequencing，OS），机器选择（Machine selection，MS）和工厂选择（Factory assignment，FA）。OS 和 MS 的长度和总工序数相等，FA 的长度和总工件数相同。图 3.3 给出了编码模式的例子，图中包含 3 个工件和 3 台机器。图中的数字表示工件号而数字出现的次数表示该工件第几个工序。图中的加工顺序为 $O_{11}\rightarrow O_{31}\rightarrow O_{21}\rightarrow O_{32}\rightarrow O_{12}\rightarrow O_{13}\rightarrow O_{22}$。工件 I_1 和 I_3 选择了工厂 F_1，工件 I_2 工厂 F_2。FA 每个位置对应的工件索引是不变的同时 MS 向量中每个位置机器选择对应的工序是一直不变的，保证不会产生不可行解。

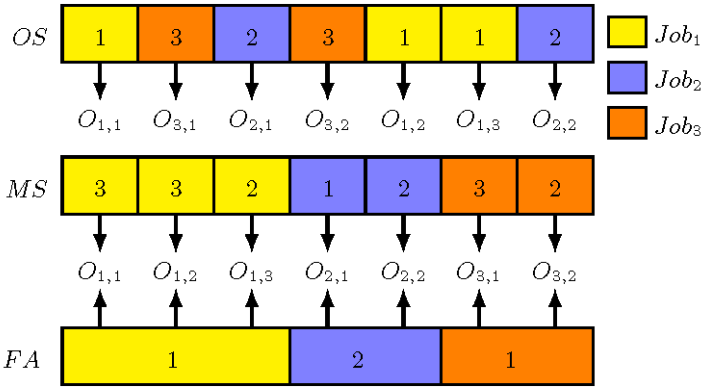

图 3.3 GFJSP 编码模式

解码模式：首先，从 FA 获取每个工厂分配的工件集合。其次，从左到右遍历向量 OS 获取加工顺序，并把每个工序分配到相应的工厂中。随后从左到右遍历 MS 向量获取每个工序的机器选择，把每个工序安排在对应工厂的机器上加工。计算每个工厂中所有工序的开始时间和完工时间，计算每个工厂中所有加工位置的等待时间，保证开工时间服从 MILP 模型中的约束条件。最后，比较每个工厂的最大完

工时间，取最大的为总最大完工时间。同时累加每个工厂的总能耗为所有工厂的总能耗。

3.5.2 种群初始化

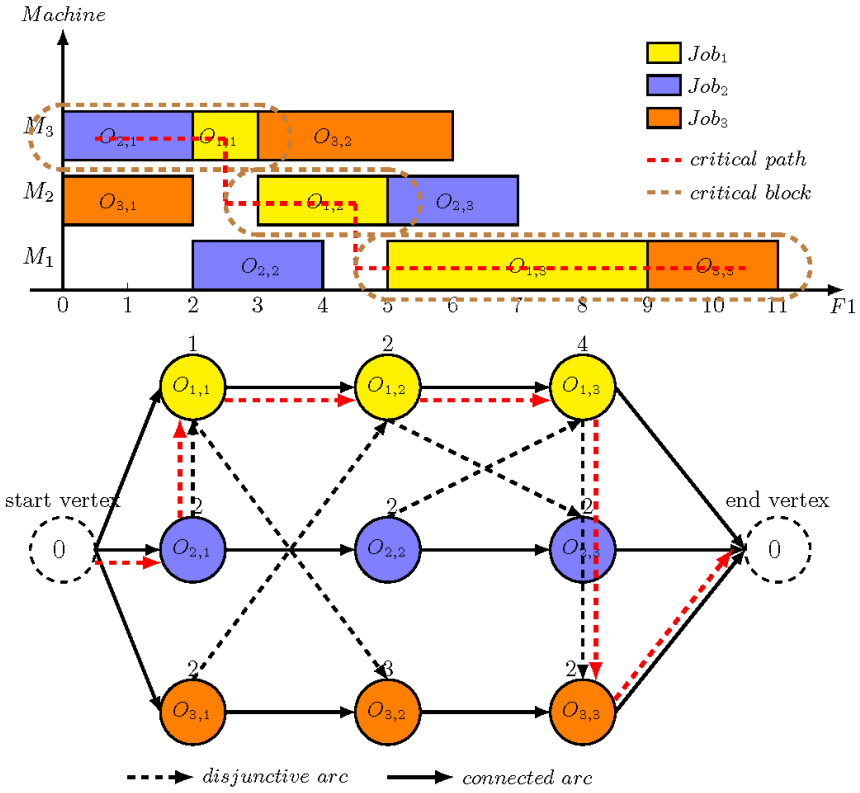

图 3.4 FJSP 的析取图编码方式

分布式车间调度问题目标空间非常复杂，为了使得算法获得良好的分布性，重复探索复杂的目标空间以便后续局部搜索获得多样性的解，本章采用随机初始化的方法。其步骤如下：

Step1：生成从 $O_{11}\rightarrow O_{1\ n1}\ldots\rightarrow O_{n\ nn}$ 的序列，随机打乱加工序列生成第一个 OS 向量 OS_1，随后在 OS_1 的基础上再随机打乱顺序生成 OS_2 ，重复以上操作每次生成当前 OS 向量都是打乱上一次的 OS 向量的顺序，直到生成 ps 个 OS 向量。

Step2：针对 *ps* 个 OS 向量，为每个工序从其备选机器集合 K_{ijf} 中随机选择加工机床生成一个 MS 向量，重复 ps 次直到生成 *ps* 个 MS 向量。

Step3：生成从{1…n}的序列，随机打乱加工序列生成第一个中间向量 T_1，同时从左到右扫描基因，每个基因对 n_f 取余，从而确定每个工件的工厂选择生成 FA_1 向量。随后在 T_1 的基础上再随机打乱顺序生成 T_2 ，同样地，取余生成 FA_2。重复以上操作，直到生成 *ps* 个 FA 向量。

3.5.3 进化算子

通过大量实验表明针对分布式车间调度问题第二章介绍的POX算子可以有效地探索目标空间。因此，本章仍采用 POX 对 OS 进行交叉。同时采用 UX 对 MS 和 FA 进行交叉。由于 MS 和 FA 从左到右对应的工序和工件不变，所以可以避免产生不可行解。突变算子与第二章相同。

算法3 - 2: 搜索FJSP中的关键路径.

输入: 一个FJSP调度序列$\prod$

输出: 关键路径π, 关键块CB

给$\prod$建立析取图数据结构DG(V, E, W).
$level \leftarrow 0$, $l \leftarrow 0$
$EST(SV) \leftarrow 0$.
while $l<length(OS)$ do
  $level \leftarrow level + 1$.
  for $i=1$ *to* n do
    $t_1 \leftarrow EST(LO) + P^{O}_{i,LO,k}$.
    if $EST(LAO)$ 目前没有被计算 then
      $EST(LAO) \leftarrow$递归获取上一个工序的EST.
    end
    $t_2 \leftarrow EST(LAO) + P^{O}_{i,LAO,k}$.
    $EST(O_{i,level}) \leftarrow \max\{t_1, t_2\}$
    $l \leftarrow l + 1$.
  end
end
$LST(EV) \leftarrow EST(EV)$.
while $l>0$ do
  $level \leftarrow level - 1$.
  for $i=1$ *to* n do
    $t1 \leftarrow LST(NO) - P^{O}_{i,level,k}$.
    if $LST(NAO)$ 目前没有被计算 then
      $LST(NAO) \leftarrow$递归获取下一个工序的LST.
    end
    $t_2 \leftarrow LST(NAO) - P^{O}_{i,level,k}$.
    $LST(O_{i,level}) \leftarrow \min\{t_1, t_2\}$
    $l \leftarrow l - 1$.
  end
end
for $i=$ *1 to length(OS* do
  if $LST(i)==EST(i)$ then
    $\pi = \pi \cup i$
  end
end
CB={}. k=0
CurrentM=$MS(\pi(1))$. $CB(k) \leftarrow CB(k) \cup \pi(1)$
for $i = 2$ *to length*(π) do
  if $MS(\pi(1))==CurrentM$ then
    $CB(k) \leftarrow CB(k) \cup \pi(i)$.
  end
  else
    k=k+1.
    $CB(k) \leftarrow CB(k) \cup \pi(i)$.
    CurrentM=$MS(\pi(i))$;
  end
end

3.5.4　基于关键路径的局部搜索算子

通过对 Cmax 更深入的分析，如图 3.4 所示将一个调度序列转换为析取图，确定析取图里的关键路径，反映为最直接影响 Cmax 的所有工序。通过，对关键路径上关键工序的扰动可以直接减少 Cmax。因此，本章针对关键路径设计了四种局部搜索算子。

(1)求解关键路径和关键块：假设析取图为 DG（V，CA，DA，W），V 表示点集合即所有工序，CA 表示直接连接边，即图 3.4 中的实线连接同一工件的相邻工序，DA 表示析取边即图 3.4 中的虚线连接相同机器上相邻位置的工序。W 表示每个工序的加工时间即权重。将一个调度序列构建成析取图后，下一步为找到 DG 中的关键路径。算法 3-2 描述如下：

首先，通过双向链表数据结构构建析取图，在 DG 中每个节点有两个出度和两个入度即相邻的工序分别为：同台机器上的相邻下一个/上一个工（Next/last adjacent operation，NAO/LAO），当前工件的下/上一个工序（Next/last operation，NO/LO）。在析取图中的所有工件都必须从开始节点（Start vertex，SV）开始同时从结束节点完结（End vertex，EV），这两个节点在图中都是虚拟节点。随后，从左到右扫描 OS 向量，把每个工件的第一个工序为第一层，每个工件的第 n 个工序作为第 n 层。采用层序遍历算法遍历所有节点，计算每个节点的最早开始时间（Earliest start time，EST）。通过比较 LAO 节点和 LO 节点的 EST 加上自身的权重，即加工时间。最大的作为当前节点的 EST。在从左到右遍历完 OS 后，开始从右到左层序遍历 OS 计算每个节点的最晚开始时间（Lastest start time，LST），通过比较当前节点相邻的 NO 和 NAO 节点的 LST 减去当前节点的权重，最小的即为当前节点的 LST。如果统计完所有节点，发现部分节点的 EST 和 LST 相等，则说明该节点在关键路径上。将关键路径上在同一机器上的相邻节点划入相同的关键块中，即图 3.4 中打圈的工序组成一个关键块。

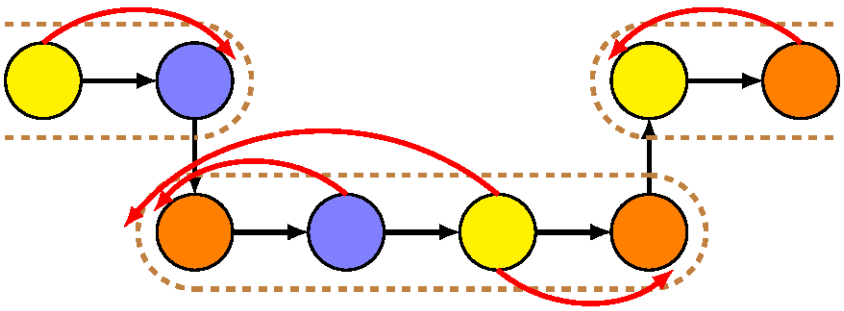

图 3.5 N6 邻域结构

(2)局部搜索算子 1 (N6 邻域结构)：在求解了关键路径和关键块后，根据前人的研究表明，在关键块中设置扰动可以有效的减少空闲时间，进而减少完工时间。而 N6 是传统作业车间调度问题中的邻域结构。经过实验测试在 FJSP 中也有良好的效果。图 3.5 详细解释了 N6 邻域结构。首先，将第一个、中间、最后一个关键块成为头块、中间块和尾块。对于头块，随机选取中间的工序插入到尾工序后面。对于中间块，随机选取两个中间工序分别插入到头工序前和尾工序后。对于尾块，随机选择一个工序插入到头工序前。

(3)局部搜索算子 2 (双点交换)：这是一个很简单的邻域结构，随机选择两个关键路径上的工序并交换其在工厂中的加工顺序，以减少完工时间。

(4)局部搜索算子 3 (随机重分配工厂)：随机选择关键路径上的工序将其所属的工件所有的工序随机移到另一个工厂。

(5)局部搜索算子 4 (随机重选机床)：随机选择关键路径上的工序，在其可选的机器范围内随机重选一个机床加工。

3.5.5 基于意外流行算法的算子选择策略

虽然设计了多种基于关键路径的局部搜索算子，但不同算子在不同阶段的表现出来的成功率不同，所以最有效的算子也不同。如何有效的组织好各类功能的算子，使得算法能自主的在不同阶段选择最好的算子成为问题的关键。本节基于之前介绍的意外流行算法，提出了基于意外流行算法的算子选择模型，该模型的优点在于能在每次进化阶段修正种群的算子选择概率，发现种群中真正有效的但由于权重较小不被重用的算子。

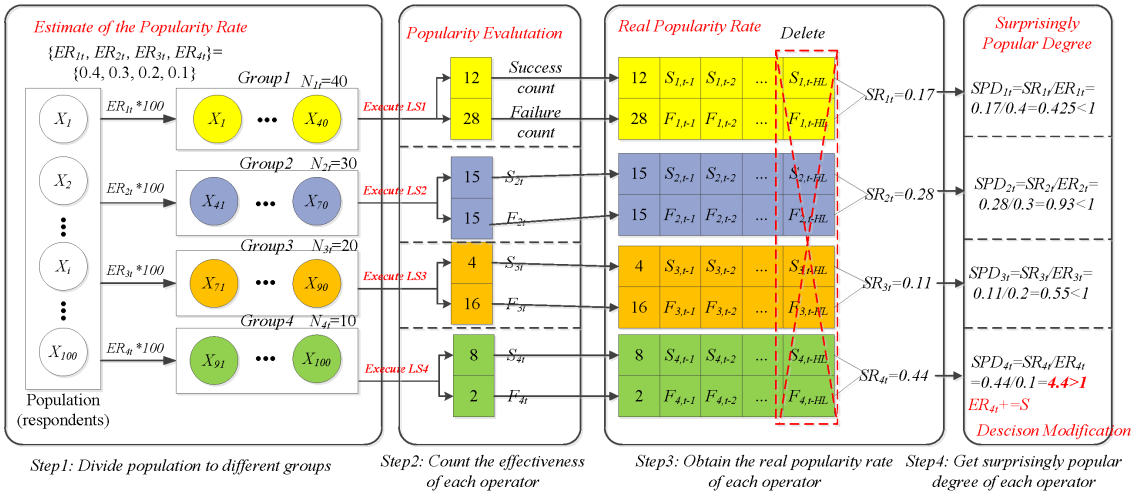

图 3.6 基于意外流行算法的算子选择模型

算法在模型中设计了两种概率来表示 SPA 中主流群众和专家群众的差别。用

预期成功率（Estimated success rate，ER_i）来表示主流人群对每个算子的成功率的期望。用实际成功率（Success rate，SR_i）来表示实际情况中，默认经过贝叶斯概率修正后的实际真实成功率。通过 SR 和 ER 的比值来计算每个算子的意外流行度。

算法3 - 3: 基于SPA的算子选择模型.

输入: 成功矩阵(NS), 失败矩阵(NF), 奖励概率(S), 预估流行度(ER), 实际流行度$(SR$, 成果次数记录SC_t, 失败次数记录$FC_t)$

输出: 算子选择概率NR

把SC_t加入NS_t的尾部，把FC_t加入NF_t的尾部.
for j= *1 to* 算子个数 do
  $SS = \sum_{i=1}^{HL} NS_t(i,j)$. $SF = \sum_{i=1}^{HL} NF_t(i,j)$.
  if $SS + SF == 0$ then
    $NR_t(i) = 0.01$.
  else
    $NR_t(i) = SS/((SS + SF) + 0.01)$.
  end
end
删掉NS_t和NF_t的第一行.
归一化NR_t使得$\sum NR_t == 1$，同时如果$NR_t(i) < 0.1$，则$NR_t(i) = 0.1$. 随后继续归一化NR_t使得$\sum NR_t == 1$.
$SR_t = NR_t$.
通过公式(3-19)计算SPD_t.
for j= *1 to* 算子个数 do
  if $SPD_t(j) > 1$ then
    $NR_t(j) = NR_t(j) + S$.
  end
end
$ER_{t+1} = SR_t$.
归一化NR_t使得$\sum NR_t == 1$.

图 3.6 展示了意外流行算法是如何运行的。假设种群中有 100 个个体，根据上一轮的真实成功率划分算子的分配权重，在模型中认为每一轮迭代都是新一轮的群体，那么经过历史成功经验的验证，上一轮的 SR_{t-1} 成为了本轮的预期成功率 ER_t。总共 4 个算子，按照 ER 分成了 4 个子种群分别执行每种局部搜索算子。在每个子种群中统计每个算子的成功和失败替换父代解的次数 $S_{i\,t}$ 和 $F_{i\,t}$, i=1，2，3，4。将每个算子的成功和失败的记录存入记忆体的第一行，同时删掉最后一行。随后，根据每一个算子历史的总成功次数和总失败次数更新其本轮迭代真实成功率 SR_t。计算公式如下：

$$SR_{i,t} = \frac{\sum_{k=0}^{HL} S_{i,t-k}}{\sum_{k=0}^{HL} S_{i,t-k} + \sum_{k=0}^{HL} F_{i,t-k}}, i = 1,2,3,4, \tag{6-18}$$

随后计算每个算子的意外流行度（Surprisingly popular degree，SPD），计算公式如下：

$$SPD_{i,t} = \frac{SR_{i,t}}{ER_{i,t}}, i = 1,2,3,4, \tag{6-19}$$

最后发现局部搜索算子 4 的意外流行度大于 1，在概率上增加 S=0.15 来增强

其在后续进化中的被选择概率，最后再对每个算子的选择概率归一化，并进入下一轮进化。算法 3-3 详细的讲述了基于 SPA 的算子选择模型的过程。

3.5.6 基于全主动调度解码的节能策略

在第二章设计的贪心节能策略主要依靠重构解来降低能耗，这样的策略有很大概率获得更差的 Cmax，使得策略低效。因此，本章在不改变 Cmax 的前提下，采用全主动调度解码减少完工时间降低能耗。在介绍节能策略之前先介绍主动调度解码的概念。半主动调度解码（Semi-active schedule，SAS）：如果不提前加工某些工序，则无法改变工序加工顺序。主动调度解码（Active schedule，AS）：如果不延迟加工某些工序，则无法改变工序加工顺序。全主动调度解码（Full active schedule，FAS）：无法通过提前加工和延后加工工序改变加工顺序。SAS 可以通过提前加工部分工序转换为 AS，AS 可以通过延迟加工部分工序转换为 FAS。因此，把调度序列转换为 FAS 可以缩小搜索空间，加速收敛。

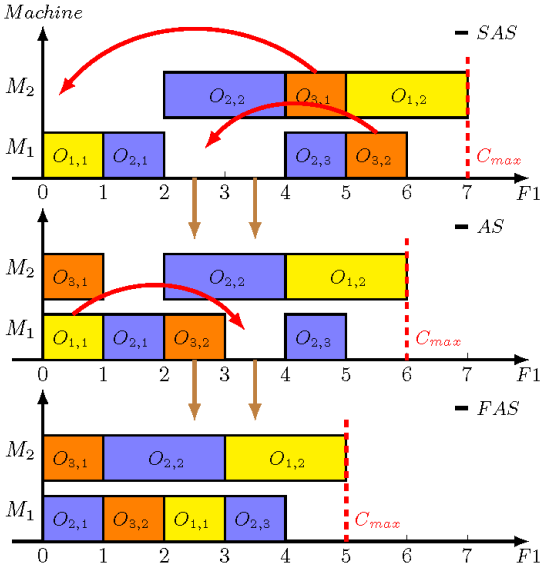

图 3.7 基于全主动调度解码的节能策略

图 3.7 展示了节能策略的过程。首先，针对当前工序在同机器上寻找可以容纳当前工序的空闲间隔，如果存在则提前加工当前工序并插入到间隔中以减少空闲时间，重复以上过程直到 SAS 转换为 AS。其次，对 AS 工序进行反向解码，从后向前遍历，针对当前工序，在同机器上向后寻找是否存在足够的空闲间隔能容纳当前工序，如果有则将当前工序延迟加工并插入到空隙中，重复以上过程直到 AS 变为 FAS。在经过工序顺序的改变后，图 3.7 中的空闲时间大幅度减少，达到降低能耗的目的。

3.5.7 SPAMA 算法流程

算法 3-4 详细说明了 SPAMA 的流程。首先，初始化种群和相关参数变量。其次，采用锦标赛算法选择交配池。随后用进化算子产生子代，并合并子代和父代种群，同时删除其中的重复解。随后，用快速非支配排序和拥挤度距离算法对合并种群排序，选择前 ps 个解作为下一代的父代种群。然后，对父代种群执行节能策略减少空闲时间且降低能耗。更多地，将父种群中非支配解集存入精英存档中，根据历史记录中不同算子的表现情况为算子分配选择概率，并采用轮盘赌选择为不同的精英解分配局部搜索算子，执行完局部搜索算子后将精英解都存入存档中。最后，采用 SPA 算子选择模型修正每个算子的选择概率。重复 4-25 行直到满足停止条件。

算法3 - 4: SPAMA的算法框架.

输入: 最大函数评价次数($MaxNFEs$), 种群大小(ps), 历史记忆体的长度(HL), 概率奖励(S), 交叉概率(P_c), 突变概率(P_m)
输出: 非支配解集PF

1 $\mathcal{P}_0 \leftarrow$ 初始化种群(ps).
2 初始化精英存档Ω, 成功和失败记忆体NS 和NF, 预估流行度ER, 和真实流行度SR.
3 $t = 1$.
4 while $NFEs \leqslant MaxNFEs$ do
5 $Q_t \leftarrow$双人锦标赛选择 ($\mathcal{P}_t$).
6 for i=1 to ps do
7 $[X_1, X_2] \leftarrow$ 进化算子 ($\mathcal{P}_t(i), Q_t, P_c, P_m$).
8 $C_t \leftarrow C_t \cup X_1 \cup X_2$.
9 $NFEs \leftarrow NFEs + 2$.
10 end
11 $C_t \leftarrow$ 删除重复解 ($C_t \cup \mathcal{P}_t$).
12 $\{F_0, ..., F_{last}, ...\} \leftarrow$ 快速非支配排序 (C_t, ps).
13 $C_t^{'} \leftarrow$计算拥挤度距离 ($F_0, ..., F_{last}$).
14 $\mathcal{P}_{t+1} \leftarrow C_t^{'}(1 : ps, :)$.
15 $\Omega \leftarrow$ 删除重复解 ($\Omega \cup F_0$).
16 $\mathcal{P}_{t+1} \leftarrow$节能策略 ($\mathcal{P}_{t+1}$).
17 $NFEs \leftarrow NFEs + ps$.
18 $StPool_t \leftarrow$轮盘赌选择 ($NR_t, |\Omega| * numSt$).
19 $[\Omega, SC_t, FC_t] \leftarrow$局部搜索 ($\Omega, StPool_t$).
20 $NFEs \leftarrow NFEs + |\Omega| * numSt$.
21 if $iter > HL$ then
22 $NR_t \leftarrow$基于SPA算子选择($NS_t, NF_t, S, ER_t, SR_t, SC_t, FC_t$).
23 end
24 $\Omega \leftarrow$ 快速非支配排序 (Ω).
25 $t = t + 1$.
26 end
27 $PF \leftarrow$ 快速非支配排序 (Ω).

3.6 数值实验

3.6.1 实验设置

在上节中已经详细介绍了算法 SPAMA 的细节，本节设计了详细的实验来验证所提出算法的有效性。在本节中，选择两个经典 FJSP 标准测试集上测试。首先，通过田口正交实验对算法的最优参数选择进行了讨论。再次，经过成分分离试验验证 SPAMA 中每个改进点的有效性。最后，通过与最新的相关算法对比验证了 SPAMA 在 GDFJSP 的求解上表现最好，并通过图表对比详细讨论了算法性能优越的原因。所有算法都用 MATLAB2016Rb 编程运行在 Intel Core i7-6700 CPU@ 3.4GHz 8G RAM Win10 上。为了公平起见，每个算法独立运行 20 次，通过收集收敛性，分布性和综合指标的均值和方差来衡量算法性能。

3.6.2 测试问题和指标

为了验证 SPAMA 的有效性，本章选择了在 Mk01-10[189]和 DP01-10[190]共 20 个测试问题上进行测试。总工厂数 n_f=2 且为同构工厂。工厂机器的加工功率 P_W=4，机器的空闲功率 P_I=1。与第二章一样，本章的实验指标采用 HV，GD，Spread 来衡量算法的综合性能，收敛性和分布性。

3.6.3 参数实验

算法参数是影响算法求解问题的性能最直接的因素。SPAMA 共有 5 个参数分别是种群大小 ps，记忆体长度 HL，奖励步长 S，交叉概率 P_c 和突变概率 P_m。本章采用田口(Taguchi)正交实验（design-of-experiment，DOE）[186]来测试最优的参数组合。不同参数的级别设置如下：ps$\in\{60,80,100\}$，$HL\in\{20,25,30\}$，$S\in\{0.05,0.1,0.15\}$，$P_c\in\{0.6,0.8,1\}$，$P_m\in\{0.1,0.15,0.2\}$。更多地，生成了正交实验表 $L_{27}(5^3)$来矫正最优参数选择。为了保证公平，每个参数组合在 20 个测试问题上独立运行 20 次。同时，收集三个指标的均值，并采用 Minitab18 进行田口实验分析，得出指标的主影响图。如图 3.8，3.9 和 3.10 所示，根据三个指标的综合评判，最后的最优参数组合为 ps=100，HL =30，S =0.15，P_c =1，P_m =0.2。

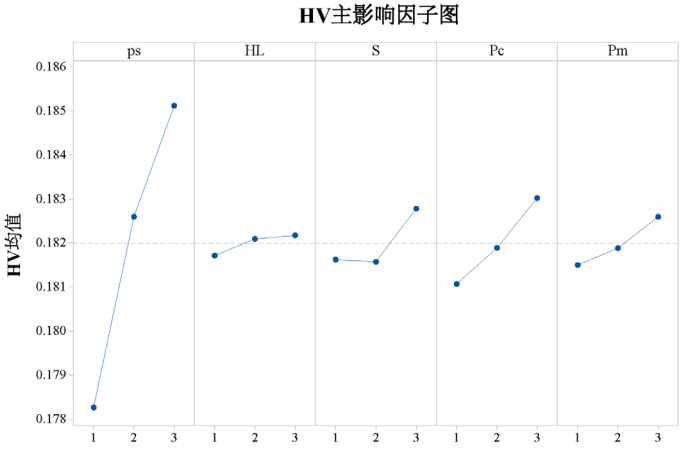

图 3.8 HV 主影响因子图

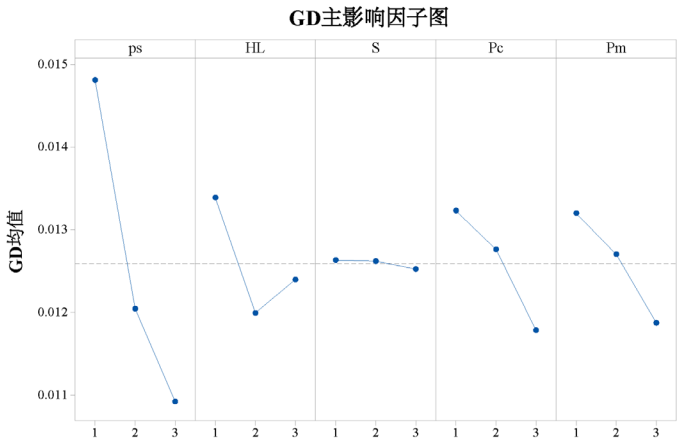

图 3.9 GD 主影响因子图

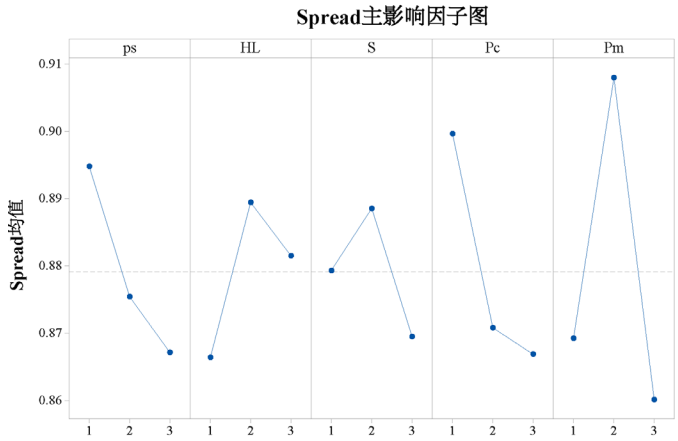

图 3.10 Spread 主影响因子图

3.6.4 成分分离实验

为了验证 SPAMA 每个改进点的有效性，本节设计了四个 SPAMA 的变种算法

分别是：EO 表示简单的带精英存档的进化算子，EO+V 表示在 EO 的基础上增加基于置信度的自学习的变邻域搜索，EO+SV 表示在 EO+V 的基础上将算子选择模型替换为基于 SPA 的算子选择模型，EO+SVE(SPAMA)表示在 EO+SV 的基础上增加节能策略。为了公平起见，每个变种算法在 20 个测试问题上独立运行 20 次，停止条件为最大迭代次数 MaxG=200 以保证相同的全局搜索次数。

表 3.1 SPAMA 变体算法 Frideman 秩和检验统计结果(置信度 α=0.05)

MOEAs	HV		GD		Spread	
	rank	*p*-value	rank	*p*-value	rank	*p*-value
EO	3.350	6.80E-06	2.750	2.17E-02	2.350	1.45E-01
EO+V	2.700		3.000		2.450	
EO+SV	2.650		2.450		**2.150**	
EO+SVE	**1.300**		**1.800**		3.050	

为了验证改进点的有效性，本章采用 KEEL 软件对变体算法之间的显著性进行了统计和分析。表 3.1 展示了 SPAMA 变体算法的 Frideman 秩和检验统计结果。如表所示**加粗**表示排名最高的算法，其中对比 EO 和 EO+V 可以验证提出基于关键路径的局部搜索算子的有效性。通过对比 EO+SV 和 EO+V，所有指标全面提升，说明算法性能有了进一步提升验证了提出基于 SPA 的算子选择模型的有效性。通过对比 EO+SV 和 EO+SVE，在 HV 和 GD 上显著提升，说明提出的节能策略有效增强算法收敛。同时 p 值小于 0.05 说明算法之间具有显著性的差异，说明 SPAMA 的每个改进点显著性地有效。同时表 3.2，3.3，3.4 详细展示了每个指标的均值和标准差在独立运行 20 次后的统计结果。灰色和加粗表示最好。可以得知 SPAMA 在所有测试上超过一半优于各变体算法，进一步证明了改进点的有效性。

表 3.2 SPAMA 变体算法 HV(max)指标对比结果

Intances	EO		EO+V		EO+SV		EO+SVE	
	mean	std	mean	std	mean	std	mean	std
DP01	0.1246	0.0052	0.1292	0.0051	0.1257	0.0073	**0.1339**	0.0084
DP02	0.1396	0.0057	0.1386	0.0073	0.1420	0.0060	**0.1466**	0.0063
DP03	0.1457	0.0068	0.1444	0.0058	0.1432	0.0052	**0.1486**	0.0070
DP04	0.1267	0.0072	0.1283	0.0043	0.1290	0.0076	**0.1360**	0.0068
DP05	0.1358	0.0068	0.1360	0.0093	0.1413	0.0060	**0.1443**	0.0090
DP06	0.1391	0.0058	0.1351	0.0077	0.1372	0.0070	**0.1396**	0.0068
DP07	0.0559	0.0052	0.0569	0.0075	0.0563	0.0057	**0.0679**	0.0066
DP08	0.0554	0.0067	0.0583	0.0061	0.0590	0.0082	**0.0631**	0.0076
DP09	0.1269	0.0075	0.1236	0.0086	0.1241	0.0079	**0.1285**	0.0074

表 3.2（续） SPAMA 变体算法 HV(max)指标对比结果

Intances	EO		EO+V		EO+SV		EO+SVE	
	mean	std	mean	std	mean	std	mean	std
DP10	0.0510	0.0071	0.0547	0.0072	0.0513	0.0068	**0.0612**	0.0050
Mk01	0.1508	0.0066	0.1521	0.0073	0.1520	0.0061	**0.1530**	0.0043
Mk02	0.1340	0.0105	0.1395	0.0078	0.1387	0.0104	**0.1421**	0.0112
Mk03	0.0811	0.0093	0.0801	0.0111	0.0841	0.0083	**0.0863**	0.0069
Mk04	0.1234	0.0095	0.1234	0.0083	0.1251	0.0085	**0.1279**	0.0076
Mk05	0.0856	0.0067	0.0861	0.0066	0.0855	0.0052	**0.0889**	0.0045
Mk06	0.0999	0.0123	0.1037	0.0170	0.1048	0.0131	**0.1119**	0.0139
Mk07	0.1103	0.0078	0.1101	0.0069	0.1092	0.0063	**0.1105**	0.0078
Mk08	0.0809	0.0091	0.0833	0.0070	0.0845	0.0070	**0.0892**	0.0055
Mk09	0.0810	0.0101	0.0839	0.0100	0.0830	0.0087	**0.0786**	0.0115
Mk10	0.0857	0.0121	**0.0975**	0.0123	0.0930	0.0108	0.0749	0.0149

表 3.3 SPAMA 变体算法 GD(min)指标对比结果

Intances	EO		EO+V		EO+SV		EO+SVE	
	mean	std	mean	std	mean	std	mean	std
DP01	0.0175	0.0068	0.0212	0.0209	0.0224	0.0147	**0.0159**	0.0103
DP02	0.0314	0.0146	0.0377	0.0152	0.0292	0.0125	**0.0261**	0.0165
DP03	**0.0257**	0.0123	0.0311	0.0099	0.0299	0.0149	0.0276	0.0182
DP04	0.0188	0.0123	0.0168	0.0062	0.0258	0.0244	**0.0070**	0.0056
DP05	0.0172	0.0067	0.0212	0.0158	0.0172	0.0104	**0.0140**	0.0088
DP06	**0.0185**	0.0115	0.0221	0.0116	0.0197	0.0093	0.0231	0.0193
DP07	0.0648	0.0190	0.0656	0.0287	0.0643	0.0166	**0.0313**	0.0166
DP08	0.0435	0.0169	0.0372	0.0154	0.0370	0.0218	**0.0235**	0.0151
DP09	0.0149	0.0098	0.0213	0.0138	0.0190	0.0115	**0.0135**	0.0083
DP10	0.0420	0.0206	0.0323	0.0203	0.0374	0.0176	**0.0151**	0.0102
Mk01	0.0117	0.0075	0.0095	0.0046	0.0101	0.0079	**0.0098**	0.0033
Mk02	0.0417	0.0153	0.0350	0.0130	0.0317	0.0120	**0.0310**	0.0139
Mk03	0.0251	0.0120	0.0276	0.0110	0.0212	0.0113	**0.0198**	0.0102
Mk04	**0.0139**	0.0072	0.0171	0.0083	0.0146	0.0071	0.0147	0.0055
Mk05	0.0107	0.0041	0.0108	0.0051	0.0101	0.0061	**0.0087**	0.0033
Mk06	0.0486	0.0181	0.0423	0.0225	0.0421	0.0250	**0.0356**	0.0148
Mk07	0.0178	0.0106	0.0200	0.0075	0.0182	0.0081	**0.0212**	0.0104
Mk08	0.0367	0.0212	0.0316	0.0144	0.0308	0.0163	**0.0194**	0.0142
Mk09	0.0336	0.0183	**0.0283**	0.0166	0.0299	0.0153	0.0355	0.0259
Mk10	0.0420	0.0218	**0.0230**	0.0163	0.0270	0.0139	0.0554	0.0249

表 3.4 SPAMA 变体算法 Spread(min)指标对比结果

Intances	EO		EO+V		EO+SV		EO+SVE	
	mean	std	mean	std	mean	std	mean	std
DP01	**1.0021**	0.0044	1.2228	0.4645	1.1245	0.3352	1.1490	0.3490
DP02	**1.0262**	0.0953	1.1548	0.3966	1.0896	0.3383	1.2614	0.4501
DP03	1.0911	0.3557	**1.0727**	0.2432	1.0966	0.2614	1.2474	0.4821
DP04	**1.1217**	0.4276	1.1323	0.4087	1.1271	0.3223	1.1275	0.2867
DP05	1.2086	0.4554	1.0032	0.0369	**0.9893**	0.0657	1.1616	0.3569
DP06	1.0561	0.1222	**1.0049**	0.0193	1.0553	0.1404	1.2216	0.4682
DP07	1.0054	0.0271	1.0175	0.0252	**1.0050**	0.0156	1.0116	0.0226
DP08	1.0384	0.0791	1.0002	0.0313	1.0387	0.0856	**0.9979**	0.0829
DP09	**0.9998**	0.0032	1.0065	0.0204	1.0029	0.0111	1.1316	0.4966
DP10	**1.0021**	0.0088	1.0235	0.0561	1.0407	0.1665	1.0138	0.0599
Mk01	0.8881	0.2706	0.9217	0.2461	**0.8143**	0.2718	0.8959	0.3101
Mk02	0.9705	0.0833	**0.9394**	0.1167	0.9448	0.0776	0.9511	0.0460
Mk03	1.0000	0.0931	**0.9560**	0.1745	1.0473	0.0618	1.0408	0.1097
Mk04	1.0181	0.1573	0.9695	0.1859	**0.9257**	0.1401	0.9953	0.1869
Mk05	1.0005	0.0608	**0.9900**	0.1236	0.9995	0.1159	0.9995	0.0623
Mk06	**0.9687**	0.0554	0.9814	0.0688	0.9794	0.0504	0.9988	0.0380
Mk07	0.9553	0.1702	0.9677	0.2848	**0.9493**	0.1926	1.0001	0.1741
Mk08	**0.9801**	0.1398	0.9992	0.1420	0.9944	0.1254	0.9877	0.1164
Mk09	0.9934	0.1087	0.9776	0.0926	**0.9686**	0.0652	0.9917	0.0363
Mk10	1.0013	0.0272	**0.9673**	0.0965	0.9905	0.0751	0.9886	0.0398

3.6.5 对比实验

为了进一步验证 SPAMA 每个改进点的有效性，本节选取了几类 MOEAs 作为对比算法分别是：（1）主流 MOEAs：MOEA/D[182]和 NSGA-II[183]；（2）近期提出的 MOEAs：AR-MOEA[187]；（3）FJSP 相关算法：TPM[60]；（4）DFJSP 相关算法：HSLFA[135]和 HMOMA[11]。每个算法的参数设置如下：所有算法的种群大小 ps=100，交叉概率 P_c=1，突变概率 P_m=0.2。MOEA/D 的邻域更新范围 T=10，TPM 的参数帝国数 Nim=6。为了对比公平，每个算法独立运行 20 次，停止条件为最大迭代次数 MaxNFEs=65000。表 3.5 展示了所有对比算法的 Frideman 秩和检验统计结果，结果显示 SPAMA 在 HV 和 GD 指标上排名第 1 显著地好于所有对比算法，在 Spread 指标上排名第 4。表 3.6，3.7 和 3.8 展示了所有指标的均值和标准差的统计结果，“-/=/+”表示各变体算法在该测试问题上显著性地“差于/无显著性/好于”SPAMA。结果显示 SPAMA 在 HV 和 GD 指标上有超过一半以上的测试集的表现

显著好于所有对比算法，验证了本章提出算法的有效性。

表 3.5 SPAMA 和对比算法 Frideman 秩和检验统计结果(置信度 α=0.05)

MOEAs	HV		GD		Spread	
	rank	*p*-value	rank	*p*-value	rank	*p*-value
NSGA-II	2.500		2.300		3.550	
MOEA/D	5.750		5.750		5.600	
ARMOEA	6.200		6.200		4.300	
HSLFA	3.200	2.58E-18	3.100	1.42E-18	3.700	8.53E-03
TPM	2.950		3.350		**2.975**	
HMOMA	5.950		5.900		4.025	
SPAMA	**1.450**		**1.350**		3.850	

表 3.6 SPAMA 与对比算法 HV(max)指标对比结果

Instances	NSGA-II		MOEA/D		ARMOEA		HSLFA		TPM		HMOMA		SPAMA	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
DP01	0.1515=	0.0049	0.1201-	0.0085	0.1227-	0.0066	0.1454-	0.0061	0.1434-	0.0079	0.1418-	0.0088	**0.1542**	0.0064
DP02	0.1777-	0.0077	0.1364-	0.0085	0.1393-	0.0096	0.1768-	0.0051	0.1738-	0.0054	0.1578-	0.0070	**0.1860**	0.0073
DP03	0.1756-	0.0061	0.1341-	0.0102	0.1304-	0.0089	0.1716-	0.0062	0.1778=	0.0059	0.152-	0.0051	**0.1822**	0.0084
DP04	0.1737-	0.0053	0.1411-	0.0095	0.1437-	0.0088	0.1717-	0.0061	0.1682-	0.0061	0.1678-	0.0066	**0.1834**	0.0081
DP05	0.1681-	0.0097	0.1297-	0.0097	0.1262-	0.0103	0.1695-	0.0063	0.1708=	0.0067	0.1479-	0.0066	**0.1778**	0.0092
DP06	0.1878-	0.0081	0.1434-	0.0101	0.1397-	0.0134	0.1942=	0.0058	**0.1949=**	0.0077	0.1579-	0.0071	0.1941	0.0095
DP07	0.1738-	0.0086	0.1292-	0.0123	0.1373-	0.0084	0.1758-	0.0065	0.1716-	0.0057	0.1648-	0.0081	**0.1850**	0.0051
DP08	0.1906-	0.0065	0.1385-	0.0135	0.1427-	0.0109	0.1939=	0.0079	0.1952=	0.0065	0.1629-	0.0092	**0.1981**	0.0092
DP09	0.1534-	0.0098	0.0998-	0.0131	0.1024-	0.0105	0.1581=	0.0062	0.1616=	0.0081	0.1285-	0.0062	**0.1623**	0.0069
DP10	0.1573-	0.0083	0.1182-	0.0104	0.1186-	0.0100	0.1567-	0.0075	0.1527-	0.0073	0.1441-	0.0065	**0.1661**	0.0068
Mk01	0.2832=	0.0093	0.2705-	0.0085	0.1698-	0.0303	0.2593-	0.0188	0.2787-	0.0070	0.1215-	0.0131	**0.2871**	0.0066
Mk02	0.3252=	0.0113	0.3011-	0.0129	0.1319-	0.0303	0.2844-	0.0228	0.3171=	0.0106	0.0968-	0.0129	**0.3237**	0.0110
Mk03	0.2907=	0.0158	0.2122-	0.0159	0.0936-	0.0199	0.2815=	0.0237	0.2849=	0.0122	0.0659-	0.0106	**0.2964**	0.0181
Mk04	0.228=	0.0103	0.1977-	0.0139	0.1385-	0.0147	0.2129-	0.0137	0.2224-	0.0082	0.119-	0.0117	**0.2358**	0.0103
Mk05	0.1452=	0.0071	0.1123-	0.0109	0.1023-	0.0117	0.1366-	0.0107	0.1396-	0.0063	0.0932-	0.0104	**0.1488**	0.0060
Mk06	0.477=	0.0182	0.4136-	0.0280	0.1413-	0.0276	0.4578-	0.0246	0.4768=	0.0184	0.1109-	0.0160	**0.4860**	0.0197
Mk07	0.3093=	0.0089	0.2763-	0.0134	0.1466-	0.0215	0.2928-	0.0134	0.298-	0.0101	0.0849-	0.0116	**0.3168**	0.0085
Mk08	0.1544-	0.0074	0.1028-	0.0135	0.0921-	0.0124	0.1611=	0.0073	0.1494-	0.0066	0.0965-	0.0096	**0.1614**	0.0077
Mk09	0.1876=	0.0121	0.0977-	0.0119	0.0817-	0.0125	**0.1935=**	0.0123	0.1886=	0.0087	0.0815-	0.0108	0.1825	0.0156
Mk10	**0.2296+**	0.0139	0.1074-	0.0183	0.0847-	0.0178	0.229+	0.0156	0.2253+	0.0138	0.0782-	0.0108	0.2079	0.0179
-/=/+	10/9/1		20/0/0		20/0/0		13/6/1		10/9/1		20/0/0			

表 3.7 SPAMA 与对比算法 GD(min)指标对比结果

Instances	NSGA-II		MOEA/D		ARMOEA		HSLFA		TPM		HMOMA		SPAMA	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
DP01	0.0241=	0.0195	0.061-	0.0173	0.0587-	0.0163	0.0264-	0.0122	0.0298-	0.0136	0.0308-	0.0166	**0.0136**	0.0068
DP02	0.0186=	0.0133	0.0683-	0.0231	0.0598-	0.0205	0.0143-	0.0054	0.0204-	0.0077	0.0372-	0.0084	**0.0093**	0.0084
DP03	0.0224=	0.0116	0.0714-	0.0224	0.071-	0.0231	0.0303-	0.0138	0.02=	0.0083	0.0497-	0.0153	**0.0185**	0.0158
DP04	0.0201-	0.0045	0.0616-	0.0190	0.0599-	0.0165	0.0231-	0.0092	0.0291-	0.0094	0.0281-	0.0081	**0.0115**	0.0092
DP05	0.0247-	0.0202	0.0596-	0.0222	0.073-	0.0205	0.0222-	0.0114	0.0233-	0.0137	0.0507-	0.0169	**0.0111**	0.0100
DP06	0.0188=	0.0094	0.0752-	0.0232	0.0805-	0.0358	0.0223-	0.0121	0.0191=	0.0143	0.0584-	0.0130	**0.0164**	0.0106
DP07	0.0271-	0.0154	0.0911-	0.0272	0.0752-	0.0174	0.024-	0.0100	0.0363-	0.0106	0.0328-	0.0121	**0.0122**	0.0136
DP08	0.0133-	0.0050	0.0758-	0.0174	0.0715-	0.0277	0.0103=	0.0075	0.0117=	0.0083	0.0487-	0.0168	**0.0079**	0.0069
DP09	0.02-	0.0106	0.0911-	0.0271	0.0939-	0.0314	0.0146=	0.0090	0.0135=	0.0109	0.052-	0.0165	**0.0153**	0.0215
DP10	0.0241-	0.0134	0.0853-	0.0349	0.0857-	0.0309	0.0294-	0.0142	0.0408-	0.0147	0.0429-	0.0150	**0.0119**	0.0078
Mk01	0.009-	0.0044	0.0198-	0.0087	0.101-	0.0388	0.0163-	0.0104	0.013-	0.0060	0.1751-	0.0276	**0.0078**	0.0034
Mk02	0.0112=	0.0042	0.0249-	0.0075	0.1809-	0.0516	0.0287-	0.0121	0.0163=	0.0077	0.2296-	0.0433	**0.0124**	0.0055
Mk03	0.0136=	0.0059	0.0701-	0.0184	0.2465-	0.0551	0.0239=	0.0167	0.0167=	0.0109	0.3169-	0.0678	**0.0153**	0.0079
Mk04	0.0168-	0.0083	0.0376-	0.0129	0.0989-	0.0192	0.0248-	0.0095	0.0212-	0.0061	0.138-	0.0307	**0.0123**	0.0054
Mk05	0.0229-	0.0135	0.0767-	0.0197	0.1022-	0.0354	0.0468-	0.0332	0.0316-	0.0123	0.1367-	0.0398	**0.0163**	0.0118
Mk06	0.0323-	0.0137	0.0825-	0.0200	0.3619-	0.0988	0.0501-	0.0255	0.0374=	0.0199	0.4718-	0.0845	**0.0234**	0.0133
Mk07	0.0089=	0.0035	0.0263-	0.0090	0.1562-	0.0562	0.0138-	0.0056	0.0194-	0.0081	0.2899-	0.0512	**0.0068**	0.0031
Mk08	0.0227-	0.0106	0.0933-	0.0285	0.1134-	0.0276	0.0157=	0.0121	0.0382-	0.0137	0.1396-	0.0295	**0.0155**	0.0119
Mk09	**0.0223=**	0.0123	0.1249-	0.0348	0.1444-	0.0368	0.029=	0.0167	0.0376-	0.0155	0.2026-	0.0558	**0.0223**	0.0145
Mk10	**0.0212+**	0.0094	0.1451-	0.0278	0.1851-	0.0591	0.0221+	0.0152	0.023+	0.0128	0.2377-	0.0496	0.0388	0.0183
-/=/+	11/8/1		20/0/0		20/0/0		14/5/1		12/7/1		20/0/0			

表 3.8 SPAMA 与对比算法 Spread(min)指标对比结果

Instances	NSGA-II		MOEA/D		ARMOEA		HSLFA		TPM		HMOMA		SPAMA	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
DP01	1.2335=	0.4841	1.0152=	0.1065	1.0628=	0.1404	1.0114=	0.0211	1.0044=	0.0098	**1.0007=**	0.0017	1.0785	0.1912
DP02	1.1624=	0.4187	1.0982=	0.1645	1.0235=	0.1606	1.0023=	0.0073	1=	0.0000	**0.9983=**	0.0052	1.1603	0.4415
DP03	1.0702=	0.2357	1.134=	0.2914	**0.9947=**	0.0185	0.9994=	0.0018	1=	0.0000	1.0069=	0.0215	1.0524	0.1828
DP04	1.1348=	0.4196	1.1084=	0.3043	1.0055=	0.0167	1.0022=	0.0058	**0.9998=**	0.0006	1.0007=	0.0018	1.2959	0.6341
DP05	1.0049=	0.0377	1.0643=	0.1286	1.0915=	0.2357	**0.9999=**	0.0002	1=	0.0000	1.0004=	0.0089	1.1244	0.3453
DP06	1.0053=	0.0205	1.0706=	0.1802	1.0093=	0.0626	**0.9997=**	0.0009	1=	0.0000	1.0003=	0.0015	1.0285	0.1908
DP07	1.0295=	0.0419	1.0889=	0.2121	1.0029=	0.0076	1.0073=	0.0151	**0.9976=**	0.0059	1.0036=	0.0213	1.1266	0.3651
DP08	1.0011=	0.0307	1.0352=	0.0656	1.0058=	0.0269	1.0291=	0.0995	**1=**	0.0000	1.001=	0.0065	1.0245	0.0532
DP09	1.0261=	0.0683	1.0163=	0.0845	1.0101=	0.0504	**0.9962=**	0.0095	1=	1.0000	0.9995=	0.0045	1.1089	0.3476
DP10	1.0169=	0.0402	1.0402=	0.1270	1.0092=	0.0182	1.0031=	0.0135	1=	0.0000	0.9977=	0.0063	**0.9596**	0.1172
Mk01	0.9316=	0.2529	1.064=	0.2551	0.9969=	0.0794	0.9758=	0.0497	0.9696=	0.1196	1.0119=	0.0924	**0.8786**	0.3074

表 3.8 SPAMA 与对比算法 Spread(min)指标对比结果

Instances	NSGA-II		MOEA/D		ARMOEA		HSLFA		TPM		HMOMA		SPAMA	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Mk02	0.9326=	0.1303	0.9836=	0.2111	0.9858=	0.0338	0.9687=	0.0635	0.9911=	0.0267	0.9958=	0.0152	**0.9062**	0.1359
Mk03	**0.9747=**	0.1408	0.9959=	0.1392	1.0013=	0.0130	1.0088=	0.0191	1=	0.0000	1.0018=	0.0043	1.1139	0.2198
Mk04	0.9653=	0.1913	1.0149=	0.1519	0.9999=	0.0209	**0.9466=**	0.1126	0.9947=	0.0905	1.0112=	0.0585	0.9523	0.1073
Mk05	0.9857=	0.1018	1.0082=	0.1037	1.0075=	0.0543	0.9923=	0.1341	1.0193=	0.0639	0.9967=	0.0087	**0.9591**	0.1610
Mk06	0.9841=	0.0519	1.0939=	0.1984	1.0053=	0.0182	1.0014=	0.0511	1=	0.0000	1=	0.0000	**0.9442**	0.1392
Mk07	**0.9798=**	0.2827	0.9838=	0.1551	0.9897=	0.0226	0.9956=	0.0709	0.9823=	0.0686	0.9902=	0.0277	0.9812	0.2390
Mk08	0.9994=	0.1251	1.0122=	0.0667	1.0034=	0.0120	1.0119=	0.0741	1=	0.0000	1.0671=	0.1162	**0.9912**	0.1346
Mk09	**0.9808=**	0.0863	1.0018=	0.0206	1.0005=	0.0192	1.0378=	0.1025	1=	0.0000	1.0036=	0.0070	0.9913	0.0468
Mk10	**0.9755=**	0.0638	1.0143=	0.0478	0.9991=	0.0178	1.0034=	0.0356	1=	1.0000	1.0196=	0.0653	0.9846	0.0616
-/=/+	10/9/1		20/0/0		20/0/0		20/0/0		20/0/0		20/0/0			

3.6.6 讨论与分析

SPAMA 算法在 GDFJSP 问题上的成功在于它的设计。首先，基于关键路径和关键块的理论发现影响 Cmax 最直接的工序，设计基于关键路径的局部搜索算子，极大地提高了局部搜索的效率。其次，提出了基于意外流行算法的算子选择模型，修正种群在算子选择决策中无法发现小权重但高效的算子的问题，使得高效的算子占比更大，加速算法收敛。再次，提出基于全主动调度解码的节能策略，该策略可以有效减少空闲时间，即减少了完工时间又有效地降低了能耗。最后，精英存档和进化过程中的精英策略收集了历史的精英解，提高了历史解的利用率，增强了种群的分布性。

为了进一步说明 SPAMA 算法的有效性，本节还通过图来展示算法性能并分析其有效的原因。图 3.11 展示了所有算法在 Mk04 测试集上求得 HV 值最好的非支配解集，说明 SPAMA 的非支配解集更能探索前沿面，表明了算法的性能最好。图 3.12 展示了 SPAMA 在 Mk01 测试集上找到的 Cmax 最小解的甘特图(Cmax=26h，T EC=693kWh)。图 3.13 展示了 SPAMA 在 Mk01 测试集上找到的 TEC 最小解的甘特图(Cmax=34h，TEC=662kWh)。综上所述 SPAMA 在求解 GDFJSP 方面具有良好的性能。

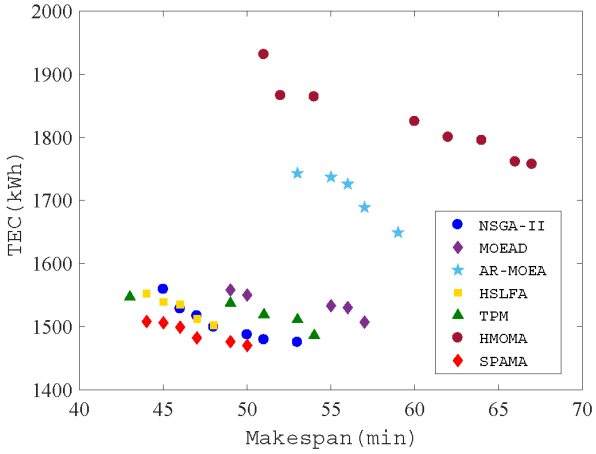

图 3.11 SPAMA 所有对比算法在 Mk04 的 PF 对比

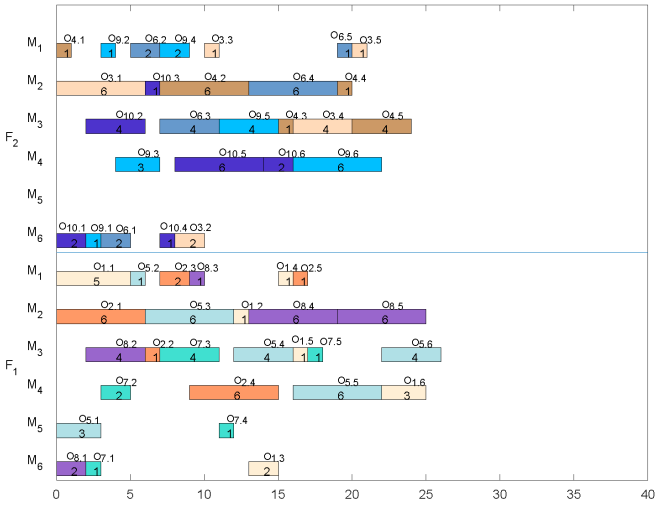

图 3.12 SPAMA 在 Mk01 上找到 Cmax 最小解的甘特图

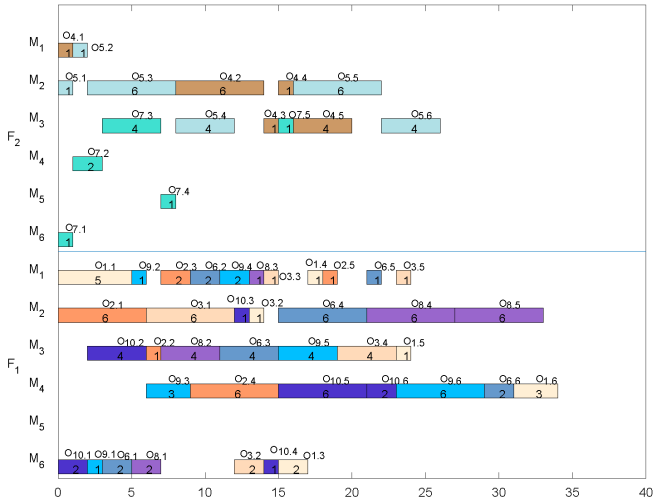

图 3.13 SPAMA 在 Mk01 上找到 TEC 最小解的甘特图

3.7 本节小结

本章根据同构绿色分布式柔性作业车间调度问题的特点，提出了基于意外流行的自适应模因算法，并在两个著名公开测试集 Mk 和 DP 共 20 个测试问题上进行测试，同时与一些最先进的算法比较。大量实验结果表明，本章所提的 SPAMA 算法在求解 GDFJSP 上取得了最好的结果。本章的贡献主要有以下几点：(1) 提出了四种基于关键路径和关键块的局部搜索算子提高了局部搜索的效率，加速算法收敛；(2) 首次将意外流行算法应用到车间调度问题中，修正种群的决策以发现小权重但高效的算子，选择出当前代数最有效的算子并分配更多的资源，加速算法收敛；(3) 提出了基于全主动调度解码节能策略，有效减少空闲时间即降低了完工时间又减少了能耗目标；(4) 设计的精英策略以及围绕精英解的局部搜索有效的节省了计算资源为下一章 MA 进化框架的改进打下基础。同构 GDFJSP 是异构 GDHFJSP 研究的基础，本章提出的 SPAMA 算法为后续章节中算法进一步的改进提供算法支持。

第四章　绿色分布式异构柔性作业车间调度问题求解

4.1 引言

在现实排产中，由于成本、工人产能、机器配置等原因，同工件在不同工厂中的加工时间是不同的。因此，在第三章的基础上，本章考虑异构 GDFJSP 即 GDHFJSP 问题。GDHFJSP 在 GDFJSP 的基础上增加了工件工厂柔性和异构的问题。因此解空间的数量在 GDFJSP 上增加了一个数量级使得 GDHFJSP 更难求解。为了解决 GDHFJSP，本章提出了基于深度强化学习的改进模因算法求解 GDHFJSP 包含以下改进：（1）为了应对复杂的决策空间使算法执行更多的全局搜索，提出改进的模因计算框架，名为寄生进化框架，以平衡全局和局部搜索；（2）针对工厂异构的特性提出了针对性的局部搜索算子，在第三章的基础上扩充到 9 种局部搜索算子，针对解决不同子问题；（3）为了使得每个解选择最合适的算子，提出了基于深度强化学习。为了验证本章算法的性能，生成了 20 个不同规模的测试问题。实验结果表明，本章提出了寄生进化框架可以有效的平衡全局和局部搜索资源，基于深度强化学习的算子选择模型可以有效地选择算子。

4.2 GDHFJSP 描述与 MILP 模型

4.2.1　GDHFJSP 描述

在绿色异构分布式柔性作业车间调度问题中有工件集合 $I=\{1,2,\ldots,i,\ldots,n\}$，每个工件有工序集合 $J_i=\{O_{i,1},O_{i,2},\ldots,O_{i,j},\ldots,O_{i,ni}\}$，有工厂集合 $F=\{1,2,\ldots,f,\ldots,n_f\}$，在工厂中有机器集合 $M=\{1,2,\ldots,k,\ldots m\}$。每个工序可以被 M 的子集 $K_{f,i,j}$ 加工。每个机器可以加工多种工件的工序。每个工序在不同工厂的相同机器上的加工时间 $T_{i,j,f,k}$ 是不同的。在工件分配了工厂后，所有工件必须都在该工厂中加工。GDHFJSP 主要解决 3 个耦合的子问题：确定每个工件的工厂选择；确定每个工厂中所有工序的加工顺序；为每个工序分配加工机床；使得所有工厂的总完工时间和总能耗最小。关于同构 GDHFJSP 的问题假设如下：（1）所有工序和机器在零时刻都是可用的；（2）所有机器在同一时刻只能加工一个工序且无法被抢占；（3）所有加工

时间和能耗相关的数据都是确定的；（4）一个工件同一时间只能被一个机器加工；（5）运输时间，启动时间以及相关能耗不考虑。（6）一个工件只能分配到一个工厂，且所有工序都要在该工厂中加工。（7）相同工序在不同工厂的机器上加工时间不同，但可选机器集合相同。

4.2.2 GDHFJSP 的 MILP 模型

在介绍 GDHFJSP 的模型之前，首先要介绍 MILP 模型中的变量如下：

索引变量：

i,i': 工件索引号 $i\in\{1,2,...,n\}$;

j,j': 工序索引号 $j\in\{1,2,...,n_i\}$;

k,k': 机器索引号 $k\in\{1,2,...,m\}$;

t,t': 机器上加工位置索引号 $t\in\{1,2,..., p_{f,k}\}$;

f: 机器上加工位置索引号 $f\in\{1,2,..., n_f\}$;

参数：

n: 总工件个数；

m: 总机器个数；

n_f: 总工厂数；

n_i: 工件 i 的工序个数；

$p_{f,k}$: 每个机器上加工位置数；

I: 工件集合 $I=\{1,2,...,n\}$;

J_i: 工件 i 的工序集合 $J_i=\{O_{i,1},O_{i,2},\ldots,O_{i,j},\ldots,O_{i,ni}\}$;

M: 机器集合 $M=\{1,2,...,m\}$;

$K_{f,i,j}$: 工序 $O_{i,j}$ 在工厂 f 机器集合;

$P_{f,k}$: 机器 k 在工厂 f 的加工位置集合 $P_k=\{1,2,..., p_{f,k}\}$;

$P_{f,k}'$: 机器 k 在工厂 f 的前 p_k -1 个加工位置集合 $P_k=\{1,2,...,p_{f,k}-1\}$;

$T_{i,j,f,k}$: 工序 $O_{i,j}$ 在工厂 f 机器 k 的加工时间;

W_O: 机器加工功率;

W_I: 机器空闲功率;

L: 充分大的整数保证不等式的一致性;

$x_{i,j,f,k}$: 0-1 变量，表明工序 $O_{i,j}$ 能否在工厂 f 的机器 k 上加工，如果可以值为 1，否则为 0；

决策变量：

$S_{f,i,j}$: 工序 $O_{i,j}$ 在工厂 f 开始加工时间；

$C_{f,i,j}$: 工序 $O_{i,j}$ 在工厂 f 完成加工时间；

$B_{f,k,t}$: 在工厂 f 机器 k 第 t 个位置开始加工时间；

E_I: 机器加工总能耗;

$X_{i,j,f,k,t}$: 0-1 变量，表明工序 $O_{i,j}$ 在在工厂 f 的机器 k 的第 t 个位置上加工，如果是值为 1 否则为 0；

$Y_{i,f}$: 0-1 变量，工件 i 被分配到工厂 f 加工，如果是值为 1 否则为 0；

GDHFJSP 的两个目标函数描述如下：

(1)最大完工时间通常被认为是车间调度问题的经济指标，车间内最大完工时间越小带来的企业受益越大。最大完工时间的优化目标的定义如下：

$$F_1 = C_{\max} = \max\{C_{f,i,n_i}\}, \forall i \in I, f \in F, \tag{7-1}$$

(2)总的能量消耗通常被认为是关键的环境指标，总能耗 TEC 反应企业在系统层面二氧化碳排放量。能耗目标主要由两个决策表达式组成分别是：机器的总加工能耗 E_W 和机器的总空闲能耗 E_I。总能耗的优化目标的定义如下：

$$F_2 = TEC = E_W + E_I, \tag{7-2}$$

$$E_W = \sum_{i\in I}\sum_{j\in J_i}\sum_{f\in F}\sum_{k\in M}\sum_{t\in P_k} W_O \times T_{i,j,f,k} \times X_{i,j,f,k,t}, \tag{7-3}$$

$$E_I = \sum_{f\in F}\sum_{k\in M}\sum_{t\in P_k'} W_I \times (B_{f,k,t+1} - B_{f,k,t} - \sum_{i\in I}\sum_{j\in J_i} X_{i,j,f,k,t} \times T_{i,j,f,k}), \tag{7-4}$$

GDHFJSP 的约束条件如下：

$$\sum_{f\in F} Y_{i,f} = 1, \forall i \in I, \tag{7-5}$$

$$\sum_{k\in K_{f,i,j}}\sum_{t\in P_k} X_{i,j,f,k,t} = Y_{i,f}, \forall i \in I, j \in J_i, f \in F, \tag{7-6}$$

$$S_{f,i,n_i} + \sum_{k\in K_{f,i,j}}\sum_{t\in P_k} T_{i,n_i,f,k} \times X_{i,n_i,f,k,t} \le C_{\max}, \forall i \in I, f \in F, \tag{7-7}$$

$$C_{f,\max} \le C_{\max}, \forall f \in F, \tag{7-8}$$

$$S_{f,i,j} + \sum_{k\in M}\sum_{t\in P_k} T_{i,j,f,k} \times X_{i,n_i,f,k,t} \le S_{f,i,j+1}, i \in I, j \in J_i - 1, f \in F, \tag{7-9}$$

$$\sum_{i\in I}\sum_{j\in J_i} X_{i,j,f,k,t} \le 1, \forall f \in F, k \in M, t \in P_{f,k}, \tag{7-10}$$

$$\sum_{i\in I}\sum_{j\in J_i} X_{i,j,f,k,t} \ge \sum_{i\in I}\sum_{j\in J_i} X_{i,j,f,k,t+1}, \forall f \in F, k \in M, t \in P_{f,k}', \tag{7-11}$$

$$B_{f,k,t+1} - B_{f,k,t} \ge \sum_{i\in I}\sum_{j\in J_i} X_{i,j,f,k,t} \times T_{i,j,f,k}, \forall f \in F, k \in M, t \in P_{f,k}', \tag{7-12}$$

$$E_{If,k,t} \geq (B_{f,k,t+1} - B_{f,k,t} - \sum_{i \in I} \sum_{j \in J_i} X_{i,j,f,k,t} \times T_{i,j,f,k}) \times W_I, \forall f \in F, k \in M, t \in P'_{f,k}, \quad (7\text{-}13)$$

$$B_{f,k,t} \geq S_{f,i,j} - L \times (1 - X_{i,j,f,k,t}), \forall i \in I, j \in J_i, f \in F, k \in M, t \in P_{f,k}, \quad (7\text{-}14)$$

$$B_{f,k,t} \leq S_{f,i,j} - L \times (1 - X_{i,j,f,k,t}), \forall i \in I, j \in J_i, f \in F, k \in M, t \in P_{f,k}, \quad (7\text{-}15)$$

$$0 \leq B_{f,k,t}, S_{f,i,j} \leq L, \forall i \in I, j \in J_i, f \in F, k \in M, t \in P_{f,k}, \quad (7\text{-}16)$$

约束(4-5)确保一个工件只能分配到一个工厂加工，约束(4-6)确保工件分配到工厂 f 后所有工序都在该工厂加工，约束(4-7)表示优化目标最大完工时间与工序完工时间的关系，约束(4-8)表示每个工厂最大完工时间和总最大完工时间的关系，约束(4-9)表示每个工序的开始加工前必须等上一个工序完工，约束(4-10)确保同一时刻每个机器的每个加工位置只加工一个工序，约束(4-11)确保每个机器上必须等前一个位置的工序完工才能加工下个位置工序，约束(4-12)表示每个机器上开始加工时间和完工时间的关系，约束(4-13)表示机器空闲能耗的计算方式，约束(4-14)和约束(4-15)是为了便于 MILP 模型采用分支界定算法搜寻解而对等式约束进行松弛，表示机器开始加工时间和工序的开始加工时间是相同的，约束(4-16)表示决策变量的取值范围。

4.3 基于 DQPEA 的 GDHFJSP 求解

本章提出了基于深度 Q 网络的寄生进化算法（Deep Q-Networks-based parasitic evolutionary algorithm，DQPEA），以 NSGA-II 算法为全局搜索算子。针对传统 MA 计算框架存在的问题，提出了寄生进化框架，将全局搜索和局部搜索划分到两个种群中执行，使得大部分计算资源分配给全局搜索，同时保持高质量的精英局部搜索。其次，在第三章的基础上，针对工厂异构的问题，对局部搜索算子进行了拓展，增加了 5 种工厂选择算子共 9 种基于关键路径的局部搜索算子，增强算法收敛。最后，提出基于深度 Q 网络的算子选择模型，通过神经网络学习解集合与算子选择概率之间的映射为每个解定制化的选择最优算子，增强算法的自主性。

4.3.1 编解码机制和种群初始化

本章采用三层编码机制，编解码方式和第三章相同，分为 OS，MS 和 FA 三维向量表示加工顺序，机器选择和工厂选择。解码细节在第三章中 3.5.1 节中已介绍。

为了给后续全局搜索提供良好的种群分布性，采用随机初始化的方式生成初

始种群，具体细节与第三章 3.5.2 节相同。

4.3.2 寄生进化框架

MA 进化框架存在的问题：传统的 MA 进化框架会在执行全局搜索后，直接对整个种群的个体执行局部搜索。但在分布式异构车间调度问题中，由于决策空间巨大，种群需要更多的计算资源执行全局搜索以充分地探索目标空间。如果直接对主种群执行局部搜索，则会极具减少全局搜索的执行次数，虽然整体种群会快速收敛，但分布性较差不利于后续对真实 PF 的探索。如图 4.1 所示，假设 P_0 表示初始化种群，此时具有良好的分布性。P_2 表示给种群每个解执行局部搜索后的种群，这时虽然收敛性更好，但种群的分布性急剧降低，不利于对潜在非支配前沿面的探索。应该不对种群执行局部搜索使得种群和 P_1 一样仅执行全局搜索，虽然不能收敛迅速，但具有良好的分布性，有利于后期对 PF 前沿的探索。

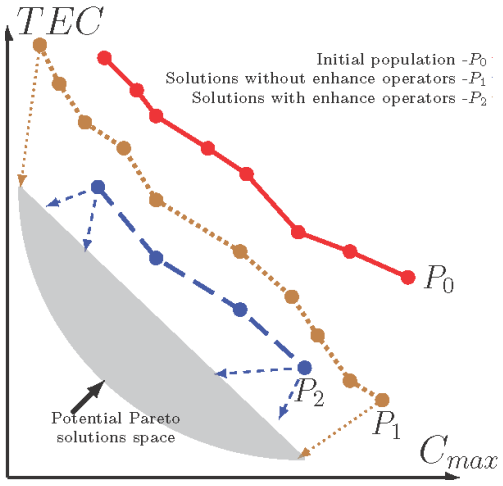

图 4.1 传统 MA 进化框架存在的问题

寄生进化框架：在分析了 MA 进化框架存在的问题后，本章提出改进的 MA 进化框架，名为寄生进化（Parasitic evolution，PE）。为了不让局部搜索影响全局搜索的种群，PE 将全局搜索和局部搜索分离到两个种群中分别为：宿主种群 H (Host)和寄生体种群 P (Parasite)。如图 4.2 所示，种群 H 主要执行全局搜索包含交配池选择，交叉变异和环境选择。种群 P 会将种群 H 中的非支配解集吸收即添加进种群 P 中。随后，P 种群中都是精英个体，在精英个体附近展开局部搜索，可以提升搜索效率。同时种群 P 主要负责局部搜索和节能策略，最后从种群 P 中输出最后的非支配解集。种群 P 在进化过程中不断进化，仅保留自身的非支配解集。因此，执行局部搜索进占少量计算资源，90%以上的资源都分配给 H 执行全局搜索。通过种群 H 和种群 P 的协同进化和分工进化使得算法即保证了充分的全局搜索和

良好的分布性以及高质量的精英局部搜索。由于种群间的信息交换方式类似于自然界中的寄生关系，受启发于目前进化计算中著名的进化框架如协同进化[193]和竞争进化[194]，同样是描述自然界中生物之间的关系，本章将改进的协同进化框架命名为寄生进化框架。PE 作为一种范式可以求解不同类型的分布式车间调度问题，解决全局搜索和局部搜索计算资源分配不配合的问题。

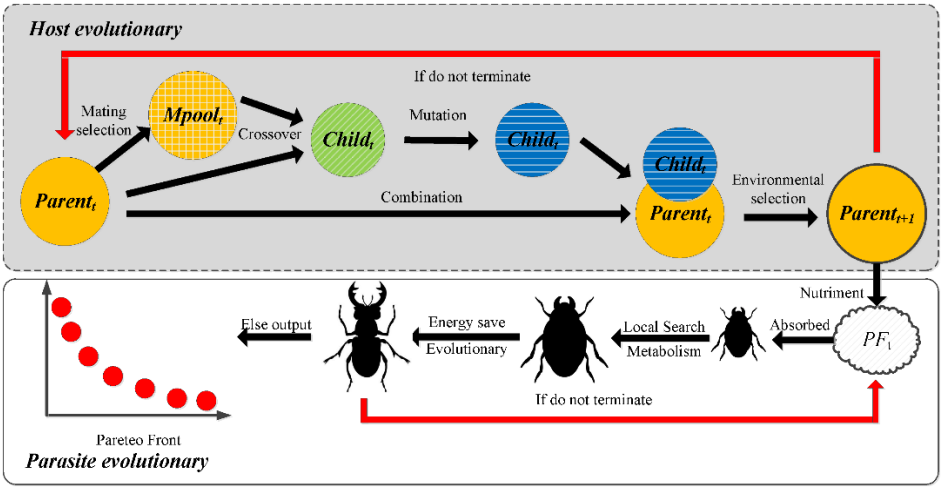

图 4.2 寄生进化框架

4.3.3 进化算子与节能策略

本章采用 POX 算子对 OS 进行交叉，采用 UX 算子对 MS 和 FA 向量交叉，算子具体细节在第二章 2.4.3 节中已介绍。突变算子与第三章 3.5.3 节中相同。

同样地，经过实验证明，第三章中提出的基于全主动调度解码的节能策略是目前针对 GFJSP 问题优化 Cmax 和 TEC 最高效的节能策略。因此，本章中仍采用基于全主动调度解码的节能策略降低能耗。

4.3.4 问题驱动的局部搜索算子

根据异构工厂中加工时间不同的特性，提炼出工厂选择知识如下：工件在总加工时间更少的工厂中加工，获得更少 Cmax 的概率更大。受制于工厂的负载以及所选机器的负载，不能确保一定收获更小的 Cmax。更多地，从机器选择中提炼了知识：当前工序选择加工时间更小的机器有可能减少 Cmax。但依据上述知识，本节设计了 2 种基于工厂选择的 2 种基于机器选择的局部搜索算子，在第三章的基础上进行了拓展，算子描述如下：

(1)局部搜索算子 1 (N6 邻域结构)：在求解了关键路径和关键块后，根据前人的研究表明，在关键块中设置扰动可以有效的减少空闲时间，进而减少完工时间。

而 N6 是传统作业车间调度问题中的邻域结构。经过实验测试在 FJSP 中也有良好的效果。图 3.5 详细解释了 N6 邻域结构。首先，将第一个、中间、最后一个关键块成为头块、中间块和尾块。对于头块，随机选取中间的工序插入到尾工序后面。对于中间块，随机选取两个中间工序分别插入到头工序前和尾工序后。对于尾块，随机选择一个工序插入到头工序前。

(2)局部搜索算子 2 (随机双点交换)：这是一个很简单的邻域结构，随机选择在所有工厂内选择两个工序并交换其在总顺序中的加工顺序，以增加分布性。

(3)局部搜索算子 3 (关键双点交换)：随机选择两个关键路径上的工序并交换其在工厂中的加工顺序，以减少关键工厂中的完工时间。

(4)局部搜索算子 4 (随机双点插入)：随机选择在所有工厂内选择两个工序并将后序工序插入到前序工序的前面，改变其在总加工顺序中的位置，以增加算法向不同区域探索的能力，增加分布性。

(5)局部搜索算子 5 (关键双点插入)：随机选择两个关键路径上的工序并将后序工序插入到前序工序的前面，以减少关键工厂中的完工时间。

(6)局部搜索算子 6 (随机重分配工厂)：随机选择关键路径上的工序将其所属的工件所有的工序随机移到另一个工厂。

(7)局部搜索算子 7 (概率重分配工厂)：为了使得加工时间更少的工厂获得更大的选择概率，将平均加工时间与选择概率对应。平均加工时间的计算公式为 $\overline{P_{i,f}}=\frac{1}{n_i}\sum_{j=1}^{n_i}(\sum_{k=1}^{K_{f,i,j}}P_{i,j,f,k})/|K_{f,i,j}|,\forall i\in I,f\in F$。随后，将所有平均加工时间归一化作为选择概率。随机在关键路径上选择一个工序，通过轮盘赌选择机制，选择加工工厂，将工序所属的工件所有的工序移到所选工厂。

(8)局部搜索算子 8 (随机重选机床)：随机选择关键路径上的工序，在其可选的机器范围内随机重选一个机床加工。

(9)局部搜索算子 9 (概率重选机床)：同样地，把一个工序所有可行机器的加工时间通过归一化转换成选择概率。随机选择关键路径上的工序，采用轮盘赌选择一个加工机器，将当前工序移到所选机器加工。

4.3.5 基于 DQN 的算子选择模型

动机：在上节中提出了 9 种基于问题知识的局部搜索算子，显然每种算子针对的优化场景和最适合优化的解是不同的。如何组织这 9 个算子使得每个解在进化过程中选择最合适的算子是关键问题。在第二章中提出的基于强化学习的自学习机制过度依赖于状态集的设计，在第三章中提出的基于意外流行算法的自学习

算子选择机制的粒度太粗。因此，为了摆脱专家知识的依赖以及最细粒度的为每个算子定制局部搜索策略。本节提出了基于深度 Q 网络的算子选择机制。

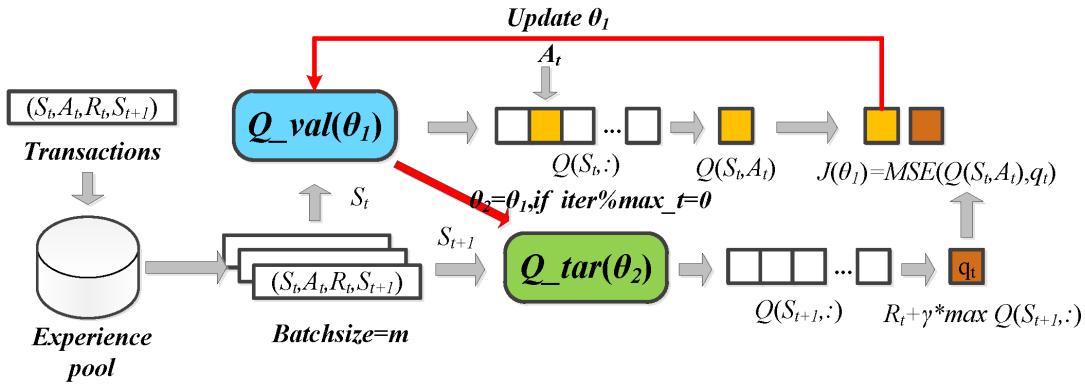

图 4.3 DQN 训练过程

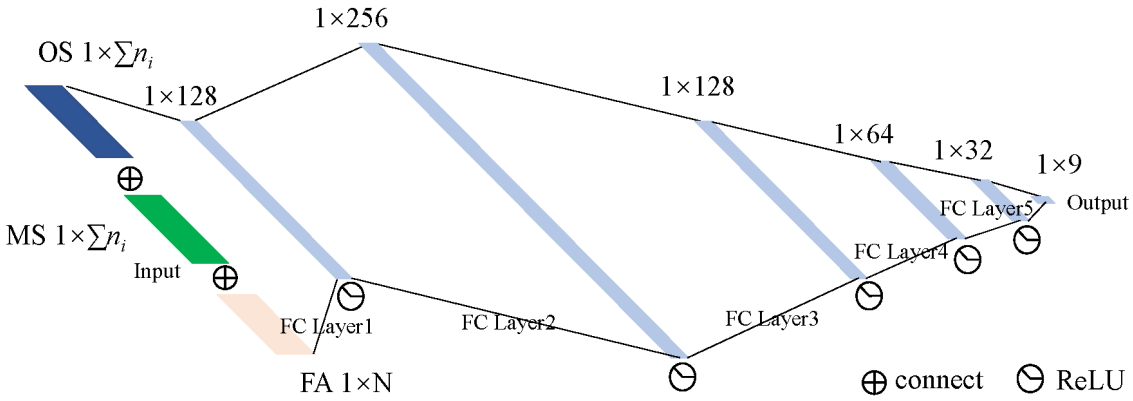

图 4.4 神经网络结构

深度 Q 网络（Deep Q-Networks，DQN）[195]算法作为一种强大的非监督的强化学习算法，可以自动学习解空间的数据分布，并根据数据分布建立起和算子选择概率之间的映射关系。在 DQN 中神经网络代替了传统 Q 学习中的 Q 表存储 Q 值信息，解决了人工设计状态集和 Q 表的困难。DQN 的训练过程如图 4.3 所示，在 DQN 中有两个带参数的多层感知机（Multilayer perceptron，MLP）分别是评价网络 Q_val(θ_1)和目标网络 Q_tar(θ_2)。在训练开始前会随机选择动作，并将每次决策后的经验(S_t,A_t,R_t,S_{t+1})存入经验池中。当经验池中的数据达到阈值，本章中设定为 50，则 DQN 开始训练。从经验池中随机选择一个批次的数据。将一个批次的 S_t 提取出，输入评价网络 Q_val(θ_1)评估得到一系列 Q 值，根据 A_t 选择对应索引的 Q 值 Q(S_t, A_t)。随后，取出批次中的 S_{t+1} 向量输入目标网络 Q_tar(θ_2)中得到下一个状态的所有动作的 Q 值 Q(S_{t+1}，:)，选择当前 Q 值最大的动作。结合奖励 R_t 和折扣因子 γ 输入以下公式计算损失函数。损失函数采用均方根误差函数，采用 Adam 优

化器以学习率 α 进行梯度下降，对是评价网络 Q_val(θ_1)的参数进行更新和优化。当目标网络 Q_tar(θ_2)已经有 max_t 代没有更新参数则将 Q_val(θ_1)的参数复制给 Q_tar(θ_2)。

$$J(\theta_1)=\frac{1}{m}\sum_{i=1}^{m}(Q(S_t,A_t)-(R_t+\gamma*\max(Q(S_{t+1},:))))^2 \tag{7-17}$$

DQN 算子选择模型：在本章中，为了表征一个调度序列的全部信息，将三个向量(OS，MS，FA)拼接为一个向量作为当前调度序列的状态 S_t。动作即为每个局部搜索算子。当执行局部搜索算子成功替换旧解时，奖励 R_t=5；找到相互不支配的解时奖励 R_t=10；否则奖励 R_t=0。执行局部搜索后找到的邻域解(OS’，MS’，FA’)作为调度序列的下一个状态 S_{t+1}。所有信息整合为一维向量(S_t, A_t, R_t, S_{t+1})存入经验池中。本章，采用的 Q 网络的结构如图 4.4 所示。总共由 5 层全连接层组成，激活函数为 ReLU，由窄到宽再到窄的结构更有利于特征提取。算法 4-1 详细讲述了基于 DQN 的算子模型的过程。在算法中扫描数据次数 epoch 设置为 1。

算法4 - 1: 基于DQN的算子选择模型.

输入: 寄生种群($\mathcal{P}$), 贪心因子(ϵ), 经验池大小S_E, epochs, and Q 网络$Q(\theta)$.
输出: Q 网络$Q(\theta)$ 和寄生种群($\mathcal{P}$)

```
for l= 1 to L do
    S_t ← [OS, MS, FA] ← P(l).
    q_t ← Q_θ(S_t).
    if rand > ε then
        随机选择一个邻域结构N_i as A_t, i ∈ [1, 9]
    else
        A_t 选择N_i 中有最大q_t(i), i ∈ [1, 9].
    end
    S_{t+1} ← [OS', MA', FA'] ← A_t(S_t).
    if S_{t+1} 支配S_t then
        P(l) ← S_{t+1}.
        R_t = 5.
    else
        if S_{t+1} 和S_t 互不支配 then
            P(l) ← P(l) ∪ S_{t+1}.
            R_t = 10.
        else
            R_t = 0.
        end
    end
    存储经验τ(S_t, A_t, R_t, S_{t+1}) 到经验池.
    if 经验池容量大于阈值S_E then
        for e= 1 to epochs do
            通过公式(4-17)训练神经网络Q(θ).
        end
    end
end
```

4.3.6 DQPEA 算法流程

算法 4-2 详细讲述了 DQPEA 的算法流程。首先初始化宿主种群 H 和寄生种

群 P，并初始化 DQN 网络。其次，经过 NSGA-II 的算法流程对 H 种群执行全局搜索和环境选择。随后将 H 种群的非支配解存入 P 中，在 P 种群中的每个个体选择局部搜索算子。并将执行算子后的解与原解比较确定奖励。将该轮决策中的当前状态 S_t，当前执行的局部搜索算子 A_t，当前获得的奖励 R_t 和下一个状态 S_{t+1}。组合成经验存入经验池中，如果经验池满了，则训练 DQN 网络。然后，对寄生种群 P 中的所有个体执行全主动解码的节能策略降低能耗。重复 5-30 行直到满足停止条件。最后从寄生种群 P 中输出最后的非支配解集 PF。

算法4 - 2: DQPEA的算法框架.

输入: 最大函数评价次数($MaxNFEs$), 种群大小(ps), 批量大小(bs), 贪心因子(ϵ), 学习速率(α), 折扣因子(γ), 参数替换阈值(max_t), 经验池大小S_E, epochs, 动作数(N_A), 交叉概率(P_c), 突变概率(P_m)

输出: 非支配解集PF

$\mathcal{H}_0 \leftarrow$ 随机初始化宿主种群(ps).
$\mathcal{P}_0 \leftarrow \varnothing$.//初始化寄生种群
$\mathbb{Q}(\theta) \leftarrow$建立DQN网络($bs, \epsilon, \alpha, \gamma, max_t, S_E, N_A$).
$t = 1$.
while $NFEs \leqslant MaxNFEs$ do
  $B_t \leftarrow$双人锦标赛选择 ($\mathcal{H}_t$).
  for i=1 to ps do
    $[X_1, X_2] \leftarrow$ 进化算子 ($\mathcal{H}_t(i), B_t, P_c, P_m$).
    $C_t \leftarrow C_t \cup X_1 \cup X_2$.
    $NFEs \leftarrow NFEs + 2$.
  end
  $C_t \leftarrow$ 删除重复解 ($C_t \cup \mathcal{H}_t$).
  $\{F_0, ..., F_{last}, ...\} \leftarrow$ 快速非支配排序 (C_t, ps).
  $C_t^{'} \leftarrow$技术拥挤度距离 ($F_0, ..., F_{last}$).
  $\mathcal{H}_{t+1} \leftarrow C_t^{'}(1:ps, :)$.
  $\mathcal{P}_{t+1} \leftarrow$ 删除重复解 ($\mathcal{P}_t \cup F_0$).
  for i to $|P_{t+1}|$ do
    $S_t \leftarrow (OS, MS, FA)$.
    $A_t \leftarrow \epsilon$-greedy算子选择($\mathbb{Q}(\theta), S_t, \epsilon$).
    $S_{t+1} \leftarrow$局部搜索 ($\mathcal{P}_{t+1}(i), A_t$).
    $R_t \leftarrow$比较非支配关系(S_{t+1}, S_t).
    $\mathcal{T} \leftarrow (S_t, A_t, R_t, S_{t+1})$.
    $\mathbb{Q}(\theta)$.存入经验池($\mathcal{T}$) if 经验池内数量大于S_E then
      训练DQN网络($\mathbb{Q}(\theta)$).
    end
  end
  $\mathcal{P}_{t+1} \leftarrow$节能策略 ($\mathcal{P}_{t+1}$).
  $NFEs \leftarrow NFEs + 2 * |\mathcal{P}_{t+1}|$.
  $\mathcal{P}_{t+1} \leftarrow$ 快速非支配排序 ($\mathcal{P}_{t+1}$).
  $t = t + 1$.
end
$PF \leftarrow$ 快速非支配排序 ($\mathcal{P}_{t+1}$).

4.4 数值实验

4.4.1 实验设置

在上节中已经详细介绍了算法 DQPEA 的细节，本节设计了详细的实验来验证

所提出算法的有效性。在本节中，生成了不同规模的测试问题共 20 个。首先，通过田口正交实验对算法的最优参数选择进行了讨论。再次，经过成分分离试验验证 SPAMA 中每个改进点的有效性。最后，通过与最新的相关算法对比验证了 SPAMA 在 GDFJSP 的求解上表现最好，并通过图表对比详细讨论了算法性能优越的原因。所有算法没有 DQN 的算法都用 MATLAB2020Rb 和并行工具箱编程，所有包含 DQN 的算法都用 Python 编程，运行环境为 python3.6，CUDA1.1，Pytorch 框架，用 Pycharm2018 运行。所有算法都在 Intel(R) Xeon(R) Gold 6246R CPU@ 3.4GHz 364G RAM，NVIDIA GeForce RTX 3090 GPU，操作系统 Win10 的工作站上运行。为了公平起见，每个算法独立运行 20 次，通过收集收敛性，分布性和综合指标的均值和方差来衡量算法性能。

4.4.2 测试问题和指标

为了验证 DQPEA 的有效性，本章生成了 20 个测试问题。n $\in \{10,20,30,40,50,100,150,200\}$ 和工厂数范围 $n_f \in \{2,3,4,5,6,7\}$，每个工厂有 5 台机器。每个工件包含 5 个工序，每个工序可选机器范围有限。每个工件的加工时间 $P_{i,j,f,k} \in [5,20]$ 且在不同的工厂中加工时间不同。工厂机器的加工功率 P_W=4，机器的空闲功率 P_I=1。与第二章一样，本章的实验指标采用 HV，GD，Spread 来衡量算法的综合性能，收敛性和分布性。所有算法的停止条件是 MaxNFEs=200* $\sum_{i=1}^{n} n_i$ 。

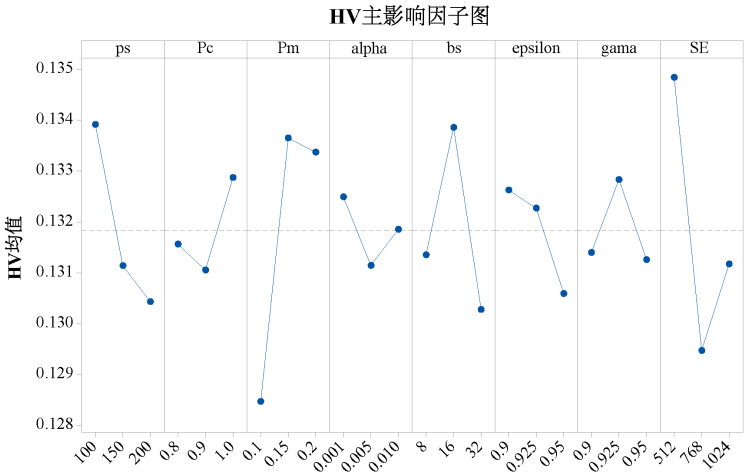

图 4.5 HV 主影响因子图

4.4.3　参数实验

算法参数是影响算法求解问题的性能最直接的因素。DQPEA 共有 8 个参数分别是种群大小 ps，学习速率α，批次大小 bs，贪心因子 ε，折扣因子 γ，经验池大小 S_E，交叉概率 P_c 和突变概率 P_m。本章采用田口(Taguchi)正交实验[186]来测试最优的参数组合。不同参数的级别设置如下：ps∈{10 0,150,200}, P_c∈{0.8,0.9,1}, P_m∈{0.1,0.15,0.2}, α∈{0.001,0.005,0.01}, ε∈{0.9,0.925, 0.95}，γ∈{0.9,0.925,0.95}，S_E∈{512,768,1024}。生成了正交实验表 $L_{27}(8^3)$来矫正最优参数选择。为了保证公平，每个参数组合在 20 个测试问题上独立运行 20 次。同时，收集三个指标的均值，并采用 Minitab18 进行田口实验分析，得出指标的主影响图。如图 4.5，4.6 和 4.7 所示，根据三个指标的综合评判，最后的最优参数组合为 ps=100，P_c=1，P_m=0.2，α=0.001，bs=16，ε=0.9，γ=0.9，S_E=512。

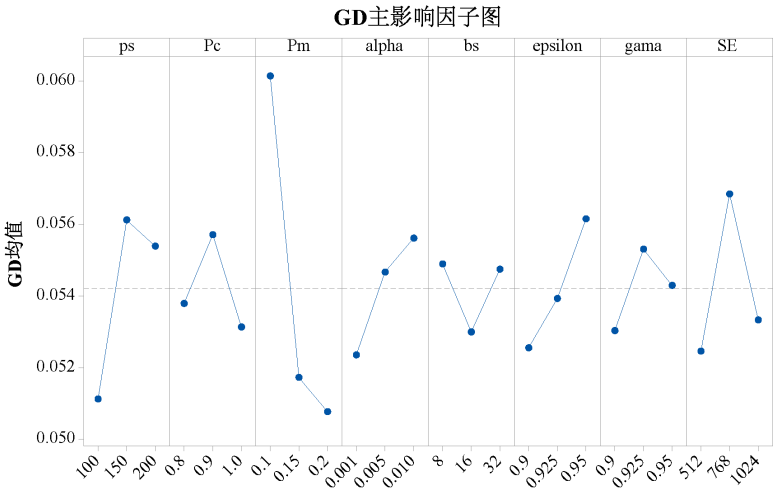

图 4.6 GD 主影响因子图

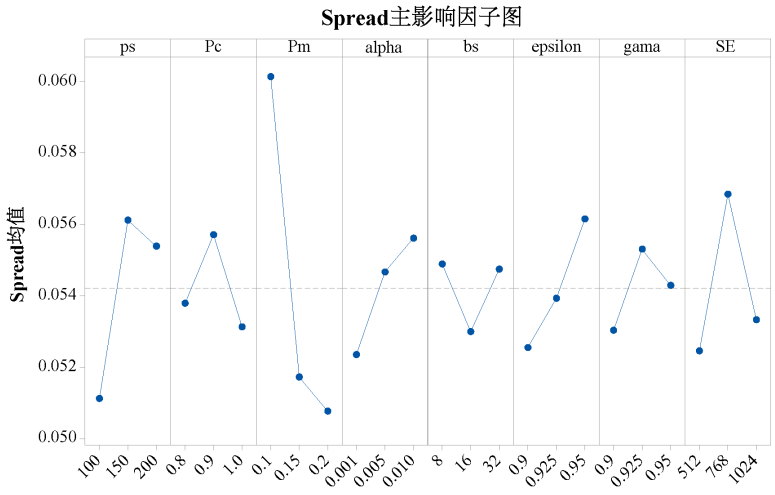

图 4.7 Spread 主影响因子图

4.4.4 成分分离实验

为了验证 SPAMA 每个改进点的有效性，本节设计了四个 DQPEA 的变种算法分别是：PEA 表示没有 DQN 算子选择模型而是随机算子选择的寄生进化算法，PEA-V9 表示在 PEA 的基础上删除随机局部搜索算子选择的变邻域搜索，PEA-ES 表示在 PEA 的基础上删除全主动调度解码节能策略，MA 表示将全局搜索和局部搜索都糅合在一个主种群执行同时对所有个体执行节能策略。为了公平起见，每个变种算法在 20 个测试问题上独立运行 20 次，所有算法的停止条件是 MaxNFEs=200* $\sum_{i=1}^{n} n_i$ 。除了 DQPEA 用 python 运行其他变体算法都用 matlab 编程和运行。

表 4.1 DQPEA 变体算法 Frideman 秩和检验统计结果(置信度 α=0.05)

MOEAs	HV		GD		Spread	
	rank	*p*-value	rank	*p*-value	rank	*p*-value
MA	5.00		5.00		3.75	
PEA-V9	3.45		3.75		3.75	
PEA-ES	2.95	1.49E-29	2.75	3.35E-29	2.20	6.80E-04
PEA	2.60		2.55		3.10	
DQPEA	**1.00**		**1.00**		**2.20**	

为了验证改进点的有效性，本章采用 KEEL 软件对变体算法之间的显著性进行了统计和分析。表 4.1 展示了 DQPEA 变体算法的 Frideman 秩和检验统计结果。如表所示**加粗**表示排名最高的算法，其中对比 PEA，PEA-V9 和 PEA-ES 可以验证提出问题模型驱动的局部搜索算子和节能策略的有效性。通过对比 PEA 和 MA，所有指标全面提升，说明所提出的寄生进化框架能更好地求解分布式调度问题，验证了传统 MA 框架的缺陷。通过对比 DQPEA 和 PEA，在所有指标上显著地提升，说明提出的 DQN 算子选择模型可以有效地通过自学习为每个解选择最合适的算子，验证了提出 DQN 算子选择模型的有效性。同时 p 值小于 0.05 说明算法之间具有显著性的差异，说明 DQPEA 的每个改进点显著性地有效。同时表 4.2 详细展示了每个指标的均值和标准差在独立运行 20 次后的统计结果。灰色和加粗表示最好。“-/=/+”表示各变体算法在该测试问题上显著性地“差于/无显著性/好于”DQPEA。可以得知 DQPEA 在所有测试上超过一半优于各变体算法，进一步证明了改进点的有效性。

表 4.2 DQPEA 变体算法 HV(max)指标对比结果

Instances	MA		PEA-V9		PEA-ES		PEA		DQPEA	
	mean	std	mean	std	mean	std	mean	std	mean	std
10J2F	0.105-	0.020	0.13=	0.015	0.135-	0.011	0.134=	0.015	**0.147**	0.015
20J2F	0.098-	0.017	0.137-	0.010	0.133-	0.017	0.138-	0.018	**0.161**	0.008
20J3F	0.183-	0.027	0.241-	0.016	0.231-	0.013	0.238-	0.024	**0.274**	0.018
30J2F	0.075-	0.014	0.095-	0.011	0.102-	0.012	0.102-	0.013	**0.132**	0.010
30J3F	0.085-	0.020	0.124-	0.016	0.125-	0.012	0.127-	0.012	**0.152**	0.014
40J2F	0.067-	0.011	0.092-	0.008	0.093-	0.005	0.089-	0.011	**0.121**	0.005
40J3F	0.073-	0.019	0.108-	0.016	0.113-	0.011	0.11-	0.015	**0.146**	0.011
40J4F	0.08-	0.015	0.132-	0.020	0.129-	0.017	0.126-	0.009	**0.172**	0.010
50J3F	0.076-	0.014	0.097-	0.015	0.109-	0.014	0.113-	0.011	**0.141**	0.013
50J4F	0.091-	0.020	0.135-	0.011	0.133-	0.017	0.135-	0.016	**0.177**	0.008
50J5F	0.112-	0.032	0.156-	0.011	0.173-	0.016	0.168-	0.014	**0.201**	0.009
100J4F	0.054-	0.011	0.089-	0.009	0.091-	0.012	0.094-	0.014	**0.126**	0.008
100J5F	0.077-	0.016	0.101-	0.012	0.113-	0.011	0.115-	0.007	**0.150**	0.010
100J6F	0.066-	0.008	0.101-	0.016	0.099-	0.010	0.098-	0.014	**0.145**	0.011
100J7F	0.07-	0.010	0.094-	0.014	0.103-	0.011	0.101-	0.011	**0.137**	0.013
150J5F	0.066-	0.008	0.087-	0.015	0.088-	0.010	0.095-	0.009	**0.135**	0.009
150J6F	0.056-	0.010	0.087-	0.008	0.084-	0.006	0.085-	0.010	**0.118**	0.007
150J7F	0.071-	0.011	0.096-	0.011	0.097-	0.009	0.1-	0.007	**0.149**	0.006
200J6F	0.053-	0.008	0.082-	0.008	0.085-	0.011	0.093-	0.005	**0.120**	0.007
200J7F	0.066-	0.007	0.083-	0.008	0.089-	0.012	0.09-	0.015	**0.130**	0.007
-/=/+	20/0/0		19/1/0		20/0/0		19/1/0			

表 4.3 DQPEA 变体算法 GD(min)指标对比结果

Instances	MA		PEA-V9		PEA-ES		PEA		DQPEA	
	mean	std	mean	std	mean	std	mean	std	mean	std
10J2F	0.074-	0.035	0.041=	0.018	0.046-	0.021	0.035=	0.020	**0.022**	0.013
20J2F	0.067-	0.024	0.03-	0.011	0.033-	0.020	0.027-	0.013	**0.008**	0.008
20J3F	0.089-	0.011	0.053-	0.019	0.052-	0.011	0.056-	0.025	**0.025**	0.013
30J2F	0.102-	0.033	0.075-	0.019	0.058-	0.014	0.058-	0.014	**0.028**	0.016
30J3F	0.067-	0.031	0.038-	0.011	0.046-	0.025	0.043-	0.018	**0.016**	0.007
40J2F	0.065-	0.023	0.042-	0.015	0.028-	0.007	0.036-	0.010	**0.007**	0.004
40J3F	0.075-	0.022	0.047-	0.020	0.037-	0.011	0.041-	0.015	**0.010**	0.009
40J4F	0.111-	0.026	0.07-	0.036	0.061-	0.019	0.07-	0.015	**0.024**	0.016
50J3F	0.085-	0.019	0.072-	0.018	0.048-	0.016	0.046-	0.020	**0.018**	0.013
50J4F	0.1-	0.040	0.064-	0.026	0.052-	0.018	0.058-	0.018	**0.016**	0.009

表 4.3 DQPEA 变体算法 GD(min)指标对比结果

Instances	MA		PEA-V9		PEA-ES		PEA		DQPEA	
	mean	std	mean	std	mean	std	mean	std	mean	std
50J5F	0.097-	0.047	0.047-	0.011	0.03-	0.011	0.044-	0.014	**0.011**	0.010
100J4F	0.116-	0.026	0.063-	0.022	0.046-	0.013	0.052-	0.023	**0.016**	0.009
100J5F	0.102-	0.045	0.059-	0.021	0.038-	0.011	0.036-	0.004	**0.010**	0.006
100J6F	0.095-	0.025	0.065-	0.030	0.061-	0.009	0.059-	0.016	**0.018**	0.016
100J7F	0.108-	0.036	0.065-	0.014	0.042-	0.017	0.047-	0.012	**0.014**	0.011
150J5F	0.079-	0.014	0.062-	0.032	0.047-	0.014	0.038-	0.010	**0.008**	0.007
150J6F	0.103-	0.042	0.056-	0.026	0.043-	0.007	0.04-	0.009	**0.012**	0.008
150J7F	0.092-	0.027	0.056-	0.018	0.049-	0.015	0.046-	0.010	**0.006**	0.004
200J6F	0.091-	0.022	0.051-	0.020	0.037-	0.015	0.03-	0.007	**0.006**	0.005
200J7F	0.094-	0.022	0.066-	0.012	0.046-	0.012	0.04-	0.012	**0.009**	0.007
-/=/+	20/0/0		19/1/0		20/0/0		19/1/0			

表 4.4 DQPEA 变体算法 Spread(min)指标对比结果

Instances	MA		PEA-V9		PEA-ES		PEA		DQPEA	
	mean	std	mean	std	mean	std	mean	std	mean	std
10J2F	0.958=	0.102	**0.928=**	0.117	0.935=	0.101	0.97=	0.162	0.974	0.113
20J2F	0.975=	0.105	0.959=	0.083	0.903=	0.209	1.001=	0.084	**0.927**	0.307
20J3F	1.014=	0.055	**0.952=**	0.118	0.975=	0.071	1.048=	0.080	0.980	0.327
30J2F	0.977=	0.060	0.979=	0.059	0.969=	0.071	**0.968=**	0.074	0.993	0.261
30J3F	0.965=	0.035	0.977=	0.105	**0.963=**	0.060	0.976=	0.075	0.993	0.144
40J2F	0.976=	0.044	0.968=	0.040	1.012=	0.163	1.032=	0.089	**0.845**	0.147
40J3F	**0.965=**	0.039	1.033=	0.070	0.966=	0.065	0.99=	0.074	0.992	0.170
40J4F	1.016=	0.077	1.048=	0.101	0.968=	0.042	0.985=	0.104	**0.968**	0.153
50J3F	1.015=	0.068	0.999=	0.047	1.011=	0.109	1.003=	0.126	**0.962**	0.242
50J4F	0.966=	0.039	0.985=	0.044	0.941=	0.078	0.961=	0.084	**0.900**	0.226
50J5F	0.996=	0.048	0.995=	0.059	0.939=	0.071	1.017=	0.085	**0.930**	0.115
100J4F	1.001=	0.021	0.992-	0.045	0.976=	0.039	0.986=	0.030	**0.814**	0.306
100J5F	0.994=	0.010	1.009-	0.029	**0.922=**	0.070	0.957=	0.060	0.967	0.046
100J6F	0.975=	0.025	0.995=	0.046	0.982=	0.063	0.983=	0.073	**0.902**	0.279
100J7F	0.986=	0.016	0.982=	0.022	0.961=	0.049	0.971=	0.039	**0.911**	0.076
150J5F	0.99=	0.013	0.997=	0.040	0.967=	0.034	0.962=	0.051	**0.895**	0.187
150J6F	0.998=	0.024	0.982=	0.032	0.982=	0.043	**0.965=**	0.080	0.976	0.224
150J7F	0.994=	0.029	1.003=	0.028	0.964=	0.040	0.987=	0.083	**0.938**	0.188
200J6F	0.988=	0.015	0.987=	0.035	0.952=	0.051	0.978=	0.073	**0.842**	0.188
200J7F	0.989=	0.014	0.997=	0.033	0.975=	0.061	**0.948=**	0.071	0.983	0.167
-/=/+	0/20/0		0/20/0		0/20/0		0/20/0			

4.4.5 对比实验

为了进一步验证 DQPEA 每个改进点的有效性，本节选取了几类 MOEAs 作为对比算法分别是：（1）主流 MOEAs：MOEA/D[182]和 NSGA-II[183]；（2）近期提出的 MOEAs：TS-NSGA-II[196]；（3） DFJSP 相关算法：HSLFA[135]；（4）基于强化学习的进化算法：LRVMA[109]和 MOEA/D-DQN[197]。每个算法的参数设置如下：所有算法的种群大小 ps=100，交叉概率 P_c=1，突变概率 P_m=0.2。MOEA/D 和 MOEA/D-DQN 的邻域更新范围 T=10，LRVMA 和 MOEA/D-DQN 的强化学习相关参数与 DQPEA 相同。为了对比公平，每个算法独立运行 20 次，停止条件为最大迭代次数 MaxNFEs=200*$\sum_{i=1}^{n} n_i$ 。表 4.5 展示了所有对比算法的 Frideman 秩和检验统计结果，结果显示 DQPEA 在 HV 和 GD 指标上排名第 1 显著地好于所有对比算法，在 Spread 指标上排名第 4。表 4.6，4.7 和 4.8 展示了所有指标的均值和标准差的统计结果，“-/=/+”表示各变体算法在该测试问题上显著性地“差于/无显著性/好于”SPAMA。结果显示 SPAMA 在 HV 和 GD 指标上有超过一半以上的测试集的表现显著好于所有对比算法，验证了本章提出算法的有效性。

表 4.5 DQPEA 所有对比算法的 Frideman 秩和检验统计结果(置信度 α=0.05)

MOEAs	HV		GD		Spread	
	rank	*p*-value	rank	*p*-value	rank	*p*-value
NSGA-II	5.75		5.25		3.50	
MOEA/D	2.80		3.00		5.40	
TS-NSGA-II	4.30		3.80		3.25	
HSLFA	4,55	2.10E-19	5.00	9.14E-19	4.65	8.26E-08
LRVMA	2.65		2.95		5.35	
MOEA/D-DQN	6.95		7.00		3.95	
DQNPEA	**1.00**		**1.00**		**1.90**	

表 4.6 DQPEA 所有对比算法 HV(max)指标对比结果

Ins	NSGA-II		MOEA/D		TS-NSGA-II		HSLFA		LRVMA		MOEA/D-DQN		DQPEA	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
10J2F	0.18-	0.023	0.166-	0.019	0.185-	0.020	0.163-	0.021	0.164-	0.019	0.146-	0.020	**0.219**	0.015
20J2F	0.136-	0.023	0.154-	0.014	0.147-	0.012	0.133-	0.017	0.16-	0.017	0.115-	0.021	**0.186**	0.008
20J3F	0.193-	0.025	0.203-	0.026	0.213-	0.026	0.185-	0.029	0.196-	0.022	0.176-	0.025	**0.271**	0.016
30J2F	0.143-	0.015	0.148-	0.013	0.156-	0.016	0.142-	0.013	0.154-	0.012	0.111-	0.021	**0.198**	0.011
30J3F	0.137-	0.022	0.148-	0.017	0.146-	0.019	0.147-	0.020	0.151-	0.019	0.108-	0.020	**0.201**	0.014
40J2F	0.121-	0.014	0.133-	0.010	0.126-	0.014	0.123-	0.013	0.136-	0.007	0.096-	0.018	**0.172**	0.007
40J3F	0.125-	0.026	0.149-	0.012	0.143-	0.016	0.131-	0.016	0.151-	0.012	0.104-	0.024	**0.196**	0.010
40J4F	0.139-	0.027	0.157-	0.015	0.152-	0.021	0.147-	0.016	0.157-	0.013	0.118-	0.023	**0.217**	0.011
50J3F	0.146-	0.017	0.172-	0.013	0.155-	0.018	0.163-	0.014	0.167-	0.013	0.133-	0.019	**0.212**	0.012
50J4F	0.125-	0.022	0.145-	0.014	0.139-	0.019	0.138-	0.016	0.144-	0.015	0.108-	0.018	**0.201**	0.011
50J5F	0.148-	0.031	0.173-	0.018	0.155-	0.016	0.165-	0.021	0.178-	0.014	0.134-	0.018	**0.229**	0.014
100J4F	0.102-	0.019	0.138-	0.008	0.12-	0.013	0.12-	0.011	0.135-	0.006	0.09-	0.017	**0.171**	0.009
100J5F	0.087-	0.019	0.105-	0.013	0.093-	0.011	0.102-	0.014	0.108-	0.008	0.079-	0.015	**0.149**	0.009
100J6F	0.096-	0.018	0.117-	0.007	0.102-	0.014	0.117-	0.014	0.123-	0.011	0.087-	0.017	**0.165**	0.010
100J7F	0.107-	0.012	0.134-	0.009	0.112-	0.016	0.13-	0.012	0.138-	0.009	0.096-	0.021	**0.183**	0.012
150J5F	0.095-	0.013	0.126-	0.006	0.101-	0.010	0.116-	0.008	0.125-	0.008	0.08-	0.019	**0.160**	0.008
150J6F	0.098-	0.013	0.136-	0.008	0.109-	0.011	0.122-	0.009	0.135-	0.008	0.089-	0.015	**0.166**	0.007
150J7F	0.091-	0.010	0.142-	0.009	0.115-	0.014	0.132-	0.010	0.143-	0.009	0.095-	0.021	**0.177**	0.009
200J6F	0.096-	0.016	0.142-	0.008	0.105-	0.008	0.125-	0.007	0.135-	0.007	0.089-	0.020	**0.167**	0.007
200J7F	0.087-	0.008	0.127-	0.012	0.099-	0.008	0.114-	0.007	0.126-	0.010	0.08-	0.018	**0.159**	0.007
-/=/+	20/0/0		20/0/0		20/0/0		20/0/0		20/0/0		20/0/0			

表 4.7 DQPEA 所有对比算法 GD(min)指标对比结果

Ins	NSGA-II		MOEA/D		TS-NSGA-II		HSLFA		LRVMA		MOEA/D-DQN		DQPEA	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
10J2F	0.047-	0.023	0.065-	0.030	0.039-	0.016	0.073-	0.028	0.068-	0.023	0.077-	0.024	**0.014**	0.008
20J2F	0.051-	0.023	0.036-	0.014	0.039-	0.010	0.059-	0.023	0.031-	0.018	0.077-	0.025	**0.009**	0.008
20J3F	0.084-	0.020	0.089-	0.029	0.066-	0.020	0.1-	0.040	0.099-	0.027	0.107-	0.028	**0.027**	0.013
30J2F	0.081-	0.026	0.089-	0.025	0.057-	0.016	0.083-	0.023	0.081-	0.022	0.122-	0.035	**0.026**	0.012
30J3F	0.06-	0.024	0.052-	0.012	0.051-	0.019	0.067-	0.024	0.047-	0.014	0.094-	0.029	**0.016**	0.007
40J2F	0.052-	0.019	0.039-	0.012	0.044-	0.012	0.055-	0.017	0.038-	0.011	0.072-	0.020	**0.008**	0.005
40J3F	0.063-	0.024	0.044-	0.016	0.044-	0.010	0.07-	0.023	0.044-	0.013	0.108-	0.060	**0.009**	0.007
40J4F	0.087-	0.024	0.086-	0.025	0.071-	0.019	0.092-	0.026	0.088-	0.019	0.126-	0.031	**0.020**	0.013
50J3F	0.074-	0.018	0.057-	0.019	0.061-	0.022	0.061-	0.020	0.059-	0.013	0.088-	0.025	**0.014**	0.009
50J4F	0.097-	0.031	0.082-	0.021	0.075-	0.023	0.097-	0.033	0.089-	0.030	0.125-	0.037	**0.022**	0.010
50J5F	0.081-	0.036	0.066-	0.025	0.07-	0.018	0.075-	0.025	0.058-	0.017	0.092-	0.026	**0.015**	0.008
100J4F	0.084-	0.030	0.042-	0.009	0.054-	0.014	0.066-	0.018	0.046-	0.008	0.105-	0.030	**0.010**	0.006
100J5F	0.08-	0.040	0.057-	0.020	0.064-	0.014	0.061-	0.021	0.049-	0.011	0.084-	0.030	**0.011**	0.006
100J6F	0.093-	0.036	0.082-	0.022	0.088-	0.028	0.083-	0.033	0.075-	0.023	0.12-	0.049	**0.018**	0.012
100J7F	0.085-	0.033	0.058-	0.021	0.074-	0.020	0.075-	0.025	0.05-	0.009	0.094-	0.033	**0.014**	0.008
150J5F	0.077-	0.017	0.048-	0.013	0.068-	0.014	0.057-	0.016	0.047-	0.014	0.105-	0.040	**0.009**	0.006
150J6F	0.078-	0.034	0.035-	0.013	0.064-	0.016	0.052-	0.013	0.037-	0.012	0.097-	0.028	**0.008**	0.005
150J7F	0.08-	0.023	0.043-	0.010	0.073-	0.026	0.059-	0.013	0.045-	0.016	0.116-	0.044	**0.009**	0.006
200J6F	0.074-	0.022	0.026-	0.008	0.057-	0.014	0.046-	0.011	0.033-	0.011	0.087-	0.032	**0.006**	0.004
200J7F	0.092-	0.021	0.052-	0.021	0.078-	0.022	0.06-	0.016	0.047-	0.015	0.107-	0.033	**0.010**	0.006
-/=/+	20/0/0		20/0/0		20/0/0		20/0/0		20/0/0		20/0/0			

表 4.8 DQPEA 所有对比算法 Spread(min)指标对比结果

Ins	NSGA-II		MOEA/D		TS-NSGA-II		HSLFA		LRVMA		MOEA/D-DQN		DQPEA	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
10J2F	0.965=	0.115	1.012=	0.102	**0.924=**	0.121	1.028=	0.079	0.987=	0.093	0.991=	0.098	0.968	0.201
20J2F	0.941=	0.107	0.971=	0.091	0.962=	0.098	1.008=	0.077	0.95=	0.083	0.994=	0.050	**0.915**	0.183
20J3F	0.99=	0.060	1.01=	0.040	0.976=	0.067	0.979=	0.062	1.005=	0.033	0.982=	0.049	**0.963**	0.256
30J2F	0.987=	0.042	1.031=	0.084	**0.968=**	0.065	0.98=	0.036	1.009=	0.044	1.01=	0.043	0.989	0.155
30J3F	0.991=	0.041	1.027=	0.098	**0.974=**	0.083	0.994=	0.043	1.004=	0.056	0.987=	0.027	1.010	0.160
40J2F	0.983=	0.038	0.965=	0.070	0.974-	0.026	0.999=	0.036	0.992=	0.063	0.972-	0.031	**0.839**	0.148
40J3F	0.977=	0.042	1.005=	0.048	0.948=	0.055	1.002=	0.022	0.989=	0.062	0.983=	0.032	**0.926**	0.172
40J4F	1.001=	0.047	0.984=	0.042	0.988=	0.046	0.996=	0.033	1.002=	0.061	0.996=	0.026	**0.980**	0.136
50J3F	1=	0.048	1.002=	0.040	0.987=	0.038	1=	0.027	0.994=	0.045	0.984=	0.027	**0.961**	0.188
50J4F	**0.98=**	0.032	1.02=	0.051	1.003=	0.047	0.997=	0.058	1.006=	0.054	0.997=	0.033	1.048	0.283
50J5F	0.993=	0.032	1.005=	0.030	0.996=	0.028	0.999=	0.023	1.025=	0.084	0.985=	0.028	**0.976**	0.090
100J4F	1.003=	0.023	1.017=	0.091	0.99-	0.024	1.006=	0.043	1.008=	0.051	1=	0.031	**0.908**	0.125
100J5F	1.003=	0.018	1.006=	0.025	0.986=	0.031	**0.971=**	0.051	1.009=	0.060	1.004=	0.055	0.976	0.110
100J6F	0.988=	0.033	0.999=	0.028	0.998=	0.031	0.999=	0.022	1.014=	0.078	0.994=	0.023	**0.956**	0.212
100J7F	1.008=	0.041	1.002=	0.041	1.002=	0.046	**0.959=**	0.039	1.024=	0.079	0.988=	0.037	0.970	0.212
150J5F	0.998-	0.015	0.995=	0.060	0.998-	0.019	1.005=	0.024	0.993=	0.028	0.99=	0.025	**0.895**	0.171
150J6F	0.988=	0.024	0.993=	0.049	1.004=	0.034	1.002=	0.036	1=	0.033	1.011=	0.038	**0.895**	0.199
150J7F	0.974=	0.026	1.018=	0.070	0.992=	0.023	1.005=	0.023	1.016=	0.041	0.993=	0.030	**0.921**	0.138
200J6F	0.992-	0.017	1.005=	0.023	0.986=	0.019	0.995=	0.025	0.999=	0.029	1.007-	0.035	**0.922**	0.149
200J7F	0.99=	0.013	0.99=	0.036	1.012=	0.033	1.018=	0.061	1.012=	0.097	1=	0.013	**0.961**	0.127
-/=/+	2/18/0		0/20/0		3/17/0		0/20/0		0/20/0		2/18/0			

4.4.6 讨论与分析

DQPEA 算法在 GDHFJSP 问题上的成功在于它的设计。首先，除了基于关键路径和关键块的理论可以有效减少 Cmax 外，针对工厂柔性选择和机器柔性选择设计了多种基于概率的局部搜索算子，将加工时间与选择概率相关联，极大地提高了工厂和机器选择的局部搜索的效率。其次，提出了改进的 MA 进化框架名为寄生进化框架，将全局和局部搜索放在两个种群中执行，使得全局搜索不受局部搜索干扰进而充分探索目标空间，同时围绕精英解进行局部搜索保证用少量的计算资源实现高质量搜索。再次，采用基于全主动调度解码的节能策略，该策略可以有效减少空闲时间，即减少了完工时间又有效地降低了能耗。最后，提出了基于 DQN 的算子选择模型，通过训练和学习实现了为每个解定制化的选择最优局部搜索算子，极大增加了算法的性能。

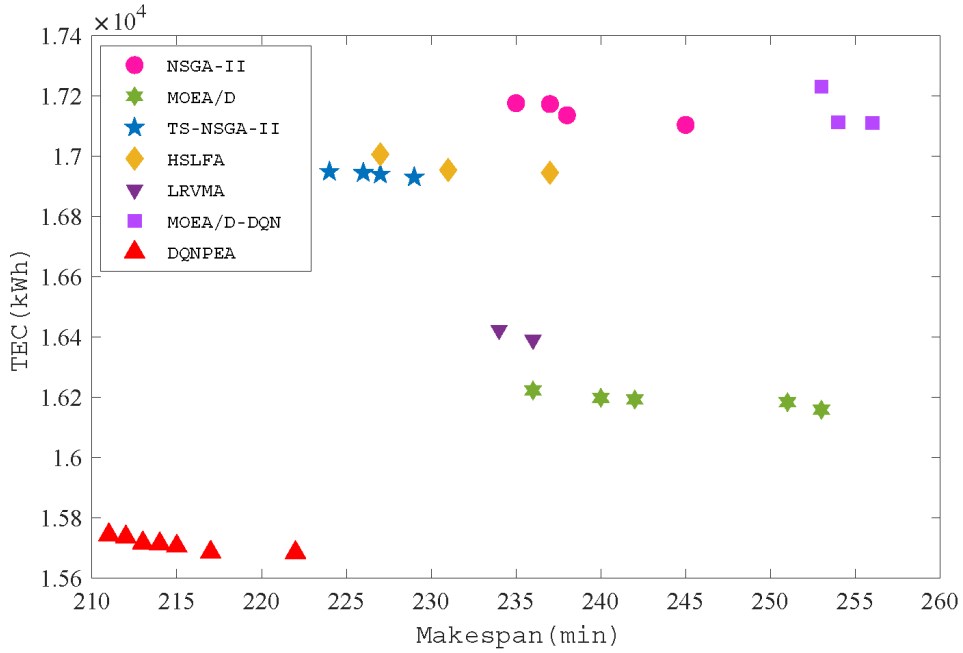

图 4.8 DQPEA 所有对比算法在 100J4F 的 PF 对比

为了进一步说明 DQPEA 算法的有效性，本节还通过图来展示算法性能并分析其有效的原因。图 4.8 展示了所有算法在 100J4F 测试集上求得 HV 值最好的非支配解集，说明 DQPEA 的非支配解集更能探索前沿面，表明了算法的性能最好。图 4.9 展示了 DQPEA 在求解 100J4F 过程中不同算子的选择比率，以每 10 代为一个单位，可以看到在不同进化阶段最优的算子选择是不同的，表示 DQN 算子选择模型可以在不同时期选择最好的算子。图 4.10 展示了 DQPEA 在 50J3F 测试集上找到的 Cmax 最小解的甘特图(Cmax=146h，TEC=7954 kWh)。图 3.13 展示了 DQP EA

在 50J3F 测试集上找到的 TEC 最小解的甘特图(Cmax=154h，TEC=7885kWh)。综上所述，DQPEA 在求解 GDHFJSP 有很好的性能。

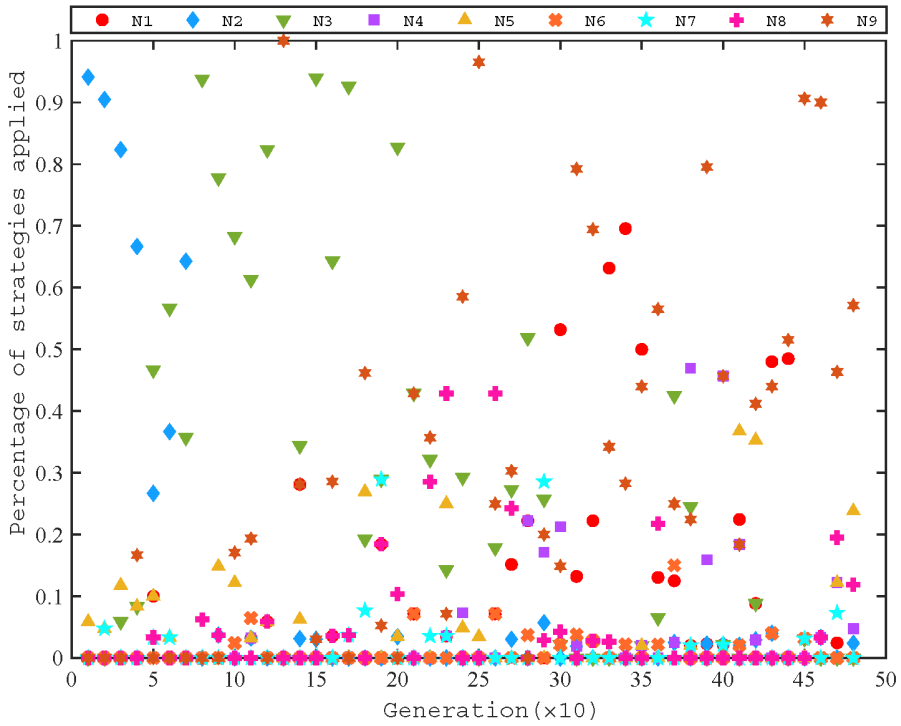

图 4.9 DQPEA 在求解 100J4F 过程中各个算子选择比率

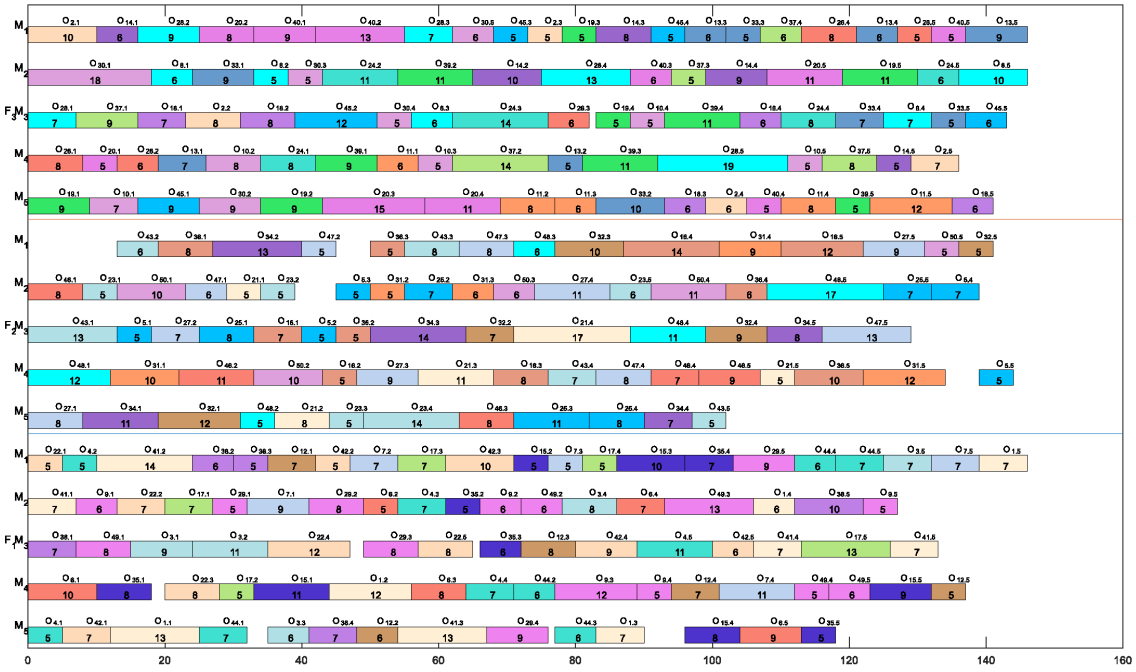

图 4.10 DQPEA 在 50J3F 求得 Cmax 最好解的甘特图

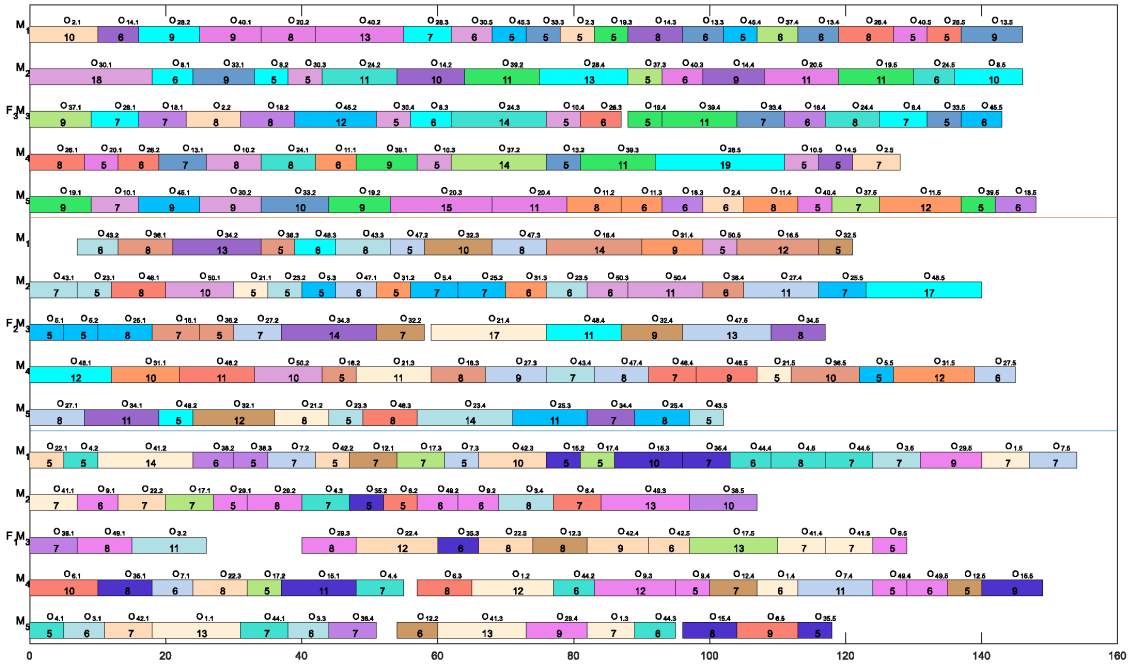

图 4.11 DQPEA 在 50J3F 求得 TEC 最好解的甘特图

4.5 本节小结

本章根据绿色分布式异构柔性作业车间调度问题的特点，提出了基于深度 Q 网络的寄生进化算法，并在生成不同规模的 20 个测试问题上进行测试，同时与一些最先进的算法比较。大量实验结果表明，本章所提的 DQPEA 算法在求解 GDHFJSP 上取得了最好的结果。本章的贡献主要有以下几点：(1) 提出了 9 种基于关键路径和关键块的局部搜索算子，根据工厂柔性和机器柔性的知识设计了基于概率的局部搜索算子，提高了局部搜索的效率，加速算法收敛；(2) 首次提出寄生进化框架求解分布式车间调度问题，将全局搜索和局部搜索放入两个种群执行互不影响，有效地平衡了计算资源，提升了算法性能，具有通用性；(3) 采用了基于全主动调度解码节能策略，有效减少空闲时间即降低了完工时间又减少了能耗目标；(4) 提出的基于 DQN 的算子选择策略，通过不断的训练和学习达到从每个解的细粒度为不同解定制算子，极大地增强了收敛性。

第五章　分布式异构柔性作业车间调度实例分析

5.1 引言

本章以重点研发项目为依托，对国内某大型装备生产厂商（A 公司）下料系统的实际生产订单作为工程案例进行分析，将本文提出的理论和方法应用到这一实际案例的生产调度中。该大型装备生产企业存在多个园区的多条下料生产线用于满足订单需求，企业根据订单量进行单个车间或者多个车间的生产，该问题可以抽象为 FJSP 和 DHFJSP 问题。采用第二章和第四章提出的算法对实例进行求解。实验结果显示本文所提方法，可以有效地提高生产效率，进一步验证了本文所提理论和算法在求解实际问题时的可行性和有效性，达到理论创新来源于实际，算法设计服务实际生产，理论与实践相互结合，相互促进的目的。

5.2 工程实例介绍

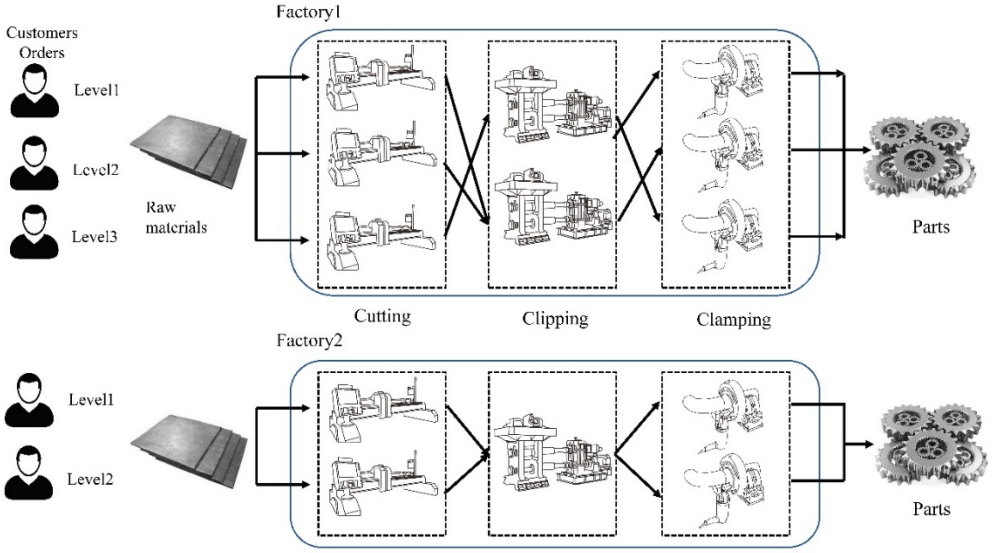

图 5.1 A 公司下料生产系统的工艺路线图

图 5.1 展示了 A 公司的下料系统工艺图。不同的园区有各自的仓库存储原材料。当接收到订单后，如何协同两个园区的产能，使得快速并行的生产出原材料是

该生产车间的关键问题。在下料系统中，每个原料需要经过三到工艺：切割，钳压和坡口打磨。最后生产的零件为后续工艺的使用。由该工艺路线图可知，每个阶段有多台机器可以选择加工零件，不同的类型的机器对零件的加工能力具有兼容性，例如机器 M1 可以加工 3m 的零件也可以加工 5m 的零件，但机器 M2 只能加工 5m 的零件。这符合 FJSP 中机器柔性的概念。原本该生产车间的类型应该是混合流水车间调度问题，但由于 A 公司提供的实际生产案例将下料的三步工序合并为一道下料工序，仅提供下料每个零件的加工时间，和不同班组的产能。因此，该实际工程问题被抽象为了全机器柔性的柔性作业车间调度问题。

表 5.1 A 公司下料系统 55 个批次的信息汇总

工件编号	重量/t	难度系数	工件编号	重量/t	难度系数
1	0.668	1	29	4	1
2	0.491	0.8	30	9.816	1
3	22.666	1	31	0.426	1
4	0.493	0.7	32	1.363	1
5	8.306	0.7	33	0.543	1
6	2.537	0.7	34	4.51	0.8
7	3.338	0.8	35	4.798	0.8
8	0.205	1	36	0.607	0.8
9	0.205	1	37	0.435	0.8
10	1.958	1	38	1	0.8
11	10.855	1	39	0.159	1
12	10.961	1	40	0.159	1
13	41.317	1	41	0.267	1
14	2.018	1	42	0.371	1
15	3.802	1	43	10.559	0.8
16	2.065	1	44	0.348	1
17	2.255	1	45	0.311	1
18	2.355	1	46	10.649	1
19	1.475	1	47	0.348	1
20	0.585	1	48	7	1
21	0.162	1	49	4	1
22	5.009	0.8	50	2.122	1
23	3.733	1	51	3.336	1
24	0.981	0.8	52	0.236	1
25	18.799	0.8	53	2.786	1
26	2.029	1	54	0.893	1
27	0.298	1	55	0.286	1
28	5	1			

表 5.2 A 公司不同园区下料资源信息汇总

班组名称	班组单位产能/KG/小时	班组总人数	工作时间
园区一下料一班	74.6KG/小时	13	8 小时/天
园区一下料二班	76.1KG/小时	16	
园区一下料三班	80.4KG/小时	14	
园区一下料四班	80.9KG/小时	17	
园区二下料一班	80KG/小时	14	
园区二下料二班	78KG/小时	15	
园区二下料三班	79KG/小时	14	
园区二下料四班	78KG/小时	15	

表 5.1 展示了该工程实例中包含的 55 个批次的重量和难度系数。表 5.2 展示了不同园区下料资源的信息，每个园区有 4 个班组，每个班组的人数和产能不同，对于同一个工件的加工时间也不同。因此，该问题特性符合分布式异构柔性作业车间调度问题，每个班组可以处理所有类型的工件，即为全柔性作业车间。每个工件在不同班组的加工时间=批次重量*1000*难度系数/班组产能。

5.3 GDHFJSP 问题求解与验证

表 5.3 DQPEA 和对比算法在实际问题上的 HV(max)，GD(min)，Spread(min)统计结果

MOEAs	HV		GD		Spread	
	mean	std	mean	std	mean	std
NSGA-II	0.167102	0.000574	0.001878	0.000917	0.943469	0.110437
MOEA/D	0.165206	0.000996	0.002287	0.00066	1.019607	**0.101734**
TS-NSGA-II	0.16741	0.000433	0.001892	0.000725	0.973863	0.109112
HSLFA	0.164324	0.001245	0.003141	0.001154	1.112472	0.309525
LRVMA	0.164192	0.001149	0.003522	0.00168	1.150408	0.327005
MOEA/D-DQN	0.166126	0.000549	0.001716	0.000726	1.090645	0.111967
DQPEA	**0.167491**	**0.000287**	**0.001497**	**0.000401**	**0.929345**	0.10818
p-value	8.60E-19		1.62E-07		5.78E-05	

为了求解该分布式异构柔性作业车间调度的实际测试问题，本节选取第四章提出了 DQPEA 算法求解。选取了第四章中的对比算法 MOEA/D，NSGA-II，TS-NSGA-II，HSLFA，LRVMA 和 MOEA/D-DQN 作为参考。同样以最小化车间总能耗和最大完工时间为优化目标，以 HV，GD，Spread 指标来衡量算法的综合性能，收敛性和分布性。每个算法独立运行 20 次，停止条件为最大评价次数为 MaxNFEs=400*55=22000。表 5.3 展示了 DQPEA 和对比算法在实际测试问题上所

有指标的对比结果。结果显示本文所提出的 DQPEA 在所有指标上皆由于所有对比算法，同时根据显著性检验的结果各项指标的 p-value<0.05，说明了本文提出的算法在实际问题上表现出良好的性能，验证了算法设计的有效性。图 5.2 为公司原调度方案(Cmax=573h，TEC=10872kWh)，图 5.3 为算法优化后的 Cmax 最小的调度方案(Cmax=517h，TEC=10752kWh)，在完工时间上缩短了 63 小时，效率提高 9.7%。图 5.4 为算法优化后的 TEC 最小的调度方案(Cmax=1229h，TEC=10656 kWh)节省了 216kWh 电量，降低了碳排放。

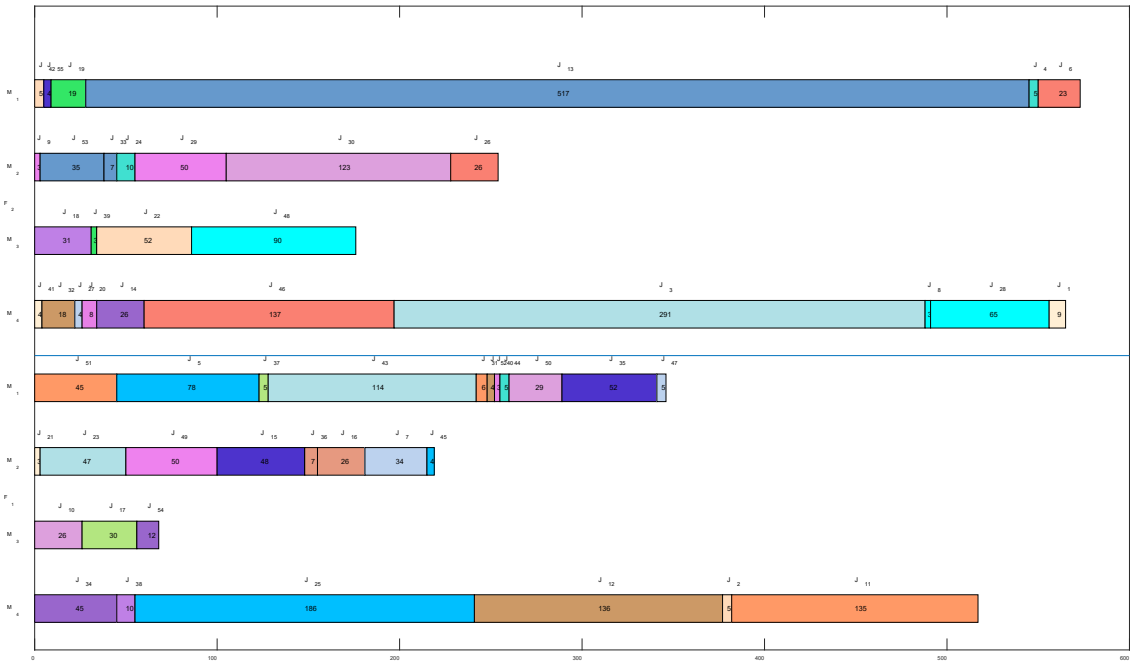

图 5.2 A 公司实际工程案例原始调度方案

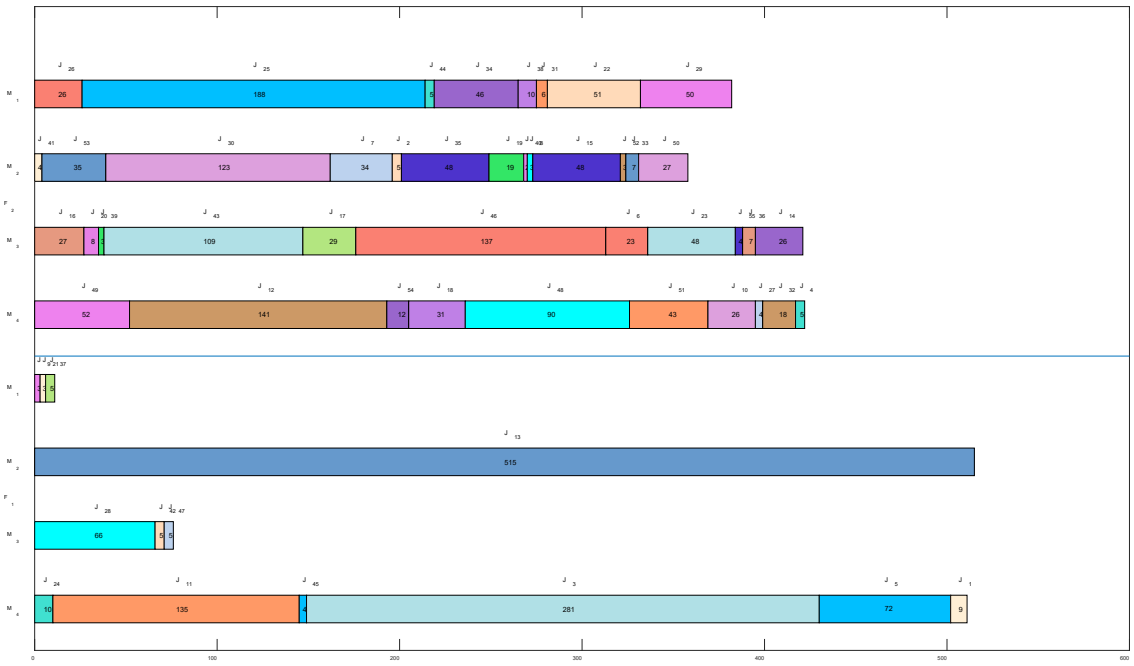

图 5.3 DQPEA 优化后在实际问题上 Cmax 最小的调度方案

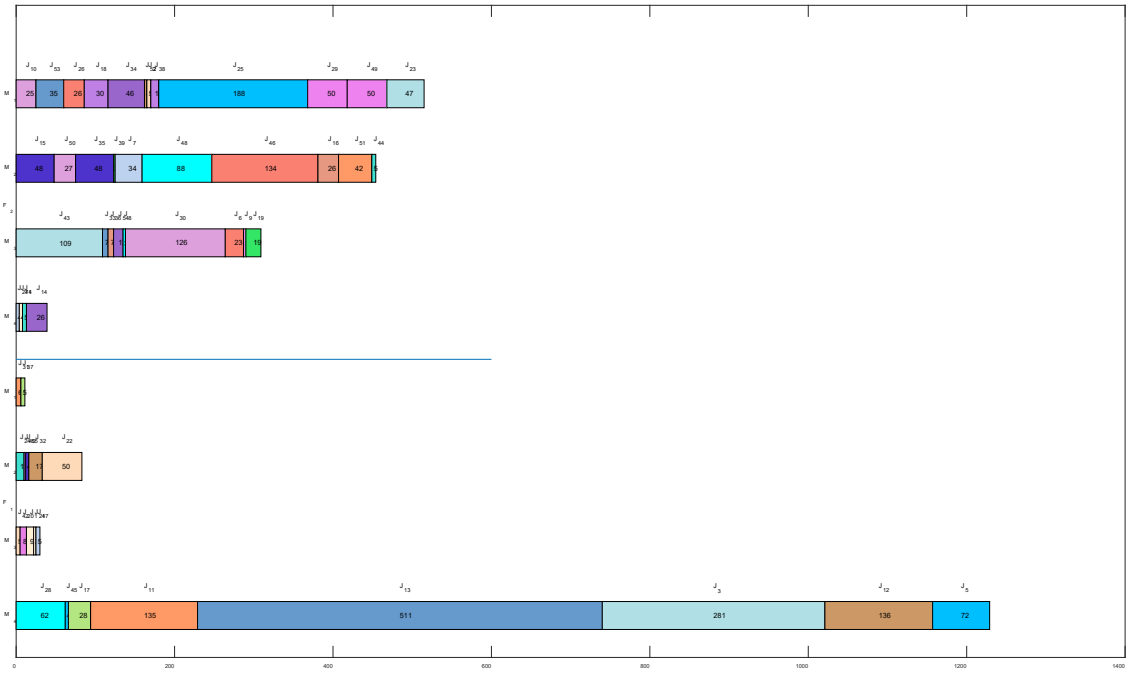

图 5.4 DQPEA 优化后在实际问题上 TEC 最小的调度方案

5.4 本节小结

本章以国内某大型工程装备生产厂商（A 公司）下料系统的实际生产问题为研究对象，首先介绍了下料车间的生产工艺、流程、实际案例的数据和参数，然后根据该企业的车间生产特征进行分析，将该企业的生产问题抽象为 DHFJSP 问题，并利用第四章提出的 DQPEA 算法进行求解，结果显示 DQPEA 算法在实际测试问题上的性能显著性好于目前相关算法，同时能够取得远好于原企业调度方案的近似最优解，可以有效提高该企业的生产效率，降低了碳排放，从而也验证了本文所提方法在求解实际问题时的可行性和有效性。

第六章 总结与展望

6.1 全文总结

本文依托国家重点研发项目，根据实际生产场景展开绿色分布式异构柔性作业车间调度问题的研究。以 GFJSP 研究为基础，到同构 GDFJSP 的研究，再到 GDHFJSP 的研究。通过对问题模型的分析，提炼解耦子问题的知识，围绕专家知识设计多种局部搜索算子，节能策略和初始化策略等增强策略，结合模因算法和不同多目标智能优化算法求解上述问题，实现知识与群体智能协同增强。由于多目标算法存在参数敏感和算子选择低效的问题。本文在方法论上，从强化学习到意外流行算法再到深度强化学习，结合指标和奖励等反馈机制，增强算法的自学习和自选择参数和算子的能力，实现学习与反馈驱动增强算法性能的目的。最后，针对传统 MA 框架在分布式车间调度问题上的缺陷，提出改进的协同进化的通用框架寄生进化算法，平衡全局和局部搜索的计算资源，在分布式车间调度问题的算法设计上提供理论参数。本文的工作总结如下：

(1)针对 GFJSP 最小化最大完工时间和总能耗的问题，提出了基于学习和参考向量的模因算法。首先，提出混合启发式初始化策略，保证算法获得收敛性和分布性良好的初始化种群。其次，提出了模型驱动的局部搜索算子，提升局部搜索的效率，加速算法收敛。再次，针对 MOEA/D 算法对邻域搜索范围的参数敏感的问题，提出了基于 Q 学习和指标反馈的参数自学习模型，增强算法的自主调节能力。随后，针对能耗指标，提出了两条启发式节能策略，通过贪心解构精英解实现降低能耗的目的。最后，在公开测试集上与相关算法进行对比，验证了提出算法和 MILP 模型的有效性。

(2)针对同构 GDFJSP 最小化最大完工时间和总能耗的问题，提出基于意外流行算法的模因算法。首先，提出了四种基于关键路径和关键块的局部搜索算子，以析取图的编码方式，有效求得关键路径并针对性扰动工序，减少最大完工时间。其次，提出了基于意外流行算法的算子选择模型，修正种群在算子选择决策中的错误，通过意外流行算法发现小权重的高效算子并分配更多选择权重，加速算法收敛。再次，提出了基于全主动调度解码的节能策略，有效减少车间中的空闲时间，降低生产能耗。最后，在著名 FJSP 测试集 Mk 和 DP 上与相关算法进行对比，验证了提

出改进点和设计的算法的有效性。

(3)针对 GDHFJSP 最小化最大完工时间和总能耗的问题，提出基于深度 Q 网络的寄生进化算法。首先，针对工厂异构和机器柔性的问题，分析模型特性，提出两条知识，依据知识设计了 9 种基于关键路径的局部搜索算子。其次，为了更细粒度地为每个解算子最优算子，提出了基于深度 Q 学习的算子选择模型，通过神经网络建立解空间与算子选择概率的映射，实现为不同解定制最优算子选择的目的。再次，采用基于全主动调度解码的节能策略降低能耗。随后，针对 MA 在求解分布式车间调度问题上的缺陷，提出了改进的协同进化框架寄生进化算法，通过将全局搜索和局部搜索放在两个种群中执行，即保证了全局搜索获得充分计算资源，又采用少量计算资源获得，良好地平衡全局和局部搜索资源，对分布式车间调度问题具有理论参考价值。最后，生成 20 个不同规模的测试集，与多种相关算法进行对比，验证了提出改进点和设计算法的有效性。

(4) 在上述理论研究的基础上，根据某国有大型装备生产企业下料车间的生产调度问题的特点，采用针对性的算法优化求解实际工程问题，有效地减少了生产时间，提升了生产效率，验证了本文所提算法和改进策略的有效性。

6.2 创新之处

本文的创新之处总结如下：

(1)为了解决前人设计算子具有随机性的问题，通过对机器负载和最后完工工序等影响因素的分析，提出了模型驱动的局部搜索算子，有效地提升局部搜索的效率。再次，提出了基于 Q 学习和指标反馈的参数自学习模型，解决 MOEA/D 算法对邻域搜索范围的参数敏感的问题，增强算法的自主调节能力。最后，针对之前研究中缺乏有效节能策略的不足，提出了两条启发式节能策略，通过贪心解构精英解实现降低能耗的目的。

(2) 为了进一步分析影响 Cmax 的关键因素，提出了 4 种基于关键路径和关键块的局部搜索算子，以析取图的编码方式，有效求得关键路径并针对性扰动工序，减少最大完工时间。其次，首次将社会科学领域的算法应用于车间调度问题优化领域。将意外流行算法用于算子自学习选择中，修正种群在算子选择决策中的错误，发现小权重的高效算子并分配更多选择权重。最后，将全主动调度策略应用到能指标优化，提出了基于全主动调度解码的节能策略，有效减少车间中的空闲时间，降低生产能耗。

(3) 首先，根据实际生产场景提出了绿色分布式异构柔性作业车间调度问题，

拓展了问题模型，使得分布式柔性作业车间调度的研究更贴近实际生产。其次，针对工厂异构和机器柔性的问题，分析模型特性，提出两条知识，依据知识设计了 9 种基于关键路径和概率的局部搜索算子。再次，首次将 DQN 算法应用于 DHFJSP，并提出了基于深度 Q 学习的算子选择模型，通过神经网络建立解空间与算子选择概率的映射，实现为不同解定制最优算子选择的目的。最后，针对 MA 在求解分布式车间调度问题上的缺陷，提出了改进的协同进化框架寄生进化算法，通过将全局搜索和局部搜索放在两个种群中执行，即保证了全局搜索获得充分计算资源，又采用少量计算资源获得，良好地平衡全局和局部搜索资源，对分布式车间调度问题具有理论参考价值。

6.3 工作展望

本文提出的模型驱动的学习型模因算法，在求解分布式异构柔性作业车间调度问题上有良好的性能。虽然，在学习型策略上提出了多种自学习模型。但在问题模型，算法设计，学习型策略等方面仍有许多问题有待解决。

(1)问题模型方面：本文研究的分布式异构柔性作业车间调度问题仅为加工时间不同且优化目标为 Cmax 和 TEC。但实际生产场景是复杂多变的，根据实际生产场景的不同，优化目标的类型也不同，例如 TTD，TWL，和 E/T 等。其次，不同类型的异构工厂也有待研究。最后，还可以在车间生产中考虑动态事件，例如订单插入，机器维修，不确定加工时间，订单交货期变化等。因此，在模型拓展方面仍有许多内容等待研究。

(2)算法设计方面：针对于不同的优化目标，需要重新分析模型知识，进而设计增强策略。更多地，缺乏理论推导从数学领域证明增强策略的设计是有效的。其次，如何根据生产数据驱动算法设计与模型知识相互结合，实现数据模型相互融合。随后，在搜索中设计合理的反馈机制，利用反馈引导算法进化是一种更关键的技术。最后，端到端的通用性的车间调度神经网络框架有待研究。

(3)学习型策略方面：本文采用的 DQN 训练策略属于在线学习，并没有对历史训练信息加以保存。每次求解需要重新训练，效率低下。具有泛化性的深度强化学习模型有待研究。其次，在训练过程中，没有考虑数据之间隐含的关联性，没有研究是否这些关联性给模型训练带来不平稳的问题。再次， DQN 模型在训练上具有滞后性，有许多对 DQN 改进的算法加速 DQN 的学习速度减少误差，如何改进 DQN 的网络结构有待研究。最后，加强深度强化学习的可解释性是研究重点。

致谢

从 2019 年保研以来，来到龚老师实验室已满四年。四年来，我不负每一寸光阴。经过三年系统地科研训练，我成长了许多，从对科研一窍不通，到第一篇反复修改数月的文章，经历过拒稿，再到投稿，修稿，到最终的录用。在“听说读写”各方面对科研都有了更深刻的理解。这一段经历，我将毕生难忘。酌水知源。如果这四年的时光不算虚度，学生首先应该诚挚的感谢诸位恩师多年来的谆谆教导：

感谢我的硕士导师龚文引教授对我学业的精心指导。龚老师是我的恩师，无论从人生启迪，学术科研还是生活上，龚老师都给予了我极大的帮助，我从龚老师身上学到了许多宝贵的知识和技能。加入龚老师的门下是我人生 24 年至今做的最正确的决定，感谢龚老师对各方面的培养，让我明确了人生的方向。

感谢我的博士导师王凌教授对我的指导和帮助，多次与王老师的交流和听王老师的汇报都令我收获颇深。有幸加入王老师的科研团队，是我的学生生涯中至为幸运的事。我以持之以恒的努力，敢于探索钻研的心和高质量的科研成果回报王老师认可和肯定。

感谢亦师亦友的卢超老师，卢老师是我开始学习车间调度优化理论的启蒙老师。卢老师总是把自己宝贵的意见倾囊相授，教会我如何做实验写论文和修改，如果没有卢老师的帮助，我也许对科研早就失去了热情。感谢本文中所有被引用文献的中外学者，为本文写作提供了思路。

感谢廖作文、李水佳、郑小操、王开、明飞、甄慧翔等师兄对我科研上的指导和帮助。感谢潘子肖师兄耐心地答疑解惑。感谢邢彩霄师妹，陈峰和黄康林师弟，感谢 632 的同学们，你们的陪伴使我的科研成为一种别样的修行，让枯燥的生活增添了无穷欢乐，与你们为伴的四年生活是宝贵的人生记忆。

我要感谢我父母对我的培养，感谢我爷爷奶奶从小对我无微不至的照顾，感谢我的姑妈和伯伯对我的关心。我还要特别感谢我的女朋友蒋书凝女士对我无微不至的关心和陪伴，没有她我将不会度过这幸福快乐的几年，没有她的宝贵建议我将不会取得今天的成绩，我相信我们今后的生活会非常美满和幸福，也要感谢她的父母蒋叔叔和舒阿姨对我的关心建议和照顾。

最后，感谢盲审专家、答辩专家们对我硕士论文提出的宝贵意见。

参考文献

[1] 白立浩. 定制化制造企业车间调度策略与仿真研究[D]. 中国矿业大学, 2015.

[2] 张枨. 绿色化转型，引领产业结构调整[N]. 人民日报, 2022-12-01: 11.

[3] 王凌,邓瑾,王圣尧.分布式车间调度优化算法研究综述[J].控制与决策, 2016, 31(01): 1-11.

[4] 袁悦. 分布式并行机调度及其智能优化算法研究[D]. 武汉理工大学, 2020.

[5] PAN Z X, LEI D M, WANG L. A Knowledge-Based Two-Population Optimization Algorithm for Distributed Energy-Efficient Parallel Machines Scheduling[J]. IEEE Transactions on Cybernetics, 2022, 52(6): 5051-5063.

[6] 孙志卫. 改进遗传算法求解分布式置换流水车间调度问题[D]. 天津理工大学，2020.

[7] SHAO Z S, SHAO W S, PI D C. LS-HH: A Learning-Based Selection Hyper-Heuristic for Distributed Heterogeneous Hybrid Blocking Flow-Shop Scheduling[J]. IEEE Transactions on Emerging Topics in Computational Intelligence, 2023, 7(1): 111-127.

[8] LI J Q, CHEN X L, DUAN P Y, et al. KMOEA: A Knowledge-Based Multiobjective Algorithm for Distributed Hybrid Flow Shop in a Prefabricated System[J]. IEEE Transactions on Industrial Informatics, 2022, 18(8): 5318-5329.

[9] ŞAHMAN M A. A discrete spotted hyena optimizer for solving distributed job shop scheduling problems[J]. Applied Soft Computing, 2021, 106: 107349.

[10] 吴锐. 面向建材装备制造企业的分布式柔性作业车间调度问题研究[D]. 武汉理工大学, 2019.

[11] SANG Y, TAN J. Intelligent factory many-objective distributed flexible job shop collaborative scheduling method[J]. Computers & Industrial Engineering, 2022, 164: 107884.

[12] CHEN X, ONG Y S, LIM M H, et al. A Multi-Facet Survey on Memetic Computation[J]. IEEE Transactions on Evolutionary Computation, 2011, 15(5): 591-607.

[13] ZHAO Z, ZHOU M, LIU S. Iterated Greedy Algorithms for Flow-Shop Scheduling

Problems: A Tutorial[J]. IEEE Transactions on Automation Science and Engineering, 2022, 19(3): 1941-1959.

[14] BRUCKER P, SCHLIE R. Job-shop scheduling with multi-purpose machines[J]. Computing, 1990, 45(4): 369-375.

[15] JOHNSON S M. Optimal two- and three-stage production schedules with setup times included[J]. Naval Research Logistics Quarterly, 1954, 1(1): 61-68.

[16] 越民义, 韩继业. n 个零件在 m 台机床上加工顺序问题（I）[J]. 中国科学, 1975, 5: 462-470.

[17] COOK S A. The complexity of theorem proving procedures[C]//Association of Computing Machinery. 1971: 151-158.

[18] PARKER R G. Deterministic scheduling theory[M]. London: Chapman and Hall, 1996.

[19] ALAN S. MANNE. On the job-shop scheduling problem[J]. Operations Research, 1960, 8(2): 219-223.

[20] BLAZEWICZ J, DROR M, WEGLARZ J. Mathematical programming formulations for machine scheduling: A survey[J]. European Journal of Operational Research, 1991, 51(3): 283-300.

[21] MARSHALL L. FISHER. Optimal solution of scheduling problems using lagrange multipliers: Part I[J]. Operations Research, 1973, 21: 1114-1127.

[22] HOOGEVEEN J A, VAN D V S L. Stronger Lagrangian bounds by use of slack variables: Applications to machine scheduling problems[J]. Mathematical Programming, 1995, 70(1-3): 173-190.

[23] CHU C, PORTMANN M C, PROTH J. M. A splitting-up approach to simplify job-shop scheduling problems[J]. International Journal of Production Research, 1992, 30(4): 859-870.

[24] HOITOMT D J, LUH P B, PATTIPATI K R. A practical approach to job-shop scheduling problems[J]. IEEE Transactions on Robotics and Automation, 1993, 9(1): 1-13.

[25] EGON B. Machine Sequencing Via Disjunctive Graphs: An Implicit Enumeration Algorithm[J]. Operations Research, 1969, 17(6): 941-957.

[26] MC MAHON G B, FLORIAN M. On scheduling with ready times and due dates to minimize maximum lateness[J]. Operations Research, 1975, 23(3): 475-482.

[27] CARLIER J, PINSON E. An Algorithm for Solving the Job-Shop Problem[J].

Management Science, 1989, 35(2): 164-176.

[28] MONTAZERI M. Analysis of scheduling rules for an FMS[J]. International Journal of Production Research, 1990, 28(4): 785-802.

[29] PANWALKAR S S, ISKANDER W. A survey of scheduling rules[J]. Operations Research, 1990, 28: 785-802.

[30] ADAMS J, BALAS E, ZAWACK D. The shifting bottleneck procedure for job shop scheduling[J]. Management Science, 1988, 34: 391-401.

[31] CARLIER J. The one-machine sequencing problem[J]. European Journal of Operational Research, 1982, 11: 42-47.

[32] DAUZERE-PERES S, LASSERRE J B. A modified shifting bottleneck procedure for job-shop scheduling[J]. International Journal of Production Research, 1993, 31(4): 923-932.

[33] DEMIRKOL E, MEHTA S, UZSOY R. A Computational Study of Shifting Bottleneck Procedures for Shop Scheduling Problems[J]. Journal of Heuristics, 1997, 3(2):111-137.

[34] WENQI H, AIHUA Y. An improved shifting bottleneck procedure for the job shop scheduling problem[J]. Computers & Operations Research, 2004, 31(12): 2093-2110.

[35] KACEM I, HAMMADI S, BORNE P. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems[J]. IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews), 2002, 32(1): 1-13.

[36] JENSEN M T. Generating robust and flexible job shop schedules using genetic algorithms[J]. IEEE Transactions on Evolutionary Computation, 2003, 7(3): 275-288.

[37] NHU BINH HO, JOC CING TAY. Solving Multiple-Objective Flexible Job Shop Problems by Evolution and Local Search[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2008, 38(5): 674-685.

[38] GAO K Z, SUGANTHAN P N, TASGETIREN M F, et al. Effective ensembles of heuristics for scheduling flexible job shop problem with new job insertion[J]. Computers & Industrial Engineering, 2015, 90: 107-117.

[39] 李尚函, 胡 蓉, 钱 斌, 等. 超启发式遗传算法求解模糊柔性作业车间调度[J]. 控制理论与应用, 2020, 37(2): 316-330.

[40] ZHANG G, HU Y, SUN J, et al. An improved genetic algorithm for the flexible job shop scheduling problem with multiple time constraints[J]. Swarm and Evolutionary

Computation, 2020, 54: 100664.

[41] DEFERSHA F M, ROOYANI D. An efficient two-stage genetic algorithm for a flexible job-shop scheduling problem with sequence dependent attached/detached setup, machine release date and lag-time[J]. Computers & Industrial Engineering, 2020, 147: 106605.

[42] GAO K, CAO Z, ZHANG L, et al. A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems[J]. IEEE/CAA Journal of Automatica Sinica, 2019, 6(4): 904-916.

[43] LI X, GUO X, TANG H, et al. Survey of integrated flexible job shop scheduling problems[J]. Computers & Industrial Engineering, 2022, 174: 108786.

[44] XIA W, WU Z. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems[J]. Computers & Industrial Engineering, 2005, 48(2): 409-425.

[45] 顾幸生, 丁豪杰. 面向柔性作业车间调度问题的改进博弈粒子群算法[J]. 同济大学学报（自然科学版）, 2020, 48(12): 1782-1789.

[46] EBRAHIMI A, JEON H W, LEE S, et al. Minimizing total energy cost and tardiness penalty for a scheduling-layout problem in a flexible job shop system: A comparison of four metaheuristic algorithms[J]. Computers & Industrial Engineering, 2020, 141: 106295.

[47] 李黎. 混合量子粒子群算法在柔性作业车间调度中的研究与应用[D]. 大连交通大学, 2019.

[48] 李俊萱, 王艳, 纪志成. 基于混合 QPSO 的模糊柔性作业车间调度问题研究[J]. 系统仿真学报, 2020, 32(10): 2010-2021.

[49] 吴晓雯, 郑巧仙. 基于改进粒子群的柔性作业车间调度问题优化研究[J]. 湖北大学学报(自然科学版), 2022, 44(5): 501-507.

[50] 陈魁, 毕利, 王文雅. 柔性作业车间 AGV 与机器双资源集成调度研究[J]. 系统仿真学报, 2022, 34(3): 461-469.

[51] LI X, PENG Z, DU B, et al. Hybrid artificial bee colony algorithm with a rescheduling strategy for solving flexible job shop scheduling problems[J]. Computers & Industrial Engineering, 2017, 113: 10-26.

[52] 郑小操, 龚文引. 改进人工蜂群算法求解模糊柔性作业车间调度问题[J]. 控制理论与应用, 2020, 37(6): 1284-1292.

[53] LI Y, HUANG W, WU R, et al. An improved artificial bee colony algorithm for

solving multi-objective low-carbon flexible job shop scheduling problem[J]. Applied Soft Computing, 2020, 95: 106544.

[54] 唐浩. 考虑运输时间的柔性作业车间调度研究[D]. 南京航空航天大学, 2021.

[55] 李俊青, 杜 宇, 田 杰, 等. 带运输资源约束柔性作业车间调度问题的人工蜂群算法[J]. 电子学报, 2021, 49(2): 324-330.

[56] GAO K, YANG F, ZHOU M, et al. Flexible Job-Shop Rescheduling for New Job Insertion by Using Discrete Jaya Algorithm[J]. IEEE Transactions on Cybernetics, 2019, 49(5): 1944-1955.

[57] CALDEIRA R H, GNANAVELBABU A. Solving the flexible job shop scheduling problem using an improved Jaya algorithm[J]. Computers & Industrial Engineering, 2019, 137: 106064.

[58] LI J Q, DENG J W, LI C Y, et al. An improved Jaya algorithm for solving the flexible job shop scheduling problem with transportation and setup times[J]. Knowledge-Based Systems, 2020, 200: 106032.

[59] CALDEIRA R H, GNANAVELBABU A. A Pareto based discrete Jaya algorithm for multi-objective flexible job shop scheduling problem[J]. Expert Systems with Applications, 2021, 170: 114567.

[60] LEI D, LI M, WANG L. A Two-Phase Meta-Heuristic for Multiobjective Flexible Job Shop Scheduling Problem with Total Energy Consumption Threshold[J]. IEEE Transactions on Cybernetics, 2019, 49(3): 1097-1109.

[61] LI M, LEI D. An imperialist competitive algorithm with feedback for energy-efficient flexible job shop scheduling with transportation and sequence-dependent setup times[J]. Engineering Applications of Artificial Intelligence, 2021, 103: 104307.

[62] GNANAVELBABU A, CALDEIRA R H, VAIDYANATHAN T. A simulation-based modified backtracking search algorithm for multi-objective stochastic flexible job shop scheduling problem with worker flexibility[J]. Applied Soft Computing, 2021, 113: 107960.

[63] CALDEIRA R H, GNANAVELBABU A, VAIDYANATHAN T. An effective backtracking search algorithm for multi-objective flexible job shop scheduling considering new job arrivals and energy consumption[J]. Computers & Industrial Engineering, 2020, 149: 106863.

[64] SUN L, LIN L, GEN M, et al. A Hybrid Cooperative Coevolution Algorithm for Fuzzy Flexible Job Shop Scheduling[J]. IEEE Transactions on Fuzzy Systems, 2019,

27(5): 1008-1022.

[65] LIN J, ZHU L, WANG Z J. A hybrid multi-verse optimization for the fuzzy flexible job-shop scheduling problem[J]. Computers & Industrial Engineering, 2019, 127: 1089-1100.

[66] WU X L, SHEN X, LI C. The flexible job-shop scheduling problem considering deterioration effect and energy consumption simultaneously[J]. Computers & Industrial Engineering, 2019, 135: 1004-1024.

[67] 夏俊红, 郑建国. 基于改进 GSO 算法的柔性作业车间 E/T 调度问题[J]. 计算机系统应用, 2019, 28(1): 119-126.

[68] 詹欣隆, 张超勇, 孟磊磊, 等. 面向高效低碳的切削参数与柔性作业 车间调度集成建模与优化[J]. 计算机集成制造系统, 2021, 27(12): 3519-3535.

[69] ZHANG S, LI X, ZHANG B, et al. Multi-objective optimisation in flexible assembly job shop scheduling using a distributed ant colony system[J]. European Journal of Operational Research, 2020, 283(2): 441-460.

[70] LI J Q, LIU Z M, LI C, et al. Improved Artificial Immune System Algorithm for Type-2 Fuzzy Flexible Job Shop Scheduling Problem[J]. IEEE Transactions on Fuzzy Systems, 2021, 29(11): 3234-3248.

[71] 姜天华, 邓冠龙, 朱惠琦. 离散猫群优化算法求解带交货期的 FJSP 问题[J]. 控制与决策, 2020, 35(1): 161-168.

[72] 吴贝贝, 张宏立, 王 聪, 等. 基于正态云模型的状态转移算法求解多目标 柔性作业车间调度问题[J]. 控制与决策, 2021, 36(5): 1181-1190.

[73] CAO Z C, LIN C R, ZHOU M C. A Knowledge-Based Cuckoo Search Algorithm to Schedule a Flexible Job Shop With Sequencing Flexibility[J]. IEEE Transactions on Automation Science and Engineering, 2021, 18(1): 56-69.

[74] LI J Q, DU Y, GAO K Z, et al. A Hybrid Iterated Greedy Algorithm for a Crane Transportation Flexible Job Shop Problem[J]. IEEE Transactions on Automation Science and Engineering, 2022, 19(3): 2153-2170.

[75] DEFERSHA F M, OBIMUYIWA D, YIMER A D. Mathematical model and simulated annealing algorithm for setup operator constrained flexible job shop scheduling problem[J]. Computers & Industrial Engineering, 2022, 171: 108487.

[76] LI H, WANG X, PENG J. A hybrid differential evolution algorithm for flexible job shop scheduling with outsourcing operations and job priority constraints[J]. Expert Systems with Applications, 2022, 201: 117182.

[77] ZHU Z, ZHOU X, CAO D, et al. A shuffled cellular evolutionary grey wolf optimizer for flexible job shop scheduling problem with tree-structure job precedence constraints[J]. Applied Soft Computing, 2022, 125: 109235.

[78] 刘璐,宋海草,姜天华等.基于改进生物迁徙算法的双资源柔性作业车间节能调度问题[J/OL].计算机集成制造系统:1-24[2023-05-12]. http://kns.cnki.net/kcms/detail/11.5946.TP.20220711.1504.002.html.

[79] DING J, DAUZERE-PERES S, SHEN L, et al. A Novel Evolutionary Algorithm for Energy Efficient Scheduling in Flexible Job Shops[J/OL]. IEEE Transactions on Evolutionary Computation, 2022: 1-1[2022-11-16]. https://ieeexplore.ieee.org/document/9953959

[80] GONG G, CHIONG R, DENG Q, et al. A two-stage memetic algorithm for energy-efficient flexible job shop scheduling by means of decreasing the total number of machine restarts[J]. Swarm and Evolutionary Computation, 2022, 75: 101131.

[81] DING H, GU X. Hybrid of human learning optimization algorithm and particle swarm optimization algorithm with scheduling strategies for the flexible job-shop scheduling problem[J]. Neurocomputing, 2020, 414: 313-332.

[82] GARCÍA-LEÓN A A, DAUZÈRE-PÉRÈS S, MATI Y. An efficient Pareto approach for solving the multi-objective flexible job-shop scheduling problem with regular criteria[J]. Computers & Operations Research, 2019, 108: 187-200.

[83] GARCÍA-LEÓN A A, DAUZÈRE-PÉRÈS S, MATI Y. An efficient Pareto approach for solving the multi-objective flexible job-shop scheduling problem with regular criteria[J]. Computers & Operations Research, 2019, 108: 187-200.

[84] YUAN Y, XU H. Multiobjective Flexible Job Shop Scheduling Using Memetic Algorithms[J]. IEEE Transactions on Automation Science and Engineering, 2015, 12(1): 336-353.

[85] PIROOZFARD H, WONG K Y, WONG W P. Minimizing total carbon footprint and total late work criterion in flexible job shop scheduling by using an improved multi-objective genetic algorithm[J]. Resources, Conservation and Recycling, 2018, 128: 267-283.

[86] DAI M, TANG D, GIRET A, et al. Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints[J]. Robotics and Computer-Integrated Manufacturing, 2019, 59: 143-157.

[87] SOTO C, DORRONSORO B, FRAIRE H, et al. Solving the multi-objective flexible

job shop scheduling problem with a novel parallel branch and bound algorithm[J]. Swarm and Evolutionary Computation, 2020, 53: 100632.

[88] TAN W, YUAN X, WANG J, et al. A fatigue-conscious dual resource constrained flexible job shop scheduling problem by enhanced NSGA-II: An application from casting workshop[J]. Computers & Industrial Engineering, 2021, 160: 107557.

[89] 刘玉婷. 高维多目标绿色柔性作业车间调度问题研究[D]. 西安交通大学, 2021.

[90] 王烨. 基于改进 NSGA-III的低碳多目标柔性作业车间调度问题研究[D]. 云南大学, 2021.

[91] AN Y J, CHEN X, GAO K Z, et al. Multiobjective Flexible Job-Shop Rescheduling with New Job Insertion and Machine Preventive Maintenance[J]. IEEE Transactions on Cybernetics, 2023, 53(5): 3101-3113.

[92] JIANG E D, WANG L. Multi-objective optimization based on decomposition for flexible job shop scheduling under time-of-use electricity prices[J]. Knowledge-Based Systems, 2020, 204: 106177.

[93] 孟磊磊. 面向高效节能的柔性作业车间调度问题建模与优化[D]. 华中科技大学, 2020.

[94] PARK M J, HAM A. Energy-aware flexible job shop scheduling under time-of-use pricing[J]. International Journal of Production Economics, 2022, 248: 108507.

[95] RAKOVITIS N, LI D, ZHANG N, et al. Novel approach to energy-efficient flexible job-shop scheduling problems[J]. Energy, 2022, 238: 121773.

[96] GAO K Z, HUANG Y, SADOLLAH A, et al. A review of energy-efficient scheduling in intelligent production systems[J]. Complex & Intelligent Systems, 2020, 6(2): 237-249.

[97] WU Z, WENG M X. Multiagent Scheduling Method With Earliness and Tardiness Objectives in Flexible Job Shops[J]. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), 2005, 35(2): 293-301.

[98] 王嘉喆. 多智能体环境下的柔性作业车间调度问题研究[D]. 沈阳工业大学, 2022.

[99] LEI K, GUO P, ZHAO W, et al. A multi-action deep reinforcement learning framework for flexible Job-shop scheduling problem[J]. Expert Systems with Applications, 2022, 205: 117796.

[100] 张凯，毕利，焦小刚. 集成强化学习算法的柔性作业车间调度问题研究[J].中国机械工程，2023, 34(02): 201-207.

[101] SONG W, CHEN X, LI Q, et al. Flexible Job-Shop Scheduling via Graph Neural Network and Deep Reinforcement Learning[J]. IEEE Transactions on Industrial Informatics, 2023, 19(2): 1600-1610.

[102] 胡一凡，张利平，白雪等. 深度强化学习求解柔性装配作业车间调度问题[J]. 华中科技大学学报(自然科学版)，2023，51(02)：153-160.

[103] ZHANG J D, HE Z, CHAN W H, et al. DeepMAG: Deep reinforcement learning with multi-agent graphs for flexible job shop scheduling[J]. Knowledge-Based Systems, 2023, 259: 110083.

[104] SU J, HUANG H, LI G, et al. Self-Organizing Neural Scheduler for the Flexible Job Shop Problem With Periodic Maintenance and Mandatory Outsourcing Constraints[J/OL]. IEEE Transactions on Cybernetics, 2022: 1-12[2022-04-05]. https://ieeexplore.ieee.org/document/9750014

[105] CHEN R, YANG B, LI S, et al. A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem[J]. Computers & Industrial Engineering, 2020, 149: 106778.

[106] DU Y, LI J Q, CHEN X L, et al. Knowledge-Based Reinforcement Learning and Estimation of Distribution Algorithm for Flexible Job Shop Scheduling Problem[J/OL]. IEEE Transactions on Emerging Topics in Computational Intelligence, 2022: 1-15[2022-02-08]. https://ieeexplore.ieee.org/document/9707606

[107] DU Y, LI J Q, LI C, et al. A Reinforcement Learning Approach for Flexible Job Shop Scheduling Problem with Crane Transportation and Setup Times[J/OL]. IEEE Transactions on Neural Networks and Learning Systems, 2022: 1-15[2022-10-10]. https://ieeexplore.ieee.org/document/9913668.

[108] PAN Z X, LEI D M, WANG L. A Bi-Population Evolutionary Algorithm with Feedback for Energy-Efficient Fuzzy Flexible Job Shop Scheduling[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2022, 52(8): 5295-5307.

[109] LI R, GONG W Y, LU C, et al. A Learning-based Memetic Algorithm for Energy-Efficient Flexible Job Shop Scheduling with Type-2 Fuzzy Processing Time[J/OL]. IEEE Transactions on Evolutionary Computation, 2022: 1-1[2022-05-17]. https://ieeexplore.ieee.org/document/9776510.

[110] LI R, GONG W Y, LU C. A reinforcement learning based RMOEA/D for bi-objective fuzzy flexible job shop scheduling[J]. Expert Systems with Applications, 2022, 203: 117380.

[111] PAN Z X, WANG L, ZHENG J, et al. A Learning-Based Multi-Population Evolutionary Optimization for Flexible Job Shop Scheduling Problem with Finite Transportation Resources[J/OL]. IEEE Transactions on Evolutionary Computation, 2022: 1-1[2022-11-03]. https://ieeexplore.ieee.org/document/9937159.

[112] LIN C R, CAO Z C, ZHOU M C. Learning-Based Grey Wolf Optimizer for Stochastic Flexible Job Shop Scheduling[J]. IEEE Transactions on Automation Science and Engineering, 2022, 19(4): 3659-3671.

[113] LI R, GONG W, LU C. Self-adaptive multi-objective evolutionary algorithm for flexible job shop scheduling with fuzzy processing time[J]. Computers & Industrial Engineering, 2022, 168: 108099.

[114] ZHANG G, LU X, LIU X, et al. An effective two-stage algorithm based on convolutional neural network for the bi-objective flexible job shop scheduling problem with machine breakdown[J]. Expert Systems with Applications, 2022, 203: 117460.

[115] 叶孙飞. 带有设置时间和资源约束的柔性作业车间调度问题的算法研究[D]. 华东师范大学, 2022.

[116] LOU H Y, WANG X P, DONG Z, et al. Memetic algorithm based on learning and decomposition for multiobjective flexible job shop scheduling considering human factors[J]. Swarm and Evolutionary Computation, 2022, 75: 101204.

[117] CHAN F T S, CHUNG S H, CHAN P L Y. Application of genetic algorithms with dominant genes in a distributed scheduling problem in flexible manufacturing systems[J]. International Journal of Production Research, 2006, 44(3): 523-543.

[118] CHUNG S H, CHAN F T S, CHAN H K. A modified genetic algorithm approach for scheduling of perfect maintenance in distributed production scheduling[J]. Engineering Applications of Artificial Intelligence, 2009, 22(7): 1005-1014.

[119] DE GIOVANNI L, PEZZELLA F. An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling problem[J]. European Journal of Operational Research, 2010, 200(2): 395-408.

[120] 周蓉. 基于改进遗传算法的分布式柔性车间调度问题研究[D]. 南昌航空大学, 2012.

[121] 黄英杰, 姚锡凡. 基于目标级联法和粒子群算法的柔性分布式车间调度[J]. 中南大学学报(自然科学版), 2012, 43(1): 151-158.

[122] ZIAEE M. A heuristic algorithm for the distributed and flexible job-shop scheduling problem[J]. The Journal of Supercomputing, 2014, 67(1): 69-83.

[123] LIU T K, CHEN Y P, CHOU J H. Solving Distributed and Flexible Job-Shop Scheduling Problems for a Real-World Fastener Manufacturer[J]. IEEE Access, 2014, 2: 1598-1606.

[124] 王 凌, 邓 瑾, 王圣尧. 分布式车间调度优化算法研究综述[J]. 控制与决策, 2016, 31(1): 1-11.

[125] 何怡. 蚁群算法及其在分布式柔性作业车间调度中的应用[D]. 合肥工业大学, 2016.

[126] MARZOUKI B, DRISS O B, GHEDIRA K. Decentralized Tabu Searches in Multi Agent System for Distributed and Flexible Job Shop Scheduling Problem[C]//2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA). Hammamet: IEEE, 2017: 1019-1026.

[127] LI J Q, DUAN P Y, CAO J, et al. A Hybrid Pareto-Based Tabu Search for the Distributed Flexible Job Shop Scheduling Problem With E/T Criteria[J]. IEEE Access, 2018, 6: 58883-58897.

[128] LU P H, WU M C, TAN H, et al. A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems[J]. Journal of Intelligent Manufacturing, 2018, 29(1): 19-34.

[129] WU X L, LIU X, ZHAO N. An improved differential evolution algorithm for solving a distributed assembly flexible job shop scheduling problem[J]. Memetic Computing, 2019, 11(4): 335-355.

[130] XIU L W, XIA J L. An Improved Differential Evolution Algorithm for Solving a Distributed Flexible Job Shop Scheduling Problem[C]//2018 IEEE 14th International Conference on Automation Science and Engineering (CASE). Munich, Germany: IEEE, 2018: 968-973.

[131] 张洪亮,徐静茹,徐公杰.分布式双资源柔性作业车间节能调度[J/OL].系统科学与数学:1-19[2023-05-12].http://kns.cnki.net/kcms/detail/11.2019.O1. 20230119.1657.005.html.

[132] LIN C S, LI P Y, WEI J M, et al. Integration of process planning and scheduling for distributed flexible job shops[J]. Computers & Operations Research, 2020, 124: 105053.

[133] LUO Q, DENG Q, GONG G, et al. An efficient memetic algorithm for distributed flexible job shop scheduling problem with transfers[J]. Expert Systems with Applications, 2020, 160: 113721.

[134] GUO S, LUO W, XU W, et al. Research on Distributed Flexible Job Shop Scheduling Problem for Large Equipment Manufacturing Enterprises Considering Energy Consumption[C]//2020 39th Chinese Control Conference (CCC). Shenyang, China: IEEE, 2020: 1501-1506.

[135] MENG L, REN Y, ZHANG B, et al. MILP Modeling and Optimization of Energy-Efficient Distributed Flexible Job Shop Scheduling Problem[J]. IEEE Access, 2020, 8: 191191-191203.

[136] MENG L, ZHANG C, REN Y, et al. Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem[J]. Computers & Industrial Engineering, 2020, 142: 106347.

[137] DU Y, LI J Q, LUO C, et al. A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations[J]. Swarm and Evolutionary Computation, 2021, 62: 100861.

[138] WANG J, LIU Y, REN S, et al. Evolutionary game based real-time scheduling for energy-efficient distributed and flexible job shop[J]. Journal of Cleaner Production, 2021, 293: 126093.

[139] XU W, HU Y, LUO W, et al. A multi-objective scheduling method for distributed and flexible job shop based on hybrid genetic algorithm and tabu search considering operation outsourcing and carbon emission[J]. Computers & Industrial Engineering, 2021, 157: 107318.

[140] 李佳磊,顾幸生.双种群混合遗传算法求解具有预防性维护的分布式柔性作业车间调度问题[J]. 控制与决策, 2023, 38(02): 475-482.

[141] MARZOUKI B, DRISS O B, GHEDIRA K. Improved chemical reaction optimization for distributed flexible job shop problem with transportation times[J]. IFAC-PapersOnLine, 2022, 55(10): 1249-1254.

[142] LI J, GU X, ZHANG Y, et al. Distributed Flexible Job-Shop Scheduling Problem Based on Hybrid Chemical Reaction Optimization Algorithm[J]. Complex System Modeling and Simulation, 2022, 2(2): 156-173.

[143] LI X, XIE J, MA Q, et al. Improved gray wolf optimizer for distributed flexible job shop scheduling problem[J]. Science China Technological Sciences, 2022, 65(9): 2105-2115.

[144] 罗文冲. 超启发式交叉熵算法求解分布式装配柔性作业车间调度问题[J]. 控制理论与应用, 2021, 38(10): 1551-1568.

[145] 李佳路. 基于改进人工蜂群算法的分布式柔性作业车间调度问题研究[D]. 安徽工程大学, 2022.

[146] LUO Q, DENG Q, GONG G, et al. A distributed flexible job shop scheduling problem considering worker arrangement using an improved memetic algorithm[J]. Expert Systems with Applications, 2022, 207: 117984.

[147] 唐红涛, 李 悦, 王 磊. 模糊分布式柔性作业车间调度问题的求解算法[J]. 华中科技大学报（自然科学版）, 2022, 50(6): 81-88.

[148] 李瑞, 王凌, 龚文引. 知识驱动的模因算法求解分布式绿色柔性调度[J]. 华中科技大学学报(自然科学版), 2022, 50(6): 55-60.

[149] LI R, GONG W Y, WANG L, et al. Two-stage knowledge-driven evolutionary algorithm for distributed green flexible job shop scheduling with type-2 fuzzy processing time[J]. Swarm and Evolutionary Computation, 2022, 74: 101139.

[150] BLACKMORE S. The Meme Machine[M]. New York: Oxford University Press, 1999.

[151] DAWKINS R. The Selfish Gene[M]. Oxford, U.K: Oxford University Press, 1976.

[152] MOSCATO P. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts - Towards Memetic Algorithms[M]. Pasadena, U.S.A: Caltech Concurrent Computation Program, 1989.

[153] ISHIBUCHI H, YOSHIDA T, MURATA T. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling[J]. IEEE Transactions on Evolutionary Computation, 2003, 7(2): 204-223.

[154] LIU B, WANG L, JIN Y H. An Effective PSO-Based Memetic Algorithm for Flow Shop Scheduling[J]. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), 2007, 37(1): 18-27.

[155] RAEESI N. M R, KOBTI Z. A memetic algorithm for job shop scheduling using a critical-path-based local search heuristic[J]. Memetic Computing, 2012, 4(3): 231-245.

[156] PAN Q K, WANG L, SANG H Y, et al. A High Performing Memetic Algorithm for the Flowshop Scheduling Problem with Blocking[J]. IEEE Transactions on Automation Science and Engineering, 2013, 10(3): 741-756.

[157] MENCÍA R, SIERRA M R, MENCÍA C, et al. Memetic algorithms for the job shop scheduling problem with operators[J]. Applied Soft Computing, 2015, 34: 94-105.

[158] WANG S Y, 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, and Cybernetics: Systems, 2016, 46(1): 139-149.

[159] DENG J, WANG L. A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem[J]. Swarm and Evolutionary Computation, 2017, 32: 121-131.

[160] KURDI M. An improved island model memetic algorithm with a new cooperation phase for multi-objective job shop scheduling problem[J]. Computers & Industrial Engineering, 2017, 111: 183-201.

[161] GONG G, DENG Q, CHIONG R, et al. An effective memetic algorithm for multi-objective job-shop scheduling[J]. Knowledge-Based Systems, 2019, 182: 104840.

[162] SHAO W S, PI D C, SHAO Z S. Memetic algorithm with node and edge histogram for no-idle flow shop scheduling problem to minimize the makespan criterion[J]. Applied Soft Computing, 2017, 54: 164-182.

[163] ENGIN B E, ENGIN O. A new memetic global and local search algorithm for solving hybrid flow shop with multiprocessor task scheduling problem[J]. SN Applied Sciences, 2020, 2(12): 2059.

[164] DING J, SHEN L, LU Z, et al. A Hybrid Memetic Algorithm for the Parallel Machine Scheduling Problem with Job Deteriorating Effects[J]. IEEE Transactions on Emerging Topics in Computational Intelligence, 2020, 4(3): 385-397.

[165] LEI C, ZHAO N, YE S, et al. Memetic algorithm for solving flexible flow-shop scheduling problems with dynamic transport waiting times[J]. Computers & Industrial Engineering, 2020, 139: 105984.

[166] ABEDI M, CHIONG R, NOMAN N, et al. A multi-population, multi-objective memetic algorithm for energy-efficient job-shop scheduling with deteriorating machines[J]. Expert Systems with Applications, 2020, 157: 113348.

[167] KURDI M. A memetic algorithm with novel semi-constructive evolution operators for permutation flowshop scheduling problem[J]. Applied Soft Computing, 2020, 94: 106458.

[168] LU C, GAO L, GONG W Y, et al. Sustainable scheduling of distributed permutation flow-shop with non-identical factory using a knowledge-based multi-objective memetic optimization algorithm[J]. Swarm and Evolutionary Computation, 2021, 60: 100803.

[169] MARICHELVAM M K, GEETHA M. A memetic algorithm to solve uncertain

energy-efficient flow shop scheduling problems[J]. The International Journal of Advanced Manufacturing Technology, 2021, 115(1-2): 515-530.

[170] WANG J J, WANG L. A Bi-Population Cooperative Memetic Algorithm for Distributed Hybrid Flow-Shop Scheduling[J]. IEEE Transactions on Emerging Topics in Computational Intelligence, 2021, 5(6): 947-961.

[171] WANG J J, WANG L. A Cooperative Memetic Algorithm with Learning-Based Agent for Energy-Aware Distributed Hybrid Flow-Shop Scheduling[J]. IEEE Transactions on Evolutionary Computation, 2022, 26(3): 461-475.

[172] ZHAO Z, LIU S, ZHOU M, et al. Dual-Objective Mixed Integer Linear Program and Memetic Algorithm for an Industrial Group Scheduling Problem[J]. IEEE/CAA Journal of Automatica Sinica, 2021, 8(6): 1199-1209.

[173] LU C, ZHANG B, GAO L, et al. A Knowledge-Based Multiobjective Memetic Algorithm for Green Job Shop Scheduling with Variable Machining Speeds[J]. IEEE Systems Journal, 2022, 16(1): 844-855.

[174] MAO J Y, PAN Q K, MIAO Z H, et al. A hash map-based memetic algorithm for the distributed permutation flowshop scheduling problem with preventive maintenance to minimize total flowtime[J]. Knowledge-Based Systems, 2022, 242: 108413.

[175] AFSAR S, PALACIOS J J, PUENTE J, et al. Multi-objective enhanced memetic algorithm for green job shop scheduling with uncertain times[J]. Swarm and Evolutionary Computation, 2022, 68: 101016.

[176] SHAO W S, SHAO Z S, PI D C. A multi-neighborhood-based multi-objective memetic algorithm for the energy-efficient distributed flexible flow shop scheduling problem[J]. Neural Computing and Applications, 2022, 34(24): 22303-22330.

[177] SHAO W, SHAO Z, PI D. A network memetic algorithm for energy and labor-aware distributed heterogeneous hybrid flow shop scheduling problem[J]. Swarm and Evolutionary Computation, 2022, 75: 101190.

[178] WANG J J, WANG L. A cooperative memetic algorithm with feedback for the energy-aware distributed flow-shops with flexible assembly scheduling[J]. Computers & Industrial Engineering, 2022, 168: 108126.

[179] ZHANG G H, LIU B, WANG L, et al. Distributed Co-Evolutionary Memetic Algorithm for Distributed Hybrid Differentiation Flowshop Scheduling Problem[J]. IEEE Transactions on Evolutionary Computation, 2022, 26(5): 1043-1057.

[180] ZHANG G H, MA X, WANG L, et al. Elite Archive-Assisted Adaptive Memetic

Algorithm for a Realistic Hybrid Differentiation Flowshop Scheduling Problem[J]. IEEE Transactions on Evolutionary Computation, 2022, 26(1): 100-114.

[181] ZHANG G H, WANG L, XING K. Dual-Space Co-Evolutionary Memetic Algorithm for Scheduling Hybrid Differentiation Flowshop With Limited Buffer Constraints[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2022, 52(11): 6822-6836.

[182] ZHANG Q F, LI H. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition[J]. IEEE Transactions on Evolutionary Computation, 2007, 11(6): 712-731.

[183] DEB K, PRATAP A, AGARWAL S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197.

[184] WHILE L, HINGSTON P, BARONE L, et al. A faster algorithm for calculating hypervolume[J]. IEEE Transactions on Evolutionary Computation, 2006, 10(1): 29-38.

[185] WANG Y N, WU L H, YUAN X F. Multi-objective self-adaptive differential evolution with elitist archive and crowding entropy-based diversity measure[J]. Soft Computing, 2010, 14(3): 193-209.

[186] RICHARD C V N. Design of Experiments Using the Taguchi Approach: 16 Steps to Product and Process Improvement[J]. Technometrics, 2002, 44(3): 289-289.

[187] TIAN Y, CHENG R, ZHANG X, et al. An Indicator-Based Multiobjective Evolutionary Algorithm with Reference Point Adaptation for Better Versatility[J]. IEEE Transactions on Evolutionary Computation, 2018, 22(4): 609-622.

[188] MING F, GONG W Y, ZHEN H X, et al. A simple two-stage evolutionary algorithm for constrained multi-objective optimization[J]. Knowledge-Based Systems, 2021, 228: 107263.

[189] BRANDIMARTE P. Routing and scheduling in a flexible job shop by tabu search[J]. Annals of Operations Research, 1993, 41(3): 157-183.

[190] DAUZÈRE-PÉRÈS S. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search[J]. Annals of Operations Research, 1997, 70(0): 281-306.

[191] PRELEC D, SEUNG H S, MCCOY J. A solution to the single-question crowd wisdom problem[J]. Nature, 2017, 541(7638): 532-535.

[192] WU X L, HAN J, CUI Q, et al. Incorporating surprisingly popular algorithm and

euclidean distance-based adaptive topology into pso[J]. Swarm and Evolutionary Computation, 2023, 76, 101222.

[193] MA X, LI X, ZHANG Q, et al. A Survey on Cooperative Co-Evolutionary Algorithms[J]. IEEE Transactions on Evolutionary Computation, 2019, 23(3): 421-441.

[194] WANG X, ZHANG K, WANG J, et al. An Enhanced Competitive Swarm Optimizer with Strongly Convex Sparse Operator for Large-Scale Multiobjective Optimization[J]. IEEE Transactions on Evolutionary Computation, 2022, 26(5): 859-871.

[195] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.

[196] MING F, GONG W Y, WANG L. A Two-Stage Evolutionary Algorithm with Balanced Convergence and Diversity for Many-Objective Optimization[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2022, 52(10): 6222-6234.

[197] TIAN Y, LI X, MA H, et al. Deep Reinforcement Learning Based Adaptive Operator Selection for Evolutionary Multi-Objective Optimization[J/OL]. IEEE Transactions on Emerging Topics in Computational Intelligence, 2022: 1-14[2022-02-11]. https://ieeexplore.ieee.org/document/9712324