分 类 号 _____ 学号 M201970707

学校代码 10487 密级 _____

华中科技大学

硕士学位论文

考虑准备时间的多资源受限柔性作业车间调度方法研究

学位申请人： 崔航浩

学 科 专 业： 工业工程

指 导 教 师： 高 亮 教授

李培根 教授

答 辩 日 期： 2021 年 5 月 17 日

Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree for the Master of Engineering

Research on Multi-Resource Constrained Flexible Job Shop Scheduling with Setup Time

Candidate ： Cui Hanghao

Major ： Industrial Engineering

Supervisor： Prof. Gao Liang

Prof. Li Peigen

Huazhong University of Science & Technology

Wuhan 430074, P.R. China

May, 2021

摘 要

柔性作业车间调度问题（Flexible Job Shop Scheduling Problem，FJSP）是近年来车间调度领域的研究热点。在经典 FJSP 的研究中，通常不考虑准备时间，或者将准备时间作为加工时间的一部分。但在许多现实的生产环境中，准备时间往往与加工机器和工艺路线相关，故不能忽视或者被当作加工时间的一部分。在实际生产活动中，除加工机器以外，一些昂贵而稀缺的辅助资源（如特种夹具、刀具、维护设备等）也需要进行合理的分配。因此，考虑准备时间的多资源受限柔性作业车间调度问题（Multi-Resource Constrained Flexible Job Shop Scheduling Problem，MRC-FJSP）具有较高的研究价值。本文所做的主要工作如下。

首先，以最小化最大完工时间为目标，构建考虑准备时间的 FJSP 数学模型，并用改进粒子群优化（Improved Particle Swarm Optimization，IPSO）算法进行求解。在 IPSO 中，给出了一种混合初始化策略，用于获得较高质量的初始种群；研究了基于关键路径的邻域结构（同机器的邻域结构和换机器的邻域结构），用于提高算法的局部开发能力；使用自适应禁忌搜索算子以避免算法陷入循环搜索。通过对比实验，验证了 IPSO 算法的优越性。

其次，以最小化最大完工时间为目标，建立了考虑准备时间的 MRC-FJSP 数学模型，并结合考虑准备时间的 MRC-FJSP 特征，对 IPSO 算法进行了调整。给出了一种半主动解码与启发式规则相结合的新型解码方式，对原有解空间进行了有效裁剪；研究了一种自适应邻域搜索算子，用于提高算法的局部开发能力；提出了基于随机网络结构的多种群策略，用来提高算法的全局搜索能力。IPSO 算法刷新了大多数标准算例的当前最好解，在收敛性和稳定性方面均具有明显优势。

然后，以最小化最大完工时间和最小化机器最大负荷为目标，创建了考虑准备时间的多目标 MRC-FJSP 数学模型，并结合多目标优化基础理论，设计多目标粒子群优化（Multi-Objective Particle Swarm Optimization，MOPSO）算法进行求解。在 MOPSO 中，引入了外部记忆库，用于存储搜索过程中找到的非支配解；应用快速非支配排序、拥挤距离等方法更新种群，保障了 Pareto 前沿的收敛性和分布均匀性；提出了面向

不同优化目标的邻域结构，用于提高算法的局部开发能力。标准测试算例的求解结果展现了 MOPSO 算法在各项性能指标上的优越性。

最后，总结了本文的主要工作，并展望了今后的研究方向。

关键词：柔性作业车间调度；多资源受限；改进粒子群优化算法；随机网络；多目标优化

Abstract

Flexible job shop scheduling problem (FJSP) is a research hot spot in the field of workshop scheduling in recent years. In the study of classical FJSP, setup time is usually not considered, or taken as a part of the processing time. However, in many realistic production environments, setup time is often related to the processing machine and process route, so it can not be ignored or regarded as a part of the processing time. In the actual production activities, in addition to processing machine, some expensive and scarce additional resources (such as special fixtures, tools, maintenance equipment, etc.) also need to be allocated reasonably. Therefore, the research of multi-resource constrained flexible job shop scheduling problem (MRC-FJSP) with setup time has high research value. The main work of this paper is as follows.

Firstly, with the goal of minimizing the maximum completion time, a mathematical model of FJSP with setup time is established, and an improved particle swarm optimization (IPSO) algorithm is proposed to solve the problems. In IPSO, A hybrid initialization strategy is proposed to obtain a high quality initial population; a neighborhood structure based on the critical path (the neighborhood structure of the same machine and the neighborhood structure of changing machine) is designed to improve the local development ability of the algorithm; An adaptive tabu search operator is designed to prevent the algorithm from falling into circular search. Through comparative experiments, the superiority of IPSO algorithm is verified.

Secondly, with the goal of minimizing the maximum completion time, a mathematical model of FJSP with setup time is established, and combined with characteristics of MRC-FJSP with setup time, the IPSO algorithm is adjusted. A new decoding method which combines semi-active decoding and heuristic rule decoding is designed, the original solution space is cut out effectively. An adaptive neighborhood search strategy is proposed to improve the local search ability of the algorithm. A multi-population strategy based on a

random network structure graph is added to improve the global search ability of the algorithm. IPSO algorithm refreshes the current best solutions of most standard calculation examples and has obvious advantages in convergence and stability.

Then, with the goal of minimizing the maximum completion time and the maximum machine load, a multi-objective MRC-FJSP mathematical model with setup time is established, and combined with the basic theory of multi-objective optimization, a multi-objective particle swarm optimization (MOPSO) algorithm is proposed to solve the problems. In MOPSO, an external memory bank is introduced to store the non-dominant solutions found in the search process; The methods of fast non-dominant sorting and crowding distance are used to update the population, which ensures the convergence and distribution uniformity of the Pareto front. Different neighborhood structures for different optimization objectives are designed to improve the local search ability of the algorithm. The test results of the standard examples show the superior performance of MOPSO algorithm.

Finally, the main work in this article is summarized, and future research directions are prospected.

Keywords: Flexible Job Shop Scheduling; Multi-Resource Constrained; Improved Particle Swarm Optimization Algorithm; Random Network; Multi-Objective Optimization

目录

1 绪论

1.1 课题来源、背景、目的与意义

1.1.1 课题来源

本文的研究工作来源于：

（1）国家杰出青年科学基金项目“车间调度的理论与方法”，项目编号：51825502。

（2）**机械有限公司横向课题“壳段加工生产线智能优化分析与决策技术研究”。

1.1.2 课题背景

考虑准备时间的 MRC-FJSP 在实际生产过程中有着广泛的应用。集成电路器件的最终测试就是一个典型的考虑准备时间的 MRC-FJSP[1]。由于产品规格的差异，集成电路器件可能需要不同的最终测试步骤[2]。机器只有装备了合适的测试机、处理机及其它配件才能对指定的工件进行测试。测试机是用于加载测试程序的中央处理单元，通过测试探头与封装好的集成电路器件相连；处理机保障了测试能够在不同的温度下进行；传送臂负责从托盘或管道中释放集成电路器件；嵌套将集成电路器件托举并装载到电气接口[3]。一个最终测试机器上所用的测试机、处理机、传送臂和嵌套的种类和数量由机器型号决定。钢铁行业的炼钢-连铸调度问题是钢铁生产中最大的瓶颈问题，也属于考虑准备时间的 MRC-FJSP 的研究范畴[4-5]。炼钢-连铸过程包含炼钢、多重精炼、连铸等环节。通常情况下，工件在多重精炼阶段的工艺路线是不同的，用于运输工件的起重机也是有限的。当前如果没有空闲的起重机，即使加工机器处于空闲状态，也必须推迟工序的开工时间。在大型数控机床加工领域也有考虑准备时间的 MRC-FJSP 的应用场景，其中的特种工装、刀具和夹具都具有高柔性的特征，一台大型数控机床往往需要在多个辅助资源的共同配合下才能完成加工任务。然而在许多情况下，辅助资源数量常常是受限的，需要进行合理的分配[6]。

1.1.3 目的与意义

FJSP 是近年来车间调度领域的研究热点。在经典 FJSP 的研究中，通常不考虑准备时间，或者将准备时间作为加工时间的一部分。但在许多现实的生产环境中，准备时间往往与加工机器和工件的工艺路线有关，故不能忽视或者被当作加工时间的一部分。在现实生产环境中，除加工机器以外，一些昂贵而稀缺的辅助资源（如特种夹具、刀具、维护设备等）也需要进行合理的分配[7]。为了使研究内容更贴合实际，本文把考虑准备时间的多资源受限柔性作业车间调度作为主要研究对象。

FJSP 是一类经典的 NP-hard 组合优化问题[8]。考虑准备时间的 MRC-FJSP 在 FJSP 的基础上增添了辅助资源的调度和准备时间，是比 FJSP 更为复杂的 NP-hard 问题，确定性算法和近似算法都无法在多项式时间内找到大规模考虑准备时间的 MRC-FJSP 的全局最优解[9]。

综上所述，考虑准备时间的多资源受限柔性作业车间调度问题具有较高的研究价值，能够为实际生产活动提供有益借鉴。

1.2 国内外研究现状

FJSP 在实际生产应用中，往往需要考虑准备时间和辅助资源约束，现存大多数文献中也是把准备时间作为 MRC-FJSP 的必备条件。为了使研究内容更贴合实际，本文以考虑准备时间的多资源受限柔性作业车间调度为主要研究对象。在此之前，首先研究一个基础问题——考虑准备时间的 FJSP，然后在这个基础问题上加入辅助资源约束，最后，同时考虑多个优化目标。考虑到粒子群优化算法易于实现、鲁棒性强的特点，本文提出的所有求解算法都使用 PSO 算法的框架结构。下面分别讨论考虑准备时间的柔性作业车间调度问题、考虑准备时间的多资源受限柔性作业车间调度问题以及粒子群优化算法的研究现状。

1.2.1 考虑准备时间的柔性作业车间调度问题研究现状

FJSP 是一个经典的 NP-hard 组合优化问题。多年来，有许多学者和工程技术人员参与到 FJSP 的研究中[10]。目前大多数 FJSP 的研究文献通常忽视准备时间或将其

视为加工时间的一部分，但是在许多实际生产环境（化工、印刷、制药和汽车制造等）中，准备时间与加工机器、工件的工艺路线有着紧密联系，故不能忽视或被当作加工时间的一部分[11]。目前，FJSP 已被广泛研究，但是考虑准备时间的文献还比较少。

在单目标优化领域，Imanipour 以最小化最大完工时间为目标，建立了非线性混合整数规划模型，并采用禁忌搜索算法（Tabu Search，TS）进行求解[12]。针对考虑准备时间的 FJSP，Sadrzadeh 设计了混合人工免疫-粒子群优化算法，对比实验结果证实了所提算法的有效性[13]。Defersha 和 Rooyani 设计了一种两阶段遗传算法。第一阶段先对工序排列向量进行遗传操作，当工序之间的加工顺序确定后，再从可选机器集中选择能够最早完成该工序的机器；第二阶段同时对工序排列向量和机器分配向量进行遗传操作，使算法有机会搜索到因第一阶段的贪婪属性被裁剪掉的解[14]。Mousakhani 对 Imanipour 的模型进行改进，建立了混合整数线性规划模型，设计了基于局部迭代搜索的元启发式算法，并将其与禁忌搜索算法和变邻域搜索算法进行比较，实验结果表明所提算法的性能更好[15]。

在多目标优化领域，Zandieh 采用线性加权的思想把多目标优化问题转换为单目标优化问题，并结合问题特性，改进了变邻域搜索算法（Variable Neighborhood Search，VNS）[16]。Li 等提出一种精英非支配排序混合算法，同时优化最大完工时间和总准备时间两个目标，最后用 39 个基准算例和 1 个真实生产案例验证了算法的有效性和应用价值[17]。最近，Zhu 等以最小化最大完工时间、机器最大负荷、总机器负荷为目标，建立了 MILP 模型，并提出一种多目标进化灰狼优化算法，设计了改进社群等级制度和多样化引导策略，用于提高算法的收敛速度和种群多样性[18]。

与此同时，考虑准备时间的 FJSP 的衍生问题也得到了许多专家学者的关注。Azzouz 等使用双层进化优化法处理具有恶化因子的考虑准备时间的 FJSP [19]。Rossi 使用带有增强信息素的蚁群算法研究了带运输时间和准备时间的 FJSP[20]。Zhang 等使用基于竞争与协作机制的候鸟优化算法研究了带有可变分批和准备时间的 FJSP，测试案例的结果展现了所提算法的优越性能[21]。

1.2.2 考虑准备时间的多资源受限柔性作业车间调度问题研究现状

考虑准备时间的 MRC-FJSP 是一类复杂而又普遍存在的车间调度问题[22]。自问题提出以来，就得到了许多国内外专家学者和工程技术人员的关注。目前解决车间调度问题的方法主要分为两类：精确方法(Exact Method)和近似方法(Approximation Method)。精确求解法是运筹学里的最优化方法，有整数规划法、混合整数规划法、拉格朗日松弛法、分枝定界法、回溯法等。由于考虑准备时间的 MRC-FJSP 具有高柔性、高计算复杂度的特征，单纯依靠精确方法求解考虑准备时间的 MRC-FJSP 会消耗大量计算资源和时间[23]。由于近似方法可以在合理的时间范围内求得一个较优解，所以在考虑准备时间的 MRC-FJSP 的科学研究和工程实践中得到了广泛应用。

2004 年 Lin 等提出了用于确定辅助资源分配优先顺序的启发式规则[24]。2004 年 Pearn 等提出了考虑工件重入的多阶段集成电路最终测试调度算法，并从最小化机器总负荷的角度对算法性能进行评估[25]。2007 年 Song 等应用蚁群算法开发了半导体最终测试厂的生产调度系统[26]。2008 年 Wu 等建立了一个具有较高的鲁棒性和通用性的 MILP 模型，并设计了一种针对此问题的遗传算法(Wu and Chien Genetic Algorithm，wcGA)，最后通过数值实验验证了 wcGA 在求解考虑准备时间的 MRC-FJSP 上的优势[9]。2012 年 Wu 等提出了一种结合新型双向量编码方式的遗传算法(bi-vector encoding Genetic Algorithm，bvGA)。不同于传统的双向量编码（工序排列向量和机器分配向量），bvGA 用获取规则（机器分配启发式规则）向量取代了机器分配向量，在确定工序的加工顺序后，依据获取规则为待加工工序分配机器[27]。2014 年 Hao 等提出采用合作分布估计算法(Cooperative Estimation of Distribution Algorithm，CEDA)求解考虑准备时间的 MRC-FJSP。在 CEDA 中，引入了分治策略以提高算法的寻优效率[2]。2015 年 Wang 等提出一种混合分布估计算法(Hybrid Estimation of Distribution Algorithm，HEDA)。在 HEDA 中，设计了一种新的编解码方式，可以有效地将解空间映射到编码空间；HEDA 通过概率抽样得到新解后，对新解执行局部搜索，提高了算法的挖掘能力[28]。2015 年 Zheng 等提出采用新型果蝇优化算法(novel Fruit-fly Optimization Algorithm，nFOA)求解考虑准备时间的 MRC-FJSP。在 nFOA

中，基于问题特征设计了嗅觉搜索算子和视觉搜索算子，提高了种群的局部开发能力；提出了一种合作搜索机制，用于模拟果蝇之间的信息交流行为[1]。2015 年 Wang 等提出采用基于知识库的多智能体进化算法(Knowledge-based Multi-agent Evolutionary Algorithm，KMEA)求解考虑准备时间的 MRC-FJSP。在 KMEA 中，使用混合初始化机制来保障初始智能体群的质量，设计了基于复杂网格模型的多智能体学习机制和竞争机制[29]。2016 年 Gao 等提出采用混合多群微候鸟优化算法(Shuffled Multi-swarm Micro-migrating Birds Optimizer，SM^2-MBO)求解考虑准备时间的 MRC-FJSP。在 SM^2-MBO 中，设计了多种群并行搜索策略、随机重组策略以及多样性维护策略[3]。2018 年 Sang 等提出采用协作式协同进化入侵杂草优化算法(Cooperative Co-evolutionary Invasive Weed Optimization，CCIWO)求解考虑准备时间的 MRC-FJSP。CCIWO 采用两个耦合的子群(一个子群处理机器分配问题，一个子群处理工序排列问题)进行迭代搜索[30]。2019 年 Cao 等提出采用基于强化学习和代理模型的布谷鸟搜索算法(Cuckoo Search Algorithm With Reinforcement Learning and Surrogate Modeling，CSRS)求解考虑准备时间的 MRC-FJSP [31-32]。2019 年付坤提出采用鲸鱼群算法(Whale Swarm Algorithm，WSA)求解考虑准备时间的 MRC-FJSP。在 WSA 中，设计了启发式初始化算子和基于贪婪前移的解码策略[33]。

1.2.3 粒子群优化算法研究现状

PSO 算法是群智能优化算法家族中的重要成员。受自然界中鸟群觅食行为的启发，Eberhart 等在 1995 年首次提出了 PSO 算法[34]。PSO 算法原理简单，需要调整的参数较少，易于实现，具有强大的鲁棒性和更快的收敛性[35-36]。基本 PSO 算法的流程图见图 1-1。

PSO 算法将现实中的鸟简化为一个没有质量和体积的粒子[37]。当粒子群寻优时，粒子会借助“个体认知”和“社会认知”到达近优解或最优解。

（1）“个体认知”是粒子的自我思考过程。它通过自身邻域的开发更新粒子个体的历史最优位置，主要承担算法的局部开发任务；

（2）“社会认知”是粒子之间信息共享、相互学习的过程。通过向最好个体学习，

提高粒子的适应度值，主要承担算法的全局勘探任务。

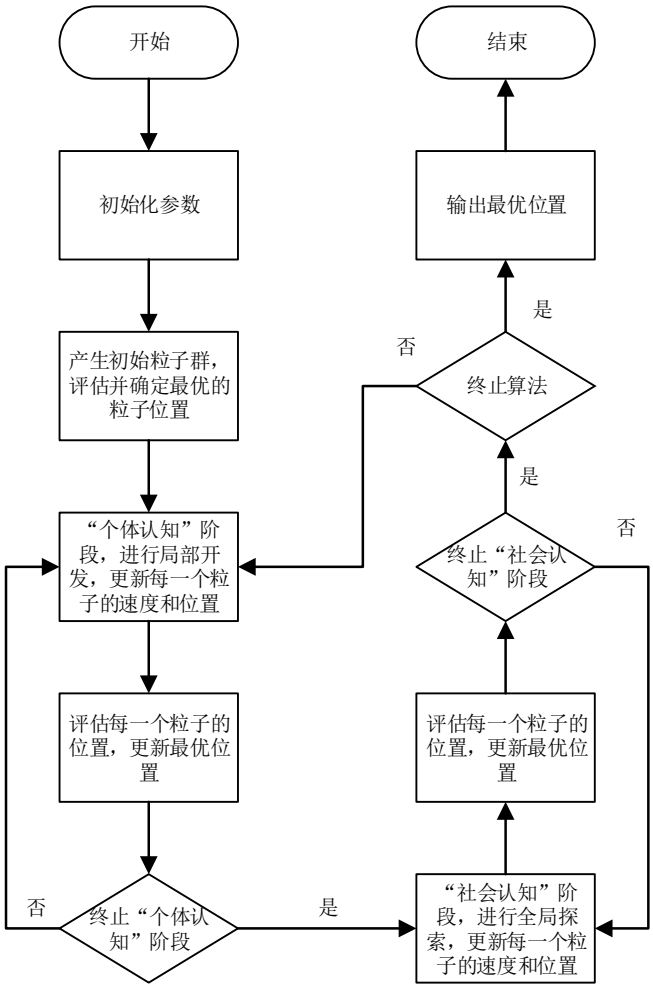


图 1-1 基本 PSO 算法流程图

PSO 算法自提出以来，受到许多专家和学者的关注，他们把 PSO 算法应用到各自的研究领域，取得了丰硕的成果。在计算机网络方面，Bonnah 等利用粒子群优化算法实现了无线传感器网络的最大化覆盖[38]。Asadi 等提出了基于 PSO 算法的僵尸网络检测技术[39]。在生态环境方面，针对双层污染路径规划问题，Qiu 等设计了一种基于模糊逻辑控制的 PSO 算法[40]。Zhao 等提出使用微分进化-交叉量子粒子群优化算法解决环境经济调度问题[41]。为了在输入数据有限的情况下估计西北干旱区的日平均蒸散量，Zhu 等提出了一种新的混合极值学习机模型，并借助 PSO 算法优化极值学习机模型的参数[42]。在生物医学方面，Li 等设计了一种基于深度学习和 PSO 算法的计算框架，用于预测物种特异性 S-谷胱甘肽化位点[43]。Cai 等提出了基于 PSO

的临床柔性穿刺路径规划技术[44]。在能源基建方面，Zhang 等应用粒子群优化方法对多形式能量中心进行优化设计[45]。Chen 等提出了基于 PSO 的全波形激光雷达回波分解方法[46]。针对具有预定几何形状和多种自应力状态的索杆结构，Chen 等设计了基于 PSO 算法的索杆结构最优可行预应力模式计算方法[47]。Zhang 等开发了基于并行 PSO 的金沙江流域水库多目标调度系统[48]。Sabanci 等借助 PSO 算法建立了 DC-DC 升压电路输出电压纹波的表达式[49]。在经济金融方面，Song 等使用 PSO 算法修正了世界上广泛使用的 KMV 模型，使之适合中国区块链金融市场[49]。Ganji 等使用多目标粒子群算法解决了绿色生产配送综合调度问题[51]。在机器视觉方面，Farshi 等提出一种基于多模态粒子群优化的图像分割方法[52]。Dadjoo 等利用 PSO 改进了图像聚类算法的初始化过程。与原初始化方法相比，聚类结果的总体精度提高了约 20%[53]。

1.3 现状分析与总结

通过对上述国内外研究现状进行分析，可以得到以下结论：

（1）到目前为止，涉及考虑准备时间的 MRC-FJSP 的研究文献相对较少，且大多数文献都集中于以最小化最大完工时间(makespan)为目标的单目标优化领域。考虑准备时间的多目标 MRC-FJSP 的研究尚处于起步阶段。

（2）近些年考虑准备时间的 MRC-FJSP 的相关研究大多局限于群智能优化算法的突破创新上，涉及考虑准备时间的 MRC-FJSP 数学模型方面的文献非常少，2012 年 Wu 等[27]建立的 MILP 模型已经不能满足当前科学研究与工程实践的需求。开发适用性强、鲁棒性高的考虑准备时间的 MRC-FJSP 的数学模型迫在眉睫。

（3）考虑准备时间的 MRC-FJSP 的求解策略经历了从启发式规则到群智能算法的转变，经历了由单一种群进化到基于分治思想的多种群协同进化的转变，求解效率得到了很大改善。但是目前依然存在以下问题制约着考虑准备时间的 MRC-FJSP 求解效率的进一步提升：

1）对考虑准备时间的 MRC-FJSP 本质特征的挖掘力度不够，没有很好地实现领域知识与群智能优化算法的有机结合，使得算法的求解效率受到很大的限制；

2）搜索过程中获得的有效信息的利用率较低。单纯依靠个体层次的信息共享，

会导致算法后期收敛缓慢，影响最终解的质量；

3）算法局部开发能力不足。算法的局部开发主要依赖个体近邻解的获取。现存考虑准备时间的 MRC-FJSP 求解算法的局部搜索还是直接使用传统的邻域结构，未设计出高效的局部搜索算子；

4）缺少面对搜索停滞的应急机制。由于考虑准备时间的 MRC-FJSP 具有高柔性和高计算复杂度的特征，算法在搜索最优解或近优解的过程中很容易陷入早熟收敛。如果没有跳出局部最优区域的应急机制，解质量的稳定性很难得到保证。

1.4 文章内容与组织结构

本文以考虑准备时间的 MRC-FJSP 为主要研究对象。首先研究了一个基础问题——考虑准备时间的 FJSP，之后在此基础上加入了辅助资源约束，最后同时考虑最小化最大完工时间和最小化机器最大负荷两个目标。针对研究问题的特征，设计了基于 PSO 框架结构的调度算法。本文的主要工作可总结如下：

（1）研究了考虑准备时间的 FJSP，建立了相应的数学模型，并提出了一种 IPSO 算法进行求解；

（2）研究了考虑准备时间的 MRC-FJSP，建立了相应的数学模型，并提出了一种基于 IPSO 的求解方法；

（3）研究了考虑准备时间的多目标 MRC-FJSP，建立了相应的数学模型，并提出了一种 MOPSO 算法进行求解；

本文的组织结构如图 1-2 所示，其中各章节的主要内容如下：

第一章，介绍课题来源、背景、目的与意义。针对考虑准备时间的 FJSP、考虑准备时间的 MRC-FJSP 以及粒子群优化算法等的研究现状进行概述、分析与总结，并进一步阐述本文的主要研究内容与组织结构。

第二章，研究考虑准备时间的 FJSP，以最小化最大完工时间为优化目标，建立相应数学模型，提出 IPSO 算法求解该问题。针对考虑准备时间的 FJSP 特性，设计基于关键路径的邻域结构，并把禁忌搜索策略应用于 IPSO 算法中。最后设计数值实验验证所提算法的有效性和优越性。

第三章，研究考虑准备时间的 MRC-FJSP，以最小化最大完工时间为优化目标，建立相应数学模型。针对考虑准备时间的 MRC-FJSP 特性，对第二章的 IPSO 算法进行修改。提出一种半主动解码与启发式规则相结合的新型解码方式，设计自适应邻域搜索算子，提出基于随机网络结构的多种群策略和子种群重新初始化策略。最后利用标准测试算例检验所提算法的各项性能。

第四章，研究考虑准备时间的多目标 MRC-FJSP，同时考虑最小化最大完工时间和最小化机器最大负荷两个目标，建立相应数学模型，并结合多目标优化基础理论，设计 MOPSO 算法求解该问题。最后利用标准测试算例检验所提算法在各项性能指标上的优越性。

第五章，对本文的主要工作进行总结，并对今后的研究方向进行展望。

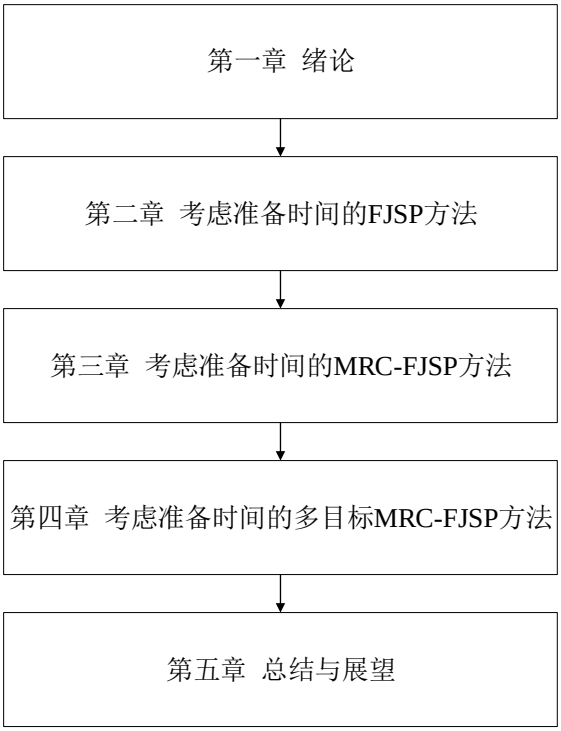


图 1-2 论文组织结构图

2 考虑准备时间的 FJSP 方法

2.1 问题描述与数学模型建立

2.1.1 问题描述

考虑准备时间的 FJSP 可以被描述如下：

（1）有 n 个工件(J_1，J_2，…，J_n)要在 m 台机器(M_1，M_2，…，M_m)上加工；

（2）每个工件包含 n_i 道工序，且工序的先后顺序是预先确定的；

（3）每道工序 O_{ij} 都可以从可选加工机器集 Ω_{ij} 中任选一台机器完成加工；

（4）工序 O_{ij} 在机器 M_k 上的加工时间是固定且已知的；

（5）每台机器在同一时刻只能加工一道工序；

（6）每个工件在同一时刻只能被一台机器加工；

（7）所有工序一旦开始加工就不允许中断；

（8）同一工件的工序之间有优先级约束，不同工件的工序之间没有优先级约束；

（9）所有工件在零时刻都可以被加工，所有机器在零时刻都可以被使用；

（10）考虑工件在不同机器上转换所需的准备时间。

2.1.2 数学模型建立

为了方便后文描述，特定义以下符号：

n：工件总数；

m：机器总数；

Ω：总的机器集；

k：机器序号，$k=1,2,3,\cdots,m$；

i：工件序号，$i=1,2,3,\cdots,n$；

l_i：工件 i 的工序总数；

j：工序序号，$j=1,2,3,\cdots,l_i$；

O_{ij}： 工件i的第j道工序；

Ω_{ij}：工序O_{ij}的可选加工机器集；

m_{ij}：工序O_{ij}的可选加工机器数；

p_{ijk}：工序O_{ij}在机器k上的加工时间；

$t_{k'k}$：工件从机器k'转换到机器k所需的时间；

s_{ij}：工序O_{ij}加工开始时间；

c_{ij}：工序O_{ij}加工完成时间；

$PM\left[O_{ij}\right]$：工序O_{ij}的机器紧前工序；

L：一个足够大的正数；

$C_{\max}$：最大完成时间；

决策变量：

X_{ijk}：0-1 决策变量，如果工序O_{ij}在机器k上加工则取 1，否则取 0；

$Y_{iji'j'k}$：0-1 决策变量，如果工序O_{ij}先于工序$O_{i'j'}$在机器k上加工则取 1，否则取 0；

目标函数：

$$\min C_{\max} = \max\left\{s_{il_i} + X_{il_ik}\, p_{il_ik}\right\} \tag{2-1}$$

式中：$i = 1,2,3,\cdots,n$； $k = 1,2,3,\cdots,m_{il_i}$。

约束条件：

$$s_{ij} + X_{ijk}\, p_{ijk} \le c_{ij} \tag{2-2}$$

式中：$i = 1,2,3,\cdots,n$； $j = 1,2,3,\cdots,l_i$； $k = 1,2,3,\cdots,m_{ij}$。

$$c_{ij} \le s_{i(j+1)} \tag{2-3}$$

式中：$i = 1,2,3,\cdots,n$； $j = 1,2,3,\cdots,l_i - 1$。

$$c_{ij} \le C_{\max} \tag{2-4}$$

式中：$i=1,2,3,\cdots,n$；$j=1,2,3,\cdots,l_i$。

$$s_{ij} + X_{ijk} p_{ijk} \le s_{i'j'} + L\left(1 - Y_{iji'j'k}\right) \tag{2-5}$$

式中：$i,i'=1,2,3,\cdots,n$；$j=1,2,3,\cdots,l_i$；$j'=1,2,3,\cdots,l_{i'}$；$k=1,2,3,\cdots,m$。

$$c_{ij} \le s_{i(j+1)} + L\left(1 - Y_{i'j'i(j+1)k}\right) \tag{2-6}$$

式中：$i,i'=1,2,3,\cdots,n$；$j=1,2,3,\cdots,l_i-1$；$j'=1,2,3,\cdots,l_{i'}$；$k=1,2,3,\cdots,m$。

$$\sum_{k=1}^{m_{ij}} X_{ijk} = 1 \tag{2-7}$$

式中：$i=1,2,3,\cdots,n$；$j=1,2,3,\cdots,l_i$。

$$\sum_{i=1}^{n}\sum_{j=1}^{l_i} Y_{iji'j'k} = X_{i'j'k} \tag{2-8}$$

式中：$i'=1,2,3,\cdots,n$；$j'=1,2,3,\cdots,l_{i'}$；$k=1,2,3,\cdots,m$。

$$\sum_{i'=1}^{n}\sum_{j'=1}^{l_{i'}} Y_{iji'j'k} = X_{ijk} \tag{2-9}$$

式中：$i=1,2,3,\cdots,n$；$j=1,2,3,\cdots,l_i$；$k=1,2,3,\cdots,m$。

$$s_{ij} \ge 0, c_{ij} \ge 0 \tag{2-10}$$

式中：$i=1,2,3,\cdots,n$；$j=1,2,3,\cdots,l_i$。

$$s_{ij} - \left(s_{PM[O_{ij}]} + p_{PM[O_{ij}]k}\right) \ge t_{k'k} \tag{2-11}$$

式中：$i=1,2,3,\cdots,n$；$j=1,2,3,\cdots,l_i$；$k=1,2,3,\cdots,m_{ij}$；$k'=1,2,3,\cdots,m_{i'j'}$。

公式（2-2）和公式（2-3）表示同一工件的不同工序之间的优先级约束；公式（2-4）用来保证每一个工件的完工时间不会超过总的完工时间；公式（2-5）和公式（2-6）表示每台机器在同一时间只能加工一道工序；公式（2-7）表示每个工件在同一时

间只能被一台机器加工；公式（2-8）和公式（2-9）表示每一台机器都存在循环操作；公式（2-10）是非负约束；公式（2-11）用来确保一道工序开始加工前，其所选机器已完成准备工作。

2.2 改进粒子群优化算法求解考虑准备时间的 FJSP

2.2.1 编码与解码

（1）编码

考虑准备时间的 FJSP 在经典 FJSP 的基础上加入了准备时间，包含工序排列子问题和机器分配子问题。编码方式采用 FJSP 最常用的基于工件编号和机器编号的双层编码。第一层为工序排列向量，使用相同的工件编号来表示同一个工件的所有工序。以图 2-1 为例，工序排列向量{1,3,2,1,2,4,1}表示的加工序列为{O_{11}, O_{31}, O_{21}, O_{12}, O_{22}, O_{41}, O_{13} }。第二层为机器分配向量，其长度与工序排列向量相同。从左到右，依次表示 1 号工件第 1 道工序至最后一道工序的机器选择、2 号工件第 1 道工序至最后一道工序的机器选择、…、*n* 号工件第 1 道工序至最后一道工序的机器选择。以图 2-1 为例，机器分配向量{1,3,1,1,2,2,3}表示的各工序与所选机器的对应关系为{(O_{11},1), (O_{12},3), (O_{13},1), (O_{21},1), (O_{22},2), (O_{31},2), (O_{41},3)}。

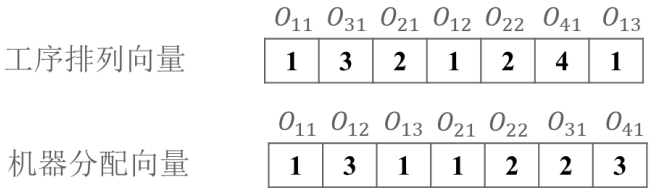


图 2-1 个体编码示意图

（2）解码

考虑准备时间的FJSP的解码过程就是确定每道工序在所选机器上的开始加工时间。解码策略需要与编码方式相适配、与待求解问题相适配。由于待求解问题的工件规模和机器规模都比较大，出于时间开销方面的考虑，本章采用半主动解码方式。即按照工序排列向量上给出的各工序的优先级顺序，顺次把相应工序附加到其所选机器的加工序列后面。对于考虑准备时间的 FJSP 而言，如果要开始加工某一道工序，

需要同时满足两个条件：

1）工件已到达所选机器；

2）所选机器处于空闲状态，且已完成加工前的准备工作。

基于以上约束条件，考虑准备时间的 FJSP 各工序的开始加工时间可以通过公式（2-12）计算得到。

$$s_{ij} = \max\left(s_{ij-1} + p_{ij-1k'}, A_k + t_{k'k}\right) \tag{2-12}$$

式中 $s_{ij-1} + p_{ij-1k'}$ 表示工件 i 的最早可用时间（工序 O_{ij} 的工件紧前工序 O_{ij-1} 的完工时间），A_k 表示机器 k 的释放时间，$A_k + t_{k'k}$ 表示机器 k 的最早可用时间。

2.2.2 种群初始化

种群初始化是群智能优化算法中的重要环节，初始解的质量对算法的求解效率和最终解的质量有着很大影响。目前，大多数文献中普遍采用方便、快捷的随机初始化方法，但这种方式获得的初始解质量偏低，要想达到最优解或近似最优解，只能通过增加算法迭代次数或种群规模进行补偿，因此时间耗费会大大增加。

基于上述考虑，为了提高初始解的质量，本章采用混合初始化方式。其中工序排列向量采用随机初始化方式，机器分配向量依照预设的概率随机从四种不同的初始化策略中选择一个。其中选择随机初始化的概率为 70%，选择最早完工时间优先规则（Earliest Completion Time，ECT）的概率为 10%，选择最短加工时间优先规则（Shortest Process Time，SPT）的概率为 10%，选择准备时间与加工时间之和最短优先规则（Shortest Setup plus Process Time，SSPT）的概率为 10%。

2.2.3 种群“社会认知”阶段

种群“社会认知”阶段是算法的全局搜索，是种群中的较差个体向较优个体靠近的过程。种群“社会认知”阶段的具体操作步骤如下：

Step1：按照最大完工时间由小到大的顺序对种群中的个体进行排序；

Step2：设置 $t=1$，重复 Step3 到 Step6 直至 $t > \frac{N}{3}$；

Step3：从排名在前 50%的个体中随机抽出一个个体 v_{o1}。从整个种群中随机抽出

一个个体v_{o2}（$v_{o2} \neq v_{o1}$）。在v_{o1}（v_{o2}）的交叉表中分别记录v_{o2}（v_{o1}），保证每次参与交叉的粒子都未曾与对方进行过信息交互；

Step4：对v_{o1}和v_{o2}执行交叉操作，其中工序排列向量采用改进优先工序交叉（Improved Precedence Operation Crossover，IPOX），机器分配向量采用多点保留交叉（Multi Points Crossover，MPX）。得到两个新个体v_{n1}和v_{n2}；

Step5：选出新个体中的较优者v_{nb}，与旧个体中的较差者v_{ow}进行比较。若v_{nb}优于v_{ow}，则舍弃v_{ow}，吸纳v_{nb}；否则，保留v_{ow}，不吸纳v_{nb}。

Step6：$t=t+1$。

IPOX 算子和 MPX 算子的具体执行过程如下，两个待交互个体为v_{o1}和v_{o2}：

（1）IPOX 算子

Step1：把所有工件随机划分到两个工件集合（J_1、J_2）中；

Step2：把J_1（J_2）所含工件拷贝到v_{n1}（v_{n2}）中，并保证与v_{o1}（v_{o2}）中的工序位点信息完全相同；

Step3：把J_2（J_1）所含工件插入到v_{n1}（v_{n2}）空缺位点中，并与v_{o2}（v_{o1}）原有的工序优先级顺序保持一致。

（2）MPX 算子

Step1：生成一个与机器分配向量等长的随机数向量，每一个随机数的取值都位于区间$(0,1)$内；

Step2：如果随机数大于 0.5，则把v_{o1}（v_{o2}）对应位点上的元素拷贝到v_{n1}（v_{n2}）相同的位点上；否则，把v_{o2}（v_{o1}）对应位点上的元素拷贝到v_{n1}（v_{n2}）相同的位点上。

IPOX 算子和 MPX 算子的操作示例见图 2-2、图 2-3。

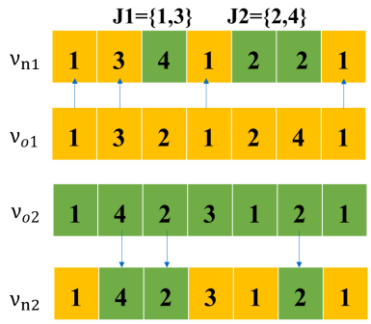


图 2-2 IPOX 算子操作实例图

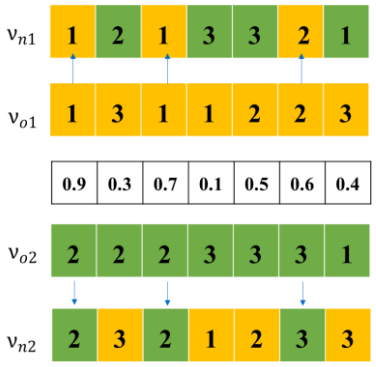


图 2-3 MPX 算子操作示例图

2.2.4 种群“个体认知”阶段

种群“个体认知”阶段是算法的局部搜索，也是单个个体的邻域搜索。

局部搜索是通过不同的邻域结构实现的。邻域结构是用来指导一个已知解移动到另一个可行解的方法论，因此邻域结构设计的好坏会直接影响到算法局部搜索的效率。现有研究表明：只有对关键路径进行扰动，才有可能减少柔性作业车间调度问题当前解的最大完工时间。考虑准备时间的 FJSP 是经典 FJSP 的拓展，也具有相同的性质。本章根据考虑准备时间的 FJSP 特征，参考基于 FJSP 关键路径的邻域结构，设计了基于考虑准备时间的 FJSP 关键路径的邻域结构。

当一个工件从一台机器转移到另一台机器上时，后面那台机器就需要一个固定且已知的时间完成加工前的准备工作。如何把准备时间和加工时间统一起来，是考虑准备时间的 FJSP 邻域结构设计的首要问题。

结论：当准备时间的结束时刻与工序的开始加工时刻相等时，时间利用率最高。

证明：当准备工作完成后，当前机器已被某个工件锁定，不能再加工其他工件，

只能等候该工件到达。如果延长这一等待时间，最大完工时间可能不会变差，但一定不会变好；如果缩短这一等待时间，最大完工时间可能不会变好，但一定不会变差。由此可知，当机器准备时间与工序加工时间无缝衔接时，时间利用率最高。

关键路径的变化是改变最大完工时间的关键，也是构造邻域结构的重要组成部分。众所周知，经典 FJSP 的关键路径是：从零时刻到终止时刻，首尾依次相接的工序所组成的路径。但经典 FJSP 中关键路径的定义并不适用于考虑准备时间的 FJSP，下面给出考虑准备时间的 FJSP 关键路径的定义：

从零时刻到终止时刻，顺次满足公式（2-13）的工序所组成的路径。

$$s_{O_{ij}} - \left(s_{KPJ[O_{ij}]} + p_{KPJ[O_{ij}]k[KPJ[O_{ij}]]} \right) \le t_{k'k} \tag{2-13}$$

其中 $KPJ[O_{ij}]$ 表示工序 O_{ij} 所在关键路径上的紧前工序，$s_{KPJ[O_{ij}]} + p_{KPJ[O_{ij}]k[KPJ[O_{ij}]]}$ 表示的是 $KPJ[O_{ij}]$ 的完工时间（开始时间+加工时间）。

换言之，如果 O_{ij} 为关键工序，则它的开始加工时间至多比 $KPJ[O_{ij}]$ 的完工时间晚一个准备时间。

本章以考虑准备时间的 FJSP 特征为向导，设计出两种基于考虑准备时间 FJSP 关键路径的邻域结构：换机器的邻域结构（Neighborhood of Changing Machine，N-CM），同机器的邻域结构（Neighborhood of The Same Machine，N-SM）。

（1）换机器的邻域结构

基于 N-CM 的邻域移动，可以归纳为以下步骤：

Step1：从当前解的某条关键路径中随机选取一道工序，判断该工序是否含有两个以上（含两个）的可选机器。若有，则进行 Step2 至 Step3；若没有，则重新选取工序，直至找到满足条件的工序 O_{ij}；

Step2：把工序 O_{ij} 从它当前机器的加工序列中删除；

Step3：从工序 O_{ij} 的其他可选机器中随机选择一个，把 O_{ij} 插入到所选机器的加工序列中，并保证插入后的解依然为可行解。

可行插入

一个可行的插入，就是要保证插入后的加工路线不存在闭合回路。为了给可行插入一个明确的界定，首先定义两个工序集合。

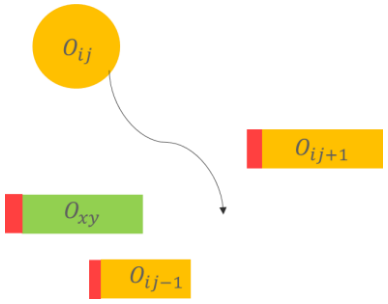
L_k： $L_k = \left(O_{xy} \in Q_k \mid s_{xy} < s_{ij+1}\right)$

R_k： $R_k = \left(O_{xy} \in Q_k \mid s_{xy} + p_{xyk} > s_{ij-1} + p_{ij-1k'}\right)$

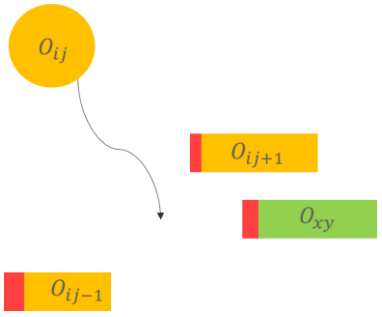
其中Q_k表示在机器k上加工的所有工序的集合。对于L_k里的工序而言，不存在从O_{ij}到O_{xy}的加工路径；对于R_k里的工序而言，不存在从O_{xy}到O_{ij}的加工路径。

定理 1：把工序O_{ij}插入到$L_k - R_k$所含工序之后和$R_k - L_k$所含工序之前，都是可行的插入[54]。

定理 1 的可视化示例见图 2-4。



（a）插入$L_k - R_k$所含工序之后



（b）插入$R_k - L_k$所含工序之前

图 2-4 可行插入的可视化示例图

最优插入

在所有可行插入中，可以获得最好解的插入就是最优插入。

Case1：当$L_k \cap R_k \neq \varnothing$时，把$O_{ij}$插入到$L_k - R_k$所含工序之后和$R_k - L_k$所含工序之前，会有一个最优插入。

Case2：当$L_k \cap R_k = \varnothing$时，把$O_{ij}$插入到$L_k - R_k$所含工序之后和$R_k - L_k$所含工序之前，都是最优插入。

（2）同机器的邻域结构

N-CM 在进行邻域移动之前，必须事先确定L_k和R_k，通过二进制搜索算法获得L_k和R_k的时间复杂度为$O(\log|Q_k|)$。为了提高算法的运行速度、降低时间开销，对于同机器的邻域移动，本章采用一种更加简单有效的方法。

基于 N-SM 的邻域移动也是在关键路径上进行的。在介绍 N-SM 之前，首先说明关键块的含义。关键块指的是由同一台机器上的关键工序组成的最大序列。

基于 N-SM 的邻域移动的可视化示例见图 2-5。

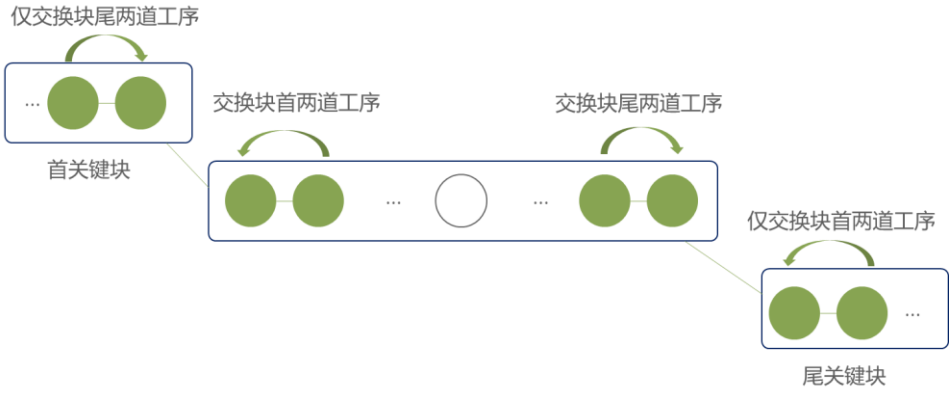


图 2-5 基于 N-SM 的邻域移动

基于 N-SM 的邻域移动，就是交换同一关键块的块首两道工序以及块尾两道工序。但是在交换过程中，需要满足下列条件：

1）首关键块只交换块尾两道工序，尾关键块只交换块首两道工序；

2）如果把工序O_{ij}移动到工序$O_{i'j'}$之后，则必须保证O_{ij+1}在$O_{i'j'}$之后；

3）如果把工序O_{ij}移动到工序$O_{i'j'}$之前，则必须保证O_{ij-1}在$O_{i'j'}$之前；

4）同一个工件的工序之间不能交换。

种群“个体认知”阶段的具体执行步骤如下：

Step1：找到当前解的一条关键路径；

Step2：执行基于 N-SM 的邻域移动。如果得到的邻域解比当前解差，转到 Step3；否则，转到 Step4；

Step3：执行基于 N-CM 的邻域移动。如果得到的邻域解比当前解差，保留当前解不变；否则，转到 Step4；

Step4：用得到的邻域解替换当前解。

2.2.5 自适应禁忌搜索

循环搜索会浪费大量的计算资源，禁忌的目的正是为了避免重复访问同一个解。禁忌的对象是存放在禁忌表中的各元素，一般会选用完整的解向量、目标函数值或解的某些特征属性。但在柔性作业车间调度问题中，一般不会选择完整的解向量作为禁忌对象，因为在判断当前候选解是否被禁忌的过程中，会消耗大量的计算资源，降低算法的求解效率。本文把移动属性$\left(O_{ij},k\right)$作为禁忌对象，其中O_{ij}为待移动工序，k表示在移动该工序之前加工此工序的机器。

禁忌表长度指的是禁忌表中可以容纳的禁忌元素的最大数量，反映了禁忌表里的元素存放时间的长短，禁忌表长度对搜索过程影响较大，需要进行合理设置。如果禁忌表长度太短，会导致循环搜索；如果禁忌表长度太长，会出现约束过多的问题。对于调度问题而言，很难找到一个既可以避免循环搜索而又不会导致约束过多的禁忌表长度，但可以通过动态设置禁忌表长度来解决这个问题。禁忌表长度可以依照公式（2-14）计算[55]。

$$TM\left(O_{ij},k\right)=|P|+|M_{O_{ij}}| \tag{2-14}$$

其中$|P|$为当前关键路径中所含关键工序的个数，$|M_{O_{ij}}|$为工序O_{ij}的可选加工机器数。

禁忌搜索具有很强的局部寻优能力，但时间消耗相对较大。算法早期搜索到的解，质量相对较差，通过禁忌搜索开发当前解的邻域，改进的余地是有限的，如果禁忌搜索的迭代次数比较大，会造成计算资源的浪费；而算法后期搜索到的解，质量相对较好，最优解或近似最优解很大可能就在当前解的邻域里，如果禁忌搜索的迭代次数比较小，会导致算法的收敛性较差。为了满足算法不同阶段的矛盾需求，设计了公式（2-15），使禁忌搜索的最大迭代次数能够自适应调整。

$$iteration_{\max}=\left\lfloor\left(1+\left(\frac{Eval_{now}}{Eval_{\max}}\right)^{ts}\right)\times\left(\frac{\sum_{i=1}^{n}l_i}{m}+\frac{\sum_{i=1}^{n}\sum_{j=1}^{l_i}m_{ij}}{\sum_{i=1}^{n}l_i}\right)\right\rfloor \tag{2-15}$$

$\lfloor x\rfloor$ 表示对 x 向下取整。其中 $Eval_{now}$ 表示解的当前评估次数，$Eval_{\max}$ 表示解的最大评估次数，ts 为调控因子，在 2.4 节中会进行校正，其余符号说明详见 2.2 节。

自适应禁忌搜索的具体执行过程如下：

Step1：选择当前解作为禁忌初始解，把它视作种子解和历史最好解，清空禁忌列表。根据公式（2-15）计算最大迭代次数 $iteration_{\max}$，作为终止条件；

Step2：令 $t=1$，重复 Step3 到 Step7，直至 $t>iteration_{\max}$；

Step3：使用 2.3.4 节中所述方法对种子解进行多次邻域搜索，得到种子解的一系列邻域解；

Step4：选择一个最好且未被禁忌的邻域解作为下一次迭代的种子解，并把它放入禁忌列表中；如果所有的邻域解都被禁忌，则特赦其中最好的一个作为下一次迭代的种子解，并把它重新放入禁忌表中；

Step5：如果禁忌表已满，则把最先放入禁忌表中的对象从禁忌表中移除；

Step6：如果新的种子解比历史最好解要好，则用它更新历史最好解；

Step7：$t=t+1$；

Step8：用历史最好解更新当前解。

2.2.6 算法流程

本章改进粒子群优化算法的流程图如图 2-6 所示。

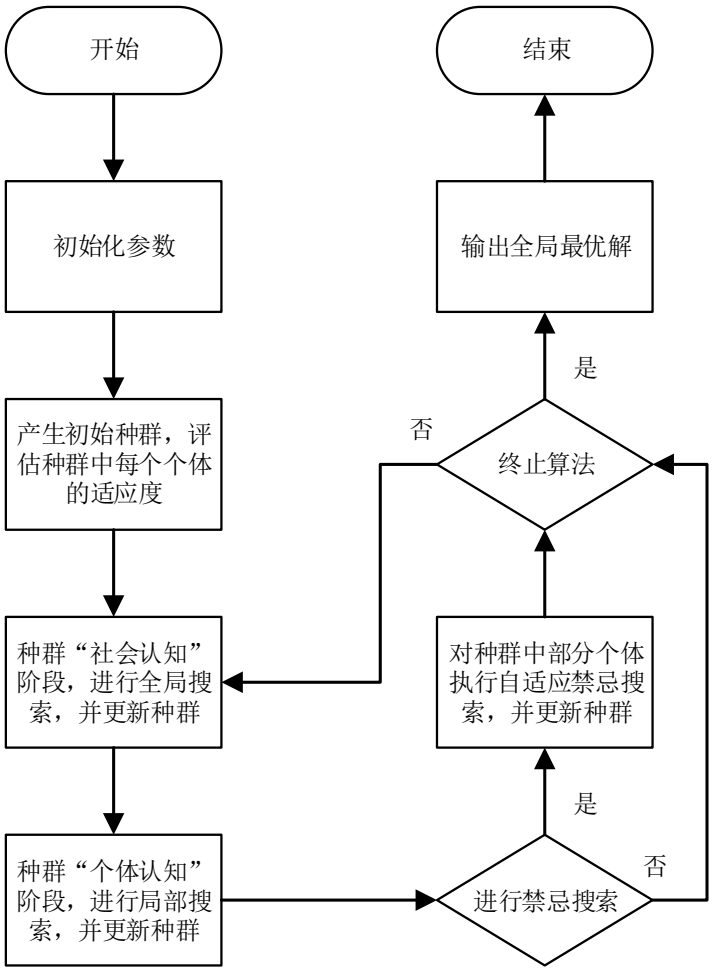


图 2-6 改进粒子群优化算法流程图

改进粒子群优化算法的具体步骤如下：

Step1（初始化）：初始化 IPSO 算法参数，按照 2.3.2 节中的方法生成初始种群，算法相关参数包含种群规模、解的最大评估次数等；

Step2（种群“社会认知”阶段）：按照 2.3.3 节中的方法完成种群的全局搜索；

Step3（种群“个体认知”阶段）：按照 2.3.4 节中的方法完成种群内单个个体的邻域搜索；

Step4（自适应禁忌搜索）：如果全局最优解在连续 Q 次迭代中都没有被更新，则

针对种群中前 10%较好的个体，按照 2.3.5 节中所述方法进行禁忌搜索；

Step5（算法终止）：重复 Step2 到 Step4 直至达到解的最大评估次数，输出全局最优解。

2.3 数值实验与结果分析

2.3.1 数值实验设计

本章中使用的基准测试算例数据来源于 Wu 等[9]的相关文献，该算例共包含 5 个大规模算例（为了和第三章区分，依次命名为 LS1-NAC、LS2-NAC、LS3-NAC、LS4-NAC、LS5-NAC）和 5 个宽范围算例（为了和第三章区分，依次命名为 WR1-NAC、WR2-NAC、WR3-NAC、WR4-NAC、WR5-NAC）。大规模算例包含 100 个工件、36 台机器，宽范围算例包含 60 个工件、36 台机器。其中宽范围算例的加工时间和准备时间的极差都大于大规模算例。

针对该算例集，将本章所提 IPSO 算法与目前在该问题求解上表现突出的两个算法进行对比。这两个算法分别是：Zheng 等[1]提出的 nFOA 算法和 Sang 等[30]提出的 CCIWO 算法。

本章中的算法在 MATLAB R2016a 上采用 MATLAB 语言编程实现，运行环境为 Core 4C+6G 1.9GHz CPU、4.00GB 内存、Windows 10 操作系统。

对比算法的参数设置与文献中保持一致，nFOA 和 CCIWO 都以解的最大评估次数作为算法的终止迭代条件。为保障对比实验的公平性，参照文献中其他算法的参数设置，IPSO 的种群规模设为 60，终止条件是解的最大评估次数达到 10000 次。IPSO 的特有关键参数通过对照实验进行确定。

为了在正交实验中综合评估算法的性能，引入了相对百分比偏差（Relative Percentage Deviation，RPD）和平均相对百分比偏差（Average Relative Percentage Deviation，ARPD）。RPD、$ARPD$ 可分别依照公式（2-16）、（2-17）进行计算。

$$RPD=\frac{1}{RT}\sum_{rt=1}^{RT}\frac{C_{\max}^{rt}-C_{\max}^{*}}{C_{\max}^{*}}\times 100 \tag{2-16}$$

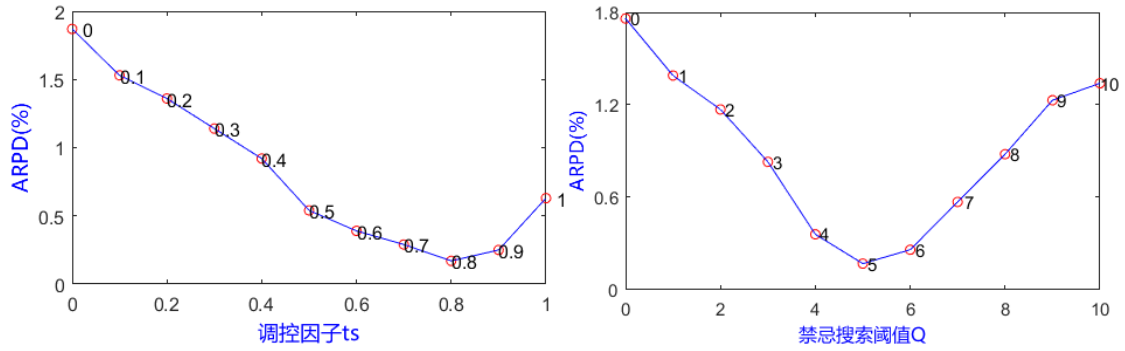
RPD 的单位是%，RT 表示算法针对该算例的独立运行总次数，$C_{\max}^{*}$ 表示目前已

知该算例可以得到的最优值，$C_{\max}^{rt}$ 为算法第 rt 次运行求得的该算例最优值。

$$ARPD=\frac{1}{BM}\sum_{bm=1}^{BM}RPD_{bm} \tag{2-17}$$

$ARPD$ 的单位是%，RPD_{bm} 表示求解第 bm 个算例的相对百分比偏差，BM 表示待求解算例的总数。$ARPD$ 越小，说明该算法的综合性能越好。

禁忌搜索的最大迭代次数调控因子 ts 依次取值为 $\{0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1\}$，IPSO 算法在不同的 ts 下针对所有算例独立运行 10 次，得到的 $ARPD$ 变化曲线图如图 2-6（a）所示。同理，可得到 IPSO 算法在不同禁忌搜索阈值 Q 下的 $ARPD$ 变化曲线图，如图 2-6（b）所示。从图 2-6 可以得知，当 $ts=0.8$、Q=5 时，IPSO 算法的综合性能达到最优。



（a）不同 ts 下的 $ARPD$ 变化曲线图　　（b）不同 Q 下的 $ARPD$ 变化曲线图

图 2-6 IPSO 关键参数校正曲线图

2.3.2 实验结果与分析

为了降低随机因素对算法求解结果产生的影响，保障对比实验的公平公正，各算法分别针对每个算例独立运行 30 次。采用以下 4 个评价指标对算法的性能进行评估。

Best：算法在针对某算例独立运行 30 次的过程中找到的最优目标函数值，由于本章选择的优化目标是最大完工时间，所以这里的最优值指的就是最小的最大完工时间，因此 *Best* 值越小，说明算法的收敛性越好。

Mean：均值，*Mean* 是一个综合性能指标，*Mean* 值越小，说明算法的收敛性和稳定性越好，*Mean* 可通过公式（2-18）进行计算。

$$Mean = \frac{\sum_{rt=1}^{RT} C_{\max}^{rt}}{RT} \tag{2-18}$$

式中 RT 和 $C_{\max}^{rt}$ 的具体含义已在公式（2-16）进行了说明。

SD：标准差，SD 值越小，说明算法越稳定，SD 可以通过公式（2-19）进行计算。

$$SD = \sqrt{\frac{1}{RT}\sum_{rt=1}^{RT}(C_{\max}^{rt} - Mean)^2} \tag{2-19}$$

$t_{CPU}(s)$：算法针对某算例独立运行 1 次平均需要花费的 CPU 时间，$t_{CPU}(s)$越小，说明算法的时间复杂度越小。

3 个算法分别针对每个算例独立运行 30 次后，把得到的 $Best$、$Mean$、SD 和 $t_{CPU}(s)$进行对比，对比结果如表 2-1 所示。

表 2-1 不同算法求解考虑准备时间的 FJSP 的对比结果

算法	nFOA				CCIWO				IPSO			
算例	*Best*	*Mean*	*SD*	t_{CPU}(s)	*Best*	*Mean*	*SD*	t_{CPU}(s)	*Best*	*Mean*	*SD*	t_{CPU}(s)
LS1-NAC	58	62.1	3.48	4.03	54	56.2	2.24	3.04	**52**	**54.4**	**2.14**	2.96
LS2-NAC	74	77.5	2.97	3.84	71	72.1	1.21	2.97	**67**	**68.7**	**1.12**	2.83
LS3-NAC	66	69.4	3.06	3.98	64	65.7	**1.84**	2.74	**61**	**63.2**	1.96	2.25
LS4-NAC	79	80.9	2.17	3.62	73	76.3	**2.16**	2.96	**71**	**73.8**	2.35	2.89
LS5-NAC	75	78.8	3.87	3.71	68	70.7	2.24	3.12	**65**	**66.7**	**1.09**	2.06
WR1-NAC	134	137.2	3.12	3.27	128	131.2	2.89	2.74	**126**	**128.9**	**2.47**	2.95
WR2-NAC	95	98.4	3.56	2.78	91	93.8	3.01	2.16	**89**	**91.4**	**2.26**	1.96
WR3-NAC	114	116.5	3.37	3.11	112	113.2	**1.12**	2.11	**110**	**111.5**	1.16	2.17
WR4-NAC	103	105.7	3.44	2.56	**102**	103.5	1.23	1.78	**102**	**103.1**	**0.91**	1.42
WR5-NAC	97	99.3	2.89	3.19	**95**	**97.2**	**2.27**	1.65	**95**	97.3	2.33	1.78

注：*Best*、*Mean*、SD 指标下的加粗字体表示该算法得到了这个算例对应指标的最优值。

从表 2-1 可以看到，IPSO 算法得到了所有算例的最优 *Best* 值，说明 IPSO 算法在收敛性方面要优于其他算法，这主要得益于基于关键路径的邻域结构的设计，增强了 IPSO 的局部开发能力。对于 *Mean* 指标而言，CCIWO 算法得到了 WR5-NAC 算例的最优值，IPSO 算法得到了除 WR5-NAC 算例外所有算例的最优值，说明在同等客观条件下 IPSO 的寻优能力更强，这主要得益于自适应禁忌搜索算子的设计，有效避免了算法陷入循环搜索。对于 *SD* 指标而言，CCIWO 得到了 4 个算例的最优值，IPSO 得到了 6 个算例的最优值，说明在算法鲁棒性方面，IPSO 也具有一定的优势。对于 $t_{CPU}(s)$ 指标而言，IPSO 和 CCIWO 没有明显差距，都要好于 nFOA，说明 IPSO 在运算速度方面，虽然没有显著优势，但仍处于中等偏上水平。究其原因在于，IPSO 算法使用的是基于关键路径信息的邻域结构，它的时间复杂度较高，相对耗时。整体而言，在考虑准备时间的 FJSP 问题求解上，IPSO 有着显著的优势。

从统计学角度出发，为了更严格地说明 IPSO 算法相较于 nFOA、CCIWO 的优越性，本章借助各算法得到的 *ARPD* 值，做了两组单侧 t 检验，试验结果见表 2-2。其中 p 值表示 IPSO 算法不具有显著优势的概率，在本次检验中，如果 $p<0.05$，说明 IPSO 算法相较于对比算法具有显著的优势；如果 $p\geq 0.05$，则说明 IPSO 算法相较于对比算法不具有显著的优势。

表 2-2 IPSO 在 95%置信区间下的显著性检验结果（考虑准备时间的 FJSP）

对比算法	t 值	p 值	IPSO 是否具有显著优势
nFOA	-4.612	0.001	是
CCIWO	-3.116	0.005	是

在表 2-2 中，所有的 p 值均小于 0.05，说明本章所提 IPSO 算法相较于其它对比算法均具有显著优势，能够有效求解考虑准备时间的 FJSP。

使用本章提出的 IPSO 算法求解算例 LS1-NAC 和 WR2-NAC 得到的最优解的甘特图分别如图 2-7、图 2-8 所示。

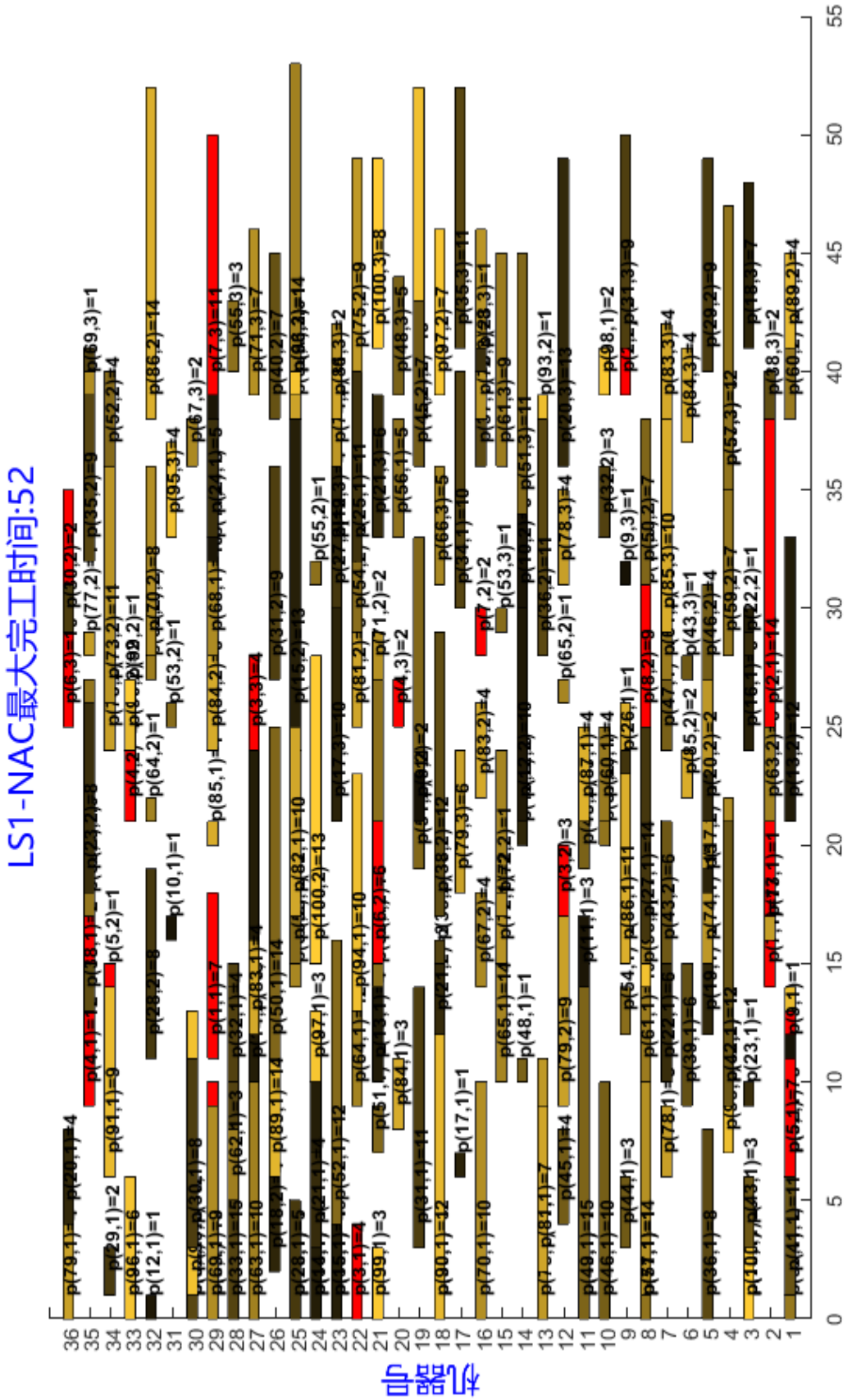


图 2-7 LS1-NAC 算例最优解的甘特图

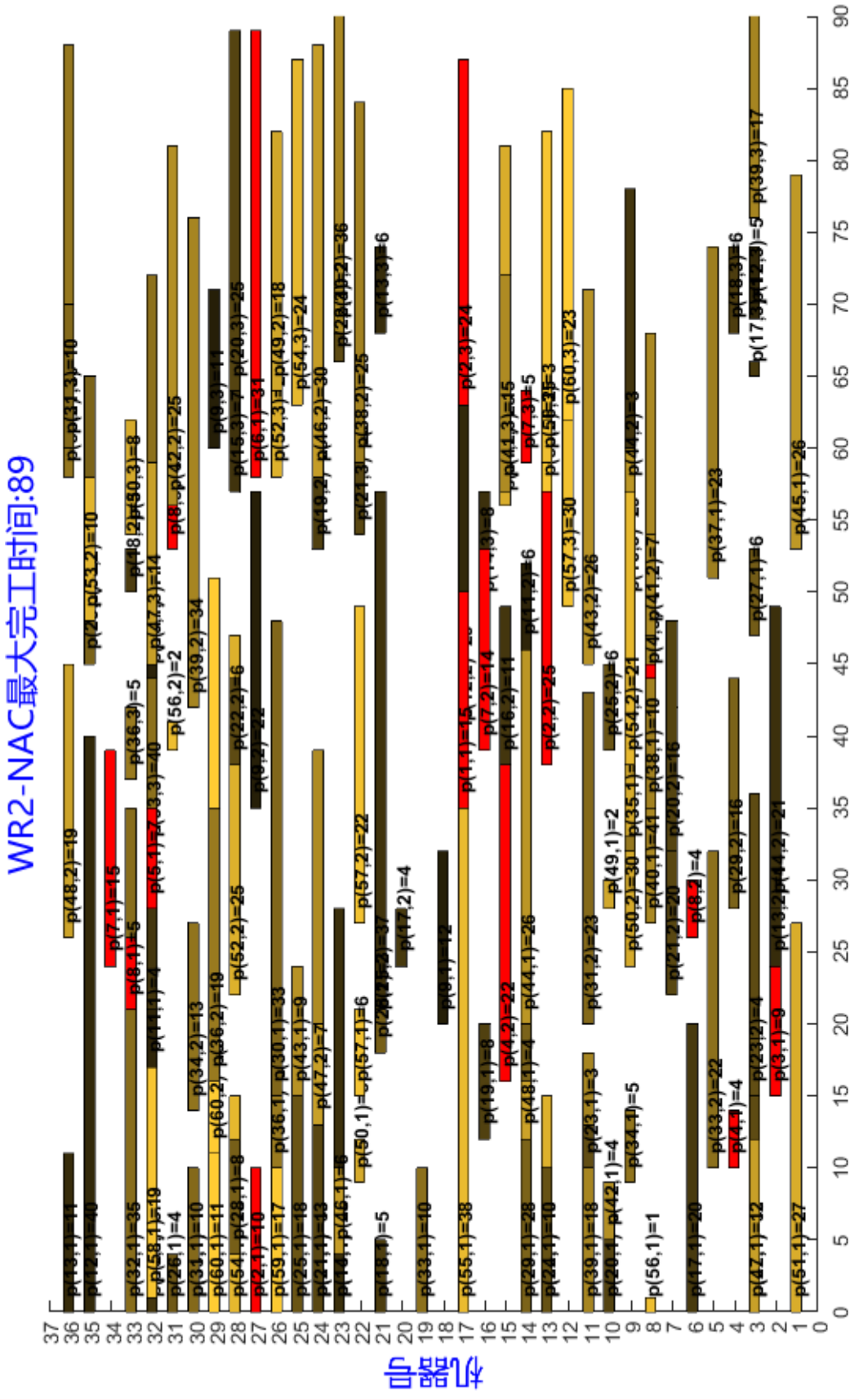


图 2-8 WR2-NAC 算例最优解的甘特图

2.4 本章小结

本章对考虑准备时间的 FJSP 进行了深入研究。首先，以最小化最大完工时间为优化目标，建立了考虑准备时间的 FJSP 数学模型。然后，结合考虑准备时间的 FJSP 特性，设计了 IPSO 算法进行求解。在 IPSO 算法中，提出了一种混合初始化策略，提高了初始种群的质量；充分挖掘考虑准备时间的 FJSP 的领域知识后，设计了两种基于关键路径的邻域结构，提高了算法的局部开发能力；设计了自适应禁忌搜索算子，避免了算法陷入循环搜索。最后，通过数值实验对 IPSO 的关键参数进行了校正，并验证了所提 IPSO 算法相较于其它对比算法的优越性。

3 考虑准备时间的 MRC-FJSP 方法

3.1 问题描述与数学模型建立

3.1.1 问题描述

考虑准备时间的 MRC-FJSP 问题描述在 2.2.1 节的基础上，增加了以下内容：

（1）在车间内有 h 类辅助资源(R^1， R^2，…， R^h)，每类辅助资源的数量是预先确定的；

（2）同一类辅助资源之间没有差异；

（3）每台机器都需要在多种辅助资源的配合下才能完成生产，因此，只有加工机器及其所需的全部辅助资源同时可得，一道工序的加工才能启动；

（4）每台机器配置的辅助资源类型是固定且已知的；

（5）同一个辅助资源单件在同一时刻只能服务一台机器；

（6）所有辅助资源在零时刻都可以被使用；

（7）不考虑辅助资源的消耗。

考虑准备时间的MRC-FJSP是一个相对新颖的研究对象，为了更直观地描述它，本章给出一个简单实例。有 4 个工件、3 台机器以及 3 类辅助资源（每类辅助资源包含 2 个完全相同的单件）。表 3-1 反映的是每道工序在对应机器上的加工时间，表 3-2 反映的是每台机器所需配置的辅助资源类型，表 3-3 反映的是准备时间。

表 3-1 加工时间表

工序	机器		
	M_1	M_2	M_3
O_{11}	7	9	11
O_{12}	∞	5	7
O_{13}	6	6	∞
O_{21}	7	∞	7

O_{22}	∞	7	6
O_{31}	∞	11	6
O_{41}	11	∞	8

注：“∞”表示该工序不能在对应机器上加工。

表 3-2 各机器所需辅助资源类型表

机器	辅助资源类型		
	R^1	R^2	R^3
M_1	Y	Y	Y
M_2	Y	N	Y
M_3	N	Y	Y

注：“Y”表示该机器需要配置对应类辅助资源；“N”表示该机器不需要配置该类辅助资源。

表 3-3 准备时间表

机器	M_1	M_2	M_3
M_1	0	3	2
M_2	3	0	2
M_3	2	3	0

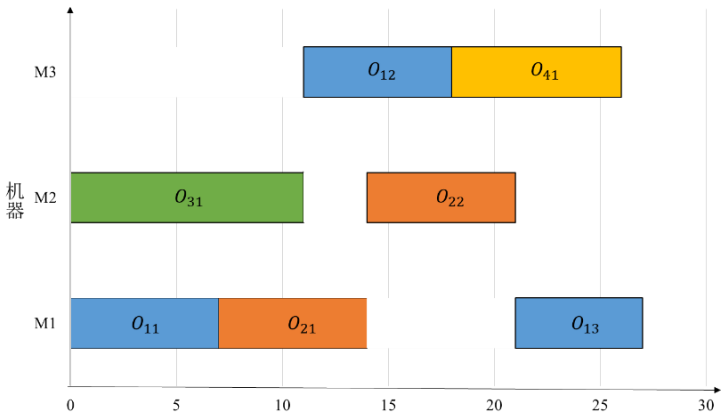


图 3-1 机器调度甘特图

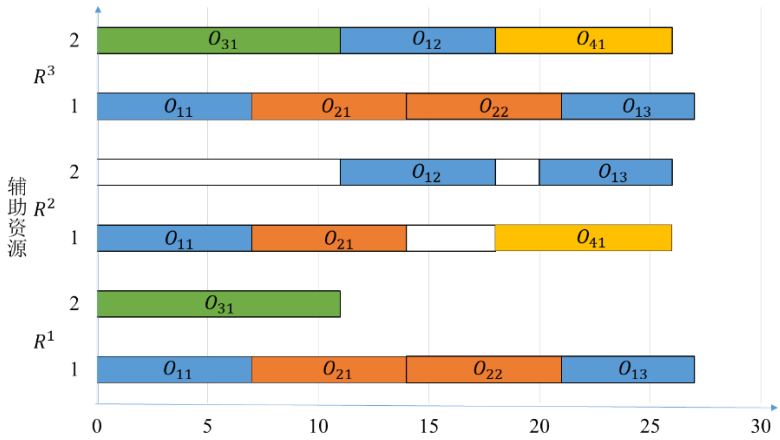


图 3-2 辅助资源调度甘特图

图 3-1、图 3-2 分别为某个解的加工机器调度甘特图、辅助资源调度甘特图。值得注意的是工序O_{11}的完工时间是时刻 7，此时机器M_3处于空闲状态，可以提前完成准备工作，但是它并没有立刻加工工序O_{12}。这是因为机器M_3需要在辅助资源R^2和R^3的配合下才能完成工序O_{12}，而辅助资源R^3都处于工作状态。直到时刻 11，机器M_2完成工序O_{31}，释放了占用的辅助资源R^3，工序O_{12}的加工才开始。

3.1.2 数学模型建立

为了方便后文描述，在 2.2.2 节的基础上增加以下符号声明：

h：辅助资源种类数；

R：总的辅助资源集合；

r：辅助资源种类序号，$r=1,2,3,\cdots,h$；

h_r：第r类辅助资源的总数，$r=1,2,3,\cdots,h$；

h_k：机器k配置的辅助资源总类数，$k=1,2,3,\cdots,m$；

g：第r类辅助资源的单件序号，$g=1,2,3,\cdots,h_r$；

在 2.2.2 节的基础上增加了以下决策变量：

Z_{ijkrg}：0-1 决策变量，如果机器k在第r类辅助资源的第g个单件的配合下加工

工序O_{ij}则取 1，否则取 0；

$B_{ijki'j'k'rg}$：0-1 决策变量，如果第r类辅助资源的第g个单件先配合机器k加工工序O_{ij}则取 1，如果先配合机器k'加工工序$O_{i'j'}$则取 0。

目标函数与 2.2.2 节公式（2-1）相同。

在 2.2.2 节公式（2-2）到公式（2-11）的基础上增加了以下约束条件：

$$\sum_{g=1}^{h_r} Z_{ijkrg} = 1 \tag{3-1}$$

式中：$i=1,2,3,\cdots,n$；$j=1,2,3,\cdots,l_i$；$k=1,2,3,\cdots,m_{ij}$；$r=1,2,3,\cdots,h_k$。

$$s_{ij} + Z_{ijkrg}\, p_{ijk} \le s_{i'j'} + L\left(1 - B_{ijki'j'k'rg}\right) \tag{3-2}$$

式中：$i,i=1,2,3,\cdots,n$；$j=1,2,3,\cdots,l_i$；$j'=1,2,3,\cdots,l_{i'}$；$k=1,2,3,\cdots,m_{ij}$；$k'=1,2,3,\cdots,m_{i'j'}$；$r=1,2,3,\cdots,h_k$；$g=1,2,3,\cdots,h_r$。

$$c_{ij} \le s_{i(j+1)} + L\left(1 - B_{i'j'k'i(i+1)k_{j+1}rg}\right) \tag{3-3}$$

式中：$i,i'=1,2,3,\cdots,n$；$j=1,2,3,\cdots,l_i-1$；$j'=1,2,3,\cdots,l_{i'}$；$k'=1,2,3,\cdots,m_{i'j'}$；$k_{j+1}=1,2,3,\cdots,m_{i(j+1)}$。

$$\sum_{i=1}^{n}\sum_{j=1}^{l_i}\sum_{k=1}^{m_{ij}} B_{ijki'j'k'rg} = Z_{i'j'k'rg} \tag{3-4}$$

式中：$i'=1,2,3,\cdots,n$；$j'=1,2,3,\cdots,l_{i'}$；$k'=1,2,3,\cdots,m_{i'j'}$；$r=1,2,3,\cdots,h$；$g=1,2,3,\cdots,h_r$。

$$\sum_{i'=1}^{n}\sum_{j'=1}^{l_{i'}} B_{ijki'j'k'rg} = Z_{ijkrg} \tag{3-5}$$

式中：$i=1,2,3,\cdots,n$；$j=1,2,3,\cdots,l_i$；$k=1,2,3,\cdots,m$；$r=1,2,3,\cdots,h$；$g=1,2,3,\cdots,h_r$。

公式（3-1）表示同一个辅助资源单件在同一时刻只能服务一台机器；公式（3-2）和公式（3-3）表示同一时刻同一台机器只能配置同一类辅助资源的一个单件；公式（3-4）和公式（3-5）表示每一类辅助资源都可以循环使用。

3.2 改进粒子群优化算法求解考虑准备时间的 MRC-FJSP

3.2.1 编码与解码

（1）编码

考虑准备时间的 MRC-FJSP 需要同时考虑工件排列、机器分配以及机器配置等三个子问题，通常情况下需要编写三个独立的混合整数向量。由于不考虑同一类辅助资源单件之间的差异性，单独编写机器配置向量会导致搜索空间过大、算法的寻优效率降低。因此，本章只编写工序排列向量和机器分配向量，各加工机器的辅助资源配置在解码阶段动态决策。工序排列向量和机器分配向量的编码方式与 2.3.1 节相同。

（2）解码

考虑准备时间的 MRC-FJSP 的解码过程同样是确定每道工序在所选机器上的开始加工时间。针对待求解问题的特征，本章设计了一种半主动解码与启发式规则相结合的新型解码方式，在保障最优解或近优解不丢失的前提下，对解空间进行了有效裁剪。对于考虑准备时间的 MRC-FJSP 而言，一道工序的开始加工要同时满足以下三个条件：

1）工件已到达所选机器；

2）所选机器处于空闲状态，且已完成加工前的准备工作；

3）所选机器需要配置的所有辅助资源均可以立刻投入使用。

基于上述约束条件，考虑准备时间的 MRC-FJSP 各工序的开始加工时间可以按照公式（3-6）计算得到。

$$s_{ij}=\max\left(s_{ij-1}+p_{ij-1k'},A_k+t_{k'k},A_{kr}\right) \tag{3-6}$$

式中 $s_{ij-1}+p_{ij-1k'}$ 和 $A_k+t_{k'k}$ 的具体含义已在公式（2-12）中进行了详细阐述，A_{kr} 表示机器 k 所需的辅助资源 r 的最早可用时间。

考虑到同一类辅助资源的单件之间没有差异这一特征，本章提出了用于处理机器配置子问题的启发式规则：在s_{ij}时刻，如果同一类辅助资源有多个单件可供选择的话，选择释放时间最接近s_{ij}的单件。因为这一选择不仅不会推迟当前工序的开始加工时刻，还为后续工序的尽早加工留出了余地。

3.2.2 种群初始化

初始解的质量对算法的求解效率和最终解的质量有着很大的影响。为了提高初始解的质量，本章依旧采用混合初始化策略，其中工序排列向量采用随机初始化方式，机器分配向量依照预设的概率随机从四种不同的初始化策略中选择一个。其中选择随机初始化的概率为 70%，选择 ECT 的概率为 10%，选择 SPT 的概率为 10%，选择 SSPT 的概率为 10%。

3.2.3 子种群“社会认知”阶段

子种群“社会认知”阶段是子种群内部的信息交互，是子种群内较差个体向较优个体靠近的过程。子种群“社会认知”阶段的具体操作步骤如下：

Step1：按照最大完工时间由小到大的顺序分别对每个子种群中的个体进行排序；

Step2：从子种群中分别抽取两个适应度值靠前个体v_{o1}（v'_{o1}）和两个适应度值靠后的个体v_{o2}（v'_{o2}）；

Step3：v_{o1}与v_{o2}交互，得到两个新个体v_{n1}、v_{n2}。交互时，工序排列向量采用 IPOX 算子，机器分配向量采用 MPX 算子。从v_{n1}和v_{n2}中随机抽取一个个体替换v_{o2}；

Step4：v'_{o1}与v'_{o2}交互，得到两个新个体v'_{n1}、v'_{n2}。交互方式同 Step3。从v'_{n1}和v'_{n2}中随机抽取一个个体替换v'_{o2}。

3.2.4 子种群“个体认知”阶段

“个体认知”阶段是个体的邻域搜索过程。如果在辅助资源约束下进行邻域移动，可行的移动方案会非常少，在寻找可行邻域解的过程中还会消耗大量的计算资

源，得不偿失。

考虑到上述问题，本章以 5 种邻域结构为基础设计了一种自适应邻域搜索算子。其中的 2 种邻域结构是 2.3.4 节中设计的 N-SM 和 N-CM，这 2 种邻域结构的使用需要分 3 步走。先使用 2.3.1 节中的解码方式对当前解进行初解码，得到一个中间过程解；再进行基于 N-SM 或 N-CM 的邻域移动；最后使用 3.3.1 节中所述解码方式完成各机器的辅助资源配置，得到一个可行邻域解。

值得注意的是，N-SM 和 N-CM 是基于考虑准备时间的 FJSP 关键路径的邻域结构，将其应用于考虑准备时间的 MRC-FJSP 时，使用效果会略差于第二章，这种差异性会随着机器辅助资源配置差异性的缩小而缩小。为了降低这种差异性对算法性能产生的影响，引入 3 种在相关文献中常用的邻域结构——交换、插入和更换机器，与 N-SM、N-CM 一起组成本章自适应邻域搜索算子的基础邻域结构。

交换邻域结构和插入邻域结构针对的是工序排列向量，更换机器邻域结构针对的是机器分配向量。下面对这三种邻域结构进行简要介绍。

（1）交换邻域结构

从工序排列向量中随机选择两个位点 site1 和 site2，互换 site1、site2 对应的元素。交换邻域结构示意图如图 3-3 所示。

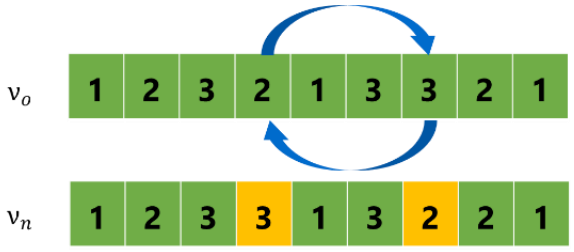


图 3-3 交换邻域结构示意图

（2）插入邻域结构

从工序排列向量中随机选择两个位点 site1 和 site2，把 site2 对应的元素移动到 site1 之前。插入邻域结构示意图如图 3-4 所示。

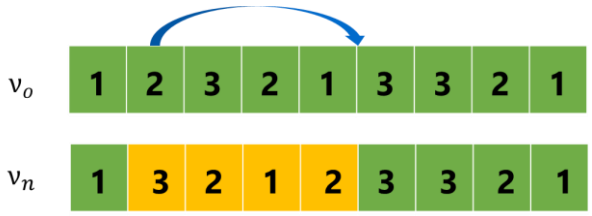


图 3-4 插入邻域结构示意图

（3）更换机器邻域结构

Step1：从机器分配向量中随机选择一个位点；

Step2：判断该位点对应的工序是否有 2 台或 2 台以上的候选加工机器，若是转到 Step3，否则回到 Step1；

Step3：更换对应工序的加工机器。

自适应邻域搜索算子的核心部分是邻域结构的自适应选择，算法依据自适应被选概率，从 5 种邻域结构中随机抽取一个使用。每个邻域结构的初始被选概率都相同，当前迭代中的被选概率按照公式（3-7）进行计算。

$$P(t+1)=\alpha\times P(t)+(1-\alpha)\times\frac{update\_num}{use\_num} \tag{3-7}$$

式中，$P(t+1)$表示该邻域结构在当前迭代中的被选概率，$P(t)$表示该邻域结构在上一次迭代中的被选概率。α 表示惯性因子，可以按照公式（3-8）进行计算，随着迭代的进行，α 从 0 逐渐趋近于 1，α 越大，说明当前迭代中的被选概率与上一次迭代中的被选概率越接近。use_num表示该邻域结构的使用次数，$update_num$表示该邻域结构更新解的次数。

$$\alpha=\left(\frac{Eval_{now}}{Eval_{\max}}\right)^{\gamma} \tag{3-8}$$

式中 $Eval_{now}$ 表示解的当前评估次数，$Eval_{\max}$ 表示解的最大评估次数，γ 根据经验取值为 0.85。

子种群“个体认知”阶段的具体操作步骤如下：

Step1：根据公式（3-7）计算各邻域结构的被选概率，依照邻域选择概率，随机

选取一个邻域结构；

Step2：使用所选邻域结构对当前个体执行邻域搜索，得到一个新个体；

Step3：如果新个体优于旧个体，则舍弃旧个体，把新个体加入子种群中；否则，保留旧个体，不吸纳新个体；

Step4：重复 Step1 到 Step3 一次。

3.2.5 子种群间信息交互

考虑准备时间的 MRC-FJSP 的解空间比较庞大，如果仅仅依靠单个种群进行迭代搜索，算法的求解效率会比较低下，且很容易陷入局部最优。多种群并行搜索机制可以满足单次迭代中搜索方向的多元化需求，保障了种群的多样性，在复杂优化问题中被广泛应用。目前文献中常用的多种群策略是离散重组和基于网络结构通信的多种群策略。

（1）离散重组

离散重组大体操作如下：在每次迭代完成后，打破原有子种群的边界束缚，将所有个体重新归为一个大种群；根据适应度值高低，将种群划分为若干个层次；在每个层次中随机选取一定数量的个体填入各子种群中；新组建的子种群再进行下一轮迭代搜索。

虽然离散重组起到了保持种群多样性的作用，但每次重组后，个体在原子种群内建立的联系被打破，需要在新子种群里重新进行磨合，这导致了算法后期收敛的精度不高。

（2）基于网络结构通信的多种群策略

在讲述这部分内容之前，需要对一些专业术语进行说明。

网络结构图：用以指导子种群交互的关系图。图中的节点代表各子种群，连接节点的边表示相连的两个子种群可以进行信息交互。

度数：从当前节点伸出的边数（或者说与当前节点相连的其他节点的个数）。

度分布：网络结构图中各节点度数的概率分布。

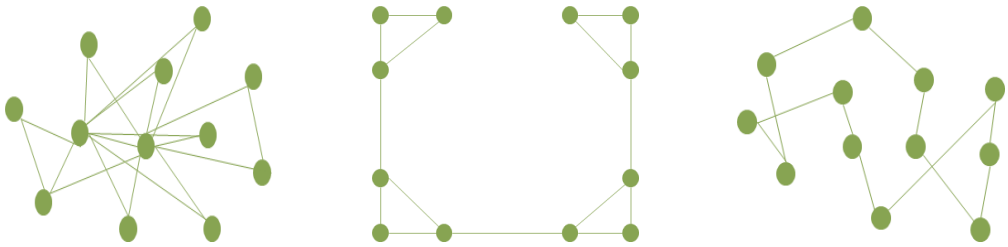
目前有三种应用比较广泛的网络结构图：无标度网络[56]、分块对角网络[56]和随

机网络[57]。

无标度网络指的是度分布满足幂律分布的网络结构，如图 3-5（a）所示。其中极少数节点占有较高的度数，而大多数节点拥有的度数很低。这些高度数节点就是绝对占优节点，能够引导整个算法的进化方向。无标度网络常用于凸优化问题，能够快速搜索到极值点。

分块对角网络把节点分成若干个集团，集团内部的节点之间频繁通信，分属不同集团的节点之间通信较少，如图 3-5（b）所示。影响分块对角网络性能的关键在于节点的分团。只有把性能相近的节点分在一个集团里，才能保证在算法求解过程中，各集团搜索方向的互异性，从而起到维护种群多样性的目的。分块对角网络常用于连续优化问题，可以事先通过聚类分析等手段对初始子种群进行分团。

随机网络是指度分布满足均匀分布的网络结构，如图 3-5（c）所示。随机网络中各节点之间的连通性通过连接概率 CP 进行调控。当 CP 取值较大时，各节点的度数普遍较高；当 CP 取值较小时，各节点的度数普遍较低。由于连接概率的可控性，随机网络适应能力较强，能够应对不同性质的复杂问题[58]。



（a）无标度网络结构图　（b）分块对角网络结构图　（c）随机网络结构图

图 3-5 网络结构图

通过分析比对，本章采用基于随机网络的多种群策略。算法早期，如果精英片段在子种群之间传播太快，会导致早熟收敛；算法后期，如果精英片段在子种群之间传播太慢，会导致收敛精度不高。为了满足算法不同阶段的矛盾需求，本章设计了自适应信息交互控制因子 P_c。P_c 用于控制子种群间信息交互的频繁程度，可按照公式（3-9）进行计算。

$$P_c = \left(\frac{Eval_{now}}{Eval_{\max}} \right)^R \tag{3-9}$$

式中 $Eval_{now}$ 和 $Eval_{\max}$ 的具体含义已在公式（3-8）作了详细阐述，R 为传播频率，在 3.4 节会进行标定。

子种群间信息交互的具体操作步骤如下：

Step1：生成一个位于区间 $(0,1)$ 内的随机数。如果该随机数小于 P_c，则进行 Step2 到 Step5；否则，本轮迭代不进行子种群间信息交互；

Step2：根据连接概率 CP 获取用以指导子种群交互的关系矩阵 A。A 是一个二维对称矩阵，里面存储着若干个 0-1 变量。当变量取值为 1 时，表示该变量横纵坐标编号对应的 2 个子种群存在交互关系；当变量取值为 0 时，表示该变量横纵坐标编号对应的 2 个子种群不存在交互关系；

Step3：按照关系矩阵 A，从两个待交互的子种群中各自抽取一个适应度值靠前的个体 v_{o1}（ω_{o1}）和一个适应度值居中的个体 v_{o2}（ω_{o2}）；

Step4：v_{o1} 与 ω_{o2} 交叉，得到两个新个体 ω_{n1}、ω_{n2}，交叉方式同 3.3.3 节。比较 ω_{n1}、ω_{n2} 和 ω_{o2} 的 $C_{\max}$，留下三者中 $C_{\max}$ 最小的个体，其余两个个体淘汰；

Step5：v_{o2} 与 ω_{o1} 交叉，得到两个新个体 v_{n1}、v_{n2}，交叉方式同 3.3.3 节。比较 v_{n1}、v_{n2} 和 v_{o2} 的 $C_{\max}$，留下三者中 $C_{\max}$ 最小的个体，其余两个个体淘汰。

3.2.6 子种群重新初始化

当某个子种群连续 T 次迭代都未提供比之前更好的解时，就重新初始化该子种群中适应度值靠后的三个个体，初始化方式与 3.3.2 节相同。

3.2.7 算法流程

本章 IPSO 算法的流程图如图 3-6 所示。

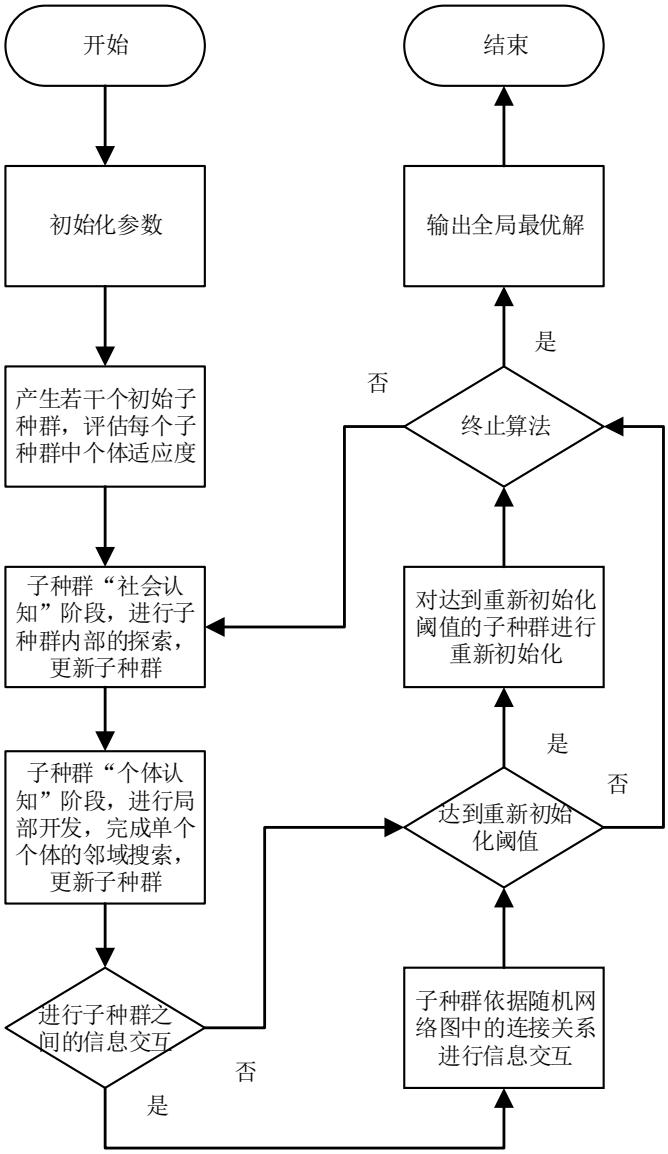


图 3-6 IPSO 算法流程图

IPSO 算法的具体步骤如下：

Step1（初始化）：初始化 IPSO 算法参数，按照 3.3.2 节中的方法生成若干个初始子种群，算法相关参数包含种群规模、子种群数量以及算法中解的最大评估次数等；

Step2（子种群“社会认知”阶段）：按照 3.3.3 节中的方法完成子种群内部的信息交互；

Step3（子种群“个体认知”阶段）：按照 3.3.4 节中的方法完成子种群内单个个

体的邻域搜索；

Step4（子种群间信息交互）：按照 3.3.5 节中的方法完成子种群之间的信息交互；

Step6（子种群重新初始化）：当一个子种群连续 T 次迭代都未提供比之前更好的解时，按照 3.3.6 节中的方法重新初始化该子种群；

Step7（算法终止）：重复 Step2 到 Step6 直至达到解的最大评估次数，输出全局最优解。

3.3 数值实验与结果分析

3.3.1 数值实验设计

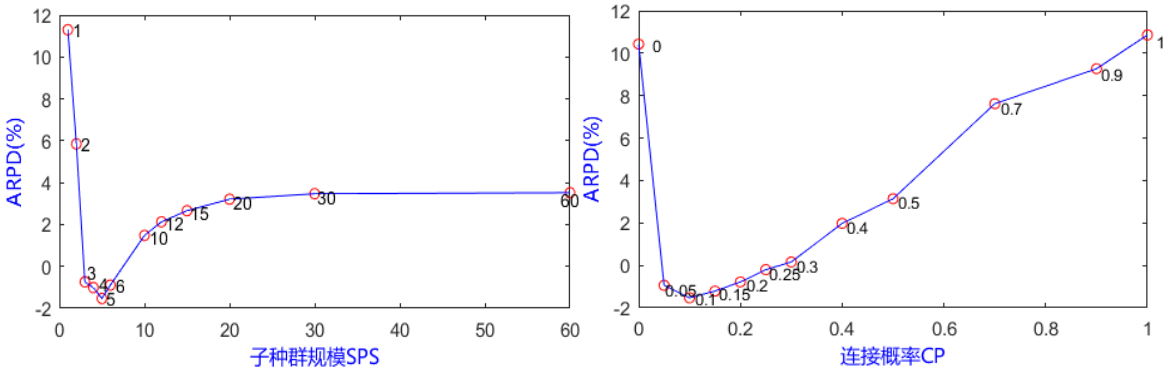
本章使用 Wu 等[9]提出的标准测试算例集，算例集的相关数据可以在决策分析实验室的门户网站（https://dalab.ie.nthu.edu.tw/newsen_content.php?id=0）上找到。该算例共包含 5 个大规模算例（即 LS1、LS2、LS3、LS4、LS5）和 5 个宽范围算例（即 WR1、WR2、WR3、WR4、WR5）。大规模算例包含 100 个工件、36 台机器，每台机器需要配齐 3 类辅助资源才能正常运行；宽范围算例包含 60 个工件、36 台机器，每台机器同样也需要配齐 3 类辅助资源才能正常运行。其中宽范围算例的加工时间和准备时间的极差都大于大规模算例。

针对 Wu 等所提的标准算例测试集，将 IPSO 算法与目前求解该问题最先进的几种算法作对比。这些算法有：Zheng 等[1]提出的 nFOA 算法，Wang 等[28]提出 HEDA 算法，Wang 等[29]提出 KMEA 算法，Cao 等[31-32]提出的 CSRS 算法和 Sang 等[30]提出的 CCIWO 算法。

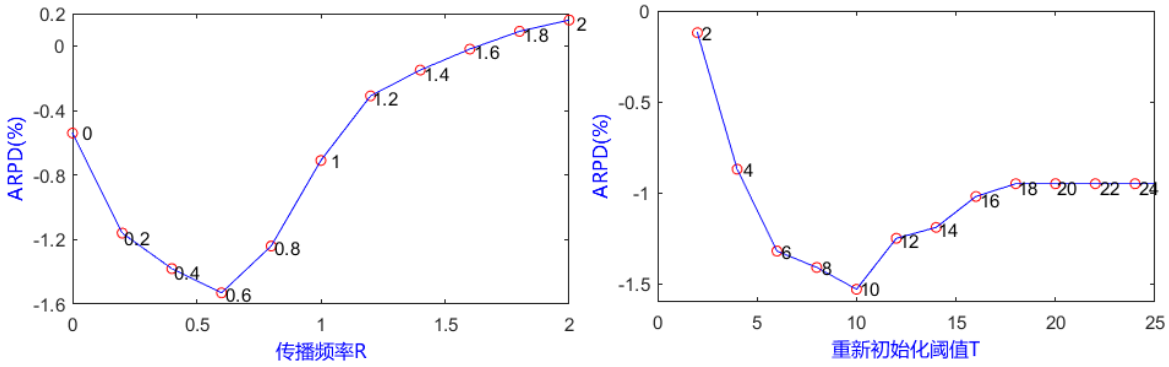
本章所提的 IPSO 算法在 MATLAB R2016a 上采用 MATLAB 语言编程实现，运行环境为 Core 4C+6G 1.9GHz CPU、4.00GB 内存、Windows 10 操作系统。

nFOA、HEDA、KMEA 和 CCIWO 等四种算法均以解的最大评估次数作为算法的终止迭代条件，为保障对比实验的公平性，参照文献中其他算法的参数设置，IPSO 的种群规模设为 60，终止条件是解的最大评估次数达到 10000 次。下面通过对照实验对 IPSO 的特有关键参数进行确定，得到的 *ARPD* 变化曲线图如图 3-7 所示。与 2.4.1

节中不同的是，在计算 *ARPD* 时，$C^{*}_{\max}$ 被定义为目前文献中出现的该算例的最优值。



（a）不同 *SPS* 下的 *ARPD* 变化曲线图　　（b）不同 *CP* 下的 *ARPD* 变化曲线图



（c）不同 *R* 下的 *ARPD* 变化曲线图　　（d）不同 *T* 下的 *ARPD* 变化曲线图

图 3-7 IPSO 关键参数校正曲线图

子种群规模 *SPS* 依次取值为 $\{1,2,3,4,5,6,10,12,15,20,30,60\}$，IPSO 算法在不同的 *SPS* 下针对所有算例独立运行 10 次，得到的 *ARPD* 变化曲线图如图 3-7（a）所示。同理，可得到 IPSO 算法在不同的连接概率 *CP*、传播频率 *R* 和重新初始化阈值 *T* 下的 *ARPD* 变化曲线图，分别如图 3-7（b）、图 3-7（c）和图 3-7（d）所示。从图 3-7 可以得知，当 $SPS=5$、$CP=0.1$、$R=0.6$、$T=10$ 时，IPSO 算法的综合性能达到最优。

为了验证本章设计的基于随机网络的多种群策略的有效性，令 IPSO 算法在不同的多种群策略下针对所有算例独立运行 10 次，得到的 *ARPD* 散点图如图 3-8 所示。从图 3-8 中可以看出，基于随机网络的多种群策略可以有效提升算法的寻优能力和鲁棒性。

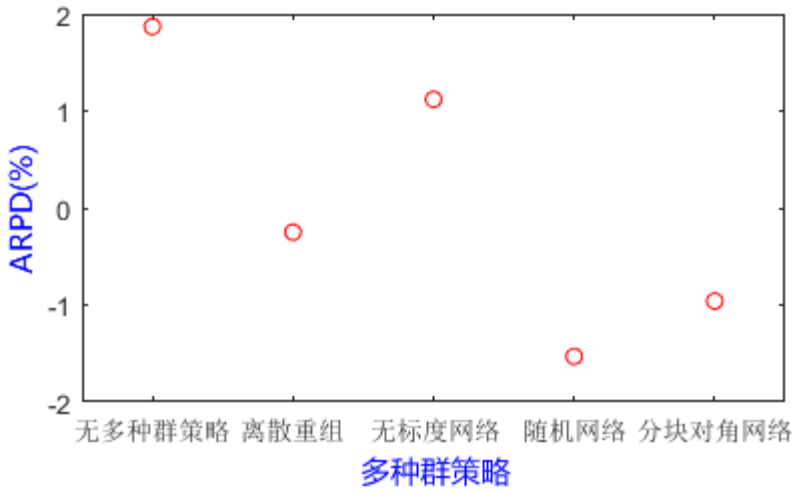


图 3-8 不同多种群策略下的 *ARPD* 散点图

3.3.2 实验结果与分析

为了尽可能降低随机因素对算法求解结果产生的影响，保障对比实验的公平公正，与其他对比算法一样，IPSO 算法也分别针对每个算例独立运行 30 次。本章从 *Best*、*Mean*、*SD* 和 $t_{CPU}\left(s\right)$ 等 4 个方面对算法的性能进行全面对比。对比结果如表 3-4 所示，nFOA、HEDA、KMEA、CSRS 和 CCIWO 的求解结果分别引自文献[1]、文献[28]、文献[29]、文献[32]和文献[30]。

从表 3-4 中可以看出，IPSO 得到了所有算例 *Best* 指标和 *Mean* 指标的最优值，说明 IPSO 的寻优能力要强于其它算法，这主要得益于自适应邻域搜索算子的设计，增强了算法的局部开发能力。对于 *SD* 指标而言，HEDA、KMEA、CCIWO 分别得到了 1 个算例的最优值，CSRS 得到了 2 个算例的最优值，IPSO 则获得了 6 个算例的最优值。说明在算法鲁棒性方面，IPSO 也具有显著的优势，这主要得益于基于随机网络的多种群策略的迁移应用，提升了算法的全局勘探能力。在 $t_{CPU}\left(s\right)$ 方面，IPSO 尚处于中等水平，这说明 IPSO 的求解速率还有待提升。整体而言，在考虑准备时间的 MRC-FJSP 求解上，IPSO 有着显著的优势。

表 3-4 不同算法求解考虑准备时间的 MRC-FJSP 的对比结果

算法	nFOA				HEDA				KMEA			
算例	*Best*	*Mean*	*SD*	t_{CPU}/s	*Best*	*Mean*	*SD*	t_{CPU}/s	*Best*	*Mean*	*SD*	t_{CPU}/s
LS1	104	108.4	3.51	4.13	102	103.7	**1.18**	4.19	98	101.4	1.56	1.56
LS2	113	117.5	3.14	3.71	111	112.9	1.37	4.06	107	113.3	2.33	2.33

LS3	102	107.8	2.89	3.63	101	104.4	1.85	3.89	99	103.2	**1.51**	1.51
LS4	119	123.6	2.10	4.00	118	121.3	1.85	4.15	117	121.1	1.80	1.80
LS5	107	115.6	4.00	3.76	112	114.3	1.55	1.55	107	112.5	1.65	1.65
WR1	245	250.4	3.57	3.46	245	251.1	3.74	3.74	229	238.0	3.83	3.83
WR2	191	204.8	3.29	3.00	202	207.2	2.99	2.99	186	198.0	4.34	4.34
WR3	220	226.8	3.43	3.10	220	225.3	3.95	3.95	210	217.8	2.74	2.74
WR4	**185**	189.8	4.04	2.73	**185**	191.2	4.33	4.33	**185**	185.6	1.74	1.74
WR5	222	227.3	3.13	3.20	222	229.2	3.92	3.92	204	214.8	3.66	3.66

算法	CSRS				CCIWO				IPSO			
算例	*Best*	*Mean*	*SD*	t_{CPU}/s	*Best*	*Mean*	*SD*	t_{CPU}/s	*Best*	*Mean*	*SD*	t_{CPU}/s
LS1	98	101.5	1.34	34.00	89	93.2	2.04	3.05	**83**	**85.3**	2.24	2.81
LS2	109	113.7	1.81	32.00	99	101.9	1.68	3.00	**94**	**95.6**	**1.08**	2.62
LS3	101	103.7	1.97	29.00	93	96.5	1.74	2.89	**87**	**89.5**	2.03	2.04
LS4	118	120.3	**1.17**	30.00	109	112.8	2.05	2.96	**103**	**105.9**	2.86	3.17
LS5	108	111.4	1.95	31.00	99	103.3	1.62	3.06	**97**	**98.6**	**1.12**	2.74
WR1	230	239.6	3.98	21.00	219	227.1	4.43	1.81	**213**	**216.8**	**2.31**	3.11
WR2	187	192.4	**2.18**	20.00	174	187.5	6.81	1.62	**170**	**172.9**	2.45	1.98
WR3	211	219.8	3.01	19.00	196	203.4	3.48	1.77	**195**	**196.3**	**1.27**	2.28
WR4	**185**	192.1	4.98	23.00	**185**	**185.0**	**0.00**	1.69	**185**	**185.0**	**0.00**	1.25
WR5	210	216.4	3.33	34.00	192	202.5	5.77	1.72	**187**	**190.2**	**2.74**	1.62

注：1. *Best*、*Mean*、*SD* 指标下加粗字体表示相应指标的最优值；

2.各算法的运行环境分别为：nFOA（Core i5 2.8GHz CPU）、HEDA（Core i5 2.3GHz CPU）、KMEA（Core i5 2.4GHz CPU）、CSRS（Core i5 2.7GHz CPU）、CCIWO（Core i7 3.6GHz CPU）、IPSO（Core 4C+6G 1.9GHz CPU）。

从统计学角度出发，为了更严格地说明 IPSO 算法相较于 nFOA、HEDA、KMEA、CSRS 和 CCIWO 的优越性，本章做了 5 组单侧 t 检验，试验结果见表 3-5。

表 3-5 IPSO 在 95%置信区间下的显著性检验结果（考虑准备时间的 MRC-FJSP）

对比算法	*t* 值	*p* 值	IPSO 是否具有显著优势
nFOA	-7.729	0.000	是
HEDA	-8.699	0.000	是
KMEA	-6.713	0.000	是
CSRS	-7.955	0.000	是
CCIWO	-5.077	0.000	是

在表 3-5 中，所有的 *p* 值均小于 0.05，说明本章所提 IPSO 算法相较于其它对比算法均具有显著优势，能够有效求解考虑准备时间的 MRC-FJSP。

使用本章提出的 IPSO 算法求解算例 LS5 和 WR5 得到的最优解的甘特图分别如图 3-9、图 3-10 所示。

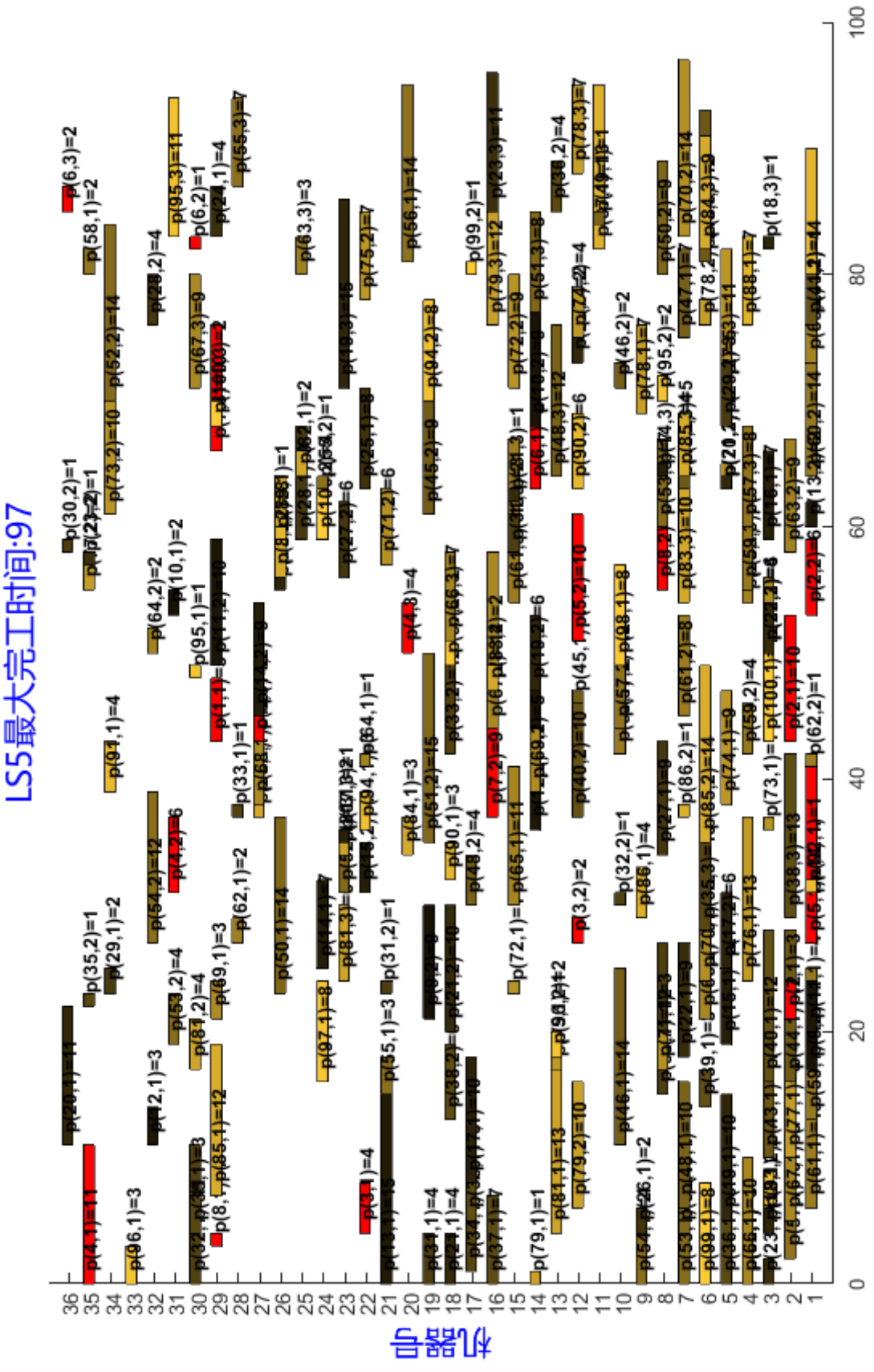


图 2-7 LS5 算例最优解的甘特图

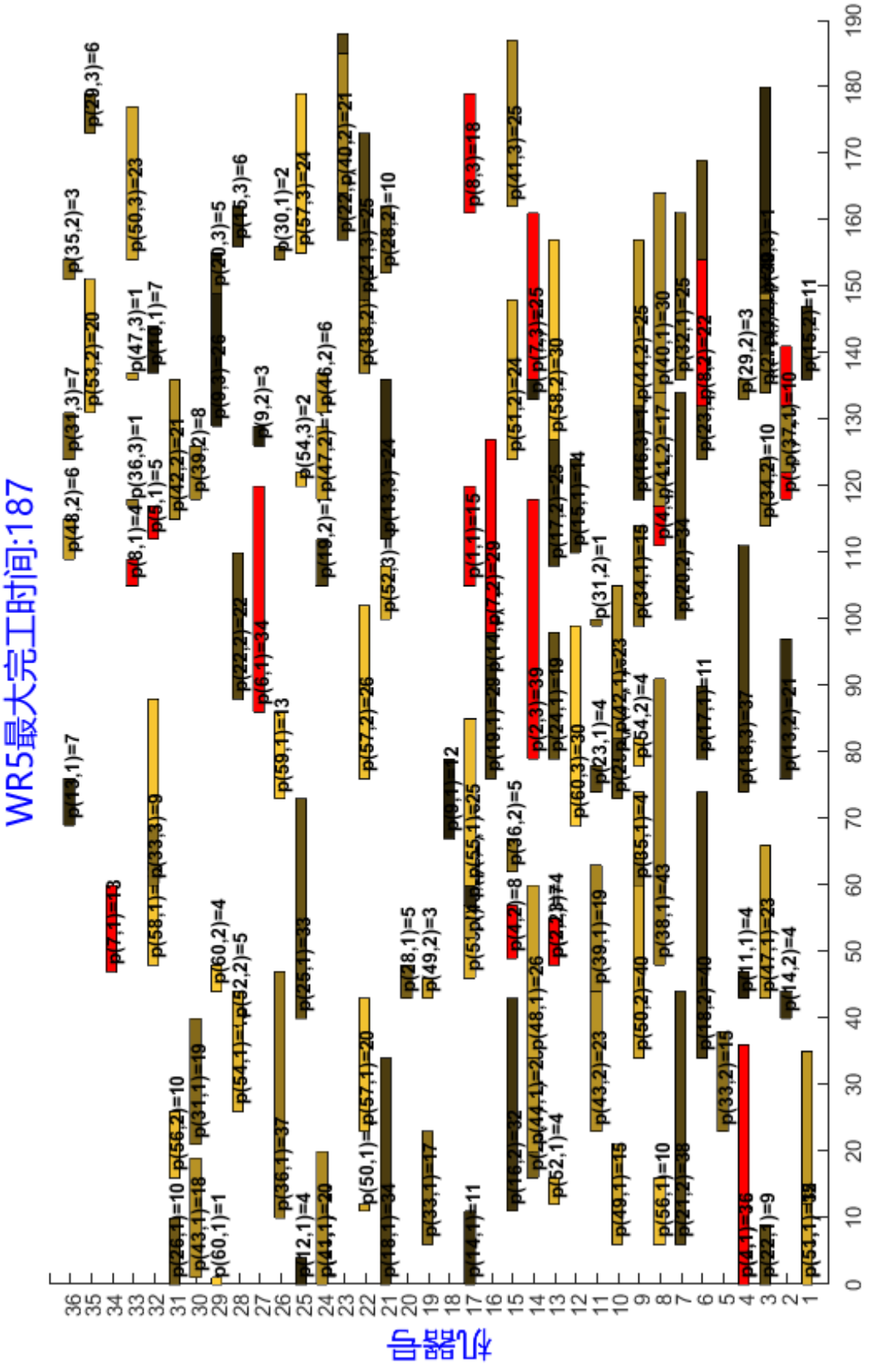


图 2-8 WR5 算例最优解的甘特图

3.4 本章小结

本章在第二章的基础上，考虑了辅助资源约束，对考虑准备时间的 MRC-FJSP 进行了深入研究。首先，以最小化最大完工时间为优化目标，建立了考虑准备时间的 MRC-FJSP 的数学模型。然后，结合考虑准备时间的 MRC-FJSP 特征，对 2.3 节的 IPSO 算法进行了修改。提出了一种半主动解码与启发式规则相结合的新型解码方式，对原有解空间进行了有效裁剪；设计了一种自适应邻域搜索算子，提高了算法的局部开发能力；提出了基于随机网络的多种群策略，提高了算法的全局勘探能力。最后，通过数值实验对 IPSO 的关键参数进行了校正，并验证了所提 IPSO 算法相较于其它对比算法的优越性。

4 考虑准备时间的多目标 MRC-FJSP 方法

4.1 多目标优化基础理论

在实际工程应用中，大多数问题是多目标优化问题。与单目标优化问题相比，多目标优化需要同时考虑多个冲突目标，一般适用于单目标优化问题的方法很难用于求解多目标优化问题。1989 年 Goldberg 提出一种求解多目标优化问题的新思路——把经济学中的 Pareto 理论与智能优化算法结合起来，这对于后续多目标优化算法的研究具有重要的指导价值。

一个具有 q 个目标函数、u 个等式约束、p 个不等式约束的多目标优化问题可以描述如下：

$$\min F(\vec{x}) = \left[f_1(\vec{x}), f_2(\vec{x}), \cdots, f_q(\vec{x}) \right] \tag{4-1}$$

$$s.t.\ \ g_i(\vec{x}) = 0, i = 1, 2, \cdots u \tag{4-2}$$

$$h_j(\vec{x}) \le 0, j = 1, 2, \cdots, p \tag{4-3}$$

其中（4-1）为目标函数集，（4-2）为等式约束集，（4-3）为不等式约束集。

下面将从 Pareto 支配、快速分支配排序、拥挤距离等对多目标优化基础理论进行简明介绍。

4.1.1 Pareto 支配

解 a 支配解 b：记作 $a \prec b$，也称 a 非劣于 b，当且仅当解 a 的所有目标函数都不差于 b，且至少有一个目标函数优于 b 时，成立。

解 a 与解 b 互不支配：记作 $a \sim b$，也称 a 无差别于 b，当且仅当 a 不支配 b 且 b 不支配 a 时，成立。

以图 4-1 为例，这是一个包含两个优化目标的多目标优化问题的目标空间，解 B 的两个目标值都要优于 G，因此解 B 支配解 G，记作 $B \prec G$。解 A 在目标 1 上优于 B，在目标 2 上差于 B，因此解 A 与解 B 互不支配，记作 $A \sim B$。

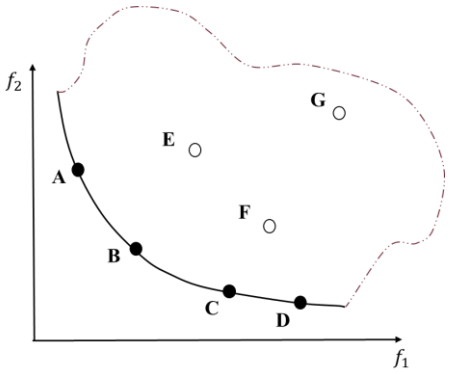


图 4-1 两个目标函数的 Pareto 前沿分布示意图

4.1.2 快速非支配排序

本章采用 Deb 等于 2002 年提出的快速非支配排序方法，该排序方法可以得到每个解的支配个体数和被支配个体数，进而推算出每个解所处的 Pareto 层级。

快速非支配排序的具体操作过程如下：

Step1：为每一个个体 a 定义一个支配个体数 np（用以记录支配个体 a 的个体数）和被支配列表 $SP\{\ \}$（用以存储被个体 a 支配的个体的下标），并对它们进行初始化；

Step2：将临时种群 R_t 内所有个体按照支配与非支配的偏序关系进行两两比较，如果个体 a 被个体 b 支配，就把个体 a 的支配个体数 np 加 1；如果个体 b 被个体 a 支配，就将个体 b 的下标值放入个体 a 的被支配列表 $SP\{\ \}$ 里；

Step3：将支配个体数为 0 的个体放入帕累托前沿的第一级，并将其序值 $rand$ 设置为 1；

Step4：遍历 Pareto 前沿第一级所有个体的被支配列表，遍历之前将被支配列表 $SP\{\ \}$ 内存储的所有个体的支配个体数 np 减 1，并将此时支配个体数 np 为 0 的个体放入 pareto 前沿第二级，将其序值 $rand$ 设置为 2；

Step5：以此类推，依次得到 Pareto 前沿各层级，完成整个种群的分层。

4.1.3 拥挤距离

拥挤距离是对处在同一 Pareto 层级上的解进行评估的依据，解的拥挤距离越大

越好。

拥挤距离 D_i 的计算公式如下所示：

$$D_i=\begin{cases}\sum_{j=1}^{q}\dfrac{|f_j(i+1)-f_j(i-1)|}{f_j^{\max}-f_j^{\min}}, i\in\{2,\cdots,|P|-1\}\\ \infty, i=1\text{或者}|P|,\text{即端点处的解}\end{cases} \tag{4-4}$$

式中，D_i 为第 i 个解的拥挤距离，$f_j^{\min}$、$f_j^{\max}$ 分别为第 j 个目标函数的最小值、最大值，$|P|$ 为相应 Pareto 层级所含解的数量。图 4-2 为拥挤距离的求解示意图。

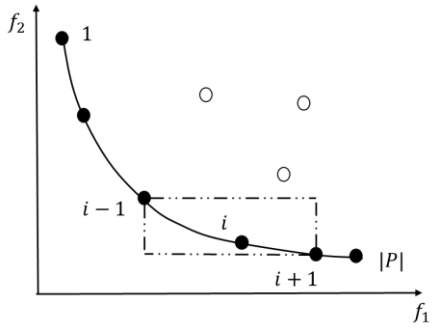


图 4-2 拥挤距离的求解示意图

4.2 问题描述与数学模型建立

综合考虑之后，本章选用了最小化最大完工时间和最小化机器最大负荷两个优化目标。这两个目标都是在生产调度领域中经常提到的，与生产效益有着密切的联系。最大完工时间是衡量调度方案优劣的基础指标，能够直接反映出车间的生产效率，决定着订单能否按时交付；在考虑准备时间的 MRC-FJSP 中，存在机器选择环节，随着调度方案的改变，每台机器的负荷也会发生变化，其中负荷最大的机器就是瓶颈机器，要想提高每台机器的利用率，就必须使各台机器的负荷尽可能小且均衡，机器最大负荷正是衡量机器负载平衡程度的指标。

本章所选的两个目标都是正规性能指标，且两目标之间具有一定的冲突性。本章结合 3.2.2 节的数学模型，给出这两个目标函数的计算公式。符号说明详见 2.2.2 节和 3.2.2 节。

（1）最小化最大完工时间 $C_{\max}$

$$\min C_{\max} = \left\{ s_{il_i} + X_{il_i k} p_{il_i k} \right\} \tag{4-5}$$

式中：$i = 1,2,3,\cdots,n$；$k = 1,2,3,\cdots,m_{il_i}$。

（2）最小化机器最大负荷 $F_{\max}$

$$\min F_{\max} = \max \sum_{j=1}^{l_i} X_{ijk} p_{ijk} \tag{4-6}$$

式中：$i = 1,2,3,\cdots,n$；$k = 1,2,3,\cdots,m_{ij}$。

约束条件与 3.2.2 节完全相同。

4.3 MOPSO 算法求解考虑准备时间的多目标 MRC-FJSP

4.3.1 编码与解码

本章的编解码方式与第三章完全相同，仅编写工序排列向量和机器分配向量，机器配置子问题放在解码过程中动态决策。

4.3.2 种群初始化

本章同时以最小化最大完工时间和最小化机器最大负荷为目标，不同于单目标优化问题，在初始解的生成过程中，需要综合考虑两个优化目标。本章中工序排列向量采用随机初始化方式，机器分配向量依照预设的概率从四种不同的初始化策略中随机选择一个。其中选择随机初始化的概率为 70%，选择最早完工时间优先规则（Earliest Completion Time，ECT）的概率为 10%，选择准备时间与加工时间之和最短优先规则（Shortest Setup plus Process Time，SSPT）的概率为 10%，选择机器负荷最小优先规则（Smallest Machine Load，SML）的概率为 10%。

4.3.3 子种群“社会认知”阶段

子种群的社会认知阶段主要完成子种群内部的信息交互。MOPSO 算法的子种群“社会认知”阶段的具体执行过程如下：

Step1：对每个子种群中的个体进行非支配排序；

Step2：从子种群中分别抽取两个 Pareto 序值靠前的个体 v_{o1}（v'_{o1}）和两个 Pareto

序值靠后的个体v_{o2}（v'_{o2}）；

Step3：v_{o1}与v_{o2}交互，得到两个新个体v_{n1}、v_{n2}。交互时，工序排列向量采用 IPOX 算子，机器分配向量采用 MPX 算子。对v_{n1}、v_{n2}和v_{o2}进行非支配排序，随机抽取其中的一个非劣解，按照非支配关系把它放入子种群相应的 Pareto 层级，其余两个个体淘汰；

Step4：v'_{o1}与v'_{o2}交互，得到两个新个体v'_{n1}、v'_{n2}。交互方式同 Step3。对v'_{n1}、v'_{n2}和v'_{o2}进行非支配排序，随机抽取其中的一个非劣解，按照非支配关系把它放入子种群相应的 Pareto 层级，其余两个个体淘汰。

4.3.4 子种群“个体认知”阶段

子种群的“个体认知”阶段主要完成单个个体的邻域搜索。由于第三章的邻域结构是在最小化$C_{\max}$的背景下设计的，虽然它们可以引导种群朝着真实 Pareto 前沿不断逼近，但是会出现算法最终求得的 Pareto 前沿过于集中、Pareto 解分布性较差的情况。所以必须改变现有的邻域搜索策略。本章加入了面向最小化机器最大负荷的邻域结构——使用机器负荷最小优先原则对一定比例个体的机器分配向量进行重置，此举在一定程度上保证了各加工机器上负荷的平衡。

MOPSO 算法中子种群“个体认知”阶段的具体执行过程如下所示：

Step1：依照邻域选择概率，从第三章中的邻域结构中随机选择一个；

Step2：使用所选邻域结构执行邻域搜索，得到一个新个体；

Step3：如果新个体支配旧个体，则舍弃旧个体，把新个体加入子种群中；如果新旧个体是非支配关系，则留下机器最大负荷小的个体；否则，保留旧个体，不吸纳新个体；

Step4：重复 Step1 到 Step3 一次；

Step5：随机抽取一个个体执行面向最小化机器最大负荷的邻域搜索，得到一个新个体；

Step6：如果新个体支配旧个体，则舍弃旧个体，把新个体加入子种群中；否则，保留旧个体，不吸纳新个体。

4.3.5 子种群间信息交互

子种群之间的信息交互，依然采用第三章中基于随机网络结构的多种群策略，但是具体操作细节有差别。MOPSO 算法的子种群间信息交互过程如下：

Step1：生成一个位于区间$(0,1)$内的随机数。如果该随机数小于P_c，则进行 Step2 到 Step5；否则，本轮迭代不进行子种群间信息交互；

Step2：根据连接概率CP获取用以指导子种群交互的关系矩阵A；

Step3：按照关系矩阵A，从两个待交互的子种群中各自抽取一个 Pareto 序值靠前的个体v_{o1}（ω_{o1}）和一个 Pareto 序值居中的个体v_{o2}（ω_{o2}）；

Step4：v_{o1}与ω_{o2}交叉，得到两个新个体ω_{n1}、ω_{n2}，交叉方式同 4.4.3 节。对ω_{n1}、ω_{n2}和ω_{o2}进行非支配排序，留下三者中机器最大负荷最小的非支配个体，其余两个个体淘汰；

Step5：v_{o2}与ω_{o1}交叉，得到两个新个体v_{n1}、v_{n2}，交叉方式同 4.4.3 节。对v_{n1}、v_{n2}和v_{o2}进行非支配排序，留下三者中机器最大负荷最小的非支配个体，其余两个个体淘汰。

4.3.6 外部记忆库更新

为了避免进化过程中得到的非劣解丢失，采用外部记忆库策略。外部记忆库更新的基本步骤如下：

Step1：标记每个个体所属的子种群，所有子种群与外部记忆库合并为一个临时种群；

Step2：对临时种群中的所有个体执行非支配排序；

Step3：统计子种群贡献的新的非劣解数量，并清除所有个体标记；

Step4：清空外部记忆库，并将临时种群中的非劣解回填到外部记忆库中。

4.3.7 子种群重新初始化

当一个子种群连续T次迭代都未向外部记忆库提供新的非劣解时，需要对该子种群进行重新初始化，以节省宝贵的计算资源。MOPSO 算法的子种群重新初始化具体步骤如下：

Step1：对子种群中的个体进行非支配排序并计算拥挤距离；

Step2：按照非支配序值由小到大、拥挤距离由大到小的顺序对子种群中的个体进行排序；

Step3：删除子种群中排在最后的三个个体，从外部记忆库中随机选择一个非劣解填入子种群，再依照 4.4.2 节中的方法生成两个新个体。

4.3.8 算法流程

本章 MOPSO 算法流程图如图 4-3 所示，具体步骤如下：

Step1（初始化）：初始化 MOPSO 算法参数和外部记忆库，按照 4.4.2 节中的方法生成若干个初始子种群，算法相关参数包含种群规模、子种群数量以及算法中解的最大评估次数等；

Step2（子种群“社会认知”阶段）：按照 4.4.3 节中的方法完成子种群内部的信息交互；

Step3（子种群“个体认知”阶段）：按照 4.4.4 节中的方法完成子种群内单个个体的邻域搜索；

Step4（子种群间信息交互）：按照 4.4.5 节中的方法完成子种群之间的信息交互；

Step5（外部记忆库更新）：按照 4.4.6 节中的方法更新外部记忆库中非劣解；

Step6（子种群重新初始化）：当一个子种群连续T次迭代都未向外部记忆库提供新的非劣解时，按照 4.4.7 节中的方法对该子种群进行重新初始化；

Step7（算法终止）：重复 Step2 到 Step6 直至达到解的最大评估次数，输出外部记忆库中的非劣解。

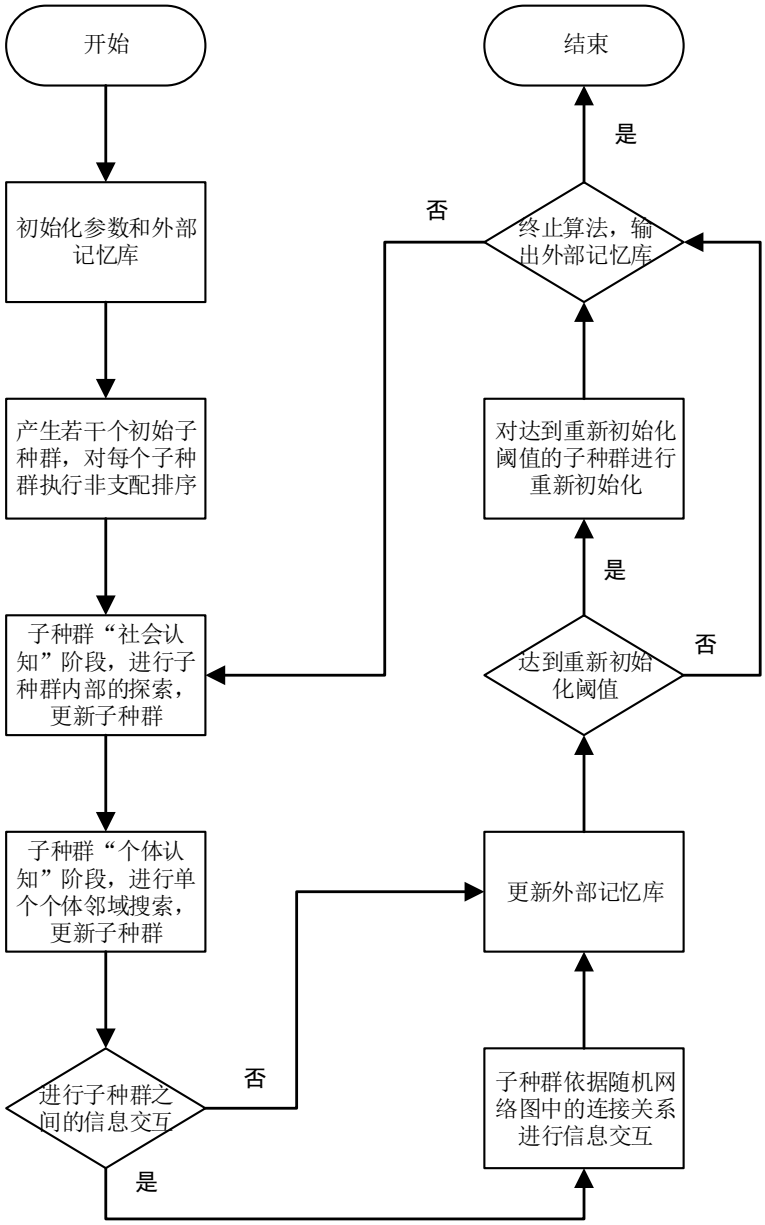


图 4-3 MOPSO 算法流程图

4.4 数值实验与结果分析

4.4.1 数值实验设计

本章将继续沿用第三章里 Wu 等[9]提出的标准测试算例，针对该算例，将本章所提 MOPSO 算法与多目标优化领域应用最为广泛的 NSGA-II算法进行对比。MOPSO 和 NSGA-II都在 MATLAB R2016a 上采用 MATLAB 语言编程实现，运行环境为 Core

4C+6G 1.9GHz CPU、4.00GB 内存、Windows 10 操作系统。

MOPSO 算法的参数设置与第三章完全相同。为保障对比实验的公平公正，NSGA-II和 MOPSO 的种群规模均设置为 60，终止条件都是解的最大评估次数达到 10000 次。

为尽可能减小偶然因素对实验结论产生的影响，各算法分别针对每个算例独立运行 30 次。并采用以下 3 个指标对算法的性能展开综合评估。

Δ：分布度（Spread），Δ被用于衡量算法得到的 Pareto 前沿（Pareto Front，PF ）的分布均匀性。Δ越小，说明 PF 分布越均匀，Pareto 解的多样性越高。Δ可以通过公式（4-7）进行计算。

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{n-1} | d_i - \bar{d} |}{d_f + d_l + (n-1) \times \bar{d}} \tag{4-7}$$

式中，d_f 和 d_l 分别表示 PF 中的边界点与实际 Pareto 前沿（ PF^* ）边界点间的距离，d_i 表示 PF 中的第 i 个解与其相邻解之间的欧式距离，$\bar{d}$ 为 d_i 的算术平均值，n 表示 PF 中 Pareto 解的数量。

GD：世代距离（Generational Distance，GD），GD 被用于衡量 PF 接近 PF^* 的程度。GD 越小，说明 PF 越接近 PF^*，算法的收敛性也就越好。GD 可以通过公式（4-8）进行计算。

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{4-8}$$

式中，d_i 表示 PF 中的第 i 个解到 PF^* 中非支配解的最小欧式距离，n 表示 PF 中 Pareto 解的数量。

IGD：反世代距离（Inverted Generational Distance，IGD），IGD 是一个综合性能指标，被用于计算 PF^* 到 PF 的最小欧式距离。IGD 越小，说明算法的收敛性越好，

算法得到的 Pareto 解的多样性越好。*IGD* 可以通过公式（4-9）进行计算。

$$IGD = \frac{\sum_{i=1}^{n} d(i, PF)}{n} \tag{4-9}$$

式中，n 表示 PF 中非支配解的数量，$d(i, PF)$ 表示 PF^* 中的第 i 个解到 PF 中非支配解的最小欧式距离。

值得注意的是，Δ、*GD* 和 *IGD* 的计算的都需要用真实 Pareto 前沿 PF^*，但 PF^* 无从知晓。所以在计算这三个评价指标时，用 NSGA-II和 MOPSO 在 10 次独立运行中得到的最优 Pareto 前沿近似替代 PF^*。同时，为消除因目标取值区间的不同造成的系统误差，对本章所用到的各目标函数值分别进行归一化处理。

4.4.2 实验结果与分析

NSGA-II和 MOPSO 分别针对每个算例独立运行 10 次后，得到的Δ、*GD*、*IGD* 对比结果分别如表 4-1、表 4-2、表 4-3 所示。对于这三个评价指标，分别统计计算它们的 *Best*、*Mean* 和 *SD*。

表 4-1 NSGA-II和 MOPSO 在Δ指标上的对比结果

算法	NSGA-II			MOPSO		
算例	*Best*	*Mean*	*SD*	*Best*	*Mean*	*SD*
LS1	6.45E-01	7.12E-01	8.15E-02	**5.13E-01**	**5.44E-01**	**4.13E-02**
LS2	5.74E-01	5.96E-01	7.53E-02	**4.86E-01**	**5.31E-01**	**4.93E-02**
LS3	5.69E-01	6.13E-01	3.74E-02	**5.53E-01**	**5.61E-01**	**2.86E-02**
LS4	4.27E-01	5.32E-01	5.41E-02	**4.00E-01**	**4.27E-01**	**2.11E-02**
LS5	4.93E-01	5.21E-01	3.96E-02	**4.15E-01**	**4.38E-01**	**1.75E-02**
WR1	5.78E-01	6.12E-01	2.79E-02	**5.23E-01**	**5.63E-01**	**2.06E-02**
WR2	6.87E-01	7.31E-01	**4.10E-02**	**5.21E-01**	**5.71E-01**	4.29E-02
WR3	6.63E-01	7.45E-01	6.59E-02	**6.44E-01**	**6.89E-01**	**3.15E-02**

WR4	4.65E-01	5.43E-01	5.52E-02	**1.85E-01**	**2.04E-01**	**2.26E-02**
WR5	**4.77E-01**	**5.03E-01**	**2.14E-02**	4.89E-01	5.37E-01	3.43E-02

注：*Best*、*Mean*、*SD*指标下加粗字体表示相应指标的最优值。

表 4-2 NSGA-II和 MOPSO 在*GD*指标上的对比结果

算法	NSGA-II			MOPSO		
算例	*Best*	*Mean*	*SD*	*Best*	*Mean*	*SD*
LS1	1.31E-02	3.32E-02	1.75E-03	**6.44E-03**	**1.11E-02**	**4.59E-04**
LS2	1.56E-02	2.10E-02	**4.95E-03**	**5.93E-03**	**1.26E-02**	5.23E-03
LS3	1.84E-02	2.56E-02	7.53E-03	**8.66E-03**	**1.00E-02**	**2.25E-03**
LS4	7.32E-03	1.24E-02	7.31E-03	**5.37E-03**	**7.22E-03**	**2.74E-03**
LS5	**1.62E-02**	**1.96E-02**	5.49E-03	1.76E-02	2.06E-02	**3.16E-03**
WR1	1.88E-02	2.65E-02	**6.82E-03**	**1.27E-02**	**1.85E-02**	6.93E-03
WR2	**2.26E-02**	**3.11E-02**	**8.72E-03**	2.36E-02	3.42E-02	1.12E-02
WR3	1.26E-02	2.21E-02	1.12E-02	**1.10E-02**	**1.93E-02**	**8.75E-03**
WR4	1.32E-02	2.75E-02	1.08E-02	**1.02E-03**	**3.61E-03**	**3.65E-03**
WR5	1.46E-02	1.87E-02	**5.27E-03**	**1.13E-02**	**1.74E-02**	6.17E-03

注：*Best*、*Mean*、*SD*指标下加粗字体表示相应指标的最优值。

表 4-3 NSGA-II和 MOPSO 在*IGD*指标上的对比结果

算法	NSGA-II			MOPSO		
算例	*Best*	*Mean*	*SD*	*Best*	*Mean*	*SD*
LS1	2.17E-02	2.79E-02	**5.79E-03**	**6.98E-03**	**1.26E-02**	5.87E-03
LS2	**7.95E-03**	**1.41E-02**	**6.23E-03**	9.56E-03	1.78E-02	7.72E-03
LS3	1.27E-02	1.86E-02	6.89E-03	**5.27E-03**	**1.15E-02**	**5.51E-03**
LS4	1.32E-02	1.62E-02	7.27E-03	**8.96E-03**	**1.24E-02**	**5.06E-03**
LS5	1.54E-02	2.31E-02	**9.31E-03**	**1.12E-02**	**1.99E-02**	1.62E-02
WR1	3.15E-02	4.27E-02	**1.13E-02**	**1.73E-02**	**2.91E-02**	1.23E-02

WR2	2.54E-02	3.51E-02	1.20E-02	**1.41E-02**	**2.43E-02**	**1.02E-02**
WR3	1.13E-02	2.03E-02	1.00E-02	**7.41E-03**	**8.12E-03**	**4.15E-03**
WR4	4.12E-03	1.11E-02	7.63E-03	**1.03E-03**	**1.99E-03**	**2.91E-03**
WR5	1.26E-02	1.95E-02	**9.82E-03**	**4.72E-03**	**9.53E-03**	1.05E-02

注：*Best*、*Mean*、*SD*指标下加粗字体表示相应指标的最优值。

对于Δ指标而言，在 10 个标准测试算例中，本章所提 MOPSO 算法得到了 9 个算例的最优*Mean*值，而 NSGA-II只得到了 1 个算例的最优*Mean*值，说明 MOPSO 求得的 Pareto 前沿分布更加均匀。这主要得益于子种群重新初始化算子的设计，避免了算法因陷入局部最优而白白浪费掉宝贵的计算资源。对于*GD*指标而言，在 10 个标准测试算例中，MOPSO 得到了 8 个算例的最优*Mean*值，而 NSGA-II只得到了 2 个算例的最优*Mean*值，说明 MOPSO 的收敛性要优于 NSGA-II。这主要得益于本章设计使用的面向不同优化目标的邻域结构，提升了算法的局部开发能力。对于*IGD*指标而言，在 10 个标准测试算例中，MOPSO 得到了 9 个算例的最优*Mean*值，而 NSGA-II只得到了 1 个算例的最优*Mean*值，说明 MOPSO 综合性能要优于 NSGA-II。

从统计学角度出发，为了更严格地说明 MOPSO 算法在各性能指标上的优越性，本章做了 3 组单侧 t 检验，试验结果见表 4-4。

表 4-4 MOPSO 在 95%置信区间下的显著性检验结果（考虑准备时间的多目标 MRC-FJSP）

性能指标	*t* 值	*p* 值	MOPSO 是否具有显著优势
Δ	-2.097	0.026	是
GD	-2.469	0.012	是
IGD	-2.025	0.029	是

在表 4-4 中，所有的 *p* 值均小于 0.05，说明本章所提 MOPSO 算法在各性能指标上均具有显著优势，能够有效求解考虑准备时间的多目标 MRC-FJSP。

4.5 本章小结

本章在第三章的基础上，进一步研究了考虑准备时间的多目标 MRC-FJSP。首先以最小化最大完工时间和最小化机器最大负荷为优化目标，构建了多目标优化数学

模型。然后，结合多目标优化基础理论，设计了 MOPSO 对该问题进行求解。在 MOPSO 中，提出了一种混合初始化策略，获得了一个分布性较好的初始种群；设计了面向不同优化目标的邻域结构，提高了算法的局部开发能力；提出了一种面向多目标的子种群重新初始化算子，可有效避免算法陷入局部最优。最后，用本章所提 MOPSO 与 NSGA-II进行对比，实验结果验证了 MOPSO 在各项性能指标上的优越性。

5 总结与展望

5.1 全文总结

本文首先研究了考虑准备时间的 FJSP。在此基础上，考虑到实际生产过程中存在辅助资源约束和待优化目标不止一个的特点，深入研究了考虑准备时间的单目标 MRC-FJSP、考虑准备时间的多目标 MRC-FJSP。针对上述问题，本文以 PSO 为主体框架，结合待求解问题的特征设计出与之相适配的调度算法。本文主要研究内容与成果如下：

（1）针对考虑准备时间的 FJSP，提出了一种 IPSO 算法。首先以最小化最大完工时间为目标，建立起考虑准备时间的 FJSP 数学模型。然后结合待求解问题特性，设计出 IPSO 算法。在初始化阶段，为了获取较高质量的初始种群，提出一种混合初始化算子；在“个体认知”阶段，充分挖掘考虑准备时间的 FJSP 的领域知识后，设计出两种基于关键路径的邻域结构——N-CM 和 N-SM，提高了算法的局部开发能力；采用自适应禁忌搜索算子，有效避免了算法陷入循环搜索。最后通过数值实验验证了所提的 IPSO 算法在求解考虑准备时间的 FJSP 上的优越性。

（2）针对考虑准备时间的 MRC-FJSP，提出了一种基于 IPSO 的求解方法。首先以最小化最大完工时间为目标，建立起考虑准备时间的 MRC-FJSP 数学模型。然后结合待求解问题特性，对第二章的 IPSO 算法进行了修改。在解码阶段，设计出一种半主动解码与启发式规则相结合的新型解码方式，在保障最优解或近优解不丢失的前提下，对原有解空间进行了有效裁剪，提高了算法的搜索效率；在“个体认知”阶段，提出了一种自适应邻域搜索算子，提高了算法的局部开发能力；设计了基于随机网络的多种群策略，提高了算法的全局勘探能力。最后通过数值实验验证了所提 IPSO 算法在求解考虑准备时间的 MRC-FJSP 上的优越性。

（3）针对考虑准备时间的多目标 MRC-FJSP，提出了一种 MOPSO 算法。首先以最小化最大完工时间和机器最大负荷为目标，建立起考虑准备时间的多目标 MRC-FJSP 数学模型。然后结合待求解问题特性，应用 Pareto 支配、快速非支配排序、拥挤距离等多目标优化基础理论，设计出 MOPSO 算法。在“个体认知”阶段，设计出

分别面向不同优化目标的邻域结构，提高了算法的局部开发能力；提出一种针对多目标的子种群重新初始化算子，有效避免了算法陷入早熟收敛。最后，用本章所提MOPSO与NSGA-II进行对比，验证了MOPSO在求解考虑准备时间的多目标MRC-FJSP上的显著优势。

5.2 研究展望

本文对考虑准备时间的FJSP、考虑准备时间的MRC-FJSP展开了研究，并基于PSO的算法框架提出了考虑准备时间的单目标FJSP、考虑准备时间的单目标MRC-FJSP和考虑准备时间的多目标MRC-FJSP求解方法。但本文的研究工作还存在一些不足之处，可以在未来继续研究，具体内容如下：

（1）在多目标方面，本文第四章仅研究了以最大完工时间和机器最大负荷为目标的考虑准备时间的多目标MRC-FJSP。针对MRC-FJSP，还有其它性能指标也需要进行考虑，比如总碳排放、总能耗等。随着人们节能减排意识的增强，多资源受限下考虑准备时间的柔性作业车间绿色调度的研究具有广阔的前景。

（2）在动态调度方面，本文研究的3个问题都只考虑了静态环境下的车间调度问题。而在工厂实际生产过程中，往往存在着一些随机因素影响着调度方案的制定，比如紧急插单、机器故障、交货期提前等。未来可以开展动态环境下考虑准备时间的MRC-FJSP相关研究工作。

（3）在分布式方面，本文研究的3个问题都只考虑了单车间环境下的车间调度问题。在经济全球化背景下，公司之间的合作生产以及兼并收购日益普遍，分布式制造已成为一种常见的生产模式。分布式车间调度技术的研究，也愈加重要。未来可以考虑研究考虑准备时间的分布式MRC-FJSP。

致谢

光阴似箭、岁月如梭，短暂而充实的硕士生生涯快要结束了。在本文即将完成之际，特向两年来在人生道路上一直陪伴着我的老师、同学和家人致以最诚挚的感谢，我取得的每一分成绩都离不开你们的帮助、鼓励和支持。

本文在高亮教授、李新宇教授和张春江老师的悉心指导下完成，从论文选题到算法设计、实验设计再到后面的论文撰写，每一个环节都倾注了老师们的大量心血。高老师渊博的学识、严谨认真的治学态度、高瞻远瞩的视野以及开朗豁达的心胸都给我留下了深刻的印象，这些精神上的财富将使我受益终生。李老师以身作则，在科研上勇攀高峰，经常为我们答疑解惑、指点迷津，他常常告诫我们细节决定成败，做事情一定要耐心、专心、细心。张老师是我们课题组的博士后，他不仅在学术上给予我许多的帮助和支持，在生活上也无微不至地关心、照顾我，既是良师，也是益友。在此，再次向各位老师表达由衷的感谢。

其次，还要特别感谢桂林师兄和王思涵师姐在做项目的过程中给予的指导与帮助，我们经常在一起交流彼此独到的见解、碰撞出了无数思想的火花。同时，感谢王光辰、李颖俐、刘齐浩、谢晋、黎阳等师兄师姐对我的照顾。感谢机械大楼西楼 D506 实验室的小伙伴们两年里的陪伴，我们一起学习、共同进步，在此祝愿大家未来可期、事业有成、生活幸福美满。

然后，还要感谢我的父母，你们二十多年来含辛茹苦地悉心培育我，始终做我最坚实的后盾，让我可以在求学的道路上奋勇前行。

最后，感谢百忙之中参与评阅和答辩工作的各位专家学者。

作者：崔航浩
二零二一年五月

参考文献

[1]Zheng X L, Wang L, Wang S Y. A Novel Fruit Fly Optimization Algorithm for the Semiconductor Final Testing Scheduling Problem[J]. Knowledge-Based Systems, 2014, 57: 95-103.

[2]Hao X C, Wu J Z, Chien C F, et al. The Cooperative Estimation of Distribution Algorithm: A Novel Approach for Semiconductor Final Test Scheduling Problems[J]. Journal of Intelligent Manufacturing, 2014, 25(5): 867-879.

[3]Gao L, Pan Q K. A Shuffled Multi-Swarm Micro-Migrating Birds Optimizer for a Multi-Resource-Constrained Flexible Job Shop Scheduling Problem[J]. Information Sciences, 2016, 372: 655-676.

[4]Tang L X, Zhao Y, Liu J Y. An Improved Differential Evolution Algorithm for Practical Dynamic Scheduling in Steelmaking-Continuous Casting Production[J]. IEEE Transactions on Evolutionary Computation, 2014, 18(2): 209-225.

[5]Anily S, Haviv M. Subadditive and Homogeneous of Degree One Games are Totally Balanced[J]. Operations Research, 2014, 62(4): 788-793.

[6]李郝林，孙栋. 数控机床加工任务与刀具的调度方法[P]. 中国专利: CN101533274A, 2009-09-16.

[7]Gargeya V B, Deane R H. Scheduling Research in Multiple Resource Constrained Job Shops: A Review and Critique[J]. International Journal of Production Research, 1996, 34(8): 2077-2097.

[8]Brucker P, Schlie R. Job-Shop Scheduling with Multi-Purpose Machines[J]. Computing, 1990, 45(4): 369-375.

[9]Wu J Z, Chien C F. Modeling Semiconductor Testing Job Scheduling and Dynamic Testing Machine Configuration[J]. Expert Systems with Applications, 2008, 35(1): 485-496.

[10]Azzouz A, Ennigrou M, Said L B. A Hybrid Algorithm for Flexible Job-Shop Scheduling Problem with Setup Times[J]. International Journal of Production Management and Engineering, 2017, 5(1): 23-30.

[11]Kim S C, Bobrowski P M. Scheduling Jobs with Uncertain Setup Times and Sequence Dependency[J]. Omega, 1997, 25(4): 437-447.

[12]Imanipour N. Modeling & Solving Flexible Job Shop Problem with Sequence Dependent Setup Times[C]. In: International Conference on Service Systems and Service Management. Troyes: IEEE, 2006: 1205-1210.

[13]Sadrzadeh A. Development of Both the AIS and PSO for Solving the Flexible Job Shop

Scheduling Problem[J]. Arabian Journal for Science and Engineering, 2013, 38(12): 3593-3604.

[14]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.

[15]Mousakhani M. Sequence-Dependent Setup Time Flexible Job Shop Scheduling Problem to Minimise Total Tardiness[J]. International Journal of Production Research, 2013, 51(12): 3476-3487.

[16]Bagheri A, Zandieh M. Bi-criteria Flexible Job-Shop Scheduling with Sequence-Dependent Setup Times—Variable Neighborhood Search Approach[J]. Journal of Manufacturing Systems, 2011, 30(1): 8-15.

[17]Li Z C, Qian B, Hu R, et al. An Elitist Nondominated Sorting Hybrid Algorithm for Multi-Objective Flexible Job-Shop Scheduling Problem with Sequence-Dependent Setups[J]. Knowledge-Based Systems, 2019, 173: 83-112.

[18]Zhu Z W, Zhou X H. An Efficient Evolutionary Grey Wolf Optimizer for Multi-Objective Flexible Job Shop Scheduling Problem with Hierarchical Job Precedence Constraints[J]. Computers & Industrial Engineering, 2020, 140: 106280.

[19]Azzouz A, Chaabani A, Ennigrou M, et al. Handling Sequence-Dependent Setup Time Flexible Job Shop Problem with Learning and Deterioration Considerations Using Evolutionary Bi-Level Optimization[J]. Applied Artificial Intelligence, 2020, 34(6): 433-455.

[20]Rossi A. Flexible Job Shop Scheduling with Sequence-Dependent Setup and Transportation Times by Ant Colony with Reinforced Pheromone Relationships[J]. International Journal of Production Economics, 2014, 153: 253-267.

[21]Zhang M, Tan Y, Zhu J, et al. A Competitive and Cooperative Migrating Birds Optimization Algorithm for Vary-Sized Batch Splitting Scheduling Problem of Flexible Job-Shop with Setup Time[J]. Simulation Modelling Practice and Theory, 2020, 100: 102065.

[22]Uzsoy R, Martin L A, Lee C Y, et al. Production Scheduling Algorithms for a Semiconductor Test Facility[J]. IEEE Transactions on Semiconductor Manufacturing, 1991, 4(4): 270-280.

[23]赵诗奎，王林瑞，石飞．作业车间调度问题综述[J]．济南大学学报(自然科学版), 2016, 30(01): 74-80.

[24]Lin J T, Wang F K, Lee W T. Capacity-Constrained Scheduling for a Logic IC Final

Test Facility[J]. International Journal of Production Research, 2004, 42(1): 79-99.
[25]Pearn W L, Chung S H, Chen A Y, et al. A Case Study on the Multistage IC Final Testing Scheduling Problem with Reentry[J]. International Journal of Production Economics, 2004, 88(3): 257-267.
[26]Yang S, Zhang M T, Yi J, et al. Bottleneck Station Scheduling in Semiconductor Assembly and Test Manufacturing Using Ant Colony Optimization[J]. IEEE Transactions on Automation Science and Engineering, 2007, 4(4): 569-578.
[27]Wu J Z, Hao X C, Chien C F, et al. A Novel Bi-Vector Encoding Genetic Algorithm for the Simultaneous Multiple Resources Scheduling Problem[J]. Journal of Intelligent Manufacturing, 2012, 23(6): 2255-2270.
[28]Wang S Y, Wang L, Liu M, et al. A Hybrid Estimation of Distribution Algorithm for the Semiconductor Final Testing Scheduling Problem[J]. Journal of Intelligent Manufacturing, 2015, 26(5): 861-871.
[29]Wang S Y, Wang L. A Knowledge-Based Multi-Agent Evolutionary Algorithm for Semiconductor Final Testing Scheduling Problem[J]. Knowledge-Based Systems, 2015, 84: 1-9.
[30]Sang H Y, Duan P Y, Li J Q. An Effective Invasive Weed Optimization Algorithm for Scheduling Semiconductor Final Testing Problem[J]. Swarm and Evolutionary Computation, 2018, 38: 42-53.
[31]Cao Z C, Lin C R, Zhou M C, et al. An Improved Cuckoo Search Algorithm for Semiconductor Final Testing Scheduling[C]. In: IEEE International Conference on Automation Science and Engineering. New York: IEEE, 2017: 1040-1045.
[32]Cao Z C. Lin C R, Zhou M C, et al. Scheduling Semiconductor Testing Facility by Using Cuckoo Search Algorithm With Reinforcement Learning and Surrogate Modeling[J]. IEEE Transactions on Automation Science and Engineering, 2019, 16(2): 825-837.
[33]付坤. 基于鲸鱼群算法的柔性作业车间调度方法研究[D]. 华中科技大学, 2019.
[34]Shi Y. A Modified Particle Swarm Optimizer[C]. In: IEEE International Conference on Evolutionary Computation Proceedings. Anchorage: IEEE, 1998: 69-73.
[35]郭文忠, 陈国龙, 陈振. 离散粒子群优化算法研究综述[J]. 福州大学学报(自然科学版), 2011, 39(05): 631-638.
[36]罗辞勇, 卢斌, 陈民铀, 等. 组合粒子群优化和分布估计的多目标优化算法[J]. 重庆大学学报, 2010, 33(04): 31-36.
[37]沈林成, 霍霄华, 牛轶峰. 离散粒子群优化算法研究现状综述[J]. 系统工程与电子技术, 2008(10): 1986-1990.

[38]Bonnah E, Ju S, Cai W. Coverage Maximization in Wireless Sensor Networks Using Minimal Exposure Path and Particle Swarm Optimization[J]. Sensing and Imaging, 2019, 21(1): 4.

[39]Asadi M, Jabraeil M A, Parsa S, et al. Detecting Botnet by Using Particle Swarm Optimization Algorithm based on Voting System[J]. Future Generation Computer Systems, 2020, 107: 95-111.

[40]Qiu R, Xu J P, Ke R M, et al. Carbon Pricing Initiatives-based Bi-Level Pollution Routing Problem[J]. European Journal of Operational Research, 2020, 286(1): 203-217.

[41]Zhao X G, Ji L, Jin M, et al. An Improved Quantum Particle Swarm Optimization Algorithm for Environmental Economic Dispatch[J]. Expert Systems with Applications, 2020, 152: 113370.

[42]Zhu B, Feng Y, Gong D Z, et al. Hybrid Particle Swarm Optimization with Extreme Learning Machine for Daily Reference Evapotranspiration Prediction from Limited Climatic Data[J]. Computers and Electronics in Agriculture, 2020, 173: 105430.

[43]Li S H, Yu K, Wang D W, et al. Deep Learning based Prediction of Species-Specific Protein S-Glutathionylation Sites[J]. Biochimica et Biophysica Acta - Proteins and Proteomics, 2020, 1868(7): 140422.

[44]Cai C X, Sun C S, Han Y, et al. Clinical Flexible Needle Puncture Path Planning based on Particle Swarm Optimization[J]. Computer Methods and Programs in Biomedicine, 2020, 193: 105511.

[45]Zhang H C, Cao Q, Gao H, et al. Optimum Design of a Multi-Form Energy Hub by Applying Particle Swarm Optimization[J]. Journal of Cleaner Production, 2020, 260: 121079.

[46]Chen R Q, Bian H, Hou C Y, et al. Echo Decomposition of Full-Waveform LiDAR based on a Digital Implicit Model and a Particle Swarm Optimization[J]. Applied Optics, 2020, 59(13): 4030-4039.

[47]Chen Y, Yan J, Sareh P, et al. Feasible Prestress Modes for Cable-Strut Structures with Multiple Self-Stress States Using Particle Swarm Optimization[J]. Journal of Computing in Civil Engineering, 2020, 34(3): 4020003.

[48]Zhang W J, Huang Y P. Using Big Data Computing Framework and Parallelized PSO Algorithm to Construct the Reservoir Dispatching Rule Optimization[J]. Soft Computing, 2020, 24(11): 8113-8124.

[49]Sabanci K, Balci S. Development of an Expression for the Output Voltage Ripple of the DC-DC Boost Converter Circuits by Using Particle Swarm Optimization Algorithm[J]. Measurement, 2020, 158: 107694.

[50] Song Y, Zhang F, Liu C C. The Risk of Block Chain Financial Market based on Particle Swarm Optimization[J]. Journal of Computational and Applied Mathematics, 2020, 370: 112667.

[51] Ganji M, Kazemipoor H, Hadji S M, et al. A Green Multi-Objective Integrated Scheduling of Production and Distribution with Heterogeneous Fleet Vehicle Routing and Time Windows[J]. Journal of Cleaner Production, 2020, 259: 120824.

[52] Farshi T R, Drake J H, Ozcan E. A Multimodal Particle Swarm Optimization-based Approach for Image Segmentation[J]. Expert Systems with Applications, 2020, 149: 113233.

[53] Dadjoo M, Fatemi S B. The Application of Spatial Domain in Optimum Initialization for Clustering Image Data Using Particle Swarm Optimization[J]. Expert Systems with Applications, 2021, 168: 114224.

[54] Mastrolilli M, Gambardella L M. Effective Neighbourhood Functions for the Flexible Job Shop Problem[J]. Journal of Scheduling, 2000(3): 3-20.

[55] 高亮, 张国辉, 王晓娟. 柔性作业车间调度智能算法及其应用[M]. 武汉：华中科技大学出版社, 2012.

[56] Shi X Q, Long W, Li Y Y, et al. Research on the Performance of Multi-Population Genetic Algorithms with Different Complex Network Structures[J]. Soft Computing, 2020, 24(17): 13441-13459.

[57] Shi X Q, Long W, Li Y Y, et al. Multi-Population Genetic Algorithm with ER Network for Solving Flexible Job Shop Scheduling Problems[J]. Plos One, 2020, 15(5): 1-23.

[58] 崔航浩, 张春江, 李新宇. 基于带随机网络的多种群粒子群优化算法求解多资源受限柔性作业车间调度问题[J/OL]. 重庆大学学报, 2021, doi: 10.11835/j.issn.1000-582X.2021.102.

附录 攻读硕士学位期间发表的学术论文目录

[1] 崔航浩，张春江，李新宇. 基于带随机网络的多种群粒子群优化算法求解多资源受限柔性作业车间调度问题[J/OL]. 重庆大学学报，2021, doi: 10.11835/j.issn.1000-582X.2021.102.（CSCD，北大中文核心，第一作者）