华南理工大学

South China University of Technology

工程硕士学位论文

基于多 Agent 的柔性生产动态调度系统研究

作 者 姓 名	梁慰乐
工 程 领 域	计算机技术
校内指导教师	张平 教授
校外指导教师	张宇 高级工程师
所 在 学 院	计算机科学与工程学院
论文提交日期	2018 年 4 月

Research on Multi-Agent based Flexible Production Dynamic Scheduling System

A Dissertation Submitted for the Degree of Master

Candidate：Liang Weile

Supervisor：Prof. Zhang Ping

South China University of Technology

Guangzhou, China

分类号：TP3 **学校代号：10561**

学　号：201521032116

华南理工大学硕士学位论文

基于多 Agent 的柔性生产动态调度系统研究

作者姓名：梁慰乐　　指导教师姓名、职称：张平 教授

申请学位级别：工程硕士　　工程领域名称：计算机技术

论文形式：□ 产品研发 □ 工程设计 □ 应用研究√ □ 工程/项目管理 □ 调研报告

研究方向： 智能技术与机器人

论文提交日期： 2018 年 4 月 20 日　　答辩日期：　　年　月　日

学位授予单位：华南理工大学　　学位授予日期：　　年　月　日

答辩委员会成员：

主席：_____

委员：_____

华南理工大学
学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

作者签名：　　日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属华南理工大学。学校有权保存并向国家有关部门或机构送交论文的复印件和电子版，允许学位论文被查阅（除在保密期内的保密论文外）；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。本人电子文档的内容和纸质论文的内容相一致。

本学位论文属于：

□保密，（校保密委员会审定为涉密学位时间：____年___月___日）于____年___月___日解密后适用本授权书。

√不保密,同意在校园网上发布，供校内师生和与学校有共享协议的单位浏览；同意将本人学位论文提交中国学术期刊(光盘版)电子杂志社全文出版和编入 CNKI《中国知识资源总库》,传播学位论文的全部或部分内容。

作者签名：　　日期：

指导教师签名：　　日期：

作者联系电话：　13660479336　　电子邮箱：403040463@qq.com

联系地址(含邮编)：广东省广州市大学城华南理工大学 510006

摘　　要

随着全球制造化和信息化的普及，制造企业面临着技术革新和竞争加剧带来的多重压力，包括生产模式的转变、市场需求的动态变化以及设备故障等多种异常因素，企业需要迫切地提高自身的生产管理水平。柔性作业车间调度问题（Flexible Job Shop Scheduling Problem, FJSP）在生产调度问题中占据越来越重要的地位，对比传统的作业车间调度问题，FJSP 在工艺规划和设备选择上具备更多的柔性，导致问题的求解难度更大。在目前生产制造领域中，企业处于复杂多变的制造环境，而动态调度系统作为生产管理的重要模块，关系着企业的核心竞争力，是企业控制生产成本，保证生产效率和质量的关键。本文在对 FJSP 的研究中，综合考虑企业所处的复杂多变的生产制造环境，应用多 Agent 理论，构建了一种柔性生产动态调度系统，本文主要工作包括：

（1）建立了基于多 Agent 的柔性生产动态调度系统的结构模型，分析了企业生产管理的实际需求，将生产管理流程中的业务进行划分，设计了包括管理 Agent、资源 Agent、算法 Agent 和监控 Agent 五类 Agent，其中资源 Agent 根据企业制造资源的分布式特点，以集中控制的分层式组成了资源 Agent 组，并与其他 Agent 构建了分布式的系统架构。

（2）设计了柔性生产动态调度系统中 Agent 间的协作机制，通过定义 Agent 间信息的交互来保证系统生产调度的高效性以及对异常因素的快速响应。该协作机制包括生产任务跨区域分解策略以及异常调度策略，前者用于综合利用企业分布在不同地区的加工工厂的资源和技术，完成生产任务在各个区域的分解和分配过程；后者在如设备故障、紧急订单等异常因素发生时，通过 Agent 之间的协商和合作，实现对动态生产环境的快速响应。

（3）应用改进的蚁群算法求解柔性生产中的调度问题。以最小化最大完工时间为优化目标，根据基本蚁群算法在传统作业车间调度问题中的应用，结合 FJSP 的柔性工艺规划特点，并基于该算法收敛速度慢和易于陷入局部最优解的缺点，对基本蚁群算法中信息素更新规则、蚁群初始位置等方面进行了改进。该算法封装在算法 Agent 中，向资源 Agent 组提供计算服务。

（4）开发了基于多 Agent 的柔性生产动态调度系统。根据系统的分布式结构模型，设计了系统开发的软件体系架构。其次，应用面对对象技术对 Agent 的属性和方法进行了封装，使用基于 TCP/IP 的 Scoket 通信机制以及线程池和消息队列实现了 Agent 的通信模型，最后在本文所开发的调度系统上验证了改进蚁群算法、生产任务跨区域分解以

及异常调度策略的有效性。

关键词：动态调度系统；多 Agent 系统；柔性作业车间调度；分布式资源；改进的蚁群算法；系统开发

Abstract

With the popularization of global manufacturing and informatization, manufacturing companies are faced with multiple pressures brought by technological innovation and increased competition, including transition of production mode, dynamic demand of market, and other abnormal factors such as equipment failures. As a result, Enterprises have to improve the quality of production management as soon as possible. Flexible Job Shop Scheduling Problem (FJSP) occupies an increasingly important position in the production scheduling problem. Compared with the traditional job shop scheduling problem, FJSP has more flexibility in process planning and equipment selection, resulting in much more difficulties among problem solving. As an important module of production management, dynamic scheduling system is the key to control production cost and ensure productivity and quality, and is helpful to improve enterprises' core competitiveness when facing complex and changeable manufacturing environment. During the research of FJSP, considering the complex and ever-changing manufacturing environment in which the company is located, this paper takes aim at constructing a flexible production dynamic scheduling system with multi-agent theory, and the main work as follows:

(1) The multi-Agent based structure model of flexible production dynamic scheduling system is established. This paper analyzes the actual requirements of enterprise production management process and divides the business into five types of agents including management agent, resource agent, algorithm agent, process agent and monitor agent. The resource agents are build as resource agent group with Hierarchical control structure base on the distribution of enterprises' manufacturing resources. Also the resource agent group and other agents are connected with a distributed structure.

(2) This paper defines cooperative mode between Agents in MAS under distributed manufacturing environment . The central idea is to ensure the high efficiency of the system's production scheduling and the rapid response to the production environment by regulating the interaction of information between agents. Therefore, it defines cross-regional production task decomposition strategy and abnormal scheduling strategy. The former strategy aims to utilize enterprise's distributed resources and technologies located in different regions through the process of decomposition and distribution of production tasks; the latter strategy defines the

consultations and cooperation between Agents when abnormal factors such as equipment failures and emergency orders occur to achieve rapid response to the dynamic manufacturing environment.

(3) This paper apply an improved ant colony algorithm to solve the scheduling problem in FJSP. The objective of the scheduling system is to minimize the makespan. Considering there are some disadvantages when using traditional ant colony algorithm to solve FJSP such as slow convergence speed, easily to stagnation, an improved ant colony optimization based on the flexible process planning features of FJSP is proposed in this paper, and the main improvements are to redefine pheromone updating rule and uniformly distribute the initial position of the ants. This improved algorithm is encapsulated in the algorithm Agent which provides computing services to the resource agent group.

(4) The flexible production dynamic scheduling system based on multi-agent is implemented. Based on the distributed structure model, this paper designs the software architecture for the scheduling system. During the implementation of the system, this paper use object-oriented technology to encapsulate the properties and function of each Agent. This paper design communication mechanism based on TCP/IP Socket. Also the communication model of Agent is implemented by thread pool and message queue. Finally, this paper verifies the effectiveness of improved ant colony algorithm, cross-regional production task decomposition and abnormal scheduling strategy via this scheduling system.

Key Words: dynamic scheduling system; multi-agent system; flexible job shop scheduling problem; distributed resource; improved ant colony optimization; system development

目录

第一章 绪论

1.1 研究背景

自我国改革开放以来，制造业一直是我国大力发展的产业，并为国民经济的发展和人民生活水平的提高做出了重大贡献[1]。近年来，在经济全球化的影响下，现代制造业拥有了更多的发展机遇和广泛的市场需求，但也意味着我国的制造业需要在全球化舞台上跟来自世界各地的企业竞争。全球化的浪潮对我国企业传统的生产方式和管理模式带来了诸多影响，这些影响包括[2,3]：

（1）世界范围内科学技术日新月异，分布式计算、云计算、物联网、智能制造、人工智能等理念应运而生，为制造业生产技术的革新提供了扎实的理论支撑，也督促企业减少对人力和人为先验知识的依赖，在生产管理上加快向智能化、自动化的方向靠拢。

（2）随着我国经济的腾飞，人们生活水平有了很大的提高，极大丰富了人民的物质文化需求，对应的，人们的消费观念和习惯发生了很大的转变，对多样化和个性化的商品给予了更多的关注，因此商家需要根据市场需求调整经营策略，不断推陈出新。丰富的产品种类要求制造企业加快转变自身的生产模式，以往的“小品种大批量”生产模式已不再适用，取而代之的则是面向订单的小批量多品种甚至是单件定制化生产，因此制造业需要摒弃传统的面向大批量的长时间跨度的静态生产方式，充分考虑市场需求的动态变化，灵活快速地响应客户的随机订单并生成满足优化目标的调度方案。

（3）个性化的产品需求在另一方面导致了产品功能和结构的升级，因此产品在工艺规划上具备更多的柔性，单一的资源配置和生产方式无法满足柔性生产的需求，这对企业在生产调度的效率和质量上有了更高的要求。

（4）经济全球化和发达的网络通信技术使制造业的生产服务对象的范围扩大到世界各地，企业在生产上的控制方式从集中控制式向分布式转变，对应地企业通常细化生产管理的各个流程，在世界各地建立加工工厂，满足不同地区的生产需求。

我国制造业面临的种种竞争和挑战，需要通过对生产管理进行革新，车间作为企业生产管理的最小单位，车间调度是企业保证生产质量、缩短生产周期和控制生产成本的关键。车间调度系统集成制造企业在生产计划制定、生产调度、生产管理、生产监控、质量保证等各个环节，是提高企业竞争力，实现企业智能化、敏捷化的核心所在。

1.2 研究目的与意义

车间调度是企业保证生产质量、缩短生产周期和控制生产成本的关键[4]，而其中的柔性作业车间调度占据了越来越重要的地位，这是由于产品在工艺规划上具备更多的柔性，与当前企业小批量，多品种和单件定制化的生产模式更加匹配，更能满足市场动态多变的产品需求。目前对柔性生产调度问题的算法研究已有一定的成果，但在在全球化背景下，企业所处的生产环境充满动态不确定性，如紧急订单、订单撤销、原料紧缺、设备故障等多种异常因素，严重干扰企业正常的生产调度[5]。其次企业为扩大生产规模，倾向于在各个地区建立加工工厂，如何综合利用企业分散的制造资源和技术对提高企业的生产效益和效率是非常关键的[6,7]。因此面对企业所处的动态复杂的分布式制造环境，若仅仅通过对传统调度理论和算法进行优化，是不足以满足企业对各种异常因素的快速响应需求的，研究人员需要从生产系统的角度出发，集成企业生产管理的各个环节，对系统的功能进行合理的划分，设计满足实际生产需求的灵活系统架构，保证企业生产调度的稳定高效进行。

传统的车间调度系统属于集中控制式的静态调度系统[8]，特点是制造资源集中、生产规模庞大、时间跨度长等，主要面向种类单一、工艺结构简单的产品制造，该形式的调度系统简单地把管理、控制、监控等功能交由人力负责，因此容错能力和稳定性低，当生产过程中发生如原料短缺或设备故障等情况时，往往需要对整个生产线进行停工处理，显然集中控制式的静态调度系统是无法满足现代制造业的要求的。近年来，多 Agent 理论作为人工智能领域的研究热点，被广泛应用于大型复杂系统的构建，通过规划系统内各 Agent 的功能，协调 Agent 间的通信交互，能够快速灵活地对复杂问题进行求解，本文针对传统调度系统的不足，应用多 Agent 理论，开展对柔性生产动态调度系统的研究，该研究对现代制造业具有以下现实意义：

（1）实现企业生活管理的信息化和自动化，提高生产效率。本文立足于柔性生产调度问题，应用基于改进蚁群算法的柔性作业车间调度算法模型，同时集成企业在数据管理、生产监控等各个环节，降低对人力的依赖，为企业实现自动化的生产管理流程提供了大力支持。

（2）提高企业对分布式资源的利用率，增强企业生产效益。企业的制造资源和技术具有分布式特征，本文在生产调度过程中，依据企业在各地区工厂的资源配置特点进行生产任务的规划，降低企业生产设备的闲置率。

（3）提高企业对动态不确定性的响应能力。动态调度系统在面临紧急订单、订单更改、订单撤销以及设备故障等异常状况时，能够通过系统内部各模块间的高效通信，及时反馈异常信息，同时快速给出异常调度方案，减少对原生产计划的影响，保证系统的稳定性和健壮性。

1.3 国内外研究现状

1.3.1 柔性作业车间调度问题的研究现状

在世界经济飞速发展的大环境下，用户需求复杂多变，产品的更新换代尤为频繁，制造业在生产调度中完成了从上世纪五十年代的单机、并行机调度到如今流水车间和作业车间调度的转变，其中柔性作业车间调度越来越受重视，这是由于工序存在机器选择，弱化了生产约束，提供了可选的加工路径，在面对客户多变的订单需求时，拥有更宽泛的生产灵活性，更适用于如今小批量多品种的生产模式。另一方面，由于约束更少，扩大了解空间的范围，提高了问题求解的难度，因此 FJSP 是比传统车间调度问题更复杂的 NP-Hard 问题[9]，传统车间调度问题所使用的如分支界定法、整数规划、拉格朗日松弛法等精确算法已无法适用于 FJSP 的求解，取而代之的是局部搜索和人工智能等新型优化方法，如局部搜索法中使用非常广泛的禁忌搜索、模拟退化和差分进化法等，以及人工智能算法中的遗传算法、蚁群算法、神经网络、免疫算法等。这些方法在求解复杂的大规模问题中拥有更好的性能，通常会结合各种算法来进行多优化目标的 FJSP 求解，这样能避免单一算法的缺点，综合各类型算法的长处，以得到更优的调度结果。

随着柔性作业车间调度使用越来越广泛，FJSP 一直备受国内外研究人员的关注。汪俊亮等在对加工时间不确定的 FJSP 研究中，以最小化最大完工时间为目标，设计了一种分阶段的遗传算法，配合顺序搜索机制加快算法的求解速度，最终通过仿真实验验证了算法的鲁棒性[10]。Zhang 和 Maier 根据 FJSP 的特点以及工件运输成本的问题，提出一种融合遗传算法和禁忌搜索的改进算法[11]。王万良等提出了一种改进的差分进化算法，该算法对柔性生产工件分批处理，在关键路径中使用随机搜索和自适应变异相结合的染色体编码方式来淘汰无用个体，加快收敛速度，最后通过仿真实验证明了算法能够有效缩短生产周期[12]。刘韵等为提高 FJSP 的求解速度以及质量，在分析禁忌搜索算法和遗传算法的优点以及不足后，在两种算法的结合基础上提出一种元启发式求解算法，并在大规模作业求解中验证了算法的可行性[13]。王雷等在 FJSP 中添加 AGV 运输约束，设

计了一种改进的遗传算法，该算法使用多段式编码以及定义了改进的交叉变异形式，淘汰无用基因，提高算法的进化速度[14]。Mosle 和 Mahnam 提出了一种应用于多目标 FJSP 的改进的粒子群优化算法，并融合局部搜索方法，获得了均匀分布的 Pereto 解集[15]。赵博选构建了一种两阶段混合 Pareto 蚁群算法应用在多目标 FJSP 求解中，该算法在第一阶段进行工序任务的机器选择，第二阶段对第一阶段的结果进行评估后再执行工序的排序操作，在仿真实验中证明了算法能够降低调度问题的规模和复杂性[16]。Rossi 在 FJSP 的求解中，当蚁群陷入局部最优或过早收敛时，通过破坏路径信息素浓度以及通过路由调整蚂蚁的转移方向[17]。Huang 在蚁群路径搜索过程中根据当前最优解和全局最优解的差异，动态调整信息素挥发比例以及信息素释放的浓度，在多目标 FJSP 仿真实验中证明了算法能有效防止蚁群停滞和早熟[18]。Chiang 等提出了一种简单高效的改进多种群进化遗传算法应用于多目标 FJSP 中，该算法只需设置两个参数，通过有效的基因操作来保持种群的多样性，并在 15 个基准实例的测试中对算法进行了验证[19]。

1.3.2 多 Agent 生产调度系统的研究现状

车间调度系统能有效协调生产管理的各个环节，对保证企业生产制造的稳定高效进行有着重大意义。调度系统需要统筹兼顾生产计划、库存管理、资源分配、人员调度、环境监控等诸多流程，涉及面广、难度高，因此制造企业对调度系统的灵活性和健壮性有很高的要求。分布式多 Agent 系统（Multi-Agent System，MAS）通过 Agent 的自治能力以及相互之间高效灵活的通信，非常适用于复杂大型系统的构建，因此国内外众多学者尝试把多 Agent 理论应用到调度系统中，并取得了一定的成果。

Pereira 在多 Agent 调度系统的构建中，为每一个 Agent 定义自我优化的目标，指定了 Agent 间的竞争协商规则来达到生产目标的优化[20]。Madureira 将一种基于元启发式的参数自配置方法应用在多 Agnet 调度系统的学习模块中，提高系统的决策管理能力[21]。Gozzi 在调度系统中应用了多 Agent 理论，通过 Agent 的相互协作来应对各种动态干扰事件的发生[22]。Lim 构建了基于多 Agent 的分层控制式的生产系统，通过层次间 Agent 的管理和监控，设计了分层迭代的通信机制实现了系统通信和管理的高效性[23]。Adhau 使用 MAS 构建了分布式的调度系统，定义了 Agent 间的组合拍卖协商模型[24]。杨陇苗使用 JADE 实现了基于多 Agent 的 MES 系统，该系统应用 PASSI（a Process for Agent Societies Specification and Implementation）方法对 Agent 进行建模，并建立了系统的多层分布式递阶控制结构，在该基础上对多个经典调度实例进行了验证，证明了系统的实

用性[25]。王芊紫在混合流水车间调度系统的构建中定义了管理 Agent、工件 Agent、策略 Agent 和机器 Agent 等四类 Agent，并将差值排序算法封装在策略 Agent 中，同时定义了 Agent 间的协调机制实现系统的动态调度[26]。宋娟根据生产车间面临的各种动态异常因素，结合 MAS 和 CNP（Contract Net Protocol）协议，构建了分布式的多 Agent 制造系统，并通过对中小规模的机械加工实验进行仿真，验证系统的有效性[27]。任海英[28]针对 FJSP 中生产环境的各种干扰信息，构建了基于多 Agent 的预先/重调度系统，该系统定义了 Agent 间的通信协商机制来应对各种动态干扰，使用遗传算法作为系统的调度算法并完成了系统的开发，通过仿真实验验证了系统动态调度的优越性。

由以上研究现状可看出，在柔性生产调度的研究中，基于 MAS 的柔性生产调度系在集成高效的调度算法模型的基础上，能够统筹兼顾企业在生产管理、数据管理、人员管理、生产监控等各方面的需求，因此在实际使用中比单纯的算法优化具有更高的实用价值。本文把多 Agent 理论应用在柔性生产动态调度系统的研究中，根据生产管理中各流程的结构和功能，把其中的物理实体或算法逻辑封装为具有不同功能和结构的 Agent，构建多 Agent 系统，通过 Agent 之间的交互协调，共同完成各项任务，更好地满足企业对灵活性、自适应性、分布式上的需求。

1.4 论文的主要研究内容和组织结构

本文在结构上分为六章，各章内容如下：

第一章介绍了论文的研究背景、目的和意义，展开了对柔性作业车间调度问题和多 Agent 车间调度系统的国内外研究现状，同时概括了本文的主要研究内容。

第二章先是介绍了关于 Agent 与多 Agent 系统的相关理论，其次基于柔性生产动态调度系统的实际需求，对系统的功能进行划分，从而设计了包括管理 Agent、资源 Agent、算法 Agent、工艺 Agent 和监控 Agent 等五类 Agent，最后根据各 Agent 的功能职责以及系统在通信效率上的需求，设计了多 Agent 系统的分布式总体结构。

第三章设计了多 Agent 柔性生产动态调度系统中 Agent 的协作机制，定义了各类 Agent 间信息的传递方向以及协作策略，包括跨区域生产任务分解策略以及异常调度策略。前者根据产品工艺特点，设计了生产任务的模型，实现了任务在多个资源 Agent 组间的任务分解流程，提高企业的资源利用率和生产效率；后者通过系统内各 Agent 的相互配合，完成系统对订单撤销、紧急订单和设备故障等各种异常状况的快速响应。

第四章提出了改进的蚁群算法在柔性生产调度系统中的应用。根据柔性作业车间调

度问题的特点以及基本蚁群算法收敛速度慢和容易陷入局部最优的缺点，基于最小化最大完工时间的优化目标，并提出了 FJSP 中改进蚁群算法的数学模型以及执行流程。

第五章开发了基于多 Agent 的柔性生产动态调度系统。该系统用于为制造企业提供信息管理、资源管理、工艺管理以及生产计划制定等功能接口。本章根据系统的结构模型设计了系统的软件构架。使用基于 TCP/IP 的 Socket 通信机制以及 Java 的面向对象和多线程技术，设计了各个 Agent 的通信模型，保证了系统的通信质量和效率。最后通过编程软件实现了对柔性生产动态调度系统的开发。

第六章验证了基于多 Agent 的柔性生产动态调度系统的可行性，通过系统各模块中 Agent 的相互配合，设计了相关实验并完成了对改进蚁群算法、基于多 Agent 的生产任务跨区域分解策略以及异常调度策略的有效性的验证。

最后是本文的结论，总结了本文的所作的主要研究以及工作，并针对本文调度系统的不足之处进行了展望。

本文的组织结构图如图 1-1 所示。

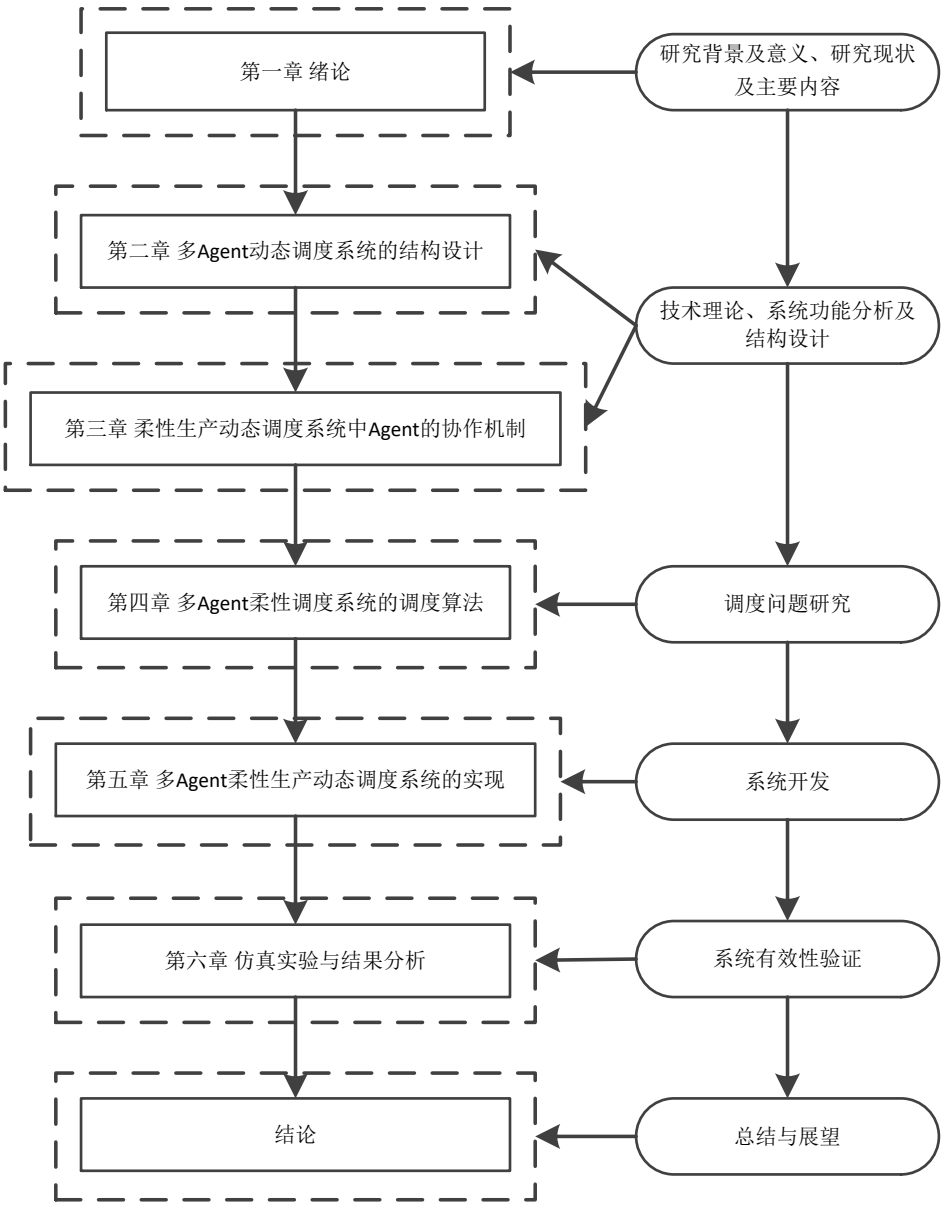


图 1-1 论文的组织结构图

第二章 多 Agent 动态调度系统的结构设计

2.1 Agent 与多 Agent 理论

2.1.1 Agent 的概念及特征

尽管 Agent 在众多研究中应用非常广泛，在理论上已渐趋成熟，但目前对 Agent 的定义尚未有统一的标准。Agent 通常也被称为智能体，属于人工智能领域，因此人们关注的主要是 Agent 的自主性，从这个角度而言，Agent 较为常见的定义是由 Jennings 和 Wooldrige 提供的：Agent 属于由硬件和软件两者综合组成的实体，能够和所处的环境进行信息交换，并能够按照特定目标自主采取相应的行动[29]。Agent 通常也被认为是“可移动的”，能够根据自身的需求并在在外界环境信息的刺激下作出相应的判断，改变自身的位置，还能够影响周边环境以及其他 Agent。尽管现在 Agent 在使用上尚未大范围推广，实际的应用实例也比较少，但是 Agent 具有大家所公认的以下一些特点[30]：

（1）自主性：拥有支撑自身行为的决策库，具备学习能力，可根据与周边环境的接触扩充、修改自身的决策库，根据决策规则选择对应的行为，可连续动态地改变自身的状态。

（2）社交性：Agent 之间能相互通信，具有通信标识和通信语言，遵循特定的通信规则可相互交换信息。

（3）反应性：能够时刻感知来自外界的信息而无需特定的信号进行唤醒，能够分析信息所表达的意图和价值，把有用的信息扩充自身数据库。

（4）能动性：不仅能接收外界的信息，同时还能够自主地发起对周边环境的探索，通过主动获取环境信息以及接收其他 Agent 的信息更改决策库以及自身状态。

根据以上 Agent 的描述，可见 Agent 具有学习能力，能够根据自身的目的获取、交换信息，从而更改自身的状态，包括决策库的决策规则、数据库信息、地理位置等等；同时还具有可移动性，在分布性和扩展性的环境中易于迁移[31]，因此由若干 Agent 组成的多 Agent 系统非常适合用于部署分布式系统，对于分布式系统中的各个功能模块或子系统的功能，可以交由单独的 Agent 或者由若干功能相近的 Agent 组成的 Agent 组负责，依靠系统内 Agent 的信息交互，来协同完成各项任务。

2.1.2 多 Agent 系统的概念及特征

多 Agent 系统 MAS 是由 Agent 组成的具有一定结构和功能的分布式系统。其中的 Agent 需要承担系统中的部分功能，功能相似或相同的 Agent 可组成一个 Agent 组。系统需指定 Agent 间的通信协议和通信方式。

MAS 能够对复杂问题进行分解，分配到相关的 Agent，各 Agent 根据自身的计算能力求解子问题，通过 Agent 间的通信进行交互协调，最终综合各个子问题的求解结果，获取对复杂大型问题的完整解决方案。与传统集中控制式、递阶控制式[32]的系统而言，多 Agent 系统部署和维护所需要的成本更低，由于系统将计算能力和资源分散到一组 Agent 中，单一的 Agent 在子问题的计算中对硬件性能的要求较低，减少了对高性能的大型服务器的依赖。同时，Agent 能独立于中央服务器运行，能并行处理消息请求以及进行问题求解，因此能有效提高系统整体的稳定性和效率。

Agent 的自主性、反应性、能动性等决定了多 Agent 系统具有更强的灵活性，对外界的变化更为灵敏，在环境动态变化频繁的场景中具有更大的优势。Agent 作为分布式系统中的节点，由于其具有可移动性，因此对于节点的增删处理反馈更及时，保证系统的高可扩展性，再者 Agent 的学习能力使其能够调整自身的状态和行为，随着时间的推进，可以优化求解过程、提高系统整体的求解质量，这相当于延长了系统的生命周期，减少人工干预、维护的成本。由此可见,MAS 非常适用于动态生产调度系统的构建，通过 MAS 对外界信息的灵敏反应以及内部 Agent 间的强大的问题求解能力，可以很好地满足调度系统对动态变化的客户需求和生产环境中各种异常因素的快速响应。

2.2 动态调度系统的功能分析

2.2.1 制造资源的分布式特点

制造资源在定义上可以分为广义和狭义两类[33]，广义的制造资源可分为物料资源、设备资源、人力资源、技术资源、运输资源等，其中物料资源表示生产加工中所需的各种原材料；设备资源表示各种加工设备；人力资源表示车间管理员、维修人员、运输工人等；知识资源表示与生产管理相关的专业知识，如产品的工艺信息、生产优化技术、环境监测技术等；运输资源表示用于在各制造单元和工厂间搬运的运输工具。狭义的制造资源专门指设备资源和物料资源，本文的制造资源属于狭义的定义。

制造全球化使我国企业面临激烈竞争的同时，也为其带来了更广阔的市场，因此现代制造业具有分布式加工的特点，倾向于在各个地区建立工厂，每个工厂拥有各自的车间结构和制造资源配置，提供对应的生产服务。本文根据企业分散式资源的异地性分布结构[34]，把每个工厂封装为一个独立的资源模块，当企业收到客户的产品订单时，以网路为媒介，获取各资源模块的加工能力，进行生产任务的分解分配流程，最终从各模块的调度结果中筛选满足客户需求的最优调度方案。通过这种方式，能够把企业分布在不同地区的资源进行合理的规划，降低设备的闲置率，提供企业的生产效率。

2.2.2 系统的功能划分以及 Agent 分类

生产调度系统最基本的功能在于满足生产任务在时间和经济效益上的需求，尽可能地给出最优的调度方案，并按照方案把任务安排到对应的加工设备上。目前市场需求动态变化、商品更新迭代的周期不断缩短，制造环境也充满诸多不确定性，因此系统的结构要能够兼顾灵活性和稳定性，当生产计划发生变化时，能够动态调整系统结构；当发生各种异常情况时，系统结构中的各单元能够灵活配合，快速反馈。

完整的车间动态调度系统应当具备以下功能：一是能够接收并管理来自客户的订单任务，包括对任务合法性和优先级的判定；二是对订单任务进行分解分配，把任务分解成工件的工序集合，根据工序在机器选择和加工路径的柔性特点计算调度方案，并向生产设备下达加工任务；三是能够快速响应如订单撤销、紧急订单、设备故障等异常因素，开启异常调度[35]，尽量降低原生产计划对交货期、加工成本和设备负荷的影响；四是对系统数据的高效管理，包括工艺信息、制造资源信息、历史工作信息、设备故障信息等等。根据以上功能，本文把动态调度系统生产管理涉及的工作流程总结为：订单的输入和处理、产品柔性工艺信息的管理、柔性生产任务规划以及异常调度。对应以上流程，本文基于多 Agent 理论，分别设计了五类 Agent，包括管理 Agent、资源 Agent、工艺 Agent、算法 Agent、和监控 Agent，其中管理 Agent 负责订单任务的处理以及协调系统内的其他 Agent 的通信和行动；资源 Agent 分为车间 Agent 和设备 Agent，设备 Agent 对应系统内的加工设备，负责记录设备的工作信息和加工序列，车间 Agent 负责管理分布其中的设备 Agent。根据企业资源的分布式特点，同属一个工厂的资源 Agent 构成了一个资源模块，封装为资源 Agent 组；工艺 Agent 负责管理系统内产品的工艺信息，为其他 Agent 提供数据查询接口并响应系统管理人员对产品工艺信息的修改；算法 Agent 是对系统与柔性作业车间调度相关的算法的封装，即基于特定的设备集合和任务集合进

行调度方案的计算；监控 Agent 对应系统的监控设备，通过分析监控数据，反馈系统内设备的工作状况，通知系统开启异常调度流程。调度系统把其中的物理实体、数据或算法逻辑封装为以上五类 Agent，构建了多 Agent 动态调度系统，其中的 Agent 各司其职的同时，通过 Agent 间灵活自由的通信，共同完成生产管理的完整流程。

2.3 各 Agent 的功能设计

调度系统中各 Agent 的具体功能设计如下：

管理 Agent：分为全局管理 Agent 和子管理 Agent。全局管理 Agent 相当于系统内部的管理员，协调 Agent 间的通信和任务配合流程。每个 Agent 在启动运行后需在全局管理 Agent 注册自身的通信标识，通信标识是 Agent 间通信和任务协调的重要依据。其次全局管理 Agent 作为系统对外的通信接口，负责接收并审核来自客户的生产订单，根据交货期对订单任务进行优先级排序，协调资源 Agent 和算法 Agent 对任务进行分解分配工作。同属一个地区工厂的资源 Agent 将构成一个资源 Agent 组，而子管理 Agent 是系统分配给一个资源 Agent 组的管理员，作为 Agent 组与全局管理 Agent 通信的主要接口。当子管理 Agent 接收到全局管理 Agent 传递的订单任务后，根据其下车间 Agent 的设备配置和加工能力来判断能否完成该任务。

资源 Agent：资源 Agent 代表系统内的制造资源，分为车间 Agent 和设备 Agent。设备 Agent 对应一个加工设备，记录设备的各项信息，包括设备名称、编号、工作状态、加工能力等；车间 Agent 对应系统内的一个车间，负责对该车间中的设备 Agent 进行管理，当有新设备添加或旧设备删除时，车间 Agent 需要与对应的设备 Agent 建立或移除关联信息。如图 2-1 所示，根据企业制造资源的分布式特点，同属一座工厂内的车间 Agent 和设备 Agent 构成一个资源 Agent 组，由子管理 Agent 负责管理和控制,当企业接受到客户的产品订单时，根据各个资源 Agent 组的生产能力完成订单任务的分解以及资源的分配。

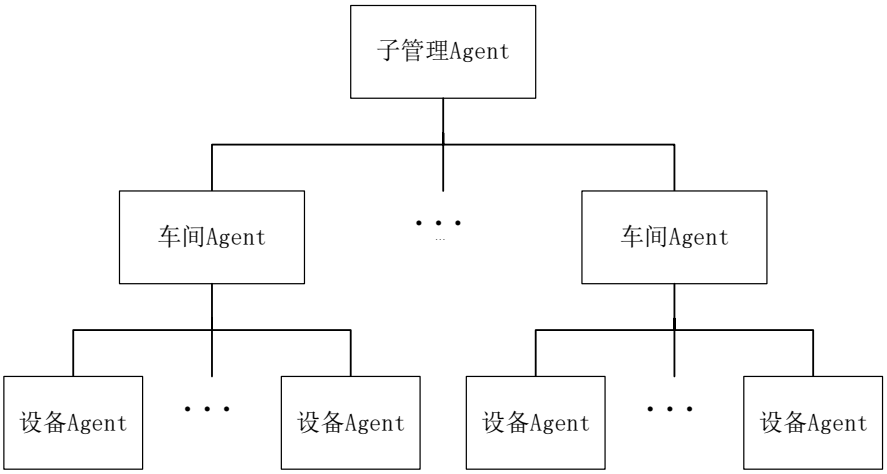


图 2-1 资源 Agent 组的结构

工艺 Agent: 负责对系统的产品工艺信息进行管理，为其他 Agent 提供数据查询服务。该工艺信息用于表示系统所能生产加工的产品种类、产品需要的原材料、产品中各工件的加工工艺流程以及其中各个工序的机器选择等信息。工艺 Agent 负责为全局管理 Agent 审核订单任务提供依据，当任务对设备、原材料、加工精度的要求能够与工艺 Agent 记录的工艺信息一一对应时，表示任务的合法性通过，可进一步进行任务的分解分配。

算法 Agent: 封装了调度系统内的算法逻辑，基于特定的设备集合和生产任务为资源 Agent 组提供调度方案的计算服务。本文的算法 Agent 选择了基于改进蚁群算法的柔性作业车间调度算法。当全局管理 Agent 把生产任务传递到各个资源 Agent 组时，资源 Agent 根据其中设备 Agent 所记录的设备加工能力和加工序列，由算法 Agent 执行调度算法计算调度方案，最终全局管理 Agent 从各个资源 Agent 组返回的调度结果中筛选完工时间最短的调度方案。

监控 Agent:代表系统内的检测设备。监控 Agent 读取监控设备采集的设备工作状态信息，当判定设备故障后，通知故障设备所属的资源 Agent 组，由对应的子管理 Agent 回收分配到故障设备的工序任务，根据任务对设备的要求，重新分配到正常设备的加工序列。

2.4 系统的结构分析

多 Agent 调度系统一般存在三种较广泛使用的系统结构，分别是层次型、分布型和混合型[36]：

（1）层次型：该结构强调的是节点间的从属关系，属于递阶控制式，Agent 间通过分层的形式进行管理，上层 Agent 负责对下层 Agent 进行管理和控制，不同层次的 Agent

间通信是必须逐层传递，位于同一层次的 Agent 之间可平等自由地通信，因此系统主要通过上层 Agent 对下层 Agent 的控制管理来完成任务。该结构具有很多优点，首先是易于扩展，层次的增删或者在某一层次中 Agent 的增删对系统其它部分影响很小。其次易于系统功能的划分和管理，层次间的从属关系能够规范下层 Agent 的行为，相同层次的 Agent 间易于数据共享。但该结构的缺点同样明显，首先是通信效率较低，当系统规模扩大，层次加深时，顶层和底层之间 Agent 的通信需要跨越多个层次，延时明显。而且当同一层次的 Agent 数量过多时，会加重上层节点控制和协调的负担，最严重的问题是，当上层 Agent 失效时，层次间的通信将被阻断，因此一般通过增加功能相近的可替换节点来预防该情况的发生，可以一定程度上提高系统的稳定性。

（2）分布型：该结构中各节点间不存在从属关系，节点间通信不需要经过消息中转，相互间可以平等自由通信，系统通过各 Agent 之间的协调完成任务。对比层次型结构，分布型结构拥有更高的灵活性和扩展性，由于 Agent 间自由通信，信息可直达，因此效率更高，并且不存在由于某一 Agent 失效导致局部或全局通信瘫痪的情况。该结构同样存在缺点，由于不存在 Agent 间的从属关系，因此 Agent 之间数据共享、状态同步更困难，当任务对数据一致性要求很高时，分布型系统效率反而会下降，因此通常会添加专门用于进行数据同步和共享操作的 Agent。再者由于信息在任意的 Agent 之间可达，提高通信效率的同时也可能导致通信风暴，这是由于 Agent 具有自主性和能动性，可以发起对环境以及 Agent 的探测，若不加以约束，可能占用过多的通信带宽，导致网络阻塞。

（3）混合型：通常情况下，可以综合两种结构的优点，兼顾两者在控制管理和通信效率上的性能，因此就有了混合型的多 Agent 系统结构，该结构为功能相近以及依赖数据共享的 Agent 组建 Agent 组，实施局部的控制管理，并且由组内具有管理职能的 Agent 发起对环境的探索，能够约束系统整体的通信量，同时功能职责分明的 Agent 之间不设置从属关系，可自由通信，保证系统的灵活性和高效性。

为了保证生产调度系统生产管理中各流程能够平滑衔接，信息高效传递，本文的调度系统的结构设计采用的是混合型，如下图 2-2 所示。根据现代企业制造资源的分布式特点，同属一个资源 Agent 组的子管理 Agent、若干车间 Agent 以及对应的设备 Agent 以层次型的结构进行呈现。由图 2-2 可知，全局管理 Agent 与各资源 Agent 组存在从属关系，上层 Agent 对下层 Agent 具有控制协调的功能，而全局管理 Agent、算法 Agent、工艺 Agent 和监控 Agent 间是平等的关系，能够自由通信，因此本文调度系统的体系结构属于典型的混合型结构，柔性生产动态调度系统应用该结构具有如下优点：

（1）易于资源管理和系统扩展。资源 Agent 组的结构符合现代制造系统的资源分布特点，制造业的资源结构基本是以“工厂-车间-设备”的方式进行呈现，一个加工工厂内设置若干车间，每个车间根据所生产加工的产品、零件来配备对应的生产设备，而企业可能在多个地理位置分布有加工工厂，制造业资源的这种分布特点是与层次型结构一致的。系统是基于各设备 Agent 的加工序列对订单任务执行分解分配过程的，该过程涉及诸多数据和状态的共享，通过上层 Agent 对下层 Agent 控制管理的能力，能够更有效率协调资源 Agent 间的工作。

（2）系统运行效率高，稳定性强。资源 Agent 组属于层次型结构，由全局管理 Agent 进行统一的管理。全局管理 Agent、算法 Agent、工艺 Agent 和监控 Agent 之间是松耦合关系，属于分布式结构，各 Agent 之间不存在从属关系，消息直接可达，无需经过中央服务器的消息中转，通信效率高，在协同解决大型复杂的调度任务时更灵活高效。

（3）系统的成本降低。车间调度系统把生产管理的各个功能进行划分并分配到对应的 Agent，相当于把系统所需的计算量和控制量分散到各个 Agent 中，降低了系统对大型高性能服务器等硬件的依赖。

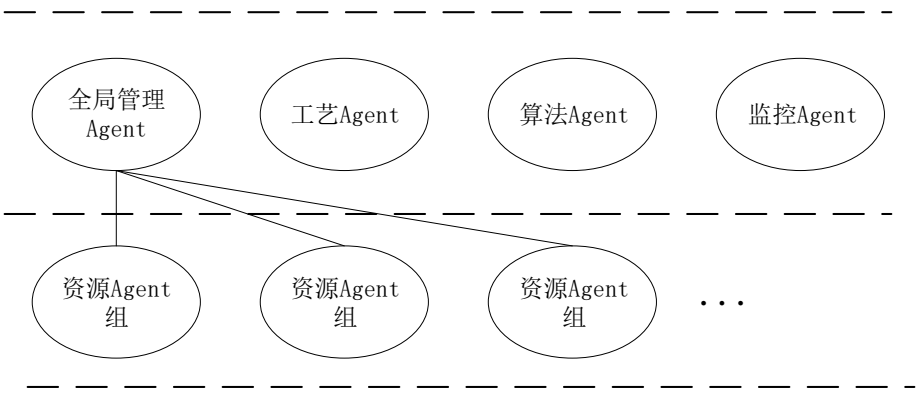


图 2-2 多 Agent 柔性生产动态调度系统的结构

2.5 本章小结

本章首先在 Agent 和 MAS 的相关理论基础上，根据动态车间调度系统的业务需求，进行了基于 MAS 的调度系统中 Agent 的职责划分，其次通过分析各 Agent 中信息的输入和输出特点，以及考虑系统通信的高效性以及管理控制的稳定性，设计了系统的体系结构。通过本章可以对本文的多 Agent 动态调度系统的功能和结构形成基本的认识，为后面系统的实现提供了理论基础。

第三章 柔性生产动态调度系统中 Agent 的协作机制

3.1 柔性作业车间调度问题

3.1.1 问题描述

柔性作业车间调度问题可总结为把 n 个工件分配到 m 台机器上进行加工，每个工件从加工到完成需要经历若干道工序，工序之间有顺序约束，并且每道工序存在一种或多种机器选择，分别对应不同的完成时间，通过定义各工序的机器选择、工序在机器上的加工顺序以及开始加工时间，来达到一个或多个的性能优化目的[37]。在传统的作业车间调度问题中，设备的加工能力、工艺的加工路线确定且唯一，即工序间的加工顺序、工序的加工时间和所需的加工设备类型是确定的。而柔性作业车间调度问题中，设备能对不同的工序进行加工，工序也有多种机器选择。如下表 3-1 呈现了两个工件的工序设备选择以及对应的加工时间。

表 3-1 2 个工件 4 台设备的 FJSP 案例

工件	工序	设备选择和对应的加工时间		
		m_1	m_2	m_3
p_1	$o_{1,1}$	–	5	3
	$o_{1,2}$	9	–	7
p_2	$o_{2,1}$	7	6	-
	$o_{2,2}$	3	–	2

图 3-1 是该调度实例对应的析取图。图中的实线箭头表示的是同一个工件中工序的顺序约束关系，虚线箭头表示不同工件的加工是相互独立的。由于 FJSP 降低了工序对机器的选择约束，当工件数目或工序的设备选择增多时，对应析取图的模型将更加复杂，对应更大的解空间，这时候需要的计算量将急剧增加，因此 FJSP 是比传统的作业车间调度难度更大的 NP-Hard 问题。

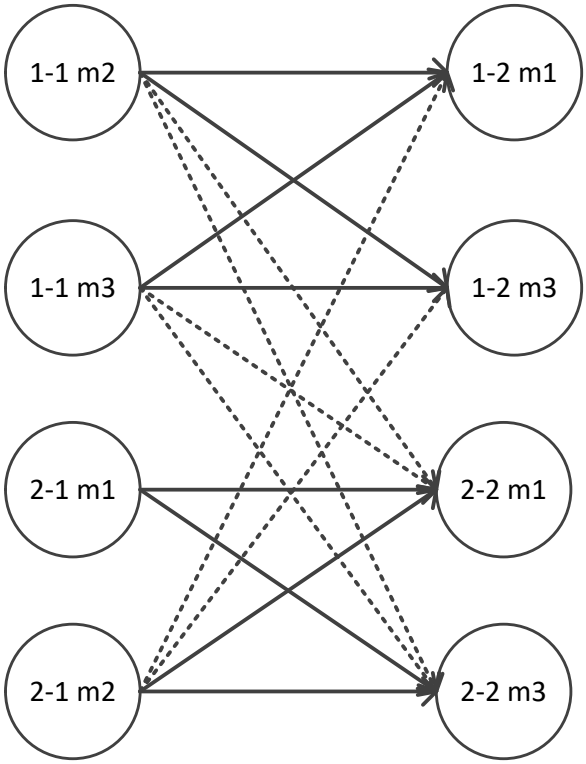


图 3-1　2 个工件 4 台机器的 FJSP 实例析取图

3.1.2 FJSP 的数学模型

柔性作业车间调度问题的研究中通常作出以下几点假设[38]：

（1）每台设备在任一时刻只能加工一道工序。

（2)每个工件在任一时刻只能有一道正被加工的工序,且工序加工过程中不得中止。若由于设备故障等原因导致必须中止加工，则工件报废，需重新安排另外的同种工件进行加工。

（3）不同工件间的加工互不影响，工件与工件之间没有顺序约束，不同工件间的工序之间也不存在顺序约束。

（4）初始时刻各设备为空闲状态，均可投入生产。

（5）忽略工件在设备间的转移时间。

本文对 FJSP 中的数学模型中各个变量的定义如下：

（1）n为工件数量

（2）m为设备数量

（3）P = $\{p_1, p_2, \dots, p_i, \dots, p_n\}$为工件集合， p_i表示第 i 个工件

（4）M = $\{m_1, m_2, \dots, m_j, \dots, m_m\}$为设备集合，$m_j$表示第 j 个设备

（5）$O_i = \{o_{i,1}, o_{i,2}, \ldots, o_{i,k}\}$ 为工件 i 的工序集合，$1 \leq i \leq n$，其中$o_{i,k}$表示工件 i 的第 k 道工序

（6）$TS_i = (ts_{i,1}, ts_{i,2}, \ldots, ts_{i,h})$ 为设备 i 的加工工序序列，$ts_{i,h}$表示安排在设备 i 上的第 h 道工序任务

（7）$MC_{i,j} = \{mc_{i,j,k}\}$为工序$o_{i,j}$的机器选择集合，$mc_{i,j,k}$表示工序$o_{i,j}$可在设备 k 上进行加工

（8）$s_{i,j,k}$为工序$o_{i,j}$在设备 k 的开始加工时间

（9）$e_{i,j,k}$为工序$o_{i,j}$在设备 k 加工完成时间

（10）$t_{i,j,k}$为工序$o_{i,j}$的在设备m_k上的加工时间

（11）c_i为工件 i 的完工时间

（12）d_i为工件 i 的交货期

FJSP 约束条件的数学表达如下：

（1）工件相邻工序间存在顺序约束，若$o_{i,j}$为$o_{i,j+1}$的前置工序，则有

$$s_{i,j+1,k} \leq e_{i,j,k'} \tag{3-1}$$

（2）设备需要按照加工序列的顺序对工序进行加工。若工序$o_{i,j}$和工序$o_{i',j'}$为设备 k 加工序列中的相邻工序，则有

$$e_{i,j,k} \leq s_{ii',j',k} \tag{3-2}$$

（3）设备在加工某道工序的过程中不得中止，必须在加工完一道工序后再加工下一道工序：

$$t_{i,j,k} = e_{i,j,k} - s_{i,j,k} \tag{3-3}$$

（4）工序在加工过程中不得转移到另外的设备：

$$\sum_{k=1}^{m} x_{i,j,k} = 1 \tag{3-4}$$

$$x_{i,j,k} = \begin{cases} 1, & o_{i,j}\text{安排在设备 k 上加工} \\ 0, & otherwise \end{cases} \tag{3-5}$$

FJSP 的优化目标通常是对完工时间、机器负荷、拖期时间和加工成本中的一个或多

个的结合。本文在 FJSP 中的调度算法的优化目标设置为最小化最大完工时间，因此目标函数如下：

$$f = \min(\max_{1\le i\le n}(c_i)) \tag{3-6}$$

3.2 基于多 Agent 的生产任务跨区域分解策略

在全球化背景下，制造企业在管理上具有社会性，在生产上具有异地性，倾向于调整资源结构和技术分布，在世界各地均设有加工工厂，各工厂根据自身的技术和设备配置生产对应的产品。当企业接收到客户的生产订单时，根据生产任务的需求以及各个工厂的生产能力决定任务的分配，若单独的工厂无法完成完整的生产任务，则企业需将任务作进一步的分解，对于其中的每一个子任务，重新进行任务的分配，从各工厂中筛选完工时间最短的调度方案。本文以通信网络为媒介，通过全局管理 Agent 对多个资源 Agent 组进行管理和控制，完成生产任务在分布式工厂车间中的分解分配流程。

3.2.1 生产任务的形式化描述

在本文的调度系统中，客户的生产订单通过合法性判定后将被封装为对应的生产任务 MT（Manufacturing Task），MT 可分为三种粒度的任务，分别是产品级任务 PRT（product task）、工件级任务 PT（part tast）和工序级任务 WPT（working procedure task），与产品的构造相似，一个产品由若干工件组成，一个工件由一组工序按照工艺约束组成。PRT、PT、WPT 分别对应产品、工件和工序。工艺 Agent 中所记录产品、工件和工序在指定生产约束后均可封装为对应的 PRT、PT 和 WPT。

本文对 MT 的形式化描述如下：

MT=<MTInfo, Constraint, Construction, Resource>

MTInfo(Manufacturing Task Information)描述了生产任务的基本信息,使用一个四元组<ID,NAME,TYPE,DESCR>表示，其中的 ID 是 MT 的唯一标识；NAME 是 MT 的名称，概括了生产对象的基本特征；TYPE 指定了 MT 的粒度，即 MT 属于 PRT、PT 或 WPT。DESCR（Description）是对生产对象的详细描述，如 PT 中 DESCR 描述了工件的精度和规格型号等信息，WPT 中描述了工序加工的注意事项等。

Constraint 描述了 MT 的生产约束，使用三元组<TIME，COST，QUALITY>来表示，其中 TIME 为 MT 对完工时间，即交货期的约束；COST 为 MT 对生产成本的约束；

QUALITY 为 MT 对生产质量的约束，如工件在规格、精度上的要求。由于 PT 是 PRT 的子任务，WPT 为 PT 的子任务，因此 TIME 由上级任务指定。

Construction 描述了生产任务的构成以及子任务的约束关系，例如 PRT 由若干 PT 组成；CPT 由一组 WPT 按照一定的约束关系组成。Construction 可以描述为：

Construction=<COMP,PROCESS>

$COMP=\{MT_1, MT_2,..., MT_i, ..., MT_n\}$

COMP（Composition）是生产任务 MT 的子任务集合，其中 MT_i 是 MT 的第 i 个子任务。当 MT 的粒度为 WPT 时，任务无法再分解，因此 COMP 为空。PROCESS 是 MT 的生产工艺，呈现了子任务完成的顺序约束，例如当 MT 的粒度为 PRT 时，PROCESS 指定了 PT 的组装顺序；当 MT 粒度为 PT 时，PROCESS 指定了工序级子任务加工的顺序约束以及机器选择。

Resource 描述了生产任务 MT 对资源的需求，表示如下：

Resource=<M,HR,E,T>

其中 M（Material）表示生产任务 MT 对原材料的需求；HR（Human Resource）表示 MT 在生产过程中对技术人员的需求；E（Equipment）表示可用于完成 MT 的设备序列；T（Time）为加工时间序列，序列中的元素与 E 中元素一一对应，表示每一个机器选择对应的加工时间。

3.2.2 基于多 Agent 的生产任务跨区域分解流程

目前主流的生产任务的调度规划通常是根据任务对设备、原材料、精度、尺寸等要求以及工件的工艺约束，建立数学模型，使用如局部搜索、人工智能算法等进行单目标或多目标的调度方案求解，以上的调度过程是基于资源集中配置的前提下进行的，即资源的分布局限在一间工厂的一个或多个车间，因此对于资源具有异地性的企业通常无法适用。

本文的基于多 Agent 的生产任务跨区域分解策略的中心思想是把任务的分解和分配过程进行一定程度的分离，首先全局管理 Agent 以网络为媒介，把生产任务 MT 广播至各个资源 Agent 组，资源 Agent 组和算法 Agent 相互协调，根据设备配置以及加工能力计算 MT 对应的调度方案；而后全局管理 Agent 从各个资源 Agent 的调度结果中筛选最优的调度方案，若不存在资源 Agent 组独立完成生产任务，则全局管理 Agent 需根据 MT 的粒度进一步分解，对于每一个子任务重新执行跨区域分解策略。在该过程中，全局管

理 Agent 与各个资源 Agent 组构成了分层式的结构，每个资源 Agent 组可视为一个子调度系统，分别计算并返回由全局管理 Agent 分配的生产任务的调度方案，

本文的基于多 Agent 的生产任务的跨区域分解策略的执行过程遵循以下原则：

（1）生产任务 MT 的分解顺序需符合产品结构，按照产品级任务 PRT、工件级任务 PT、工序级任务 WPT 从上往下逐层分解，其中 PRT 是若干 PT 的集合，PT 是若干 WPT 的集合。

（2）生产任务 MT 分解过程中的传递顺序需符合全局管理 Agent 与资源 Agent 组之间的分层式结构，即按照全局管理 Agent、子管理 Agent、车间 Agent 的顺序从上往下逐层传递。

（3）MT 的粒度与资源 Agent 组的结构需一一对应。PRT、PT、WPT 分别与子管理 Agent、车间 Agent、设备 Agent 对应。当 MT 粒度大于 Agent 级别时，资源 Agent 需对 TM 进一步分解；当 TM 粒度小于自身粒度时，直接把 TM 发布至其下的各个资源 Agent；当两者级别相同方可进行能力判定，即判断自身的设备配置能否满足 MT 对制造资源的需求。

（4）尽量保证生产任务 MT 的粒度。随着生产任务 MT 分解程度的加深，最终参与分解的任务粒度将越来越小，极端情况下会导致一个工件级别的生产任务 PT 包含的多个 WPT 被分配到不同地区的生产设备上，在实际生产中，这需要把工件在多个地区的工厂车间之间进行转移，将大幅提高了各种费用和成本。因此，当某个粒度的任务分解所得到的设备组合能够满足任务的交货期时，不再进行进一步的分解。其次，除非征得客户的同意，传递到各个资源 Agent 组的生产任务的最小粒度为工件级任务，即同属一个工件的工序任务不可被分散到不同地区的工厂中。

基于多 Agent 生产任务跨区域分解策略的执行过程中，各资源 Agent 组的协商目标为最小化生产任务 MT 的完工时间，对该目标的数学表达式以及相关的数学符号说明如下：

MT为全局管理 Agent 接收的来自客户的生产任务

n为生产任务 MT 的子任务数量

m为全局管理 Agent 其下的资源 Agent 组数量

$\mathrm{COMP} = \{MT_i\}$为生产任务 MT 的子任务集合，$1 \le \mathrm{i} \le \mathrm{n}$

$t_{i,j}$为子任务MT_i在第 j 个资源 Agent 组的预计完工时间，$1 \le \mathrm{j} \le \mathrm{m}$

t_i表示工件 i 的完工时间，对于一个工件任务 i，每个资源 Agent 组均返回一个预计完工时间$t_{i,j}$，全局管理 Agent 最终把工件分配到完工时间最短的资源 Agent 组中，即有：

$$t_i = \min_{1\le j\le m}\left(t_{i,j}\right) \tag{3-7}$$

各资源 Agent 组的协商目标为最小化生产任务 MT 的完工时间：

$$f = \min\left(\max_{1\le i\le n}(t_i)\right) \tag{3-8}$$

在最小化生产任务 MT 的完工时间的共同目标下，生产任务跨区域分解策略中各 Agent 的协作流程如下图 3-2 所示，可以细分为以下步骤：

（1）封装生产任务 MT。全局管理 Agent 接收到客户的生产订单后，进行订单合法性判定。合法性通过后，根据产品结构按照工序、工件、产品的顺序从下往上进行封装，得到生产任务 MT。

（2）任务发布。全局管理 Agent 把 MT 广播至其下各个资源 Agent 组。

（3）资源 Agent 组内进行粒度判定以及任务分解。上层 Agent 接收到 MT 后，若 MT 粒度与 Agent 级别相同，进行能力判定，执行步骤（4）；若 MT 粒度小于 Agent 的级别，直接把 TM 广播至其下各个子 Agent，由子 Agent 进行能力判定；若 MT 粒度大于资源 Agent 的级别，需对 MT 的子任务集合 COMP 进一步分解，得到粒度更低的若干子任务，并逐一把子任务广播至其下的子 Agent，即执行步骤（2）。

（4）能力判定。当资源 Agent 的级别与 MT 粒度相同时，进行能力判定，即根据自身的原材料和设备配置与 MT 对资源的需求进行对比，若满足需求执行步骤（5），否则向上层 Agent 返回 MT 无法完成的结果，执行步骤（6）。

（5）计算调度方案。资源 Agent 把 MT 和设备集发送至算法 Agent，由算法 Agent 基于 MT 的工艺约束和设备 Agent 记录的加工序列计算调度方案，并返回到上层 Agent。

（6）结果筛选。上层 Agent 收集到下层子 Agent 对 MT 的判定结果，分两种情况进行处理：

a）所有子 Agent 均返回无法完成的判定结果。若 MT 的粒度可进一步分解，则对分解得到的子任务集，逐个发布到下层子 Agent，即执行步骤（2）；若 MT 无法再分解（粒度为 WPT），则向上层 Agent 返回无法完成的判定结果，若此时 Agent 位于最上层，即为全局管理 Agent，表明系统无法完成任务，向客户返回判定结果，结束分解流程。

b）存在子 Agent 能够完成任务，从返回的结果中选择用时最少的调度方案，返回到

上层 Agent，若 Agent 为全局管理 Agent，进行交货期的判定。若该方案满足任务的交货期，则结束分解流程，该方案作为最终的调度方案；若该方案不满足 MT 的交货期且 MT 可进一步分解，分解任务后执行步骤（2），若 MT 不可分解，表明系统无法完成任务，向客户返回判定结果，结束分解流程。

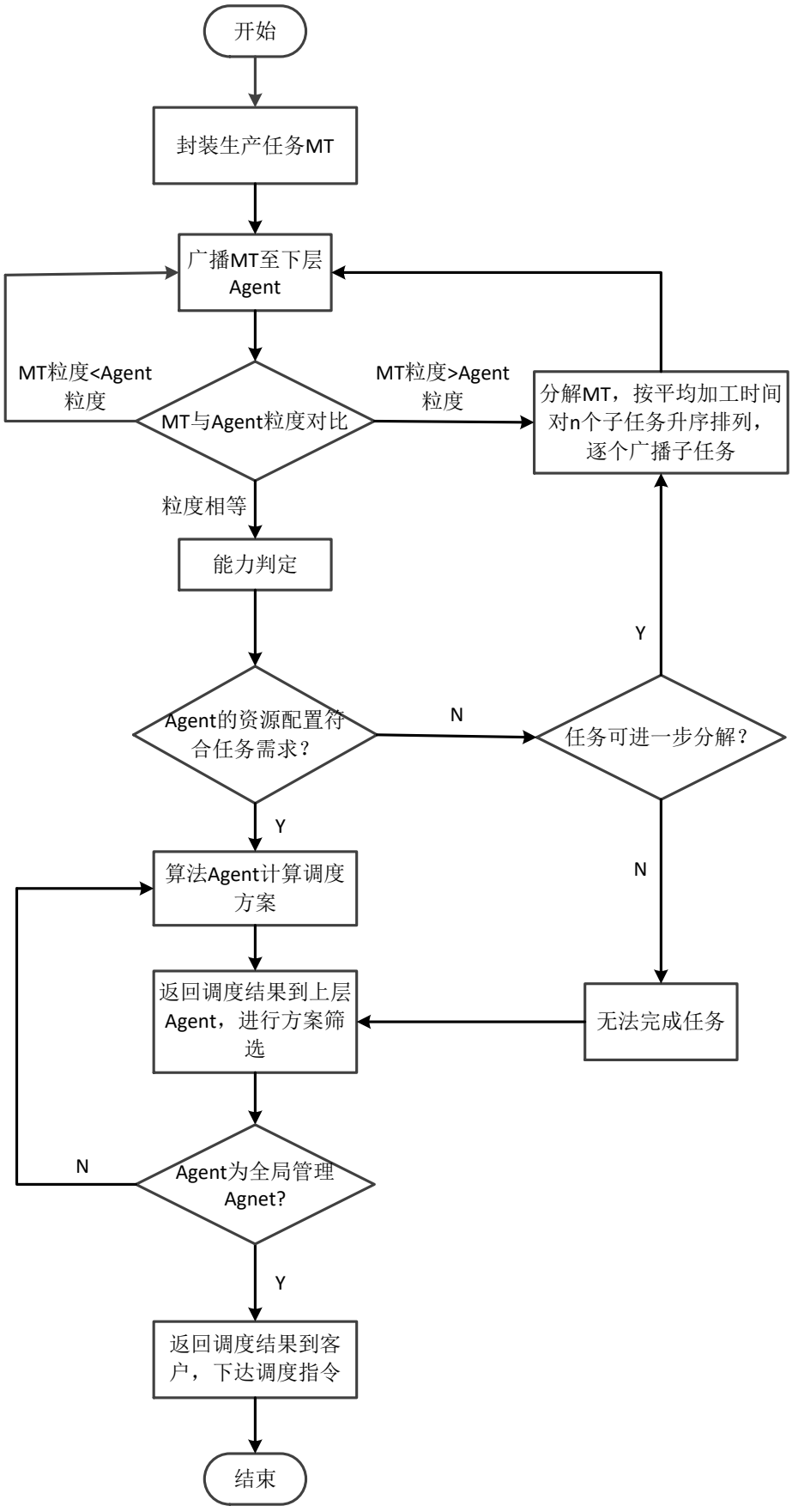


图 3-2 生产任务跨区域分解流程

3.3 基于多 Agent 的异常调度策略

当生产过程出现异常因素时，系统需要及时。快速地进行响应并采取对应的措施，进行调度方案的调整，即开启异常调度过程。系统面临的异常因素通常包括设备故障、订单取消、紧急订单、原材料短缺等等。由于全局管理 Agent 在订单任务审核中对任务所要求的原材料和系统库存原材料进行比对，因此本文主要对设备故障、紧急订单以及订单取消三种情况的异常调度策略进行说明。

3.3.1 紧急订单条件下的异常调度策略

当生产订单被标记为紧急订单时，对应的生产任务拥有最高优先级，本文把该最高优先级的作用范围分为两种，即局部最高优先级和全局最高优先级，系统根据两种优先级从而采用以下对应的异常调度策略：

（1）局部最高优先级是指紧急订单对应的生产任务在全局管理 Agent 的任务队列中具备最高优先级。紧急任务被插入到任务队列的最前端，按照生产任务跨区域分解策略的执行流程完成紧急订单在分布式资源的分配，该过程是基于系统内设备 Agent 原有的加工序列进行的，不对原生产计划造成影响。

（2）全局最高优先级是指紧急订单在系统所有生产任务中具有最高优先级，包括已安排在设备加工序列中的生产任务。该情况下系统先按照局部最高优先级来对紧急任务进行分解分配，若此过程得到的调度方案满足紧急订单的交货期，表明能够兼顾紧急任务和原生产计划的需求。若无法满足，对系统中所有设备 Agent 所记录的加工序列进行备份后清空，而后系统在各设备空闲的前提下进行生产任务的跨区域分解流程，完成紧急订单的任务分配，而后对设备 Agent 原有的工序任务的备份按照相对顺序放回到对应设备的工序序列中。

3.3.2 设备故障条件下的异常调度策略

当设备由于零部件磨损、断裂等导致设备故障无法运行时[39]，将导致原本安排给该设备的生产任务无法进行加工，同时由于工件中工序存在顺序约束，同属一工件的安排在正常设备上的工序任务也无法按照原生产计划进行，因此必须调整原有的调度方案。

在对设备故障过程中作出如下设定：

（1）工序加工过程不可中断。若当设备正在加工某道工序时发生故障从而必须停工

时，则对应的工件报废，不可继续加工，需安排工件重走完整的加工流程。

（2）工件任务由所属地区的工厂优先处理。当工件任务执行过程中由于设备故障导致后续工序无法按照原本的分配方案加工时，先查看该工件所处工厂的设备配置是否满足后续工序的机器选择，若满足，把工件转移到替换设备上加工，否则交由全局管理 Agent 重新进行工件任务的跨区域分解分配流程。

（3）若异常调度后所得的若干调度方案均无法满足生产任务的交货期，从中选择拖期最小的方案，并提交到客户说明原因。

设备故障处理的异常调度策略执行过程如下：

（1）监控设备检测到设备故障，对应的监控 Agent 读取故障信息并进行封装，发送到故障设备所属资源 Agent 组的子管理 Agent。

（2）工序回收与工件任务封装。首先子管理 Agent 通知车间管理 Agent 把故障设备 Agent 中尚未执行的工序任务回收，并回收同属一工件的安排在正常设备上的工序任务。如图 3-3 为三台设备的生产计划甘特图，假设时间 t_1 设备 m2 发生故障，则原本安排在 m2 上的工序 $o_{3,2}$ 和 $o_{2,2}$ 需要回收，同时由于 FJSP 中同属一工件的工序间存在顺序约束，因此正常设备上的工序 $o_{3,3}$ 和 $o_{2,3}$ 同样需要回收，回收后各设备的加工序列甘特图如图 3-4 所示。工序任务回收后需把这些任务重新封装为新的工件任务，封装过程为：首先确定安排在故障设备上的工序所属的工件任务集合$P = \{p_1, p_1, \ldots, p_n\}$，然后把集合内的工件剩余的后续未加工工序按照工序约束顺序重新封装，得到新的工件任务集合$P' = \{p'_1, p'_2, \ldots, p'_n\}$。

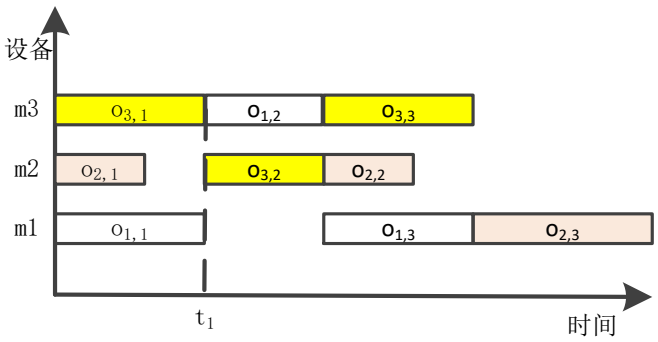


图 3-3 工序回收前的设备-时间甘特图

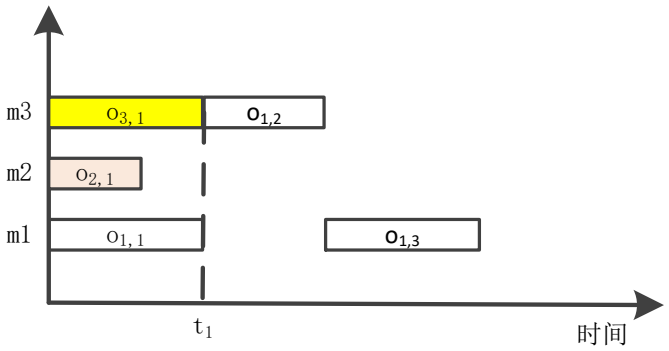


图 3-4 工序回收后的设备-时间甘特图

（3）车间 Agent 移除故障设备对应的设备 Agent，子管理 Agent 根据其下各车间 Agent 的设备配置与集合中P'中各工件任务的设备需求进行对比，若存在可替换设备，则该资源 Agent 组按照生产任务分解策略重新计算调度方案。若不存在可替换设备，则子管理 Agent 需把工件任务集合P'发送至全局管理 Agent，由全局管理 Agent 和其他资源 Agent 组中进行任生产任务的跨区域分解流程，求解新的调度方案。

（4）若求解的调度方案无法满足生产任务的交货期，需向客户返回延迟时间并进行原因说明。

3.3.3 订单取消条件下的异常调度策略

当全局管理 Agent 接收到取消生产任务的请求后，若该任务处于全局管理 Agent 的任务队列中，尚未进行任务的分解分配过程，则可直接从任务队列中删除该任务。若该任务已经完成任务分解流程，并安排到不同设备 Agent 的加工序列中，则全局管理 Agent 需向各个资源 Agent 组发送撤除请求，由子管理 Agent 通过层次间信息传递来通知其中的设备 Agent 移除属于该订单任务的工序任务，此时各设备 Agent 的工序队列中将出现空闲的时间片，可在满足工序顺序约束的前提移动各生产任务，能够减小各生产任务的交货期。如图 3-5，假设 t_1 时刻全局管理 Agent 接收到取消工件 1 的生产订单，则删除各设备上工件 1 的工序任务后，工序 $o_{2,3}$ 的预计开始加工时间可从 t_3 提前至 t_2，如图 3-6 所示。

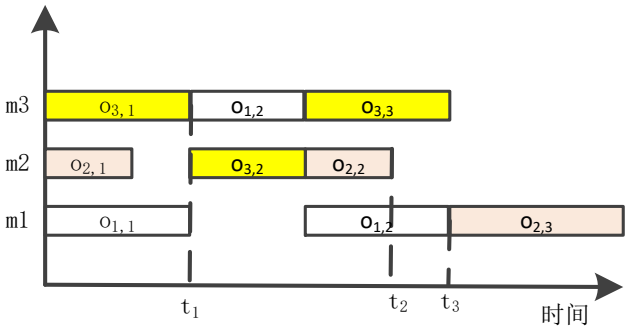


图 3-5 订单取消前的设备-时间甘特图

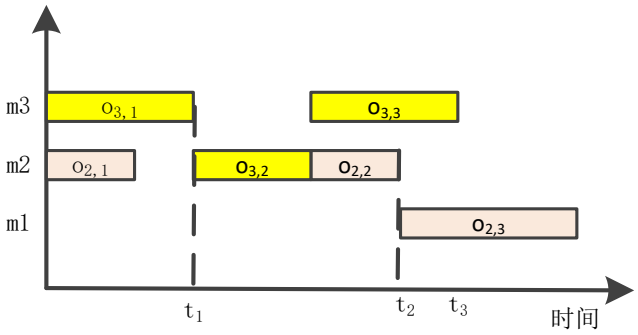


图 3-6 订单取消后的设备-时间甘特图

3.4 本章小结

本章主要介绍本文的动态调度系统中 Agent 的协作机制，通过定义各类型 Agent 间信息传递和反馈流程，来提高系统对生产调度的求解能力以及对各种异常因素反应的灵敏度，其中重点描述了生产任务的数学表达模型，以及生产任务在多个资源 Agent 组之间的分解流程，来提高制造企业对分布式资源的利用率。同时还定义了在订单取消、紧急订单和设备故障等异常状况发生条件下 Agent 之间的协作流程，通过 Agent 间灵活的通信与交互，来完成系统对异常因素的快速响应，保证系统生产管理的平稳高效进行。

第四章 多 Agent 柔性生产调度系统中的调度算法

在基于多 Agent 的生产任务跨区域分解过程中，每个资源 Agent 组对应为一个子调度系统，根据自身的设备配置与生产任务对制造资源的需求进行匹配，当匹配成功后，交由算法 Agent 执行调度算法计算调度方案，最终由全局管理 Agent 从各个资源 Agent 组筛选完工时间最短的调度结果。本文使用改进的蚁群算法作为动态调度系统中的调度算法，由算法 Agent 进行封装，为资源 Agent 组提供计算服务。

4.1 基本蚁群算法的引进

M.Dorigo 等意大利学者在观察蚁群觅食过程中发现，蚁群在没有任何信息提示和指引的情况下，依然能够寻找出食物源和蚁巢之间的一条最短路径[40]。学者们通过分析蚂蚁觅食行为以及相互间的作用关系，总结了影响蚂蚁行动决策的因素，并由此提出了蚁群算法。蚁群算法属于仿生学的启发式算法[41]，多用于图论中最短路径的搜索问题，算法通过模仿蚁群觅食过程中的行为以及蚂蚁间、蚂蚁与环境间的相互作用规律，使问题求解能够逐渐收敛，得到最优解。

蚁群在觅食过程中存在如下的行为和规律[42]：

（1）蚂蚁从蚁巢出发寻找食物时，会沿途洒下一种化学物质，即信息素。信息素对其他蚂蚁有一定的指引作用，信息素浓度越高的方向，蚂蚁选择的概率越大。

（2）蚂蚁的行为具有一定的随机性。当周围环境没有信息素指引时，蚂蚁会概率性地选取觅食方向；当环境中有信息素指引时，蚂蚁并不完全遵循信息素浓度最高的方向，而是存在一定的概率探索信息素浓度较低的方向。

（3）信息素会自行挥发。随着时间的进行，信息素会挥发直至完全消失。因此离蚁巢越远，信息素浓度越低。

因此，当一只蚂蚁沿某条路径寻找到食物后，沿途洒下的信息素会吸引其他蚂蚁，这些蚂蚁在行动过程中同样会洒下信息素，吸引更多的蚂蚁选择该路径，随着时间的推移，路径上的信息素浓度将越来越高，这是一个正反馈的过程，因此蚁群总能找到一条连接蚁巢和食物源的路径。又由于蚂蚁的行动具有随机性，存在少数蚂蚁独辟蹊径，选择信息素浓度低甚至没有信息素的方向，当蚂蚁寻找到一条通往食物源更短的路径，吸引其他蚂蚁的同时，由于路径短，所花时间少，残留信息素更多，因此选择该路径的蚂蚁会逐渐增多，最终蚁群会收敛到更短的路径。由此可看出，蚁群觅食过程中信息素的

正反馈作用能够指引蚁群寻找到食物，而蚂蚁行动中所具备的一定随机性可以使蚁群能够概率性地聚集到更优的路径。根据蚁群算法的原理，可以总结出该算法具有如下的优点[43]：

（1）逻辑清晰简洁、易于实现

蚁群在自然环境中的觅食规则非常简单，所受干扰和限制少，遵循“初始化——状态转移——洒下信息素——目标判定——更新信息素”的简洁行为模式，对比传统的经典调度算法，蚁群算法的数学模型简单，计算量较小，易于实现。

（2）较强的鲁棒性

蚁群算法最终的输出结果是通过多次迭代中对每一只蚂蚁的全局搜索结果进行比较、筛选来得到的，因此即使存在蚂蚁在较低概率下求解得到较差的结果，对当前的全局最优结果影响较小，同时蚂蚁间信息素的相互指引作用能有效降低搜索过程中接受坏选择的几率。

（3）正反馈

蚁群觅食过程中依靠信息素相互作用、配合，呈现出很强的社会性。每只蚂蚁在路径搜索过程中洒下的信息素会对全体蚂蚁当前以及未来的行动具有指引作用。受到指引的蚂蚁同时也会洒下信息素，进一步增强信息素浓度，循环反复，将导致越来越多的蚂蚁集中到某条路径。因此对应的算法中，蚂蚁根据状态转移概率公式决定可达范围的位置的同时，也会更新环境信息素，两者相互影响，直至达到共同的优化目标。

4.2 基本蚁群算法在 FJSP 中的应用

基本蚁群算法在根据搜索目标、搜索规则以及信息素浓度等进行概率性位置转移的特点非常适用于旅行商问题（Traveling Salesman Problem, TSP）的求解[39]。TSP 定义了 n 座城市，其中任意两座城市之间均可达，推销员需要从某个起点城市出发，仅访问每个城市一次后回到起点，要求推销员所经过的路程最短[44]。TSP 等同于在一个完全图中寻找一条路径最短的哈密顿回路，这是典型的 NP-Hard 问题，随着完全图中节点数目增加，经典的精确算法是无法在多项式时间内完成求解的。

FJSP（以最小化最大完工时间为单优化目标）与 TSP 非常相似：两者均属于图论中的最短路径搜索问题，其中 TSP 等同于要求在完全图中寻找一条最短回路，而 FJSP 要求在若干工件的工艺规划图中寻找一条用时最少的的加工路径；TSP 中每个城市只能被访问一次，FJSP 中每个工序只能被加工一次；FJSP 和 TSP 同为 NP-Hard 问题，精确算

法难以求解。

另一方面，FJSP 和 TSP 之间存在如下几点差异：

（1）TSP 中任意两个城市之间可达，若干城市构成了一个完全图。FJSP 中同属一个工件的工序之间存在顺序约束，工序的加工顺序必须满足约束条件。

（2）TSP 求解的路径是一个回路，达到最后一个城市时需返回到起点城市。FJSP 所求的加工路径要求包括每个工件所有的工序，不要求是回路。

（3）TSP 求解的路径要求包含每一个城市，FJSP 中蚂蚁同样要求包含每一个工序，但工序存在机器选择，因此在进行状态转移，即从一个节点转移到另一个节点时，蚂蚁需要面临工序的机器选择问题。

为阐述基本蚁群算法在 FJSP 的执行过程，先对其中的数学符号进行说明：

w：蚂蚁数量

N：算法迭代次数

Q：每轮迭代中蚂蚁洒下的信息素总量

$\tau_{i,j}(t)$：t 时刻工序 i 和工序 j 路径上的信息素浓度。

τ_0：各工序节点间信息素的初始浓度

ρ：信息素挥发因子，其中 $0 \leq \rho < 1$

$\Delta\tau_{i,j}(t)$：本轮迭代中在工序 i 和工序 j 的路径间洒下的信息素增量

$\eta_{i,j}(t)$：t 时刻蚂蚁从工序 i 转移到工序 j 的启发式

α：信息素权重系数

β：启发式权重系数

$P_{i,j}^{x}(t)$：t 时刻蚂蚁 x 从工序 i 转移到工序 j 的概率

$Tabu_i^x$：蚂蚁 x 在工序 i 时的禁忌池，记录了蚂蚁 x 选择工序 i 时走过所有工序

$allowed_i^x$：蚂蚁 x 在工序 i 时的可选池，记录了蚂蚁 x 完成工序 i 时下一步可选的工序

下面根据 FJSP 和 TSP 中的差异对上述符号表示进行细节的说明：

（1）FJSP 中每个工序存在机器选择，每个选择对应一种加工时间。例如假设某工件存在两个工序，工序 1 和工序 2，以及三个设备，m_1、m_2 和 m_3。两个工序的机器选择如下表 4-1。由表可知工序 1 可在设备 1 和设备 3 上进行加工，对应的加工时间是 3

和 6；工序 2 可在设备 1、2 和 3 上加工，对应的加工时间分别是 2、5 和 1。因此工序 1 的设备选择序列 $MC_1=\{m_1,m_3\}$。同理可得 $MC_2=\{m_1,m_2,m_3\}$。

表 4-1 工件所包含的两个工序的机器选择

	设备 m_1	设备 m_2	设备 m_3
工序 1	3	-	6
工序 2	2	5	1

（2）TSP 和 FJSP 中禁忌池 Tabu 都是用于记录蚂蚁已经走过的节点，但两者中的可选池 allowed 有所差异。TSP 中由于任意两个节点间均可达，因此可选池包含除禁忌池中节点外的所有节点；FJSP 中由于工件的工序间存在顺序约束，除工件的起始工序外，所有工序都必须等待前置工序完成后方可进行加工，因此可选池中包含的节点只能是禁忌池节点中的后续节点。假设 FJSP 需要对如下图 4-1 中两个工件进行调度，算法执行过程中某只蚂蚁已走过节点 $o_{1,1}$，$o_{1,2}$ 和 $o_{2,1}$，即 $tabu=\{o_{1,1},o_{1,2},o_{2,1}\}$,则根据工序间的约束关系此时 $allowed=\{o_{1,3},o_{2,1}\}$。

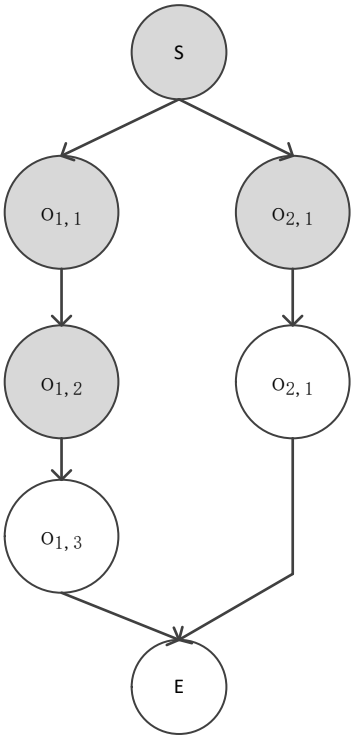


图 4-1 两个工件的工艺网络图

（3）蚂蚁在经过的路径需要洒下信息素，随着时间的推移，信息素会自行挥发直至完全消失，因此需要对信息素矩阵进行更新，更新公式如下[45]：

$$\tau_{i,j}(t+1)=(1-\rho)\tau_{i,j}(t)+\Delta\tau_{i,j}(t) \tag{4-1}$$

$$\Delta\tau_{i,j}(t)=\sum_{x=1}^{m}\Delta\tau_{i,j}^{x}(t) \tag{4-2}$$

其中，$\Delta\tau_{i,j}^{x}(t)$为本次迭代蚁群 x 在工序 i 和工序 j 之间洒下的信息素浓度增量。

蚂蚁更新信息素的时机有两种，一种是每次进行状态转移时洒下信息素，另一种是在每轮结束时更新信息素，本文选择后者，如下[45]：

$$\Delta\tau_{i,j}^{x}(t)=\begin{cases}\dfrac{Q}{L_x}, & \text{蚂蚁 x 经过路径}(i,j)\\ 0\ , & otherwise\end{cases} \tag{4-3}$$

其中 Q 是定值，在 TSP 中，L_x是蚂蚁 x 经过的路径总长度，在 FJSP 中 L_x代表蚂蚁 x 走过的路径对应的完工时间。因此完工时间越长的路径，单位长度撒下的信息素量就越少。

（4）蚂蚁 x 从工序 i 转移到工序 j 的概率计算公式为[45]：

$$P_{i,j}^{x}(t)=\begin{cases}\dfrac{\tau_{i,j}^{\alpha}(t)\eta_{i,j}^{\beta}(t)}{\sum_{s\in allowed_i^x}\tau_{i,s}^{\alpha}(t)\eta_{i,s}^{\beta}(t)}, & j\in allowed_i^x\\ 0\ , & otherwise\end{cases} \tag{4-4}$$

其中，$\eta_{i,j}(t)$是蚂蚁从工序 i 转移到工序 j 的启发式，其中信息素权重系数α的大小体现了环境信息素浓度对蚂蚁状态转移的影响程度，能见度权重系数β的大小体现了蚂蚁可达范围内节点的距离对蚂蚁的影响程度。当$\alpha=0$时，表明蚂蚁的行动完全忽略信息素的影响，退化为贪心算法；当$\beta=0$时，表明蚂蚁的行动完全由先验知识决定。在 TSP 中，$\eta_{i,j}$取决于城市 i 和城市 j 之的距离，通常设置为距离的倒数；在 FJSP 中，启发式由工序 j 的完工时间c_j决定，由于工序 j 存在机器选择，因此$\eta_{i,j}$(t)的计算表达式如下：

$$\eta_{i,j}(t)=\begin{cases}\sum_{k\in MC_{i,j}}\eta_{i,j,k}^{x}(t), & j\in allowed_i^x\\ 0\ , & otherwise\end{cases} \tag{4-5}$$

$$\eta_{i,j,k}^{x}(t)=\frac{1}{c_{i,j,k}},\quad j\in allowed_i^x, k\in MC_{i,j} \tag{4-6}$$

其中，$MC_{i,j}$为 3.1.2 中定义的工序$o_{i,j}$的机器选择集合，$\eta_{i,j,k}^{x}(t)$为蚂蚁 x 在工序 i 时转移到工序 j 的设备选择$mc_{i,j,k}$的启发式，$c_{i,j,k}$是工序$o_{i,j}$在设备m_k上的预计完工时间。

根据以上数学模型，基本蚁群算法在 FJSP 中的执行流程如图 4-2 所示，总结如下：

（1）初始化。定义蚂蚁数量、初始信息素浓度、优化目标（完工时间等）、最大迭

代次数 N，当前迭代次数 Nc 以及为每只蚂蚁设置可选池和禁忌池、当前最优路径等。

（2）当前迭代次数 Nc=Nc+1，在本轮循环开启前，清空所有蚂蚁的可选池和禁忌池，把所有工件的起始工序添加到每只蚂蚁的可选池中。蚂蚁根据状态转移概率公式从可选池中选择下一个加工的工序，选择完成后，把该工序加入到禁忌池中，同时把其后续工序添加到可选池中。

（3）信息素更新。当蚂蚁完成所有工序的遍历，即禁忌池为空时，按照信息素更新公式对蚂蚁的搜索路径进行信息素更新。当所有蚂蚁完成搜索后，从中筛选出本轮的最佳路径并与全局最优路径进行对比，若前者更优，则进行替换。

（4）终止条件判定。若 Nc 已达到最大迭代次数 N，执行步骤（5），否则跳转至步骤（2）。

（5）输出全局最优解，包括各设备上的加工序列以及序列上各工序的预计开始加工时间和结束时间，最后结束算法流程。

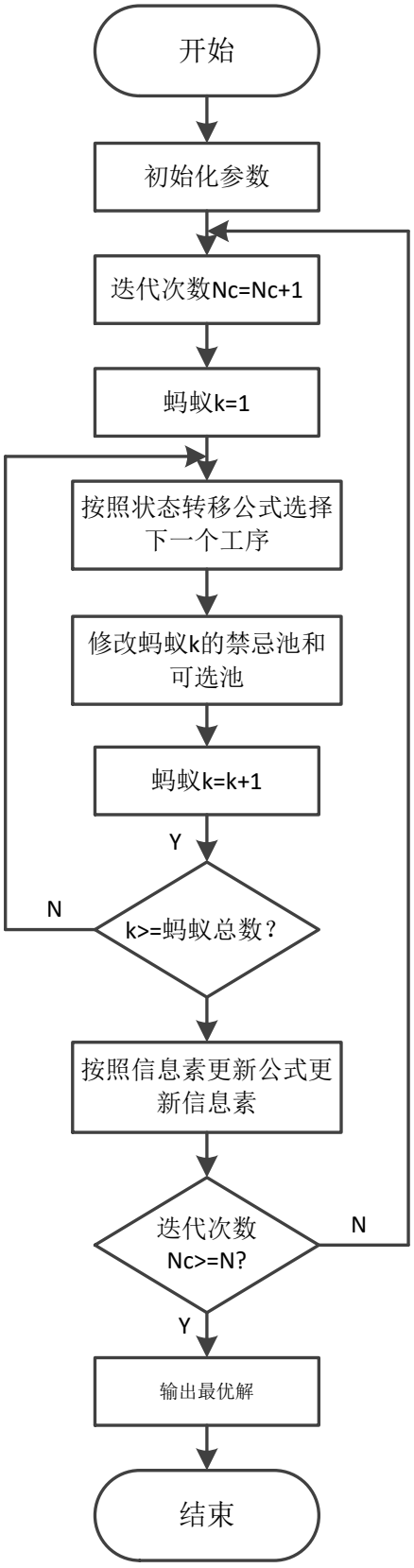


图 4-2 基本蚁群算法在 FJSP 的执行流程

4.3 基本蚁群算法在 FJSP 中的改进

基本蚁群算法的缺点主要有两个：收敛速度较慢和较容易偏离最优解，陷入局部最优。算法初始由于迭代次数少，环境信息素初始浓度低，各路径信息素浓度无差异，对蚂蚁的指引作用弱，蚂蚁更多是根据可达范围内工序的完工时间来概率选择转移的工序，而后需要经过多次迭代累积信息素，当不同工序节点的信息素浓度存在较大差异时，才能指引大多数蚂蚁集中到某一条路径，因此算法初始收敛速度较慢，需要的执行时间较长。另一方面，蚁群间信息素的相互指引作用使算法具备了正反馈特性，保证算法最终能收敛到某个可行解，但同时 FJSP 中工件的工序间存在顺序约束，导致蚂蚁的能见度低，在初始信息素浓度较低的情况下容易受可见范围内工序的完工时间影响而选择较差的路径，而后信息素的释放又会彼此相互吸引，从而蚂蚁搜索过程中易陷入局部最优解。若算法迭代次数较少，可能导致蚂蚁没有机会搜索其他路径，无法跳出局部最优解。

目前最大最小蚂蚁系统（Max-Min Ant System, MMAS）是使用较为广泛的对基本蚁群算法的改进策略，经众多研究人员的实验，MMAS 在 TSP 的求解有着理想的优化性能[46]。MMAS 主要改进了信息素更新规则，在每次迭代中，只对全局或本轮迭代的最佳路径进行信息素的更新，该方法能够更大程度地提高蚁群算法的正反馈，加快收敛速度，本文在参考 MMAS 的改进策略的基础上，提出如下的信息素更新公式：

$$\tau_{i,j}(t+1) = (1-\rho)\tau_{i,j}(t) + \Delta\tau_{i,j}(t) \tag{4-1}$$

$$\Delta\tau_{i,j}(t) = \sum_{x=1}^{m} \Delta\tau_{i,j}^{x}(t) \tag{4-2}$$

$$\Delta\tau_{i,j}^{x}(t) = \begin{cases} \dfrac{Q}{L_x \times e^{(L_x - L_{best})}}, & \text{蚂蚁}x\text{为最佳蚂蚁且}(i,j) \in L_x \\ 0, & otherwise \end{cases} \tag{4-7}$$

其中，L_{best}为蚁群的全局最优路径，假设蚂蚁 x 为本次迭代的最佳蚂蚁，对应的最佳搜索路径为L_x，由上式可知，当L_x优于L_{best}，且优于的程度越高，信息素增量越大，这有助于提高蚁群路径搜索的收敛速度；若L_x不如L_{best}，说明L_x路径上信息素浓度偏高，且偏离最优解，因此需要降低该路径上的信息素增量，能降低蚁群陷入局部最优的概率。

同时，为了避免某一路径上信息素过度积累，导致蚁群无法探索其他的路径，过于早熟，为节点间的信息素浓度设置一个最大值τ_{max}和最小值τ_{min}，这样可以避免某条路径通过正反馈不断积累信息素导致蚁群无法跳出局部最优解，也可避免某些路径信息素挥发导致信息素浓度过低，蚁群的搜索空间缩小。

在蚁群进行路径搜索过程中会记录全局的最佳路径L_{best}，因此若存在蚂蚁当前的路径时间大于全局最优路径时，可提前结束该蚂蚁的搜索，减少算法的计算量，这有助于加快蚁群的搜素速度。同时，基本蚁群算法初始化时是把所有工件的起始工序添加到各只蚂蚁的可选池中，由于初始各节点的信息素浓度相同，对蚂蚁没有指引作用，因此由于能见度的局限，大部分蚂蚁可能选择同一个工序作为搜索路径的起点，将较大概率集中于某一条局部最优路径中。同时由于初始信息素浓度低，蚂蚁搜索过程中主要受可选池中工序的完工时间的影响进行状态转移，当某条路径信息素与其他路径产生较大差异时，同样容易导致算法早熟。因此，本文在前 10%的迭代中，把蚁群均匀分布到各个工件的起始工序中，分布方式为：把工件集合以随机的顺序排序，得到一个工件序列，而后按照序列的顺序，把工件的起始工序添加到一只蚂蚁的禁忌池中，然后把该工序的后续工序以及其他工件的起始工序添加到该蚂蚁的可选池中，如此循环往复直至每只蚂蚁都完成禁忌池和可选池的初始化。

为避免环境初始信息素浓度差异过低，蚂蚁过早集中于某条局部最优路径，因此本文的蚁群算法在前 10%的迭代次数中忽略信息素的指引作用，即信息素权重系数α满足以下公式：

$$\alpha = \begin{cases} 0, & Nc < 0.1N \\ \alpha_0, & otherwise \end{cases} \tag{4-8}$$

其中α_0为算法设置的信息素权重系数，Nc 为算法当前的迭代次数。随着迭代次数的增加，环境信息素逐渐积累并产生差异，对蚂蚁的行动具有更强的指导意义，能够扩大蚂蚁的搜索范围，因此在 10%的迭代后接受信息素的指导作用。

本文改进的蚁群算法的执行流程可总结如下图 4-3，详细过程如下：

（1）初始化算法参数，包括环境初始信息素浓度，迭代次数、信息素权重系数、能见度权重系数等。

（2）初始化蚁群，包括蚂蚁数量、蚂蚁的禁忌池和可选池。

（3）判断迭代次数 Nc<0.1N 是否成立，若成立，均匀分布蚂蚁的初始位置，即把工件的起始工序均匀地分配到每一只蚂蚁的禁忌池中，同时把该工序的后续工序以及其他工件的起始工序添加到蚂蚁的可选池中，同时设置$\alpha = 0$；若不成立，把所有工件的起始工序添加到每只蚂蚁的可选池中并设置$\alpha = \alpha_0$。

（4）状态转移概率计算。通过状态转移公式从可选池的工序中选择下一个工序。

（5）把当前已选择的工序添加至禁忌池，并把其后续工序添加到可选池。

（6）计算蚂蚁当前路径的时间是否大于全局最佳路径，若是，结束该蚂蚁在本轮的搜索。

（7）判断蚂蚁的可选池是否为空，若不为空，转至步骤（4）。

（8）从蚁群中筛选本轮迭代中的最佳路径，并与全局最佳路径进行对比，若本轮最佳路径更优，则进行替换。

（9）信息素更新。按照公式(4-7)对本轮最佳路径进行信息素更新。

（10）判断迭代次数已达最大值 N，若是，输出全局最优路径并结束算法，否则跳转至步骤（2）。

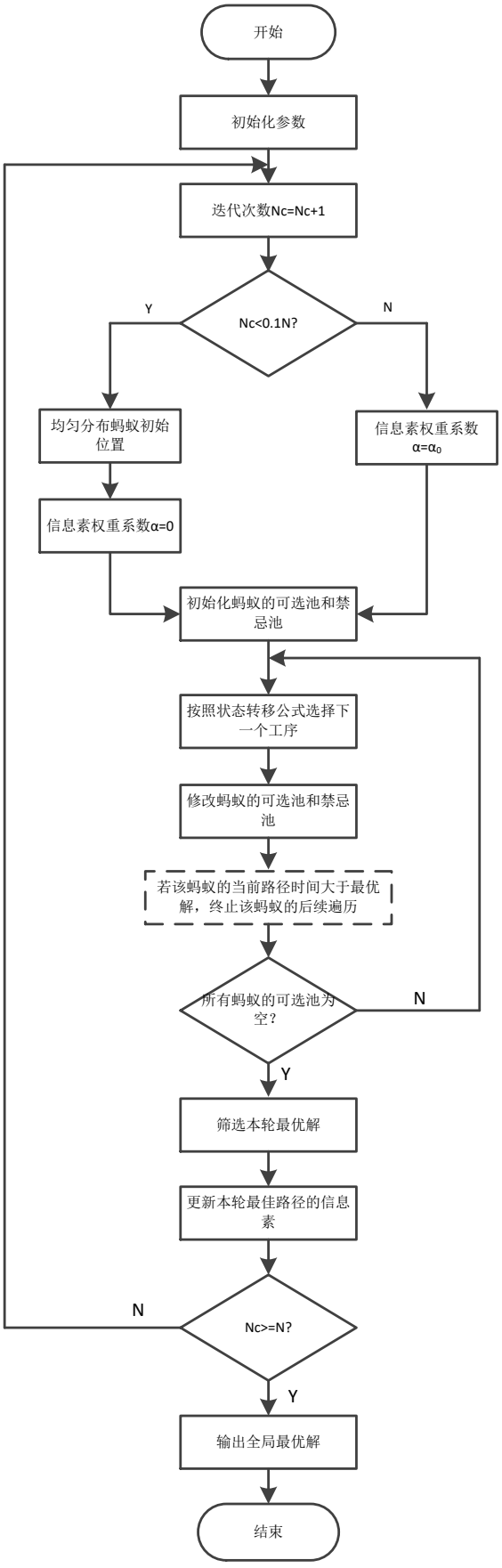


图 4-3 改进的蚁群算法执行流程

4.4 本章小结

本章应用了蚁群算法作为系统的调度算法，提出了基本蚁群算法在 FJSP 中的数学模型以及信息素更新规则和状态转移公式，并根据算法收敛速度慢，易于陷入局部最优的缺点，提出了对基本蚁群算法的改进，改进的方面包括均匀分布蚂蚁的初始位置、只对本轮最佳蚂蚁的路径进行信息素更新、在前 10%的迭代中忽略信息素的影响、当蚂蚁的当前解时间大于全局最优解时放弃该蚂蚁的后续搜索等，最后总结了改进的蚁群算法在 FJSP 中的执行流程。

第五章 多 Agent 柔性生产动态调度系统的实现

5.1 制造企业的生产管理流程

调度系统的实现需要符合制造企业实际的生产管理需求，制造企业传统的生产管理流程如下图 5-1 所示，当客户下达生产订单时，由销售部负责接收并处理，销售部根据订单内容提炼客户对产品数量、精度、规格、交货期等要求，并转交到工艺部；工艺部根据产品的技术指标，从数据库中获取符合要求的工件加工工艺信息，而后由生产计划部转接；生产计划部的车间调度员根据工件的加工工艺、车间的设备配置以及订单的交货期，完成工序的机器分配和顺序的确定工作，然后统计生产过程所需的物料清单并提交到采购部；采购人员检查库存，确定物料充足后，通知生产计划部安排生产。

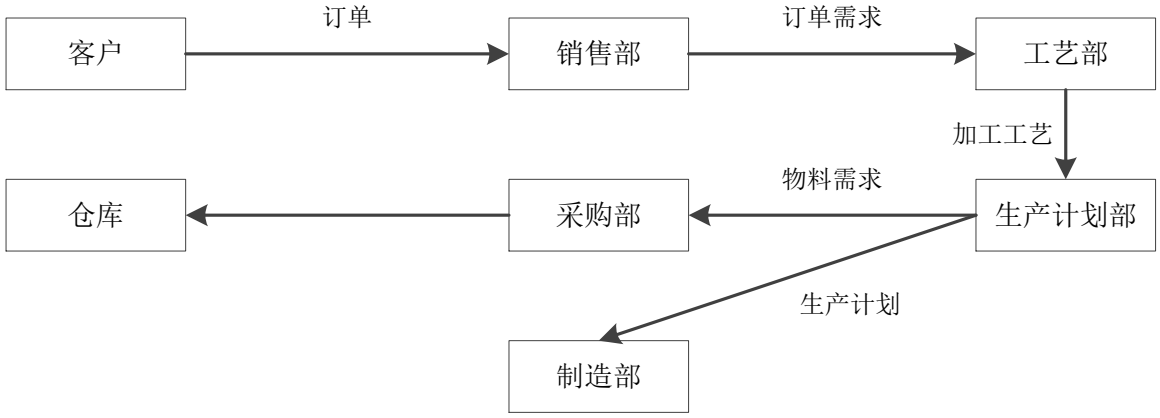


图 5-1 企业传统的生产管理流程

5.2 柔性生产动态调度系统需求分析

制造企业传统的生产管理流程高度依赖人力，生产计划的制定依靠个人的先验知识，在生产订单频繁变更、工艺复杂的产品生产中难以保证生产计划的质量和效率。其次信息由人为传递，可能出现传递不及时甚至传递错误信息的情况，当出现设备故障时，信息人为传递的滞后性会干扰正常的生产流程。针对以上不足，柔性生产动态调度系统应能满足企业的自动化和信息化需求，集成生产管理的各个环节，提供先进的调度机制，保证高质量和高效率的生产计划的同时能够实时监测系统的生产环境，快速响应各种动态变化。因此本文的调度系统设置如图 5-2 所示的几个功能模块：

登录模块：提供用户的登录和注册功能，根据用户权限提供对应的系统操作。

信息管理模块：提供对员工信息和物料信息的增删查改，当用户是系统管理员时，

可以修改其他用户的使用权限。

资源管理模块：可用于对各个地区的工厂中车间结构以及设备配置进行查看和修改，同时可查看设备的工作状态。

工艺管理模块：对系统内产品的工艺信息进行管理，可查看产品的组成，工件的加工工艺等。当用户是工艺管理员时，可对工艺信息进行增删查改。

生产调度模块：用于接收客户的生产订单并制定对应的调度方案，提供对改进蚁群算法的参数设置以优化调度结果。也可由系统管理员直接设置参与调度的生产任务并制定生产计划。

系统帮助模块：提供系统操作指南帮助用户快速掌握系统的使用流程。

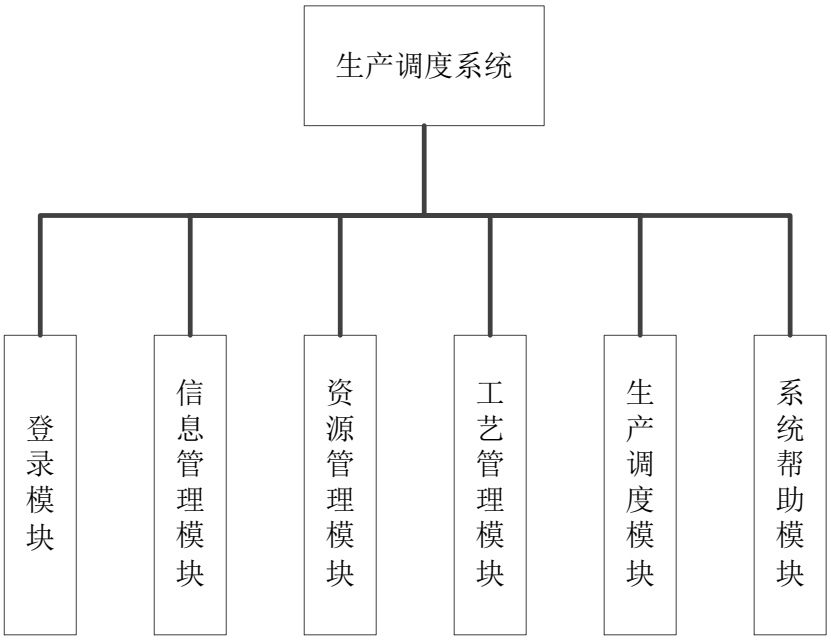


图 5-2 生产调度系统功能模块划分

5.3 多 Agent 系统的实现

5.3.1 开发环境及工具

操作系统：Microsoft Windows 10 Professional

开发环境：JDK1.8.0_144

开发平台：Eclipse4.7.0

开发语言：Java

数据库：MySQL5.6.17

5.3.2 系统软件架构设计

本文的调度系统属于混合型的结构。系统根据企业各地区工厂的数量建立对应的资源 Agent 组，每个资源 Agent 组中包含一个子管理 Agent，负责管理和协调其中的车间 Agent 和设备 Agent。其次各个子管理 Agent 之间通过全局管理 Agent 相互联系，由此可见全局管理 Agent 和各个资源 Agent 组之间组成了分层式的结构，上层 Agent 通过协调下层 Agent 的通信来完成生产任务的分解分配。根据以上特点，全局管理 Agent 和各资源 Agent 组之间非常适合使用 C/S 架构，即客户端/服务器模式。全局管理 Agent、算法 Agent、工艺 Agent 和监控 Agent 之间相对独立，以分布式结构联系在一起，在通信上处于平等的地位，因此适合使用点对点通信。因此，根据系统中各 Agent 之间的合作模式，最终把系统软件架构分为三层：表现层、应用层、数据层，如图 5-3 所示。

（1）表现层：提供友好人机交互界面，是管理员下达指令、监控生产流程的窗口。

（2）应用层：完成调度系统生产管理的全流程，如制定生产计划、任务分解、资源分配、数据管理、异常检测等，是实现自动化、信息化和敏捷化的关键。具体表现为各类型 Agent 独立运行，承担由系统所分配的各项职能和任务，各司其职相互协商配合，维持系统生产调度工作的稳定有序进行。

（3）数据层：为应用层中各 Agent 提供数据支持，当 Agent 响应管理员的指令以及相互配合达到共同目标的过程中，数据层需要提供数据的增删查改操作，该层存储管理的数据主要包括：各设备的工作状态、加工序列、历史工作数据、产品工艺信息、库存信息、设备监控信息等等。

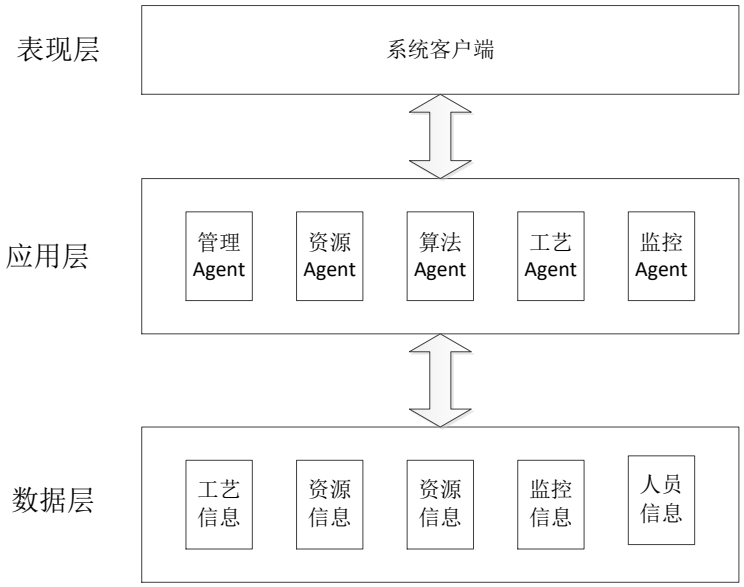


图 5-3 系统体系结构图

5.3.3 多 Agent 系统的面向对象设计

本文的调度系统是通过多个 Agent 的相互协调配合来实现生产管理的各个流程，Agent 是对系统内物理实体或算法逻辑的封装，被赋予了特定的功能职责，彼此间相对独立的同时，能够通过消息传递相互关联，这些特点与面对对象设计中继承、封装和多态的特征非常匹配。Agent 可看作是对对象进行更深层次的发展和完善，是一种粒度更大，具有智能性的对象[47]。因此 Agent 适合使用面对对象的思想来进行功能的封装和结构的设计，可使用任意的面向对象的编程语言实现，如 C++、Java、C#等。考虑到 Java 语言具有平台无关性，易于移植，Java 程序能够在任意一台安装 JRE 环境的主机上运行，这与 Agent 的移动性以及 MAS 易扩展的分布式特点相符，因此本文使用 Java 的面向对象技术来进行各类 Agent 的实现。如下图 5-4 是 Agent 的系统类图。

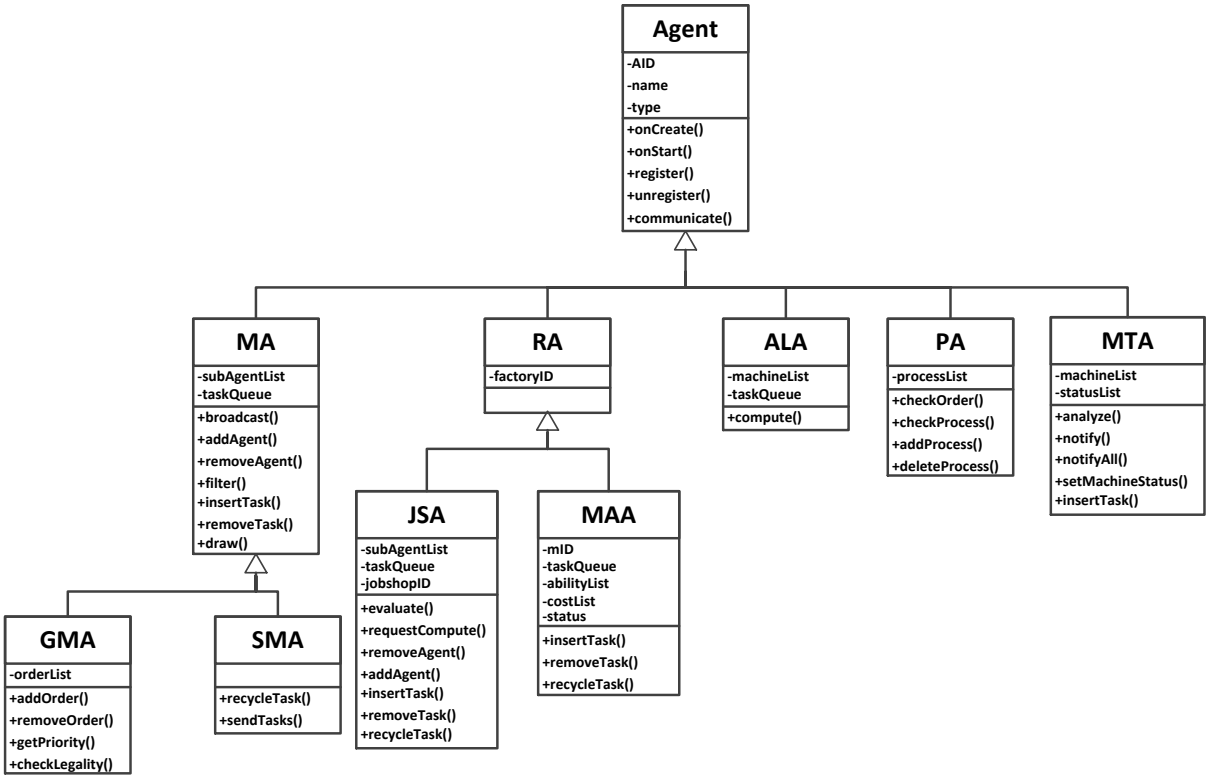


图 5-4 系统类图

下面对系统中各 Agent 的属性定义、方法封装以及继承关系进行说明：

（1）Agent 是基类，定义了 Agent 标识（AID）、名称（name）、类型（type）属性，同时定义了 Agent 初始化、启动、注册/注销以及通信相关的方法。

（2）MA（Manger Agent）为管理 Agent，继承自 Agent。管理 Agent 和各资源 Agent 之间是分层式架构，因此 subAgentList 记录了管理 Agent 的子 Agent，broadcast()方法用

于把生产任务广播至下层子 Agent，filter()方法用于在跨区域生产任务分解过程中筛选下层 Agent 返回的调度方案。

（3）GMA（Global Manager Agent）为全局管理 Agent，继承自 MA，维护一个生产任务队列，队列中任务按照优先级进行排序。当接收到客户的订单时，GMA 使用 checkLegality()审核订单，而后封装为生产任务，使用 computePriority()确定任务优先级，最后插入到生产队列中。

（4）SMA（Sub-Manager Agent）为子管理 Agent，继承自 MA，接收到 GMA 的生产任务时，广播至其下的各个车间 Agent，当接收到监控 Agent 的设备故障信息时，使用 recycleTasks()回收故障设备上的工件任务。

（5）RA（Resource Agent）为资源 Agent，继承自 Agent。FactoryID 定义 Agent 所属的工厂，ResourceAgent 由有两个子类，分别是车间 Agent（JSA）和设备 Agent（MAA）。在生产任务跨区域分解流程中，当 JSA 接收到生产任务时，使用 evaluate()判断其下的设备集是否满足任务对制造资源的需求，使用 requestCompute()通知算法 Agent 进行调度方案的计算；MAA 记录对应设备的名称、加工能力、加工序列、工作状态等。当 GMA 确定最终的调度方案后，MA 需要在加工序列中添加对应的工序任务。在故障发生时，故障设备对应的 MAA 需要使用 recycleTask()进行工序回收，并把回收结果返回到 JSA。

（6）ALA（Algorithm Agent）为算法 Agent，继承自 Agent，内部封装了与调度算法相关的参数，使用 compute()提供调度方案的计算服务。

（7）MTA（Monitor Agent）为监控 Agent，继承自 Agent，使用 analyze()分析采集数据，判断是否存在设备故障；当设备发生故障时，使用 notify()通知 SMA。

（8）PA（Process Agent）为工艺 Agent，继承自 Agent，processList 记录了系统工件的工艺信息，使用 checkOrder()为 GMA 审核订单的合法性，checkNewProcess()为工艺管理员新增工件的合法性进行审核。

在系统运行期间，各个 Agent 对象根据自身的属性、数据以及方法函数发起对系统内部或外部环境的探索以及响应，其中方法函数被编译为 JAR 包，JAR 包可看做是 Agent 的决策库，其中的函数作为决策规则为 Agent 提供行动指南。在生产任务跨区域分解流程以及异常调度协商流程中，各 Agent 根据接收到的信息类型执行 JAR 包中对应的函数，如下表 5-1 是 GMA 需要处理的部分消息类型。

表 5-1 GMA 处理的消息类型

消息类型	含义
ORDER_ENQUEUE	接收到新的生产订单
ORDER_URGENT	紧急订单
ORDER_CANCEL	订单取消
SMA_ADD/SMA_DELETE	新的资源 Agent 组添加/移除请求
ORDER_CHECK_REPLY	PA 返回的订单审核结果
PRT_REPLY	SMA 返回的对工件集合的调度结果
PT_REPLY	SMA 返回的对单个工件的调度结果
MT_ABNORMAL	SMA 发出的异常调度请求

5.3.4 Agent 的通信模型设计

多 Agent 系统作为分布式的系统，对复杂大型问题的求解依赖其内 Agent 的相互配合，因此对 Agent 间通信效率有很高的要求，尤其当频繁发生订单更改、设备故障等动态变化时，将出现大量的消息请求，为了能及时处理反馈，需要保证系统的通信质量。本系统使用基于 Socket 的通信机制完成 Agent 间的信息交互，其中选择底层通信协议为 TCP/IP。Agent 的通信模型由线程池和消息队列实现。

Socket 通信属于消息传递的通信方式。Agent 通信双方需事先确定请求消息的格式，发送方在进行消息的封装时需严格遵循规则约束，接收方接收到消息后同样按照约定的方式对消息的字段逐个进行解析，从中分析请求方的意图并采用对应的处理方案。本系统采用基于 TCP/IP 的 Socket 通信，底层的 TCP 协议是基于字节流的安全可靠通信协议，不保留传输数据的边界，为了能够让通信双方接收到字节流时能够正确识别数据，本系统使用 Java 的序列化和反序列化技术实现对象和字节流之间的双向转换。顾名思义，序列化能够把 Java 对象转换为字节序列，之后可以存储在数据库或在网络上进行传输；反序列化能够把序列化后的字节流还原为 Java 对象。通过该方法，接收方 Agent 获得反序列化的对象后，通过获取对象的属性信息便可解析出发送方的请求意图以及请求数据。

为保证系统的通信效率，Agent 需要能够并发处理来自其他 Agent 的请求，因此通常采用多线程技术，即每当接收到一个消息请求，Agent 就新建线程建立 Socket 连接。

该方式能够充分利用多核 CPU 的性能，并发处理大量请求，提高 Agent 的数据处理能力，从而保证了系统的通信效率。该方式同时也存在一个明显的缺点，当过多的请求蜂拥而至时，Agent 需要在极短时间内创建大量的线程。虽然线程的粒度较小，创建和销毁的开销远比进程小，但过多的线程十分占用 CPU 资源，导致 Agent 无法响应后续的请求，因此本系统使用线程池缓存线程，实现线程的复用，具体过程是：设定线程池和消息队列，线程池初始线程数目n_0，最大线程数目n_{max}，消息队列的容量为c。如图 5-5 所示。当 Agent 接收到一个请求时，从线程池中分配一个空闲线程进行请求的处理；若无空闲线程，检查当前线程池内线程数目n_c,若$n_c < n_{max}$，新建线程，否则请求进入消息队列，直至线程池中出现空闲线程。若 Agent 消息队列为空且当前线程数目大于n_0时，销毁多余的线程。若有新的请求到来且 Agent 消息队列已满时，Agent 将丢弃该请求并向发送方发送消息，请求发送方延后重新发送。

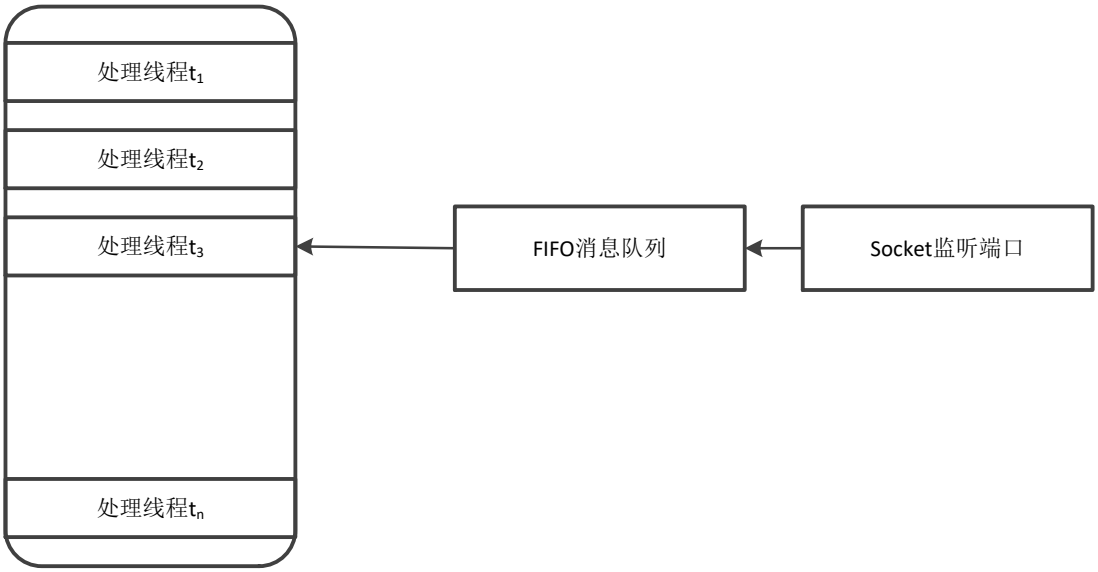


图 5-5 线程池和消息队列

5.4 动态调度系统实现与界面设计

5.4.1 系统登录模块

多 Agent 调度系统成功启动后首先呈现给用户的是登录模块，由管理 Agent 进行界面的绘制和渲染，登录界面如图 5-6 所示。在该界面中，用户需要输入自己的用户名和密码，管理 Agent 根据此信息验证用户的合法性以及权限。

图 5-6 系统登录界面

用户合法性通过后，系统将进入如图 5-7 所示的系统主界面，该界面包括六个选择按钮作为各模块的操作入口，从左至右分别为系统信息管理模块、工艺管理模块、登录管理模块、资源管理模块、生产调度模块以及信息帮助模块。

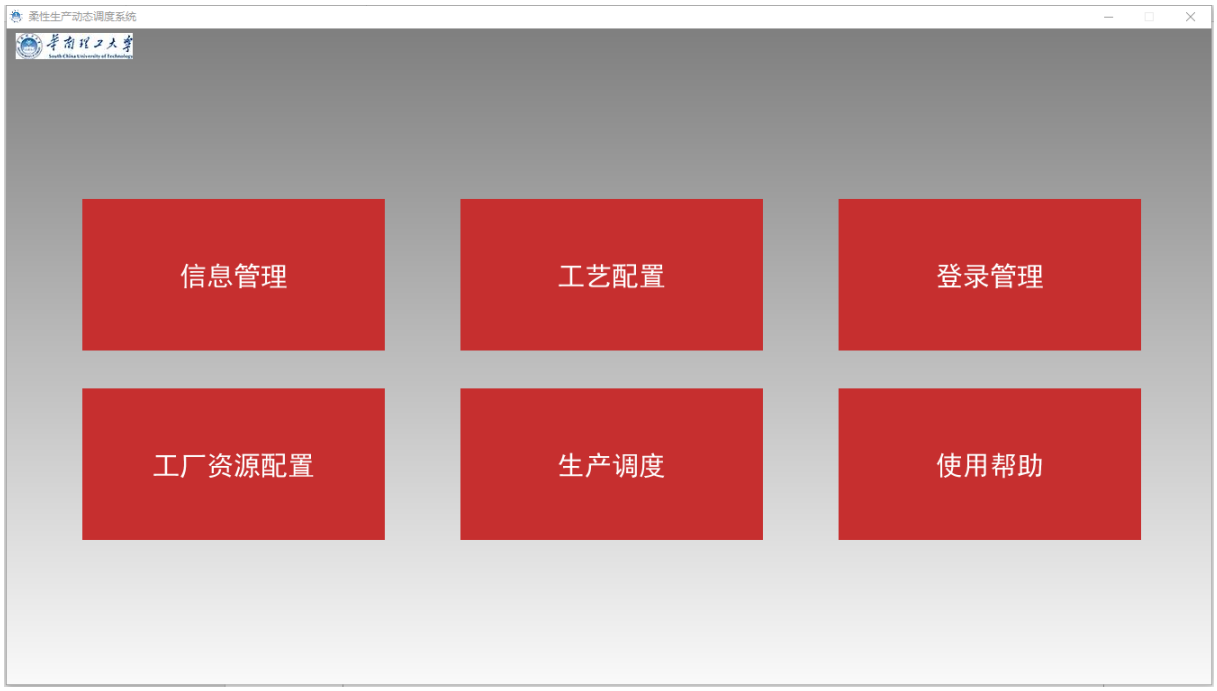


图 5-7 系统主界面

5.4.2 系统信息管理模块

系统信息管理模块用于管理用户信息以及企业的物料信息，其中用户信息存储于全局管理 Agent 的数据库中，物料信息存储在工艺 Agent 的数据库。下图 5-8 是系统的用户信息展示界面，表格中记录了用户的基本信息，包括用户名、年龄、性别和操作权限等。用户的权限分为三个等级，从高到低排列分别是系统管理员、子系统管理员和工艺管理员。系统管理员对应全局管理 Agent，可查看各个工厂的资源配置，并下达面向各工厂的生产计划。子系统管理员对应于一间工厂的管理员，只能查看并制定所属工厂的生产计划。工艺管理员负责管理系统的产品工艺，能够对工艺信息进行增删查改。其中，仅有系统管理员可修改其他用户的信息，如图 5-8 展示了系统管理员对用户信息的修改过程。图 5-9 呈现了企业的物料信息，包括物料名称、型号规格以及对应的数量，系统管理员和工艺管理均有权限对此数据表进行操作。

图 5-8 用户信息管理界面

图 5-9 物料管理界面

5.4.3 资源管理模块

资源管理模块可以为系统管理员提供资源配置的功能，点击主界面的工厂资源配置选项，可进入如图 5-10 所示界面，该界面呈现了系统的工厂分布、各工厂的车间结构以及设备配置，由图 5-10 可知，系统目前包含七间工厂。其中工厂一包含一个车间，名称为车间 1。

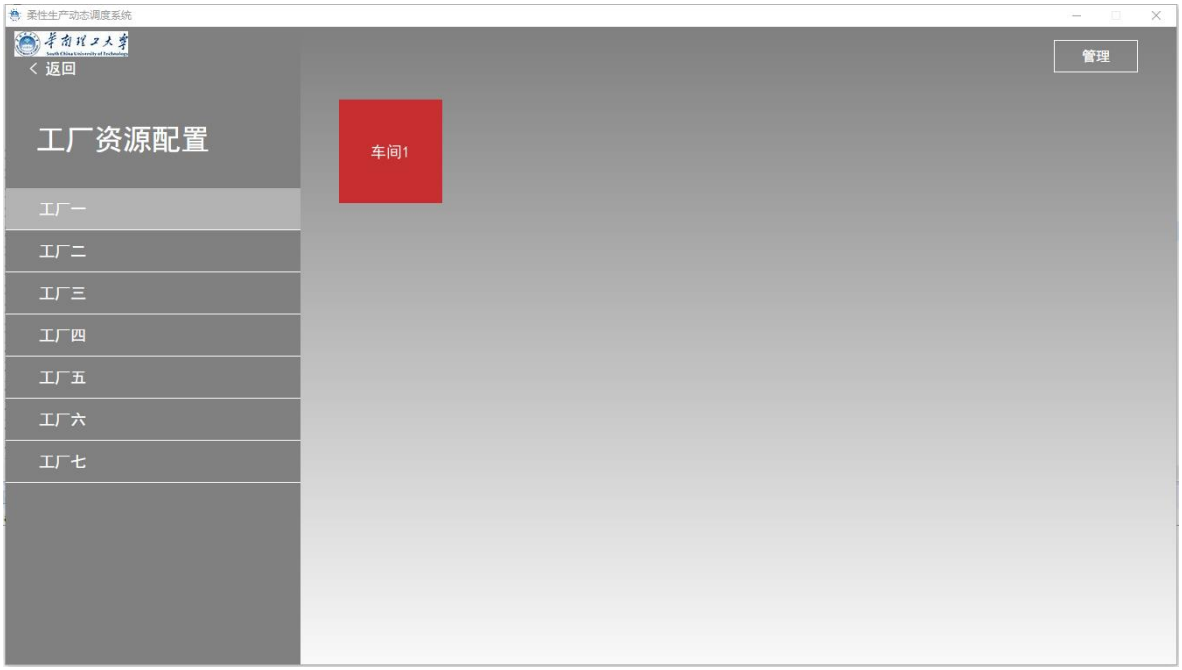


图 5-10 工厂资源配置页面

点击图 5-10 的管理按钮后可进入如图 5-11 所示的管理模式，可进行车间进行增删查改操作，子系统管理员仅可对自身所属的工厂进行此操作。在非管理模式下点击车间，可查看车间的设备配置，如图 5-12 表明车间 1 中包含十个设备，设备的数量均为 1，状态均为正常运作。管理员同样能对设备进行增删查改等操作，该操作由对应的全局管理 Agent 和子管理 Agent 执行。

工厂和车间的增删操作对应与系统内资源 Agent 对象的创建和销毁操作。当系统管理员新增车间和设备信息时，系统需要生成对应的 JSA 和 MAA 对象。同理，当系统内部根据业务需求创建新的 RA 时，Agent 将通知 MA 进行界面的重绘，展示与 RA 相关的资源信息。

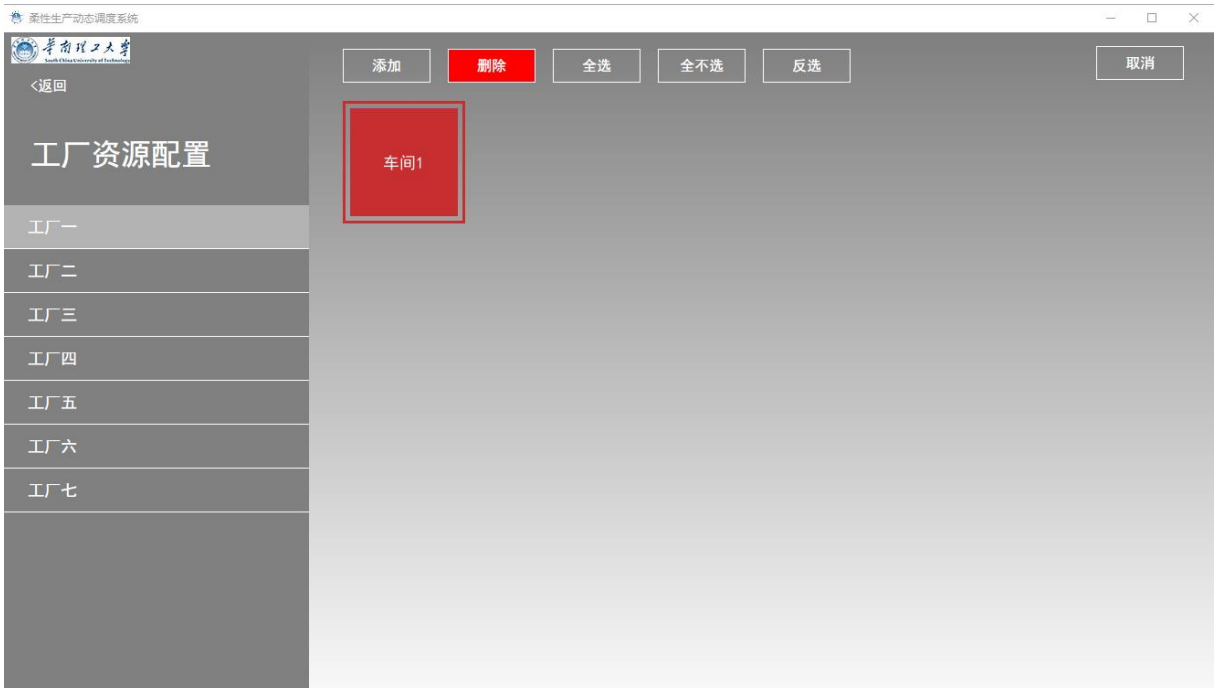


图 5-11 工厂一的资源管理界面

图 5-12 车间 1 的设备配置

5.4.4 工艺管理模块

在主界面点击工艺管理选项后可进入工艺管理模块，该模块为工艺管理员提供对工艺信息的管理接口，如图 5-13 呈现了系统所持有的产品种类，可见目前为系统配置的产品有五种。点击右上方的管理按钮可进入管理模式，能够对产品进行删除和添加。

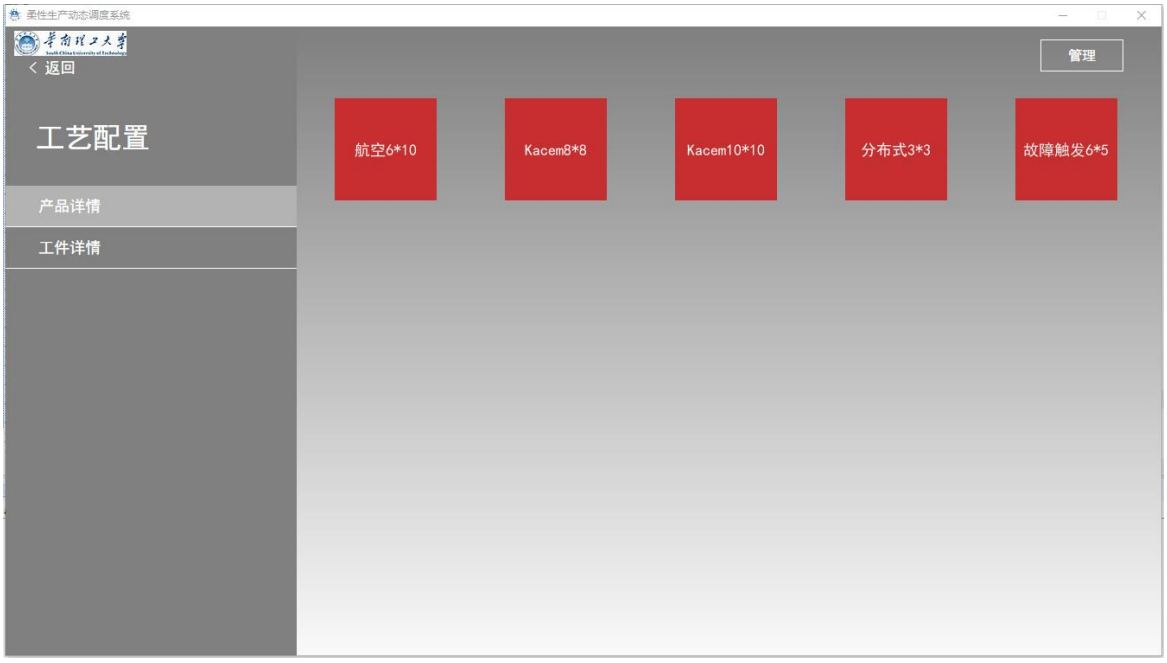


图 5-13 工艺管理界面

如图 5-14 是在非管理模式下，点击了产品“航空 6*10”后进入的产品详情页，此时

可看出该产品包含六种工件，每种工件各一个。

图 5-14 产品的工件组成

点击“工件详情”选项后，可查看系统中包含的工件种类，如图 5-15 所示。在非管理模式下，选择工件“航空-p1”后将呈现工件的加工工艺信息，如图 5-16 所示，从中可看出该工件包含六道工序，可安排在十台设备上进行加工。在系统内部实现中，每个工件、工序都被封装为对象，当工艺管理员新增工件时，工件的工序组成需要与数据库中原有工序集合匹配，该匹配过程由工艺 Agent 执行。

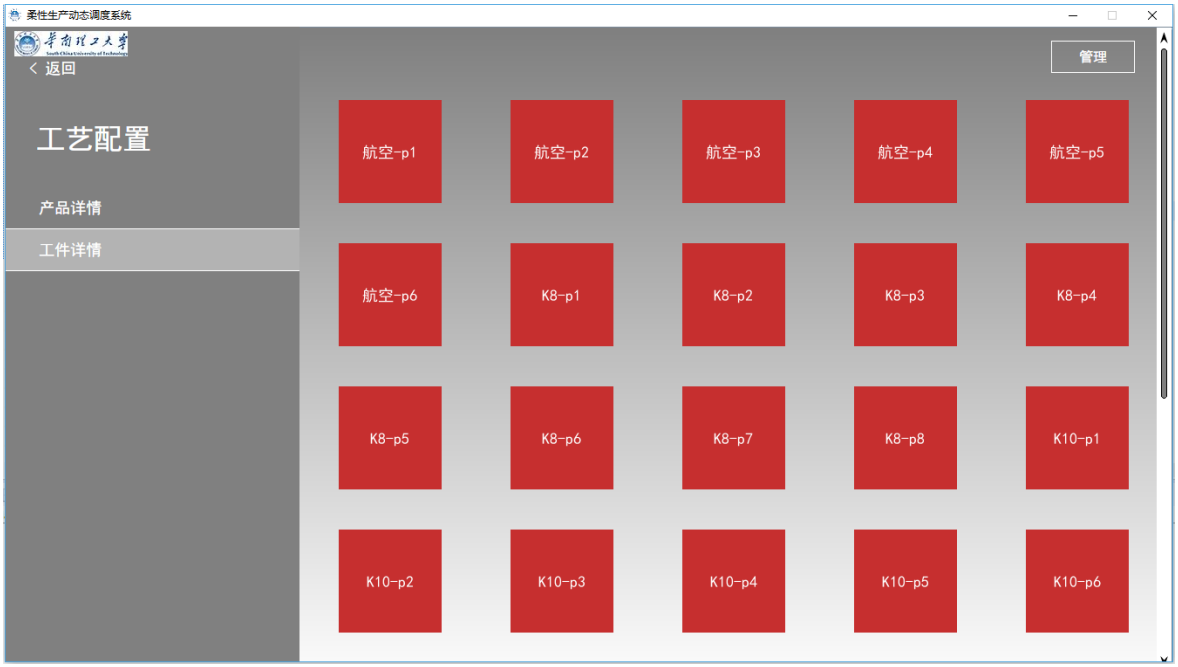
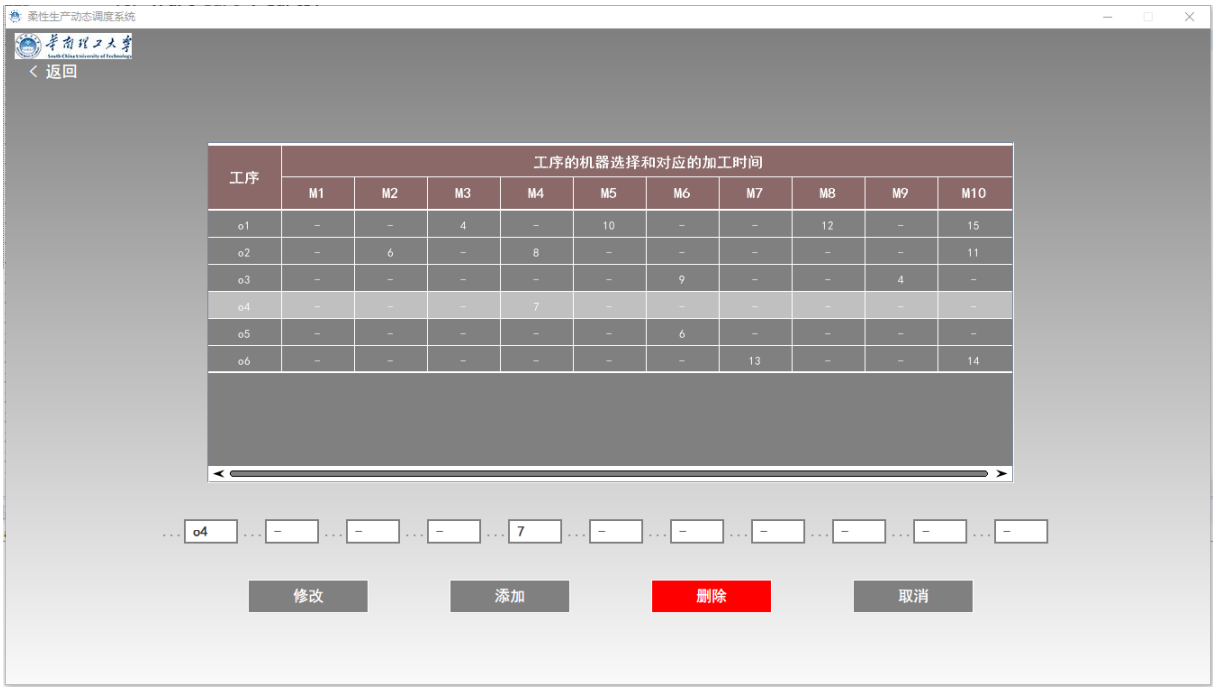


图 5-15 系统的工件集合



5-16 工件航空-p1 的加工工艺信息

5.4.5 生产调度模块

在主界面选择生产调度选项后可进入生产调度模块，当系统接收到客户订单时，由全局管理 Agent 协调各个工厂对应的资源 Agent 组进行任务分配，由算法 Agent 计算调度方案，最终由全局管理 Agent 筛选并输出最优解。因此该模块是用于计算并呈现分配给各工厂的生产计划，同时为系统管理员提供生产计划的制定，如图 5-17 为生产调度的主界面，在该界面上管理员可以设置参与调度的工厂、工件任务以及与调度算法相关的参数。

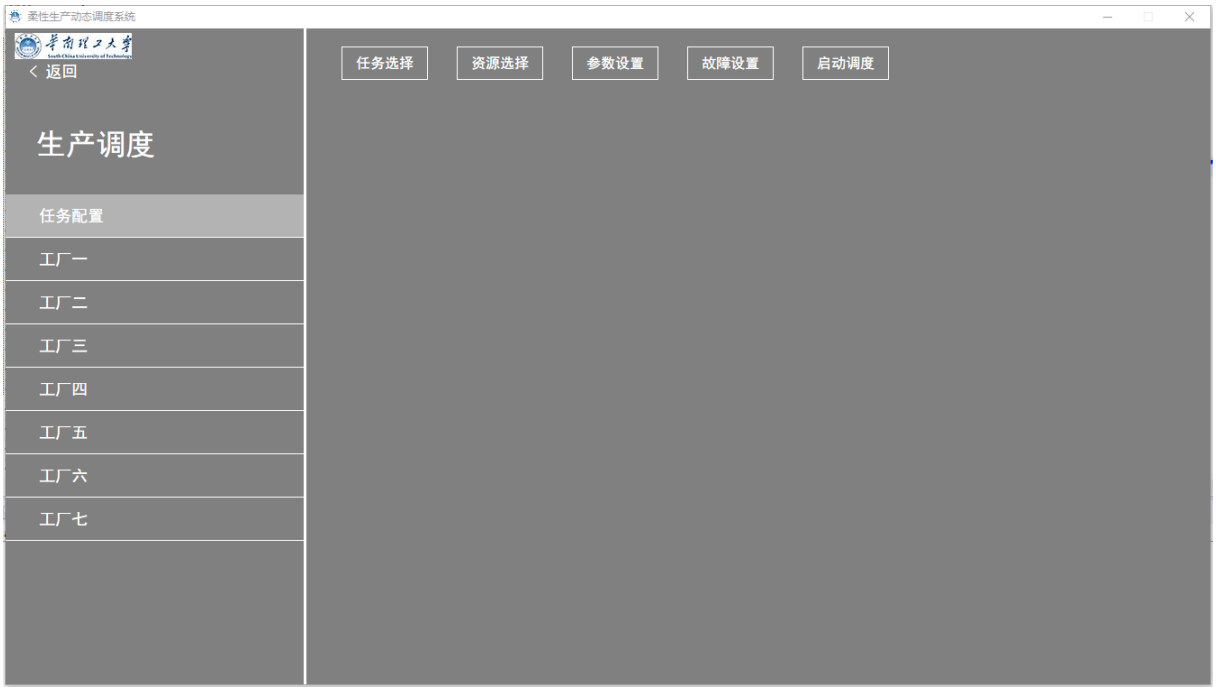


图 5-17 生产调度模块主界面

当选择生产调度主界面的任务选择按钮和资源按钮时，可以分别选择参与调度的产品和工厂，如图 5-18 和 5-19 所示，管理员选择了工厂一接收生产计划，生产任务为产品航空 6*10，其中包含六个工件。

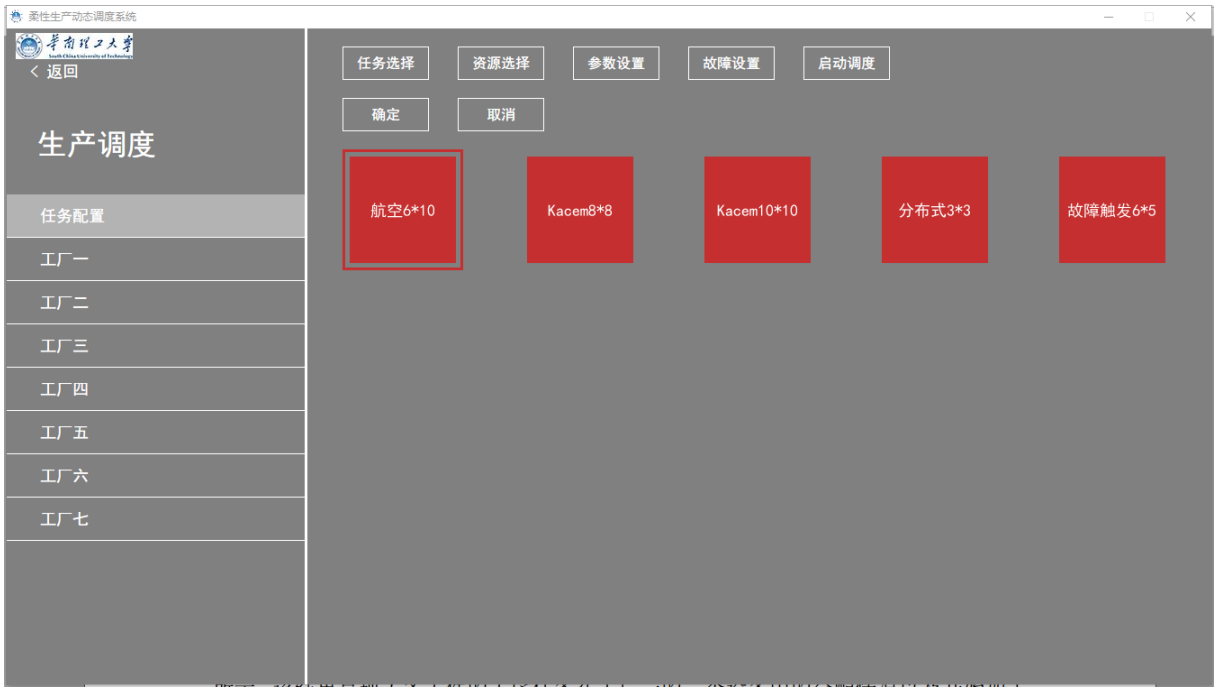


图 5-18 任务选择界面

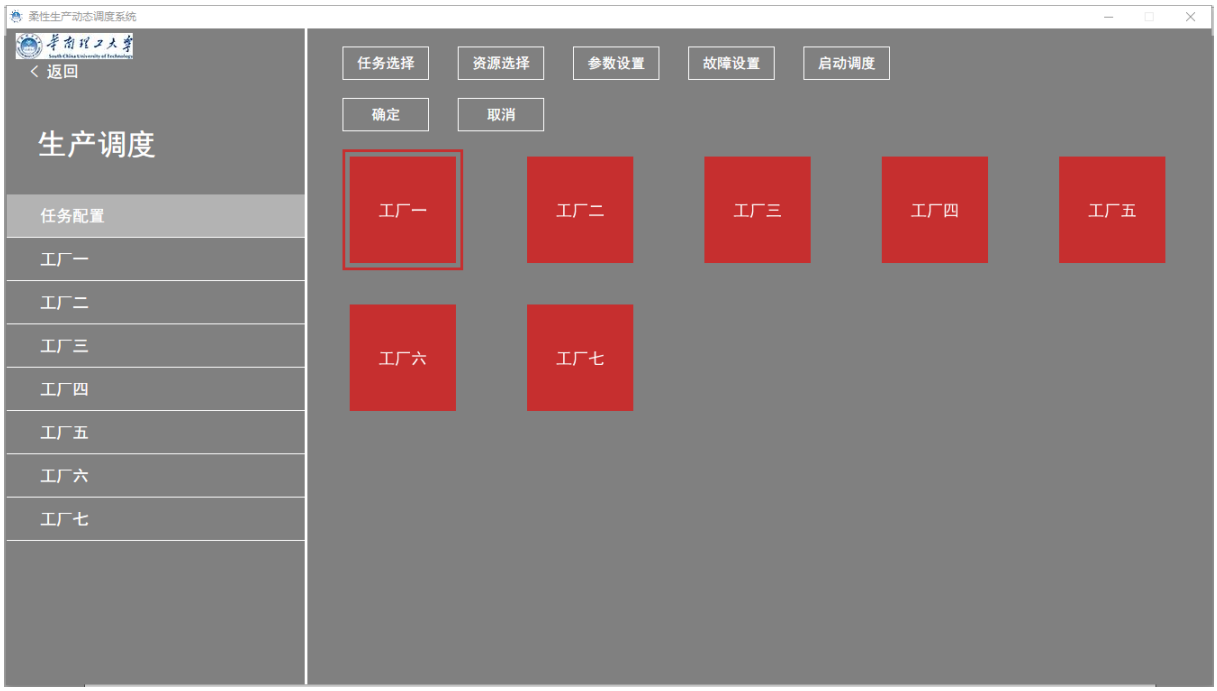


图 5-19 资源选择界面

完成参数设置和任务配置后，点击启动按钮，全局管理 Agent 将对产品任务封装为对应的生产任务 MT，由于此时参与调度的是工厂一，因此跨区域生产任务分解过程将只在工厂一对应的资源 Agent 组中进行，最后由算法 Agent 计算调度方案并由全局管理 Agent 返回并显示工厂一的调度结果，图 5-20 所示，该结果呈现了各工件的工序任务在工厂一的十个设备中的分配情况以及预计的开始加工时间。

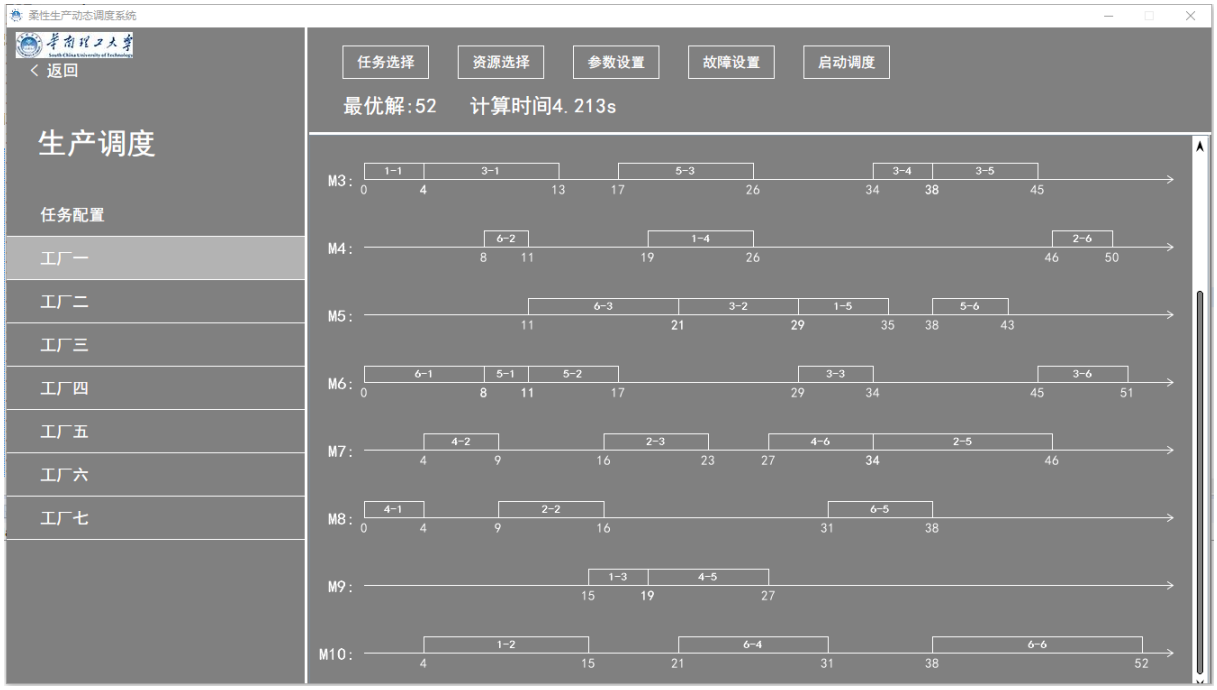


图 5-20 调度结果

5.5 本章小结

本章主要介绍了基于多 Agent 的柔性生产动态调度系统的实现与开发，先是介绍了企业传统的生产管理流程，针对其中的不足提出了调度系统的需求，并根据系统的 Agent 划分设计了系统的体系结构，同时使用 Java 的面向对象技术封装了 Agent 的属性和功能，使用线程池和消息队列构建了 Agent 的通信模型，最后介绍了调度系统的界面设计和使用流程。

第六章 仿真实验与结果分析

6.1 实验环境

本文的实验环境为：Intel(R) Celeron(R) G530 CPU @ 2.40GHz，RAM 2.0GB，操作系统 Microsoft Windows 10 Professional 32 位, JDK 版本 jdk1.8.0_144

6.2 实验一：验证改进的蚁群算法

该实验用于验证本文的基于改进蚁群算法的调度策略在 FJSP 中的可行性，优化目标为最小化最大完工时间。文献[48]提出了一种基于多种群蚁群算法用于 FJSP 求解中，该文献对从某航空发动机公司提取的 6×10 柔性生产调度实例对算法进行了验证，该实例包含 6 种工件，每种工件各包含 6 道工序，调度的设备集共有 10 个设备。各工件的工序组成、工序的设备选择以及对应的加工时间如下表 6-1 所示：

表 6-1 6 个工件 10 个设备的 FJSP 调度实例[48]

工件	工序	加工设备集									
		m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
p_1	$o_{1,1}$	−	−	4	−	10	−	−	12	−	15
	$o_{1,2}$	−	6	−	8	−	−	−	−	−	11
	$o_{1,3}$	−	−	−	−	−	9	−	−	4	−
	$o_{1,4}$	−	−	−	7	−	−	−	−	−	−
	$o_{1,5}$	−	−	−	−	−	6	−	−	−	−
	$o_{1,6}$	−	−	−	−	−	−	13	−	−	14
p_2	$o_{2,1}$	10	−	−	−	−	15	−	−	−	−
	$o_{2,2}$	−	−	8	−	−	−	−	6	−	−
	$o_{2,3}$	−	−	−	5	−	−	7	−	−	−
	$o_{2,4}$	−	−	−	4	−	8	−	−	3	−
	$o_{2,5}$	−	10	−	−	−	−	12	−	−	−
	$o_{2,6}$	−	−	−	4	−	−	7	−	−	−
p_3	$o_{3,1}$	−	−	9	−	−	−	−	−	−	14
	$o_{3,2}$	−	−	−	−	8	−	−	−	−	−

表 6-1 6 个工件 10 个设备的 FJSP 调度实例[48]（续）

p_3	$o_{3,3}$	−	9	−	−	−	−	−	6	−	−
	$o_{3,4}$	−	−	4	−	−	6	−	−	−	−
	$o_{3,5}$	−	−	7	−	−	−	−	−	−	9
	$o_{3,6}$	−	−	−	−	−	6	−	−	9	−
p_4	$o_{4,1}$	−	−	−	5	−	−	4	−	−	−
	$o_{4,2}$	−	−	6	−	−	−	5	−	-	−
	$o_{4,3}$	5	−	−	8	−	−	11	−	−	9
	$o_{4,4}$	−	3	−	−	11	12	−	5	−	10
	$o_{4,5}$	−	−	−	−	−	8	−	−	8	−
	$o_{4,6}$	1	−	−	5	−	−	7	−	−	8
p_5	$o_{5,1}$	6	−	−	−	−	3	−	−	−	−
	$o_{5,2}$	3	−	−	−	−	6	−	−	−	8
	$o_{5,3}$	−	−	9	−	−	−	−	−	−	11
	$o_{5,4}$	−	7	−	−	1	−	−	−	−	9
	$o_{5,5}$	5	−	−	14	−	−	−	−	−	−
	$o_{5,6}$	−	−	−	−	5	−	−	8	−	−
p_6	$o_{6,1}$	−	−	−	−	10	8	−	−	−	−
	$o_{6,2}$	−	−	−	3	−	−	−	−	−	3
	$o_{6,3}$	−	−	−	−	10	−	−	−	−	−
	$o_{6,4}$	−	−	−	−	−	9	−	−	−	10
	$o_{6,5}$	−	−	−	4	−	−	−	7	−	−
	$o_{6,6}$	−	−	13	−	−	15	−	−	−	14

本文的基于改进蚁群算法调度策略的参数设置如下表 6-2：

表 6-2 改进的蚁群算法的参数设置

参数名称	参数值
工件数量	6
蚂蚁数量 m	30
迭代次数 N	100

表 6-2 改进的蚁群算法的参数设置（续）

忽略信息素影响迭代次数	10
信息素权重系数α	2
能见度权重系数β	2
初始信息素浓度τ_0	1
每轮迭代释放信息素总量 Q	20
节点间信息素浓度最小值τ_{min}	0.1
节点间信息素浓度最大值τ_{max}	8
信息素挥发因子ρ	0.1

根据该调度实例，本文的基于蚁群算法的调度策略在最小化最大完成时间的单优化目标下执行，在本文的调度系统中的工厂一中设置 10 个设备对 6 个工件进行调度方案的计算，所得最优解 52，而文献[48]得到的最优解为 53，可见本文算法具有较强的求解性能。图 6-1 为本文调度系统的计算的最优解，对应的甘特图如图 6-2 所示。文献[48]的最优解对应甘特图如图 6-3 所示。

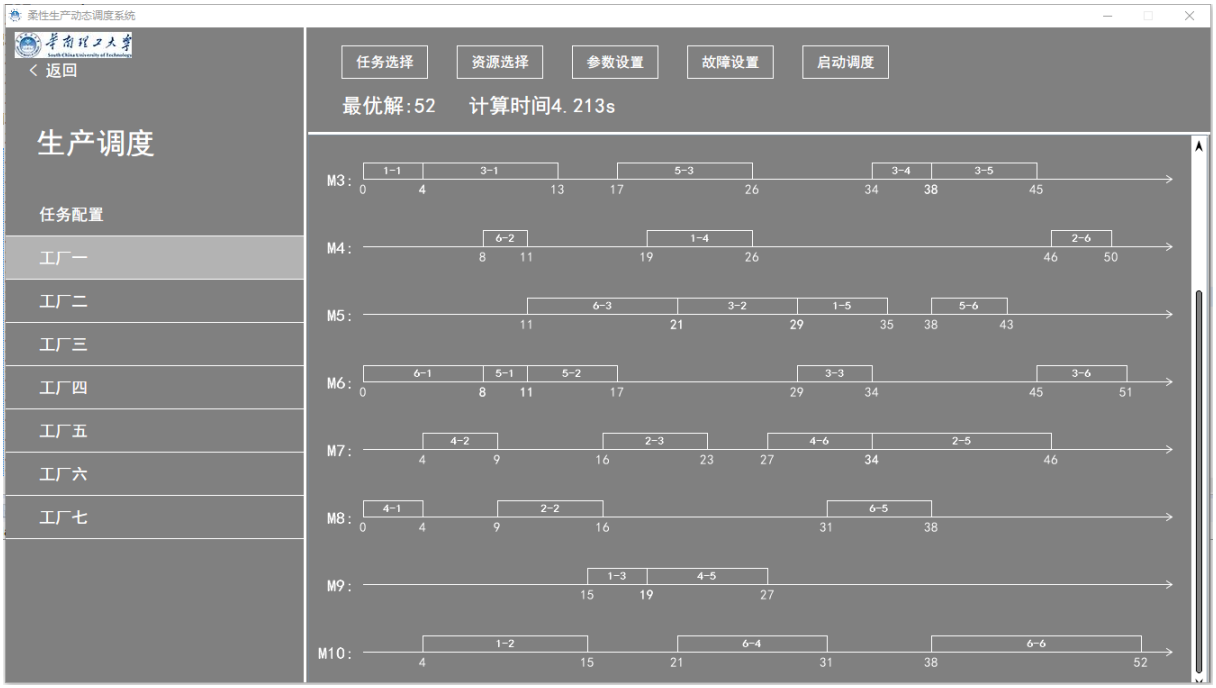


图 6-1 100 次迭代所得的最优解

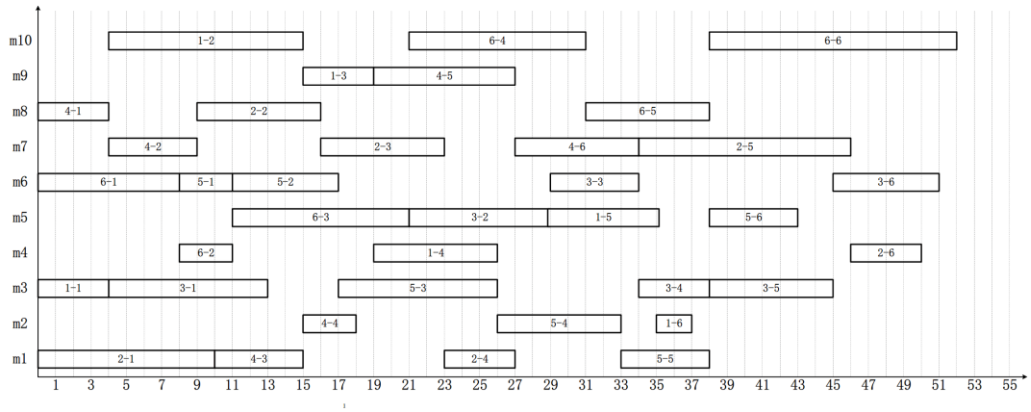


图 6-2 本文调度系统所求最优解对应的设备-时间甘特图

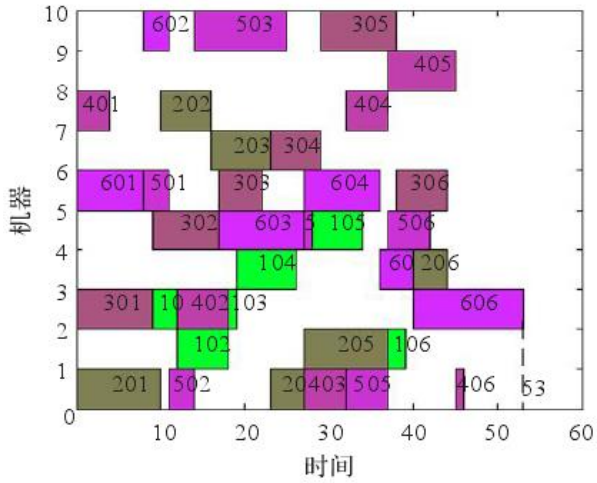


图 6-3 文献[48]的最优解对应的设备-时间甘特图[48]

本文使用基本蚁群算法（ACO）在相同的参数条件下进行 20 次试验，每次试验迭代次数为 100，所得结果求平均值，与本文改进的调度算法比较结果如下表 6-3 所示。20 次试验中平均数值的收敛曲线如下图 6-4 所示。

表 6-3 改进的蚁群算法与基本蚁群算法 20 次实验的结果对比

算法	最优解	前 100 次迭代的平均解	最差解
ACO	53	53.45	55
本文的算法	52	52.789	54

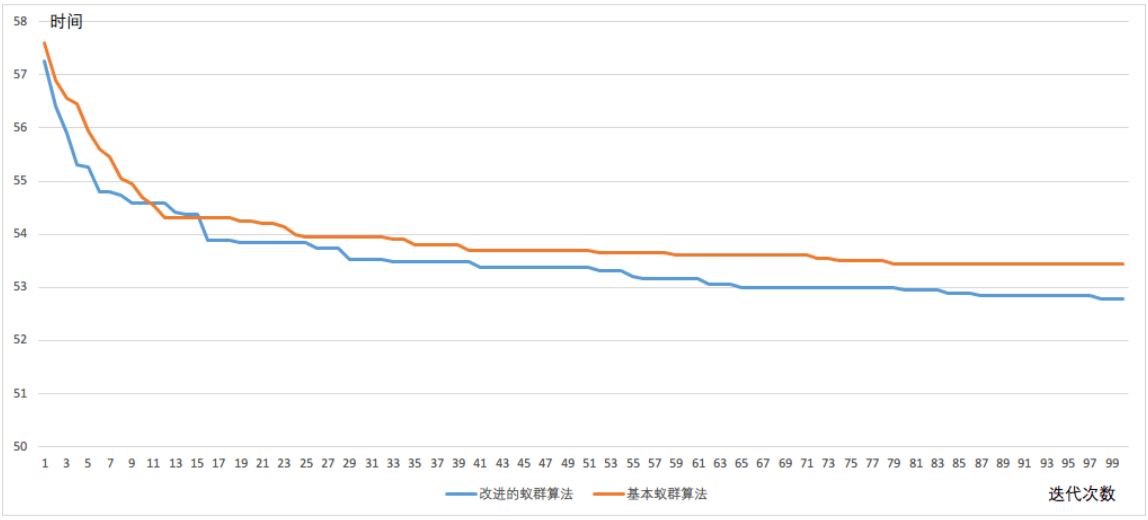


图 6-4 改进的蚁群算法与基本蚁群算法 20 次实验的平均收敛曲线

由以上数据和图表可知，本文的算法在初始收敛速度上稍慢于 ACO，但具有更好的求解性能，能收敛得到更优的最优解。由此可见，本文对于基本蚁群算法的改进是有效的，能够避免算法早熟的同时，能保证较好的求解质量。

为了进一步验证本文调度算法的求解质量，需要与其他文献进行对比。本文选取了 Kacem 基准问题[49]中的 8×8 和 10×10 调度实例进行实验，与其他文献的比较结果如下表 6-4 所示。

表 6-4 Kacem 算例的对比结果

算例	指标	ACO	GA	文献[50]	文献[51]	文献[52]	文献[53]	本文算法
8×8	最优解	15	16	16	14	14	14	14
	平均解	17.627	18.3	-	-	15.5	15.3	15.413
10×10	最优解	7	7	-	7	7	7	7
	平均解	8.35	8.167	-	-	7.6	7.5	7.357

本文的调度系统在 8×8 和 10×10 两个算例的调度结果如下：

（1）Kacem 基准问题的 8×8 算例

该实例中，本文调度系统设置蚂蚁数量为 54，迭代次数为 100，其他参数不变，所得最优解为 17，与表中各文献的最优解相同，调度结果以及该结果对应的甘特图分别如图 6-5 和 6-6 所示。图 6-7 显示了最优解对应的收敛曲线，可知算法在第 23 次迭代收敛得到最优解。

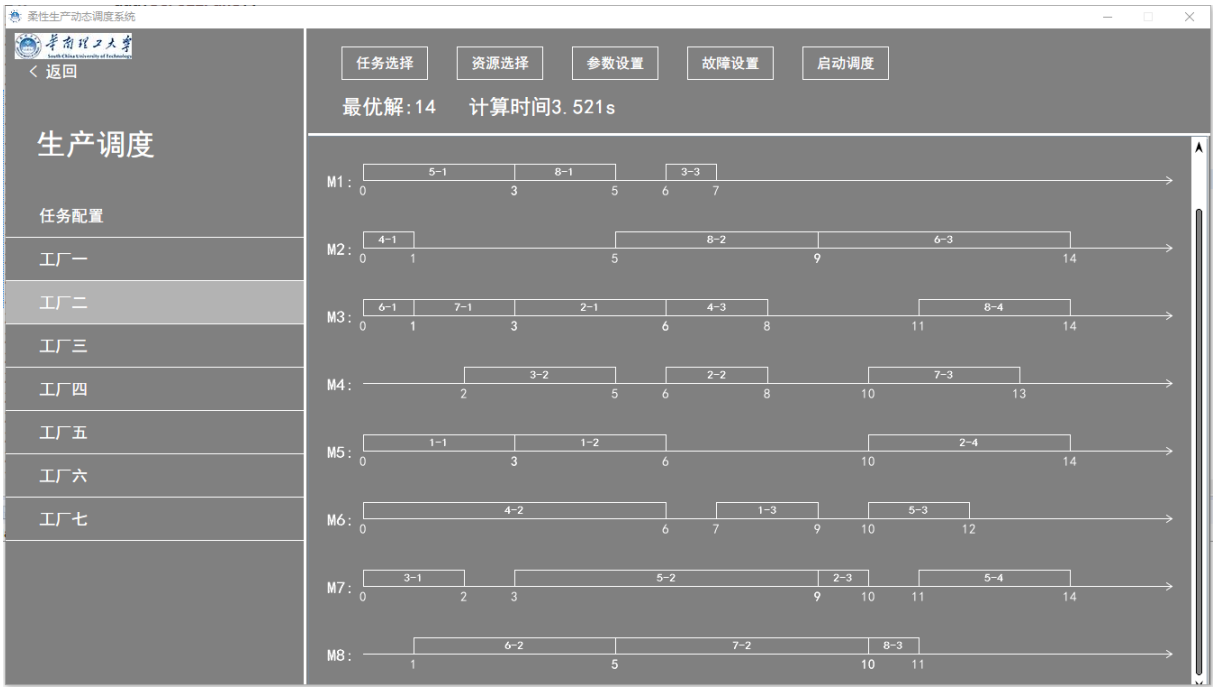


图 6-5 8×8 算例的最优解

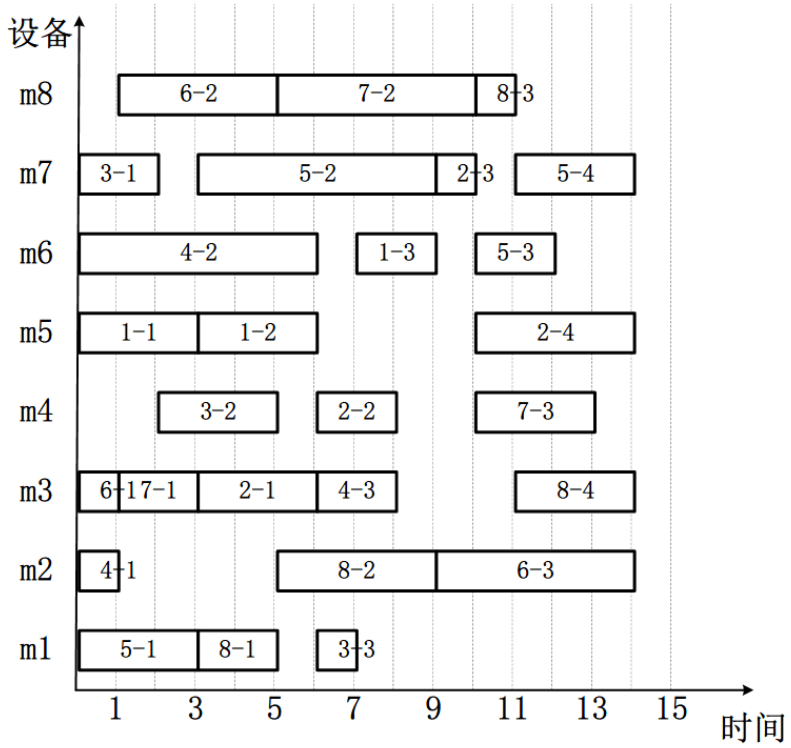


图 6-6 Kacem 的 8×8 算例最优解的甘特图

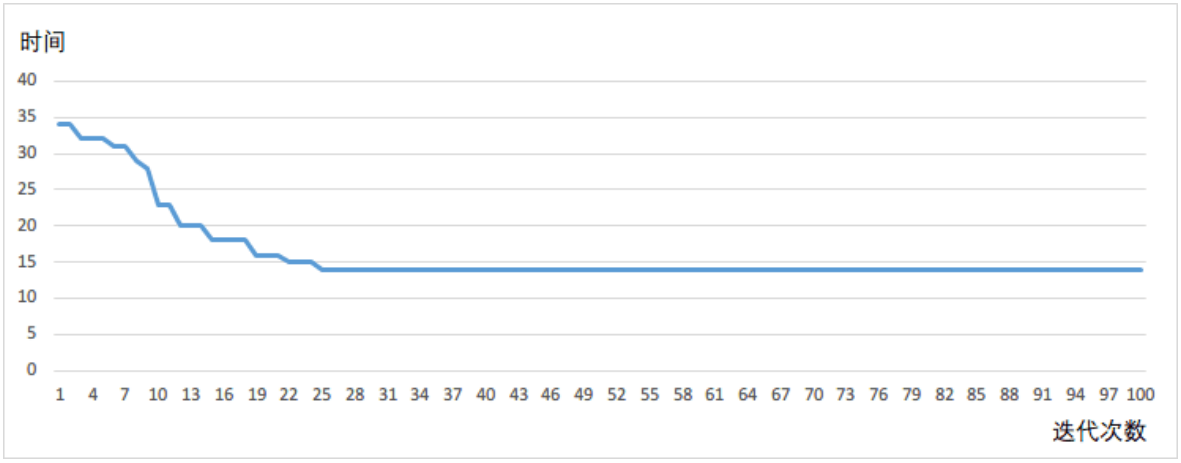


图 6-7 8×8 算例最优解的收敛曲线

（2）Kacem 基准问题的 10×10 算例

该实例中，本文调度系统设置蚂蚁数量为 60，迭代次数为 100，所得最优解为 7，调度结果以及该结果对应的甘特图分别如图 6-8 和 6-9 所示。图 6-10 显示了最优解对应的收敛曲线，从中可知算法在第 15 次迭代收敛得到了最优解。

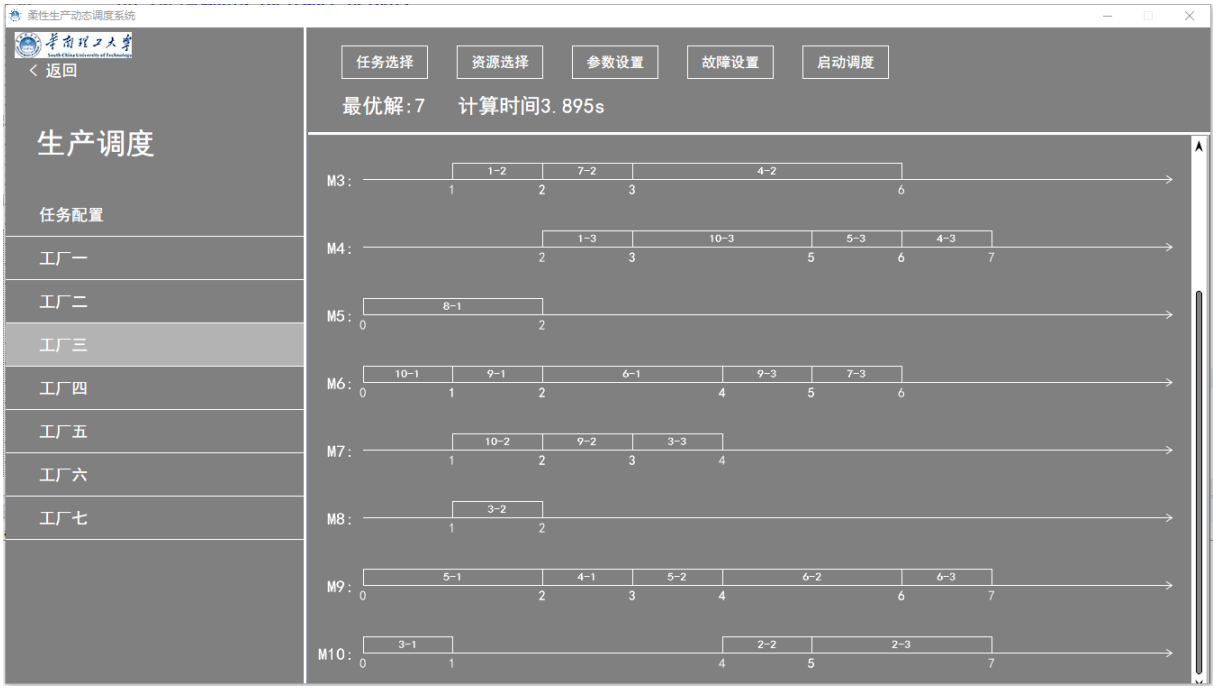


图 6-8 Kacem 的 10×10 算例的最优解

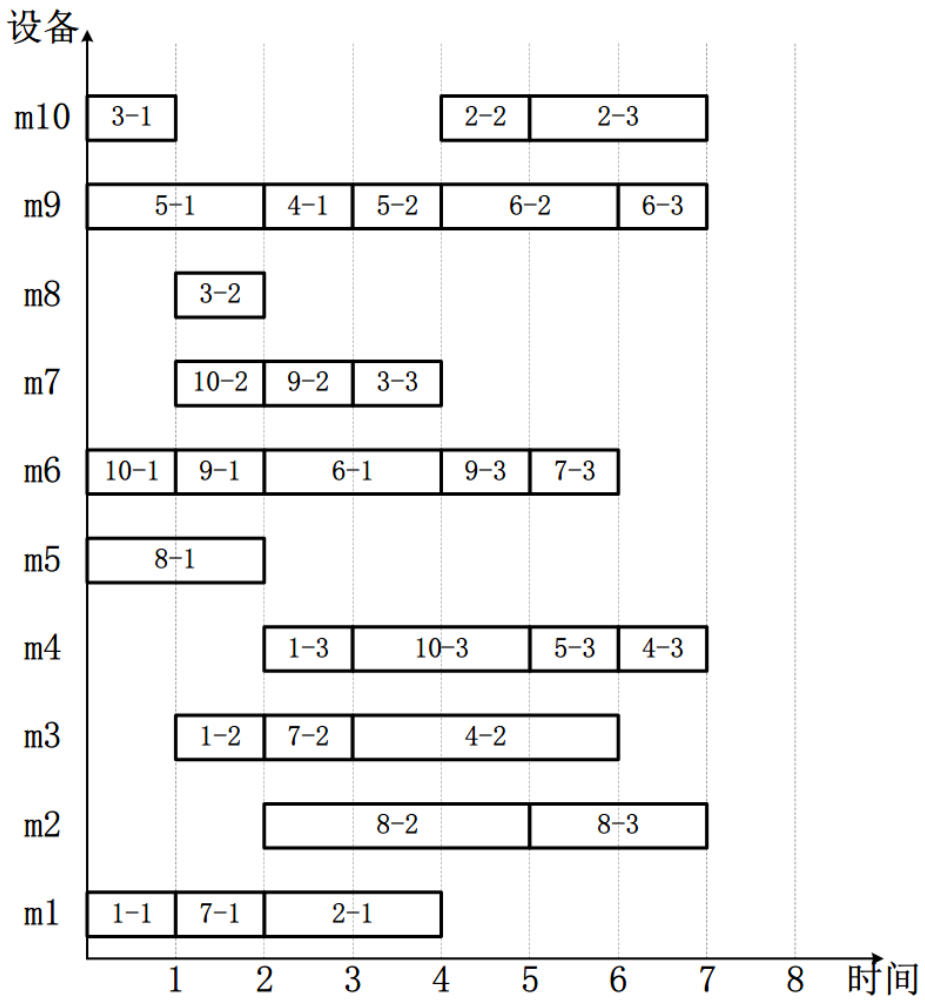


图 6-9 Kacem 的 10×10 算例最优解的甘特图

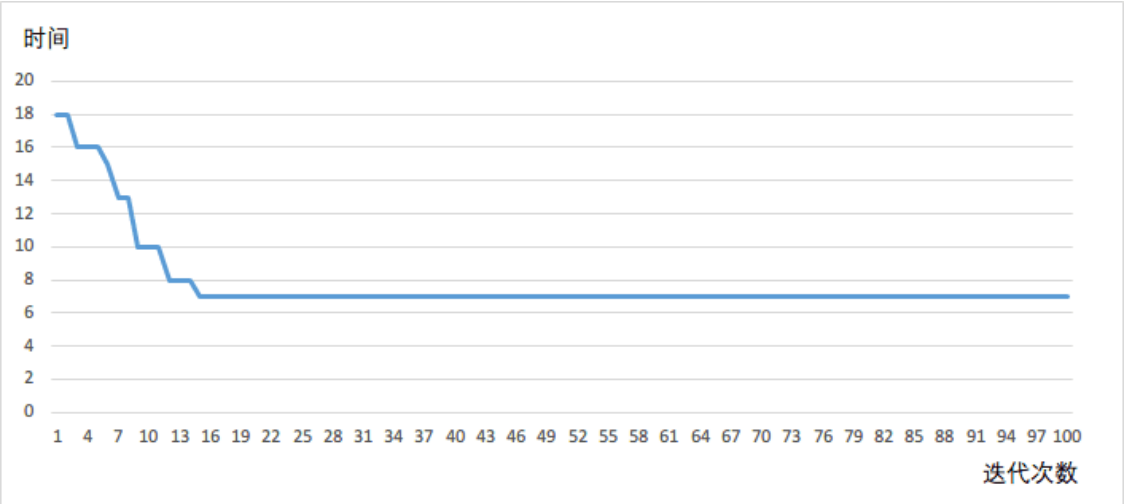


图 6-10 Kacem 的 10×10 算例最优解的收敛曲线

根据对 Kacem 基准问题的两个实例的实验结果可知，本文改进的蚁群算法能保证较高的可行性和效率，能够在较低的循环迭代中计算得到最优解。

6.3 实验二：验证基于多 Agent 的生产任务跨区域分解策略

本文在车间调度过程中把生产任务的分解和分配进行一定程度的分离，其中通过跨区域生产任务分解流程，在保持工件级任务粒度的前提下，得到若干满足任务需求的设备集，每个设备集中的设备配置均有能力完成对完整任务加工的能力。该过程根据任务粒度和制造资源的地区分布特性，减少了调度设备集中设备的规模，取而代之的是多个规模较小的设备集。而后通过本文改进的蚁群算法基于每个设备集进行求解，最终从中筛选完工时间最短的调度方案。因此该实验用于跨区域分解策略配合本文改进的蚁群算法能否完成生产任务在各地区工厂中的最优分配，从而合理利用企业的分布式制造资源。

文献[54]根据云制造环境下资源的分布式特征，建立资源虚拟化框架，并根据 FJSP 的调度特点建立工序和设备选择约束，以加工时间和最小为目标，使用启发式-蚁群算法进行资源的配置。该文献基于两家企业的制造资源配置，提供了一个柔性作业车间调度实例，该实例包含 3 个工件，3 个设备，其中每个工件包含三道工序，工序的设备选择及对应的加工时间如下表 6-5。

表 6-5 3 个工件 3 台设备的 FJSP 调度实例[54]

工件	工序	设备选择和对应的加工时间		
		m_1	m_2	m_3
p_1	$o_{1,1}$	2	3	2.5
	$o_{1,2}$	1	1	0.5
	$o_{1,3}$	3	1.5	0.5
p_2	$o_{2,1}$	1	1.5	2
	$o_{2,2}$	1	2	0.5
	$o_{2,3}$	3	1.5	0.5
p_3	$o_{3,1}$	2	1	1
	$o_{3,2}$	1	0.5	2
	$o_{3,3}$	0.5	1	1

该实例中，两家企业的资源配置相同，均配备完整的三个设备，其中企业 1 的设备 m1 和设备 m3 分别在[2,2.5]和[2,3]的时间段被占用，如图 6-11 所示。企业 2 的设备均为空闲，加工序列为空。

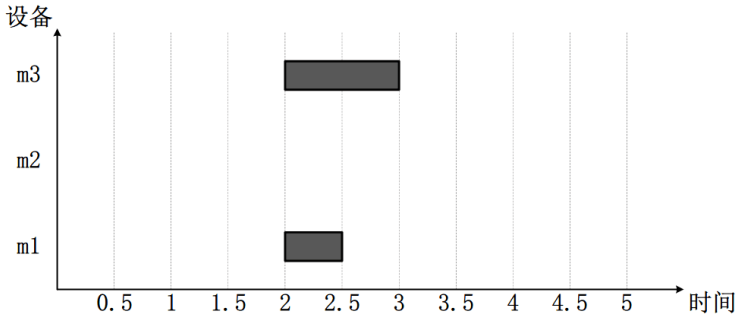


图 6-11 企业 1 设备原始加工序列甘特图

在本实验中，基于以上调度实例把两家企业对应为位于不同地区的两间加工工厂，对应本文调度系统中的工厂四和工厂五，两间工厂各包含三个设备 m1,m2 和 m3，其中对工厂四的设备设置如图 6-11 的初始工序的占用。实验初始化时，把三个工件任务封装为一个完整的产品任务，由全局管理 Agent 广播至两个子管理 Agent。由于两间工厂设备配置相同且均有三个设备，因此在跨区域分解过程中两个资源 Agent 组均会得到设备集，而后把设备集中 Agent 模拟的设备加工序列以及生产任务发送到算法 Agent，算法 Agent 计算调度方案后返回到对应的资源 Agent 组。工厂四和工厂五最终所得的最优解分别如图 6-12 和图 6-13 所示，对应最优解甘特图如图 6-14 和 6-15 所示。其中工厂四的最优解为 4，工厂五的最优解为 3.5；由于工厂四的设备存在工序占用，调度方案在完工时间上大于工厂五，因此最终全局管理 Agent 从中选择工厂五执行生产计划。

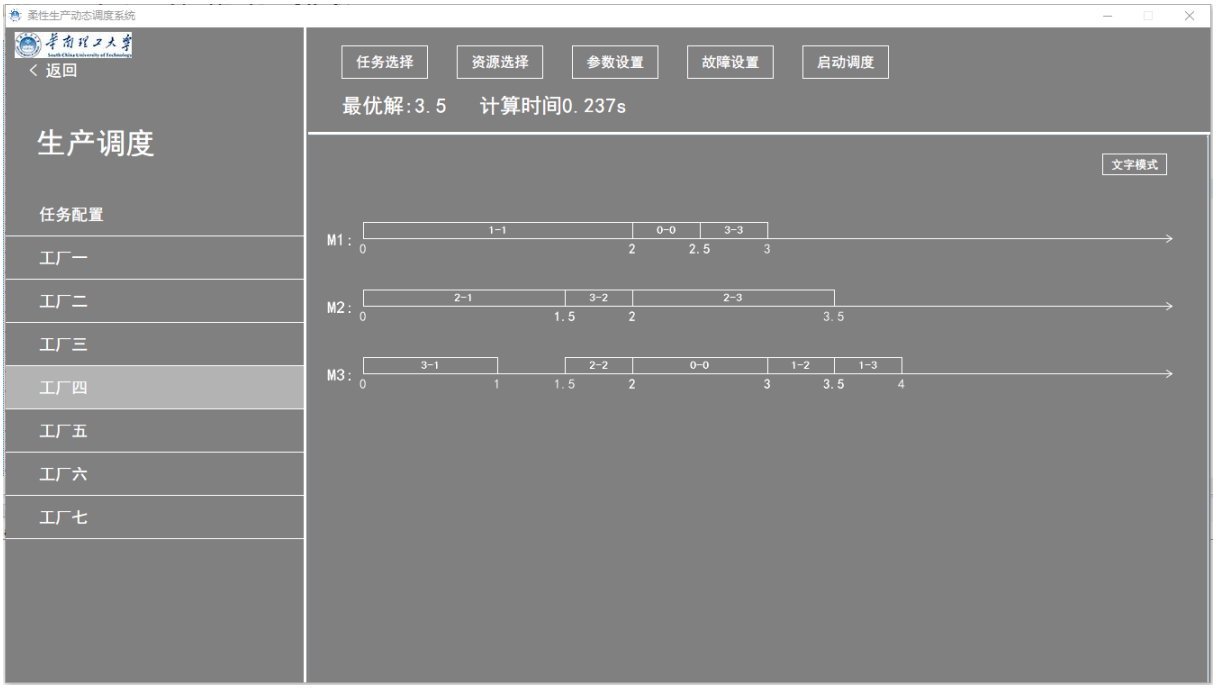


图 6-12 工厂四的调度结果

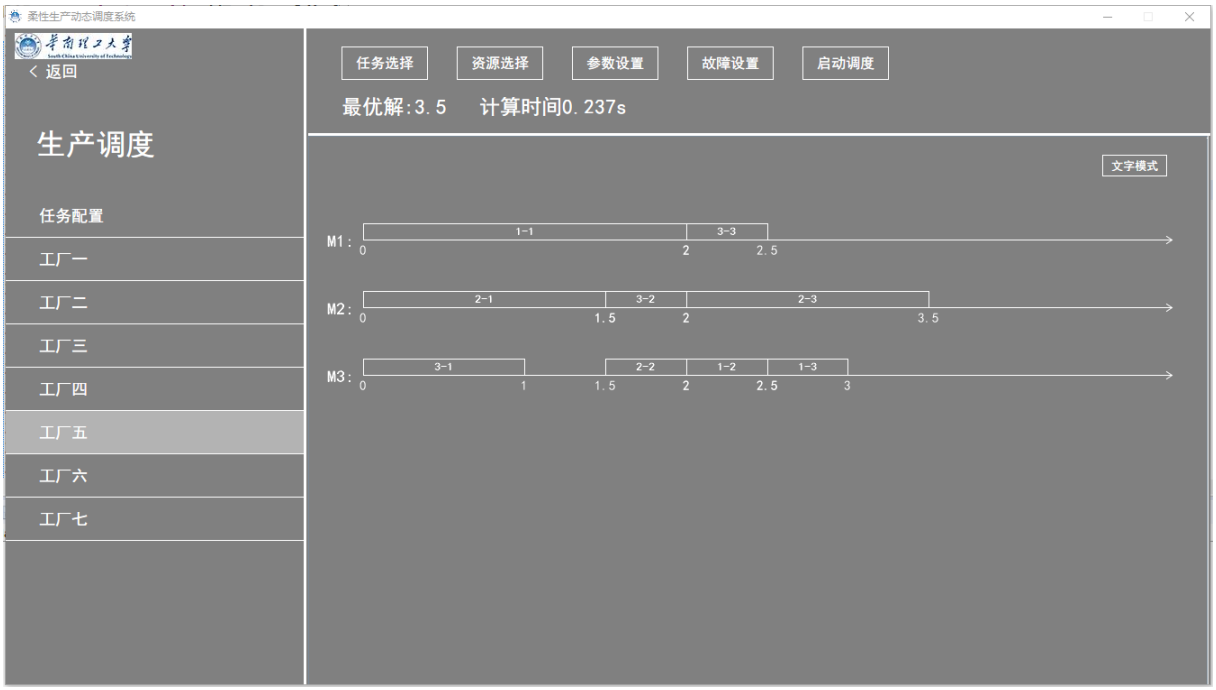


图 6-13 工厂五的调度结果

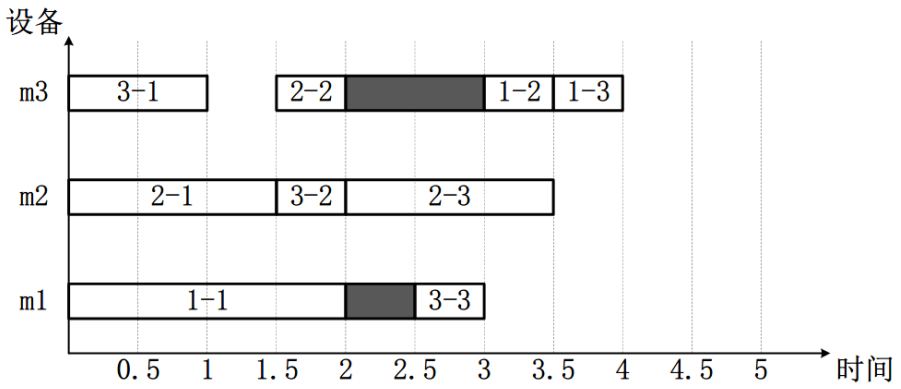


图 6-14 工厂四的最优解甘特图

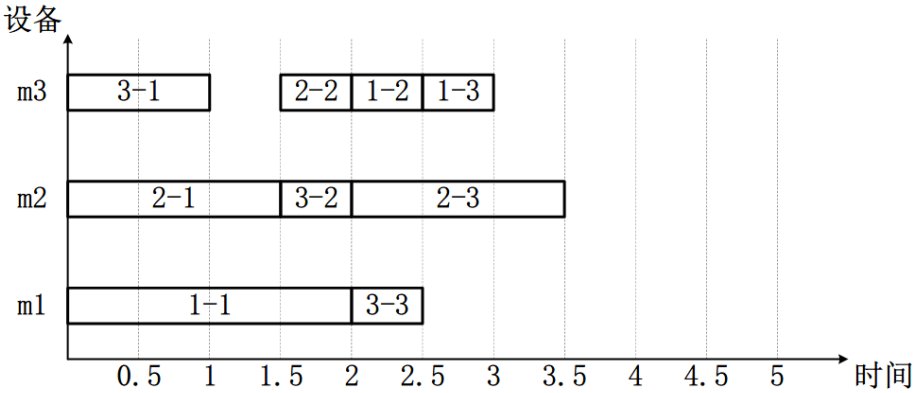


图 6-15 工厂五的最优解甘特图

由此可见，本文的基于多 Agent 的跨区域生产任务分解策略能够根据生产任务对制

造资源的要求以及企业跨地区的资源分布式特点得到对应的设备集合，配合本文改进的蚁群调度算法，能够提高企业分布式资源的利用率。

6.4 实验三：验证设备故障条件下的多 Agent 异常调度策略

本文的基于多 Agent 的动态调度系统是通过各 Agent 之间的协作来完成共同的任务，在异常状况发生，Agent 接收异常信息后，通过 Agent 间高效的通讯，信息在各类型 Agent 中进行传递以及采取对应的行为，从而达到快速及时响应异常状况的目的。该实验通过比较在异常情况发生时，根据系统的反应时间、异常调度所需的时间以及跟正常调度方案的差异来验证系统的重调度协商策略的可行性。

本实验基于文献[55]提供的 6×5 车间调度实例，该实例的设备集包含 5 个设备，工序的设备选择和对应的加工时间如下表 6-6 所示。文献[55]的作者在基于多 Agent 的动态制造系统中，定义了资源 Agent、工件 Agent 和管理 Agent 三类 Agent，当故障发生时，通过基于合同网的 Agent 间协商机制完成故障资源 Agent 上工序任务的转移。

表 6-6　6 个工件 5 台设备的 FJSP 调度实例[55]

工件	工序	设备选择以及对应的加工时间				
		m_1	m_2	m_3	m_4	m_5
p_1	$o_{1,1}$	3	4	–	–	–
	$o_{1,2}$	–	8	–	9	–
	$o_{1,3}$	–	–	9	–	7
	$o_{1,4}$	–	5	–	4	–
	$o_{1,5}$	–	3	–	–	3
p_2	$o_{2,1}$	6	–	–	8	–
	$o_{2,2}$	–	8	–	6	–
	$o_{2,3}$	5	–	4	–	–
	$o_{2,4}$	–	–	2	–	3
	$o_{2,5}$	–	3	–	–	4
p_3	$o_{3,1}$	–	–	4	–	6
	$o_{3,2}$	5	–	7	–	–
	$o_{3,3}$	–	7	9	–	–

表 6-6 6 个工件 5 台设备的 FJSP 调度实例[55]（续）

p_3	$o_{3,4}$	5	5	–	–	–
	$o_{3,5}$	–	–	–	6	6
p_4	$o_{4,1}$	–	9	–	–	7
	$o_{4,2}$	–	3	5	–	–
	$o_{4,3}$	–	4	–	6	–
	$o_{4,4}$	–	–	–	3	–
	$o_{4,5}$	–	3	–	–	7
p_5	$o_{5,1}$	6	4	–	–	–
	$o_{5,2}$	-	-	7	5	–
	$o_{5,3}$	–	–	–	7	9
	$o_{5,4}$	8	–	7	–	–
	$o_{5,5}$	–	–	5	5	–
p_6	$o_{6,1}$	–	3	7	–	–
	$o_{6,2}$	9	–	-	7	–
	$o_{6,3}$	–	–	6	–	9
	$o_{6,4}$	–	–	6	7	–
	$o_{6,5}$	–	4	–	4	–

根据该实例，该实验在本文的调度系统工厂六的车间 1 中设置 5 台设备，分别为 m_1,m_2,m_3,m_4 和 m_5，用于对该实例的 6 个工件进行调度方案的计算，调度结果如图 6-16，对应的甘特图如图 6-17 所示。

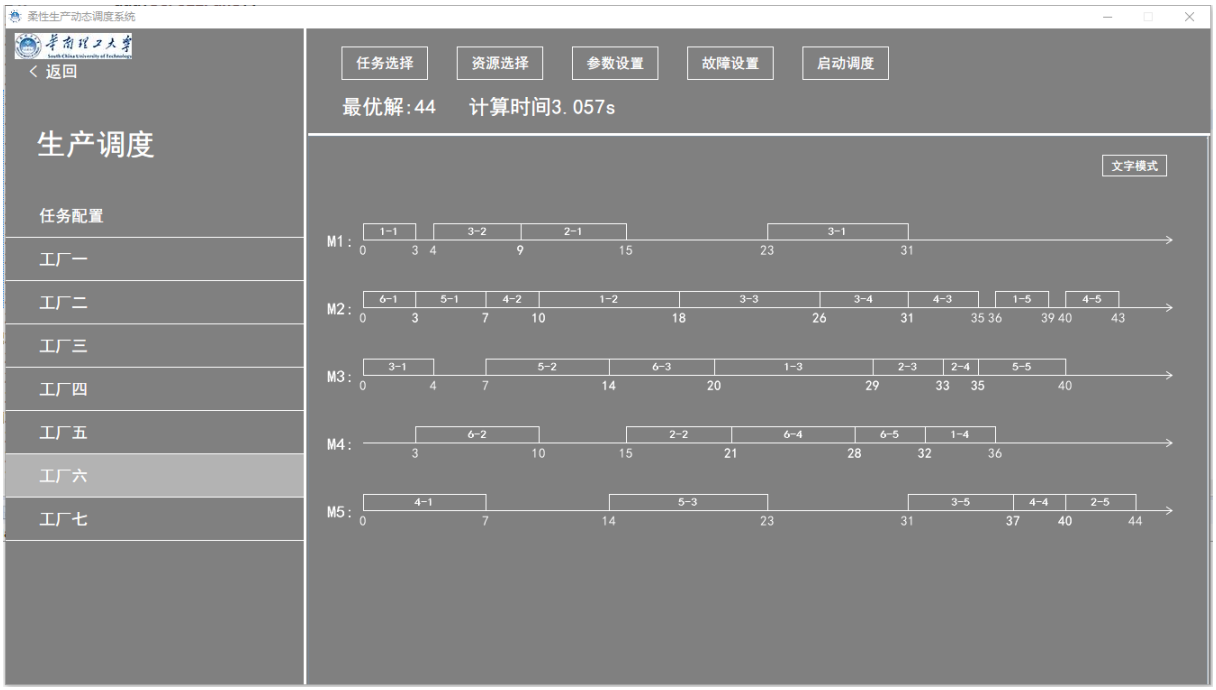


图 6-16 系统调度结果

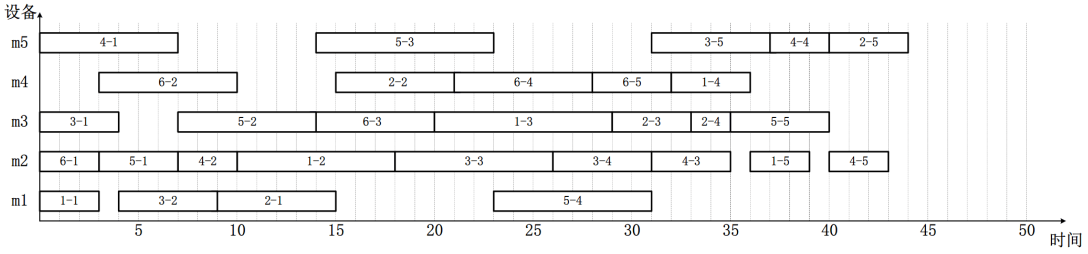


图 6-17 系统调度结果对应的甘特图

该实验中，设备故障由监控 Agent 触发，而后把故障信息发送到子管理 Agent，子管理 Agent 通知其下的车间 Agent 进行故障设备 Agent 的移除，而后进行工序任务的回收工作，最终重新封装为新的工件任务在资源 Agent 组间进行分解分配，得到重调度方案。本文把故障发生到新的工件任务封装完成定义为系统的反应时间，反应时间的大小体现了系统对设备故障是否灵敏；工件任务封装完成到异常调度方案生成定义为异常调度时间，体现系统调度算法的效率；异常调度方案与正常调度方案的完工时间差值用于衡量系统异常调度方案的有效性。

根据对 FJSP 中车间调度的假设，工序加工过程中不可中断，若由于设备故障原因导致设备停工，工序加工不完整，则对应的工件报废，原有的工件任务作为加急工件任务参与异常调度，因此该实验在设备故障触发上进行如下设置：

（1）仅在设备空闲时间触发设备故障，若工序$o_{i,j}$，$o_{i',j'}$为某设备 k 加工工序中的相邻

工序，当时间 t 满足以下条件时，可触发设备故障:

$$e_{i,j,k} \leq t \leq s_{i',j',k} \tag{6-1}$$

（2）任意时刻只有一台设备触发设备故障。

（3）当设备加工序列为空时，即完成最后一道工序的加工后，不触发设备故障。

本实验基于甘特图 6-17 对应的调度方案进行，以 50%的概率随机触发空闲设备故障，在 100 次故障中测试的异常调度的反应时间、异常调度时间，时间差异的平均值如下表 6-7 所示。

表 6-8 设备故障条件下系统异常调度结果

时间段	故障次数	平均反应时间/s	平均异常调度时间/s	平均时间差
0-5	11	0.638	2.764	13.327
5-10	15	0.569	2.863	12.802
10-15	12	0.471	2.368	10.518
15-20	5	0.587	2.472	9.362
20-25	18	0.483	1.806	7.434
25-30	16	0.431	1.342	8.260
30-35	11	0.472	0.879	7.713
35-40	12	0.433	0.203	5.251
40-45	0	-	-	-

由上表数据可计算得到该实验平均反应时间为 0.511s，平均异常调度时间为 1.837s, 平均时间差为 9.252 并可得出以下结论：

（1）故障发生时间越早，所需的异常调度时间越长，这是由于后续未开始加工的工序较多，时间越早封装所得的新工件越接近原始工件任务，调度算法所需的计算量越大。

（2）反应时间与故障发生时间相关性小，这是由于反应时间主要由 Agent 之间的 Socket 通信决定，依赖 Agent 之间的通信质量和效率。工序回收以及新工件任务的封装所需时间基本是固定的，影响小

（3）当故障时间发生越早，异常调度方案与正常调度时间差越大，这是由于回收的工序多，且故故障设备无法用于任务分配，需由其他可替换设备用于工序任务的转移，因此异常调度结果对应的时间较长，所需要的计算时间也更长；若故障发生时间接近调度计划尾声，由于回收工序少，计算时间大幅降低，时间差对应也降低；

根据以上分析，系统在故障设备发生时反应时间和异常调度时间短，异常调度方案与正常调度时间差保持在较小的范围（平均时间差 9.25）内，说明系统对异常状况反应灵敏（平均反应时间 0.511s），异常调度执行时间短（平均异常调度时间 1.837s），对原生产计划影响小，具有实用价值，从而验证了在故障条件下的基于多 Agent 的异常调度策略的有效性。

6.5 本章小结

本章主要设计了三个实验用于验证基于多 Agent 的柔性生产调度系统的有效性，其中实验一验证本文改进的蚁群算法在 FJSP 中的可行性，实验二验证在各资源 Agent 组以及算法 Agent 的相互配合下，能否实现生产任务的跨区域分解与分配，实验三验证在设备故障条件下，各 Agnet 之间通过信息交互，能否快速完成异常调度流程，调整原调度方案。最后根据各个实验的实验结果和数据，证实了系统在分布式环境下能够合理地分配资源，快速响应异常状况，具有实用价值。

总结

制造业是国民经济的重要组成部分，为我国的经济的腾飞作出了巨大的贡献。如今市场需求复杂多变，制造业的生产模式已经逐渐向小批量多品种甚至是单件定制化生产转变，所生产加工的产品通常具备柔性，这需要企业具备高效高质量的车间调度算法。同时还需要注意到企业所处的生产环境通常充满诸多不确定性，包括客户订单的频繁更改、撤销以及设备故障等等，这些具有不可预料性的异常因素严重干扰企业正常的生产管理流程，为了保证企业生产的平稳高效进行，仅仅通过研究高效的调度算法是不足够的，还需从系统的角度出发，集成高效的调度算法，设计灵活的系统架构，提高系统对异常因素的响应速度，衔接企业生产管理的各个流程，实现企业生产管理的自动化与信息化。

本文在研究柔性生产调度调度系统中，考虑到企业所处的动态变化环境以及 FJSP 的复杂性，应用了多 Agent 理论构建柔性生产动态调度系统，所做工作如下：

（1）对 FJSP 的特点、研究现状以及现代制造企业所处的生产环境进行了深入的分析，对企业生产管理过程中的功能需求进行了划分，同时根据多 Agent 理论在复杂大型问题求解的优越性，构建了基于多 Agent 的柔性生产动态调度系统，该系统由管理 Agent、资源 Agent、算法 Agent、监控 Agent 以及工艺 Agent 组成，五类 Agent 根据企业资源的分布式特点构成了混合式的结构，在保证生产调度高效的同时，提高系统对异常信息的响应速度。

（2）设计了企业分布式制造环境下 Agent 的协作流程，通过定义了生产任务跨区域分解策略来帮助企业对异地性资源和技术的整合；定义了基于多 Agent 的异常调度策略来保证系统对异常因素的快速响应，维持企业生产管理的稳定性。

（3）把蚁群算法作为系统的调度算法，同时根据 FJSP 的特点以及基本蚁群算法收敛慢易早熟的确定进行了改进，主要通过更改基本蚁群算法的信息素更新规则以及蚂蚁的均匀分布来提高蚁群算法的适用性。

（4）开发了基于多 Agent 的柔性生产动态调度系统。根据企业生产管理的需求构建了系统的体系架构，使用 Java 的面向对象技术和多线程技术进行 Agent 模型的实现，最终提出系统各模块的设计与实现。该系统能够实现对用户信息、物料信息、工艺信息等的管理，同时提供生产调度模块进行柔性生产调度的仿真验证。

在基于多 Agent 的柔性生产动态调度系统的研究中，本文的系统仍然存在许多有待

完善的地方：

（1）由于条件所限，本文的调度系统是在单机环境下运行的，通过面向对象和多线程技术对 Agent 进行封装来模拟分布式环境，今后需加深对多 Agent 理论的研究，以互联网为媒介，实现真正的分布式多 Agent 系统。

（2）本文应用的是基于最小化最大完工时间的单目标优化的蚁群算法作为系统的调度算法，实际生产环境中还需考虑成本、设备负载等各种因素，因此今后可结合多种调度算法对多目标优化的柔性生产调度问题开展研究，提高动态调度系统的实用性。

（3）需要把本文的调度系统真正用于实际的生产管理中，方可从企业实际生产面临的难题中获取需求，提高系统的实用性，例如工厂车间中的设备故障所造成的影响应当是多样的，需要联合实际采取对应的措施。

参考文献

[1] 徐梦周, 贺俊. 第三次工业革命的特征及影响[J]. 政策瞭望, 2012(10):46-47.

[2] 吕铁, 邓洲. 第三次工业革命的技术经济特征[J]. 中国党政干部论坛, 2013(10):6-10.

[3] 纪建强等. 第三次工业革命:特征、影响及应对战略[J]. 管理现代化, 2015, 35(1):127-129.

[4] 黄群慧,贺俊.中国制造业的核心能力、功能定位与发展战略——兼评《中国制造 2025》[J].中国工业经济,2015(06):5-17.

[5] 左乐. 不确定环境下柔性作业车间的多目标动态调度研究[D].北京交通大学,2015.

[6] 陶永,王田苗,李秋实,赵罡.基于“互联网+”的制造业全生命周期设计、制造、服务一体化[J].科技导报,2016,34(04):45-49.

[7] 刘想德. 作业车间实时调度若干关键问题研究[D].重庆大学,2013.

[8] 苏凯凯. 云制造环境下的制造资源优化配置方法研究[D].北京交通大学,2017.

[9] 周恺,纪志成.关于柔性作业车间调度问题的仿真研究[J].计算机仿真,2016,33(03):282-287+375.

[10]汪俊亮,张洁,秦威,银莉,陈定方.加工时间不确定的柔性作业车间鲁棒调度方法[J].中国机械工程,2015,26(05):627-632.

[11]Zhang Q, Manier H, Manier M A. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times[J]. Computers & Operations Research, 2012, 39(7):1713-1723.

[12]王万良,范丽霞,徐新黎,赵燕伟,张静.多目标差分进化算法求解柔性作业车间批量调度问题[J].计算机集成制造系统,2013,19(10):2481-2492.

[13]刘韵,胡毅,房超,罗企.解决柔性车间作业调度问题的侦查包围搜索算法[J].组合机床与自动化加工技术,2015(11):124-128.

[14]王雷,蔡劲草,唐敦兵,李明.基于改进遗传算法的柔性作业车间调度[J].南京航空航天大学学报,2017,49(06):779-785.

[15]Moslehi G, Mahnam M. A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search[J]. International Journal of Production Economics, 2011, 129(1):14-22.

[16]赵博选,高建民,陈琨.求解多目标柔性作业车间调度问题的两阶段混合 Pareto 蚁群算

法[J].西安交通大学学报,2016,50(07):145-151.

[17]Rossi A, Dini G. Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method[J]. Robotics and Computer-Integrated Manufacturing, 2007, 23(5):503-516.

[18]Huang R H, Yang C L, Cheng W C. Flexible job shop scheduling with due window—a two-pheromone ant colony approach[J]. International Journal of Production Economics, 2013, 141(2):685-697.

[19]Chiang T C, Lin H J. A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling[J]. International Journal of Production Economics, 2013, 141(1):87-98.

[20]Pereira I, Madureira A. Self-Optimizing A Multi-Agent Scheduling System: A Racing Based Approach[J]. 2016.

[21]Pereira I, Madureira A, Oliveira P M. Case-based Reasoning for Meta-heuristics Self-Parameterization in a Multi-Agent Scheduling System[C]// International Symposium on Computational Intelligence for Engineering Systems. 2013.

[22]Gozzi A, Paolucci M, Boccalatte A. A multi-agent approach to support dynamic scheduling decisions[C]// International Symposium on Computers and Communications, 2002. Proceedings. ISCC. IEEE, 2002:983-988.

[23]Lim M K, Zhang Z. A multi-agent based manufacturing control strategy for responsive manufacturing[J]. Journal of Materials Processing Technology, 2003, 139(1–3):379-384.

[24]Adhau S, Mittal M L, Mittal A. A multi-agent system for distributed multi-project scheduling: An auction-based negotiation approach[J]. Engineering Applications of Artificial Intelligence, 2012, 25(8):1738-1751.

[25]杨陇苗,王玉洁.基于多 Agent 的 MES 系统调度[J].信息系统工程,2017(02):120-121+124.

[26]王芊博,张文新,王柏琳,吴子轩.基于 Agent 的混合流水车间动态调度系统[J].计算机应用,2017,37(10):2991-2998.

[27]宋娟.多 Agent 分布式车间动态调度仿真系统研究[J].制造业自动化,2011,33(24):57-58+74.

[28]任海英,邹艳蕊.基于多 Agent 的柔性作业车间预先/重调度系统[J].武汉理工大学学报

(信息与管理工程版),2012,34(01):69-73.

[29]Wooldridge M, Jennings N R. Agent theories, architectures, and languages: A survey[M]// Intelligent Agents. Springer Berlin Heidelberg, 1994:408-431.

[30]Jennings N R, Wooldridge M J. Agent Technology: Foundations, Applications and Markets.[J]. Renewable Energy Technologies, 1998, 2(1):43–51.

[31]刘勇. 多 Agent 系统理论和应用研究[D]. 重庆大学, 2003.

[32]赵龙文, 侯义斌. 多 Agent 系统及其组织结构[J]. 计算机应用研究, 2000, 17(7):12-14.

[33]李成海, 黄必清. 基于属性描述匹配的云制造服务资源搜索方法[J]. 计算机集成制造系统, 2014, 20(6):1499-1507.

[34]张执南, 李响, 陈斌, 等. 企业分布式资源环境的概念设计[J]. 机械设计与研究, 2013, 29(5):1-3.

[35]张利平. 作业车间预反应式动态调度理论与方法研究[D].华中科技大学,2013.

[36]杨娜. 多 agent 生产调度系统的设计与实现[J]. 信息技术与信息化, 2014(9):191-192.

[37]Ponsich A, Coello C A C. A hybrid Differential Evolution-Tabu Search algorithm for the solution of Job-Shop Scheduling Problems[M]. Elsevier Science Publishers B. V. 2013.

[38]张国辉, 王永成, 张海军. 多阶段人机协同求解动态柔性作业车间调度问题[J]. 控制与决策, 2016, 31(1):169-172.

[39]刘轩,尚鋆,白翱.设备故障驱动的作业车间生产任务重调度方法研究[J].制造业自动化,2016,38(12):26-30+60.

[40]Dorigo M, Gambardella L M. Ant Colony System: A cooperative learning approach to the traveling salesman problem[C]// IEEE Transactions on Evolutionary Computation. 1996:53 - 66.

[41]田松龄,陈东祥,王太勇,刘晓敏.一种异步蚁群算法求解柔性作业车间调度问题[J].天津大学学报(自然科学与工程技术版),2016,49(09):920-928.

[42]王硕. 基于改进蚁群算法的作业车间调度研究[D].华东理工大学,2013.

[43]刘志虎. 基于改进蚁群算法的柔性车间调度研究[D].安徽工程大学,2016.

[44]袁豪. 旅行商问题的研究与应用[D].南京邮电大学,2017.

[45]Dorigo, Marco, Di C, et al. Ant algorithms for discrete optimization[J]. Artificial Life, 1999, 5(2):137.

[46]黄厦. 基于改进蚁群算法的柔性作业车间调度问题研究[D].昆明理工大学,2015.

[47]王媛. 多 agent 生产调度系统的设计与实现[D].大连理工大学,2005.

[48]薛宏全,魏生民,张鹏,杨琳.基于多种群蚁群算法的柔性作业车间调度研究[J].计算机工程与应用,2013,49(24):243-248+261.

[49]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 & Cybernetics Part C Applications & Reviews, 2002, 32(1):1-13.

[50]王小蓉, 李蓓智, 周亚勤,等. 基于混合遗传算法的柔性作业车间调度研究[J]. 现代制造工程, 2015(5):39-42.

[51]刘巍巍, 马雪丽, 刘晓冰. 面向柔性作业车间调度问题的改进变邻域搜索算法[J]. 计算机应用与软件, 2015(4):234-238.

[52]刘志勇, 吕文阁, 谢庆华,等. 应用改进蚁群算法求解柔性作业车间调度问题[J]. 工业工程与管理, 2010, 15(3):115-119.

[53]董蓉, 何卫平. 求解 FJSP 的混合遗传—蚁群算法[J]. 计算机集成制造系统, 2012, 18(11):2492-2501.

[54]何林燕. 云制造环境下柔性作业车间调度算法的研究[D].哈尔滨理工大学,2017.

[55]宋娟. 基于多 Agent 的制造车间动态调度系统研究[D].沈阳工业大学,2004.

攻读硕士学位期间取得的研究成果

一、已发表（包括已接受待发表）的论文，以及已投稿、或已成文打算投稿、或拟成文投稿的论文情况：

序号	作者（全体作者，按顺序排列）	题　　目	发表或投稿刊物名称、级别	发表的卷期、年月、页码	相当于学位论文的哪一部分（章、节）	被索引收录情况

注：在“发表的卷期、年月、页码”栏：
1 如果论文已发表，请填写发表的卷期、年月、页码；
2 如果论文已被接受，填写将要发表的卷期、年月；
3 以上都不是，请据实填写“已投稿”，“拟投稿”。
不够请另加页。

二、与学位内容相关的其它成果（包括专利、著作、获奖项目等）：

发明专利:《基于多 Agent 的面向订单的柔性生产动态调度系统》申请号：201810271536.5 申请时间：2018.3.29 申请人：张平，梁慰乐，陈昕叶，李方

致谢

三年的研究生学习生涯即将结束了，回想其中的点点滴滴，不禁感慨万千。首先，我由衷地感谢我的导师，张平教授，这篇论文以及各种相关实验都是在张老师的悉心指导下完成的。张老师严谨的学习态度，丰富的知识储备让我在科研学习中少走了很多弯路。在这三年中，我一直比较任性，给张老师添了很多麻烦，但老师一直不厌其烦地纠正我的学习态度，教导了我很多为人处世的道理，在我日常生活出现困难的时候给予了我很多鼓励和关怀，我心怀愧疚的同时又无比感恩，必将珍惜这份宝贵的人生经验。

同时还需要感谢实验室的李方老师和杜广龙老师。两位老师积极向上，对我的科研工作给予了很多指导，给我的生活带来了很多帮助，对我的成长意义重大。

此外，我还要感谢我的室友何子平和徐伟杰。两位室友非常优秀，非常善良，三年来我们一起上课，一起在实验室开展工作，没有出现一次矛盾，他们平时分享我的快乐，分担我的忧愁，一起成长，能够遇到两位志同道合的好朋友，我感到非常幸运。

另外我还要感谢我的师姐陈昕叶，我平时的科研工作以及论文的完成少不了师姐的功劳，师姐教会了我很多宝贵的学习方法，在算法理论和实验设计上给予了很多指导，在此我同样感到无比的感恩。同时，我还要感谢我的师弟庄焕嘉、赵俊宇、邵亨康，在团队实验过程中，他们给予了我很多技术支持。

感谢我的爸爸、妈妈、姐姐，他们一直是我成长路上的避风港，为我付出了很多，承担了很多，他们也一直包容我的任性，我的缺点，成长至今，我唯有通过不断的努力来回报家人对我无微不至的关怀。

最后，向各位论文评审专家致以深深的敬意，感谢你们在百忙之中抽出宝贵的时间对本文进行审阅，并给我提出宝贵的修改意见。

IV - 2 答辩委员会对论文的评定意见

<table>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">论文答辩日期：_____年____月____日
答辩委员会委员共_____人，到会委员_____人
表决票数：优秀（ ）票；良好（ ）票；及格（ ）票；不及格（ ）票
表决结果（打“√”）：优秀（ ）；良好（ ）；及格（ ）；不及格（ ）
决议：同意授予硕士学位（ ） 不同意授予硕士学位（ ）</td></tr>
<tr><td>答辩委员会成员签名</td><td>_____（主席） _____ _____
_____ _____ _____</td></tr>
</table>