**基于多Agent的柔性生产动态调度系统的设计与研究**

**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挑战，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管理和服务水平。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。

近年来，多Agent理论作为分布式人工智能的研究热点之一，被广泛地应用于复杂大型系统的构建中。在多Agent系统（MAS）中，每个Agent在功能和结构设计过程中被赋予了自主性、社会性和能动性，通过各个Agent之间的信息交流，能够稳定高效地进行复杂问题的求解。本文在对柔性作业车间调度问题的研究中，综合考虑企业所处的复杂多变的生产制造环境，引用了多Agent理论，构建了一种车间动态调度系统，本文主要工作包括：

（1）建立了基于多Agent的柔性作业车间动态调度系统的结构模型。本文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，通过对Agent决策库的设计发挥了Agent的自主性和社会性。其中资源Agent根据企业制造资源的分布特点以集中控制的分层式组成了资源Agent组，并与其他Agent构建了分布式系统的系统架构。

（2）提出了基于多Agent的任务协商策略，包括跨区域的生产任务分解策略以及异常状况下的重调度协商策略。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之间的协商和配合，实现对动态生产环境的快速响应。

（3）提出了基于改进蚁群算法的柔性作业车间调度算法。本文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、设备选择等方面进行了改进，以得到更优的调度结果。该算法封装在算法Agent中，向资源Agent组提供计算服务。

（4）实现了基于多Agent的柔性作业车间动态调度系统。本文根据系统定义的结构模型，设计了系统开发的软件架构并根据Agent的功能分配定义了Agent的内部模型。使用基于TCP/IP的Scoket通信机制以及Java的多线程和消息队列实现了Agent的通信机制，其中资源Agent组内部使用C/S通信模式，保证了Agent之间的通信质量和效率。

关键字：多Agent系统，柔性作业车间调度，动态调度，重调度

目录

[1 绪论 2](#_Toc510452456)

[1.1 研究背景 2](#_Toc510452457)

[1.2 研究目的与意义 2](#_Toc510452458)

[1.3国内外研究现状 3](#_Toc510452459)

[1.3.1 柔性作业车间调度问题的研究现状 4](#_Toc510452460)

[1.3.2 多Agent生产调度系统的研究现状 4](#_Toc510452461)

[1.4 论文的主要研究内容和组织结构 5](#_Toc510452462)

[2 基于多Agent的柔性作业车间调度系统结构的设计 6](#_Toc510452463)

[2.1 Agent与多Agent理论 6](#_Toc510452464)

[2.1.1 Agent的概念及特征 6](#_Toc510452465)

[2.1.2 多Agent系统的概念以及特征 7](#_Toc510452466)

[2.1.3 多Agent系统的结构 7](#_Toc510452467)

[2.2 动态车间调度系统功能与结构分析 8](#_Toc510452468)

[2.2.1 系统功能分析 8](#_Toc510452469)

[2.2.2 系统的结构分析 8](#_Toc510452470)

[2.3 各Agent的功能分析 9](#_Toc510452471)

[2.4 各Agent的决策库设计 11](#_Toc510452472)

[2.5 本章小结 12](#_Toc510452473)

[3 基于多Agent的任务协商策略 12](#_Toc510452474)

[3.1 基于多Agent的生产任务跨区域分解策略 13](#_Toc510452475)

[3.1.1 生产任务的形式化描述 13](#_Toc510452476)

[3.1.2 基于多Agent的订单任务跨区域分解策略的设计 14](#_Toc510452477)

[3.2 基于多Agent的重调度协商策略 15](#_Toc510452478)

[3.2.2 设备故障处理的重调度策略 16](#_Toc510452479)

[3.2.3 订单取消处理的重调度策略 17](#_Toc510452480)

[3.3 本章小结 17](#_Toc510452481)

[4 改进蚁群算法在柔性作业车间调度问题的应用 17](#_Toc510452482)

[4.1 制造系统中的车间调度问题 17](#_Toc510452483)

[4.1.1 车间调度问题的描述和分类 18](#_Toc510452484)

[4.1.2 柔性作业车间调度的工艺特点 18](#_Toc510452485)

[4.1.3 柔性作业车间调度问题的数学模型 21](#_Toc510452486)

[4.2 基本蚁群算法 22](#_Toc510452487)

[4.2.1 基本蚁群算法的原理 22](#_Toc510452488)

[4.2.2 基本蚁群算法在柔性作业车间调度问题中的执行流程 22](#_Toc510452489)

[4.2.3 基本蚁群算法在柔性作业车间调度问题中的改进 26](#_Toc510452490)

[4.4 本章小结 27](#_Toc510452491)

[5 基于多Agent的柔性作业车间调度系统的实现 27](#_Toc510452492)

[5.1 系统需求分析 27](#_Toc510452493)

[5.2 系统设计与实现 28](#_Toc510452494)

[5.2.1 开发技术和工具 28](#_Toc510452495)

[5.2.2 系统软件架构设计 28](#_Toc510452496)

[5.3 Agent间的通信机制与实现 29](#_Toc510452497)

[5.3.1 Agent与Java的多线程技术 29](#_Toc510452498)

[5.3.2 Agent通信模型设计 30](#_Toc510452499)

[5.4 Agent的模型设计 30](#_Toc510452500)

[6 仿真实验与结果分析 31](#_Toc510452501)

[6.1 实验环境 31](#_Toc510452502)

[6.2 实验一：基于改进蚁群算法的调度策略验证 32](#_Toc510452503)

[6.3 实验二：验证基于多Agent的跨区域生产任务分解策略 36](#_Toc510452504)

[6.4 实验三：验证设备故障条件下的多Agent重调度协商策略 39](#_Toc510452505)

[6.5 本章小结 42](#_Toc510452506)

# 1 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[6]，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不动态确定性[4-5]，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。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。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相应需求的，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，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。。针对以上不足，本文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，同时快速给出重调度方案，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。同时，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难问题，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使用中FJSP，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；人工智能算法中遗传算法、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，综合各类型算法的优点，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。李莉[9]等提出一种二阶式蚁群粒子群优化算法（TSAPO）应用于FJSP，该算法把求解过程分为两个阶段，分别使用改进的蚁群算法和粒子群算法基于不同的优化目标进行计算，该算法在保证了最优解的同时，具有收敛快，运行效率高的特点。Zhang和Maier[10]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。王万良[11]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。王晓娟[12]在FJSP的研究中把遗传算法和禁忌搜索算法融合，设计了禁忌搜索算法的邻域结构，发挥局部搜索能力来调整遗传算法中个体的行为，通过在实际生产中的应用，验证了算法在生产周期，成本和机器负载等多优化目标调度的可行性。王雷[13]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，提高进化速度。Mosle G和Mahnam M [14]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。陈成[15]在以最小化最大完工时间的单优化目标的FJSP研究中，把机器选择和工序排序进行分离，分别使用遗传算法和蚁群算法进行操作，并在优化过程中提取机器分配知识，加快算法的收敛。Rossi[16]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。Huang[17]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。

### 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。

WIENDH[18]把大型复杂的生产系统根据功能划分为多个Agent，通过定义Agent之间的结构和Agent之间的通信机制来完成生产制造任务。Duffie[19]把多Agent技术应用于分布式实时制造系统，定义了分层控制式结构以及基于合同网的通信规则来协调Agent的交互。Gozzi[20]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。Lim[21]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。Sycara[22]在分布式生产调度系统中，定义了多个监控Agent对生产环境、设备运行状况进行监控，当异常发生时发起警报，同时Agent还负责检测其他Agent的运行，当Agent陷入死锁、中断通信时通知系统管理员。肖晴[23]把多Agent系统用于企业集成，设计了混合型的系统架构以及垂直层次型的Agent模型，通过定义Agent的感知、知识、行为单元来保证Agent的自主性和社会性。郭文胜[24]等建立了基于MAS的虚拟车间调度作业方法，把设备、生产任务、工艺信息等虚拟化为对应的Agent，并应用了改进的蚁群算法用于求解资源的分配问题，在实际应用中取得了良好的结果。乔兵[25]应用MAS对工业制造系统建模，在Agent中以合同网的投标/竞标机制完成任务在各制造单元的分配。廖强[26]提出一种基于现场总线的多Agent调度系统建模方法，根据设备在车间的分布特点，建立了主从两类Agent，由主Agent向从Agent发布任务，根据各Agent的调度结果，从中选出最优的调度方案，最终实验结果证明了方法的可行性。

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互，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，根据生产管理中各个流程的结构和功能，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，其次根据柔性作业车间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，同时指定了各Agent的职责以及决策库设计。其次根据系统的业务流程，设计了基于多Agent的调度系统的分布式结构，其中资源Agent根据企业制造资源在各个地区的分布，构建为呈现分层控制式的资源Agent组。

第三章根据系统中Agent间的信息的传递方向，提出了基于多Agent的任务协商策略，包括跨区域生产任务分解协商策略以及重调度协商策略。前者根据产品工艺特点，设计了生产任务的模型，实现了生产任务在多个资源Agent组间的任务分解流程，该过程最终得到若干满足任务需求的设备集；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响应。

第四章基于最小化最大完工时间的优化目标，应用基本蚁群算法求解传统作业车间调度问题，并根据柔性作业车间调度问题的特点以及蚁群算法收敛速度慢和容易陷入局部最优的缺点，提出了FJSP中改进蚁群算法的模型。

第五章提出了基于多Agent的柔性作业车间动态调度系统的实现方式。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。同时根据各个Agent的功能设计了对应的内部模型。

**2 多Agent动态调度系统功能与结构设计**

**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并未有统一的标准。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Wooldrige和Jennings[23]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。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[24]：

（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：具有通信标识和通信语言，Agent之间具有通信的能力，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息进行唤醒，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位置。

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，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，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，通过系统内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, Multi-Agent System）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[24]的系统而言，多Agent系统部署和维护所需要的成本更低，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，对外界的变化反应更为灵敏，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非常适用于动态车间调度系统，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，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分布结构，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**

车间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要求，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，当生产任务发生变化时，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，根据工序在设备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，开启重调度，尽量降低原有调度方案对交货期、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，负责模拟设备的加工序列，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，通知系统开启重调度流程。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。GMA相当于系统内部的管理员，协调Agent间的通信和任务配合。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。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2-2 资源Agent组结构

工艺Agent: 负责对系统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设备选择等信息。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封装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[25]：

（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间通信是必须通过层次间逐层传递，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，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间的控制从属关系，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[25]，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-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，柔性车间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的仿真数据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，有全局管理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，无需经过层次间或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2-1系统结构图

## 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，定义了基于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，设计了系统的体系结构，而后为了发挥Agent的优越性，为各种类型的Agent进行了决策库的设计。

**3 柔性生产任务规划**

**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设备选择，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，来达到一个或多个优化目的。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，。如下表呈现了两个工件中工序的设备选择以及对应的加工时间。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 工件 | 工序 | 设备选择和对应的加工时间 | | |
| m1 | m2 | m3 |
| p1 | o1,1 | - | 5 | 3 |
| o1,2 | 9 | - | 7 |
| p2 | o2,1 | 7 | 6 | - |
| o2,2 | 3 | - | 2 |



图4-2 2\*2调度实例的析取图

图4-1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是典型的NP难问题。

**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：

（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）工序在同属一个地区的设备上加工时忽略转移时间。

数学模型中的各个变量定义如下：

（1） 工件数量

（2） 设备数量

（3） 工件集合，表示第i个工件

（4） 设备集合，表示第i个设备

（5） 工件i的工序集合，，其中表示工件i的第k道工序

（6） 工序的设备选择序列，表示工序能在设备上进行加工

（7） 工序的加工时间序列，与设备选择集合中元素一一对应，表示工序在设备上的加工时间。

（8） 设备i的加工序列，表示设备i上的第k道加工工序

（9） 工序的开始加工时间，表示工序在设备k的加工序列上的开始加工时间

（10） 工序的加工结束时间，表示工序在设备k的加工序列上的加工结束时间

（11） 工件i的完工时间

（12） 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，若必须在完成后才能开始加工，则有

（2）每个设备在任一时间只能加工一道工序，若工序和工序为设备k加工序列中的相邻工序，则有

（3）工序在加工过程中不可中止：

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：

**3.1 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。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。

**3.1.1 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以及工序间约束组成。PRT、CPT、WPT分别对应产品、工件和工序，工艺Agent中所记录产品、工件和工序在制定生产约束后均可封装为对应生产任务。PRT、CPT和WPT。

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CPT中DESCR描述了工件的精度和规格等信息，PT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,CRAFT}

COMP={MT1, MT2,..., MTi, ..., MTn}

COMP（Composition）是生产任务MT的子任务集合，其中MTi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。CRAFT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，CRAFT可指定了工件级子任务的生产对象的组装顺序；当MT粒度为CPT时，CRAFT指定了工序级子任务加工的顺序约束。

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。

**3.1.2 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、尺寸等要求以及产品的工艺约束，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的生产任务的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，根据设备的加工能力计算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需对MT进一步分解，对于每一个子任务重新执行跨区域分解策略。

全局管理Agent与资源Agent、资源Agent组内的各个Agent间均是分层式结构，在生产任务分解过程中，通过上层Agent对MT的分解，协调下层子Agent对子任务的匹配，最后从各个匹配结果中筛选最优的调度方案，因此当管理Agent或资源Agent接收到MT，且根据MT的COMP进行分解得到子任务MT1, MT2,..., MTn,其下第i个子Agent和算法Agent对子任务MTk的调度方案定义为Ti,k，则跨区域生产任务分解过程中，各Agent间协商的目标函数可表示如下：

其中n为MT的子任务集合COMP中任务数量，n为上层Agent的子Agent数量。

本文的基于多Agent的订单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的结合。

（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工厂车间之间进行转移，从而大大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流程如下图所示，可以细分为以下步骤：

步骤一：封装生产任务。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。

步骤二：任务发布。全局管理Agent把MT广播至其下各个资源Agent组。

步骤三：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级别对应，进行能力判定，执行步骤四；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，得到若干子任务MTk，并逐一把子任务广播至其下的子Agent，即执行步骤二。

步骤四：能力判定。当资源Agent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，若匹配成功执行步骤五，否则向上层Agent返回MT无法完成的结果，执行步骤六。

步骤五：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的加工约束和设备的加工能力以及工作状态计算并返回调度方案到上层Agent。

步骤六：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，即执行步骤二；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二，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。



**3.2 多Agent动态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。

**3.2.1 基本蚁群算法的引进**

1992年M.Dorigo等研究人员通过观察蚁群觅食过程中的行为，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，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：

（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，而蚂蚁之间行动中所具备的一定随机性可以使蚁群聚集到最优的路径。根据蚁群算法的原理，可以总结出该算法具有如下的优点：

（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，同时对应的数学模型简单，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循环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相互影响，直至达到共同的优化目标。

3.2.2 基本蚁群算法在FJSP中的应用

基本蚁群算法遵循蚁群觅食过程中体现的社会性以及蚂蚁和环境间的相互作用，其执行流程如图。可以分为以下几个步骤

（1）定义觅食环境以及初始化蚁群

初始化环境中的起始信息素浓度，定义蚁群的规模、蚂蚁的搜索规则、各蚂蚁的起始位置等。

（2）定义搜索目标

定义蚁群的搜索目标，如蚁巢距离食物的路径最短、单位距离中食物量最大等等。

（3）蚂蚁位置移动

蚂蚁根据环境中各个方向的信息素浓度以及当前环境约束（如障碍物）等概率性地选择自身下一步的位置。

（4）撒下信息素

蚂蚁每进行一次位置转移都需要在所经过的路径中洒下一定量的信息素。

（5）终止判定

蚁群终止搜索的判断依据一般有两个：是否达到搜索目标以及搜索时间。当蚁群搜索时间达到一定值时，终止搜索。搜索时间在算法中一般使用迭代次数表示。

（6）信息素挥发

随着时间的推移，蚂蚁洒下的信息素会按照一定的比例挥发直至完全消失。

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。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回路路程最短。TSP等同于在一各完全图中寻找一条路径最短的哈密顿回路，这是典型的NP完全问题，随着完全图中节点数目增加，经典的精确算法无法在多项式时间内进行求解。

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，FJSP要求在若干工件的工艺规划网络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完全问题，精确算法难以求解。

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严格按照约束。

（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工艺规划网络图的起始节点S和终止节点E，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：

:蚂蚁数量

:算法循环次数

:每轮循环中蚂蚁洒下的信息素总量

:t时刻信息素浓度矩阵, 1≤i,j≤n。表示从工序i转移到工序路径上的信息素浓度。

: 各工序间初始信息素浓度

: 信息素挥发因子

: t时刻在工序i和工序j的路径间洒下的信息素增量

: t时刻蚂蚁从工序i转移到工序j的启发式，由工序j的完工时间决定

：t时刻蚂蚁x从工序i转移到工序j的概率

：蚂蚁x在工序i时的禁忌池，记录了蚂蚁k选择工序i时走过所有工序

：蚂蚁x在工序i时的可选池，记录了蚂蚁k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。例如假设存在两个工序，工序1和工序2，以及三个设备，设备m1、设备m2和设备m3。两个工序的机器选择如下表。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MC1=(m1,m3),加工时间序列为T1=（3,5）。同理可得MC2=(m1,m2,m3),T2=(2,5,1)。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 设备m1 | 设备m2 | 设备m3 |
| 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，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。例如，假设FJSP需要对如下图4-4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o1,1，o1,2和o2,1，即tabu={o1,1,o1,2,o2,1},则根据工序间的约束关系此时allowed={ o1,3,o2,1}。



图4-4 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：

其中，为t时刻蚁群x在工序i和工序j之间洒下的信息素浓度

蚂蚁洒下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：

其中Q是定值，Lk是蚂蚁经过的路径总长度，在FJSP中Lk代表蚂蚁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：

其中，是蚂蚁从工序i转移到工序j的启发式，由工序j的完工时间决定，由于工序j存在机器选择序列MCi，因此的计算表达式如下：

为蚂蚁在工序i时转移到工序j的设备选择的启发式，是蚂蚁选择后工序j的完工时间。

根据以上数学模型，基本蚁群算法在FJSP中的执行流程总结如下：

（1）初始化。定义蚂蚁数量m、初始信息素浓度、优化目标（完工时间等）、最大循环次数N，为每只蚂蚁设置可选池和禁忌池、当前最优路径等。

（2）执行N轮循环。在每次循环开启前，清空所有蚂蚁的可选池和禁忌池，把每个工件网络图的起始节点添加到每只蚂蚁的可选池中。蚂蚁根据环境信息素浓度和状态转移概率公式从可选池中选择下一个移动的节点。选择某个节点后，把该节点加入到禁忌池中，同时把其后续节点添加到可选池中。当且仅当蚂蚁遍历所有工序后，结束循环。循环结束前需把本轮最优路径与全局最优路径进行对比，若前者更优，则进行替换。

（3）结束一轮循环后，进行终止条件判定。若当前最优路径达到优化目标或者循环次数到达N，可结束算法流程；否则进行信息素更新，并开启下一轮循环。

（4）结束算法流程，输出全局最优路径。

3.2.4 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；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，MMAS主要改进了信息素更新规则，在每次迭代中，只对本轮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：

其中，Lbest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Lx，由上式可知，当Lx优于Lbest，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Lx不如Lbest,说明Lx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和最小值，在本文中，设置为，设置为。

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，此时各节点的信息素浓度相同，对蚂蚁没有指引作用，因此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，把蚁群平均分配到各个工件的起始工序的各个设备选择中，同时忽略信息素的指引作用。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，获得更优的搜索路径

## 3.3 本章小结

本章通过定义各类型Agent间信息的传递和反馈，来提高系统对外界信息的灵敏度，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组之间的分解流程以及在异常发生条件下Agent之间的合作流程。

4 多Agent柔性生产动态调度系统的实现

4.1 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：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。

4.2 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下几个功能模块：

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：提供对员工信息和客户信息的增删查改，当用户是系统管理员时，可以修改其他用户的使用权限。

工厂管理模块：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和进度。

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。

车间调度模块：用于设置参与调度的生产任务并制定生产计划，提供对改进蚁群算法的参数设置以优化调度结果。

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。

4.3 多Agent系统的实现

4.3.1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

4.3.2 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。

（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、工作序列、历史工作数据、产品工艺信息、库存信息、设备监控信息等等。

4.3.2 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。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。如下图是Agent的系统类图。

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（JA）和设备Agent（MA）。JA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；MA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。

（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()为工艺部新增工件的合法性进行审核。

4.3.3 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同步阻塞队列实现。

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原本的对象。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线程小，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n0，最大线程数目nmax，消息队列的容量为c。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c,若c<cmax，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c0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。

4.4 动态调度系统实现与界面设计

4.4.1 系统登录模块

登录模块用于向用于展示登录界面，如图4-1所示。用户需输入用户名和密码，经系统验证身份成功后方可进入系统，登录成功后进入系统的主页面，如图4-2所示。



图4-1 系统登录界面



图4-2 系统主界面

4.4.2 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。下图4-3是系统的用户信息展示界面。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。系统管理员可查看各个工厂的资源配置，并下达面向各工厂的生产任务。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修改。图4-3中，系统管理员可以修改其他用户的个人信息，包括姓名、年龄、性别以及操作权限。图4-4呈现了企业的物料信息，包括物料名称以及对应的数量。

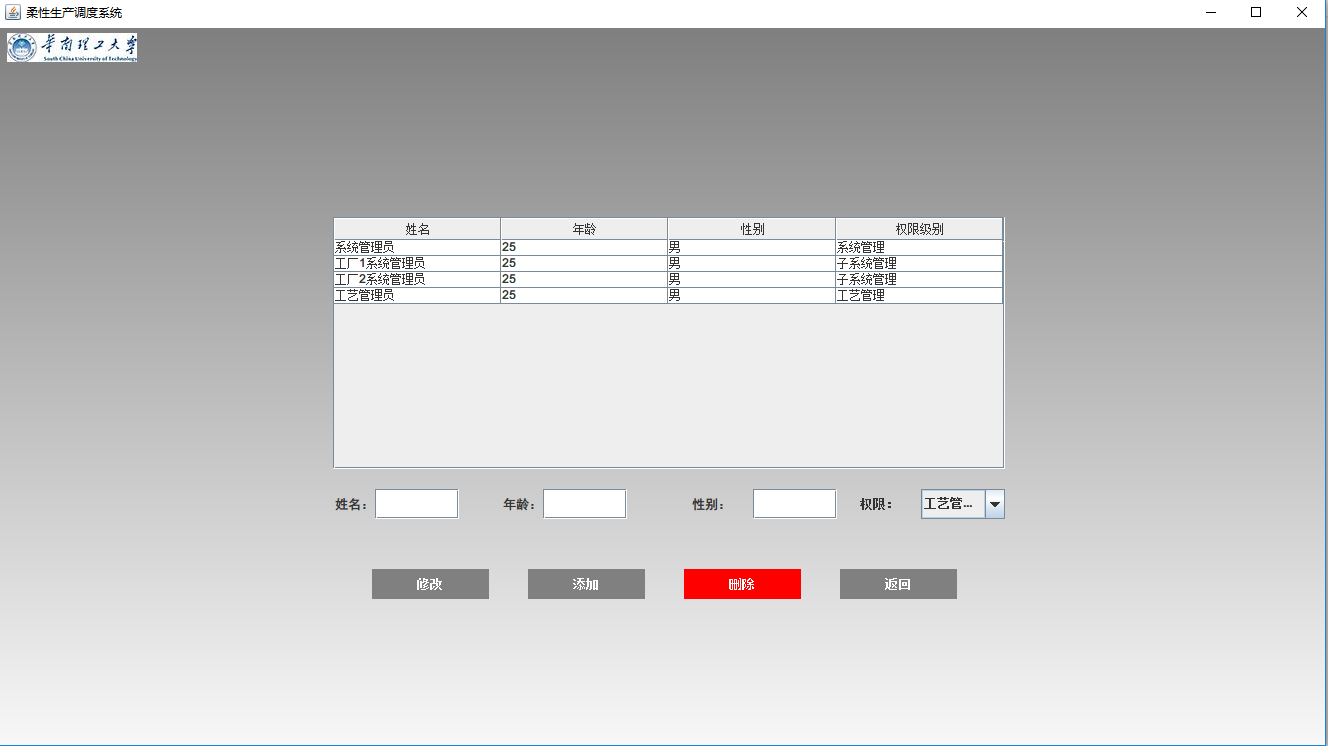


图4-3 用户信息管理界面



图4-4 物料管理界面

4.4.3 工厂管理模块

工厂管理模块可以为系统管理员提供资源配置的功能，如图4-5所示，系统中分布两间工厂，分别为工厂一和工厂二。其中工厂一有两个车间，如图4-6，点击管理按钮后进入管理模式，可进行车间的添加或者删除操作。在非管理模式下点击车间，可查看车间的设备配置，如图4-7表明车间1中包含三个设备m1,m2和m3，管理员同样能对设备进行增删查改等操作。

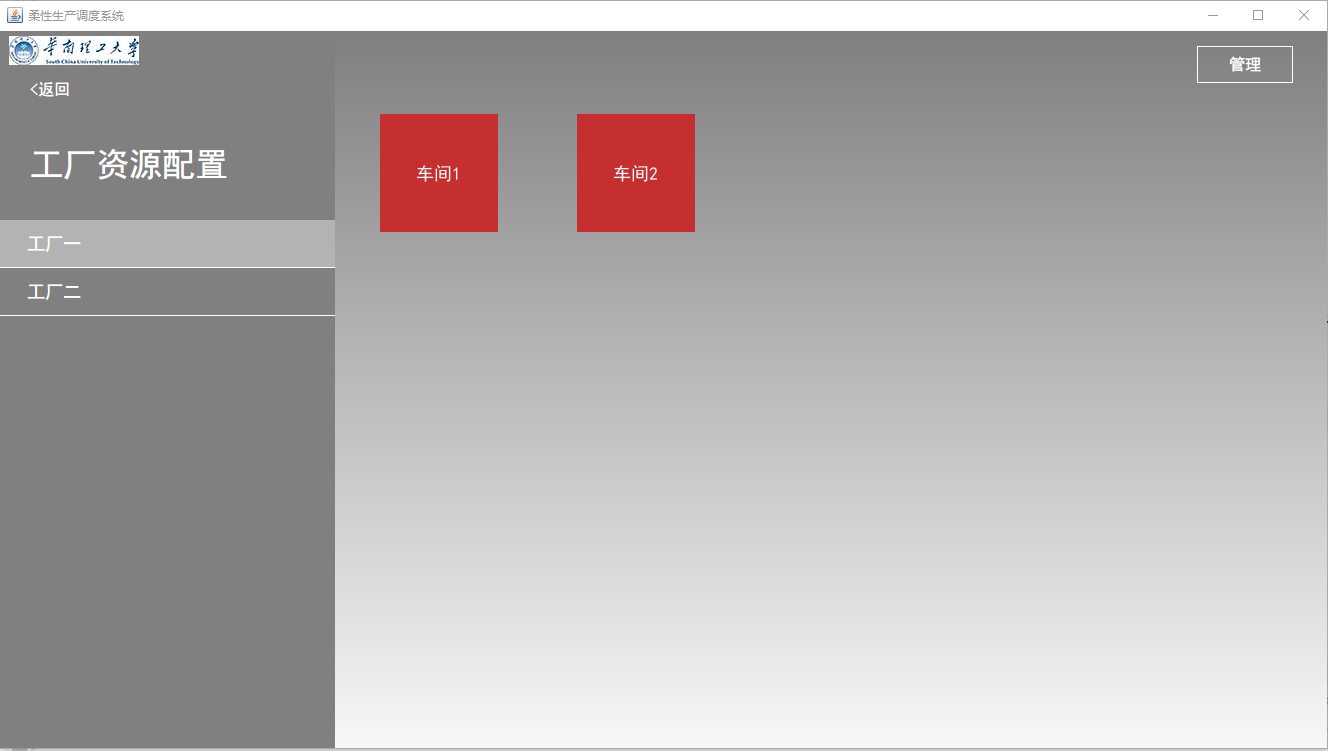


图4-5 工厂资源配置页面

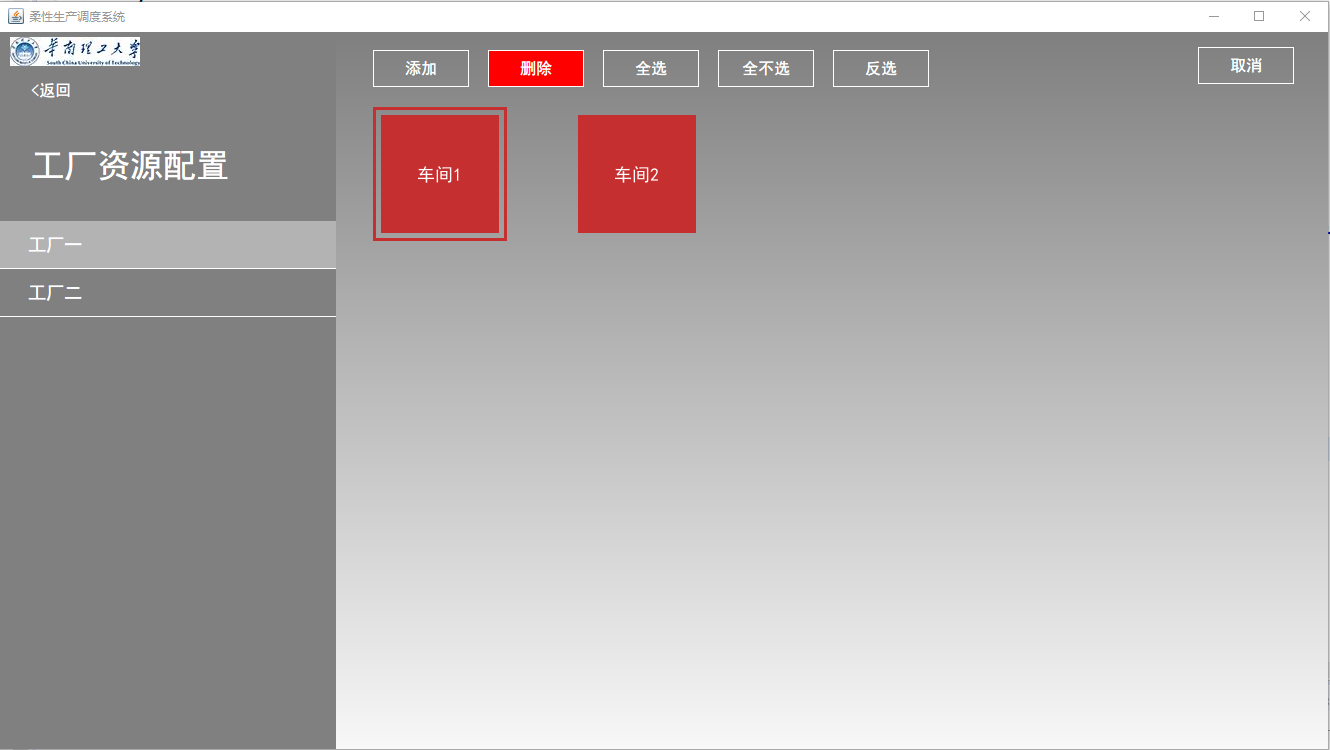


图4-6 资源管理界面



图4-7 车间1的设备配置

4.4.4 工艺管理模块

工艺管理模块为工艺管理员提供对产品工艺信息的管理接口，如图4-8呈现了系统的工件种类，可见系统目前可生产加工的工件公有六种。

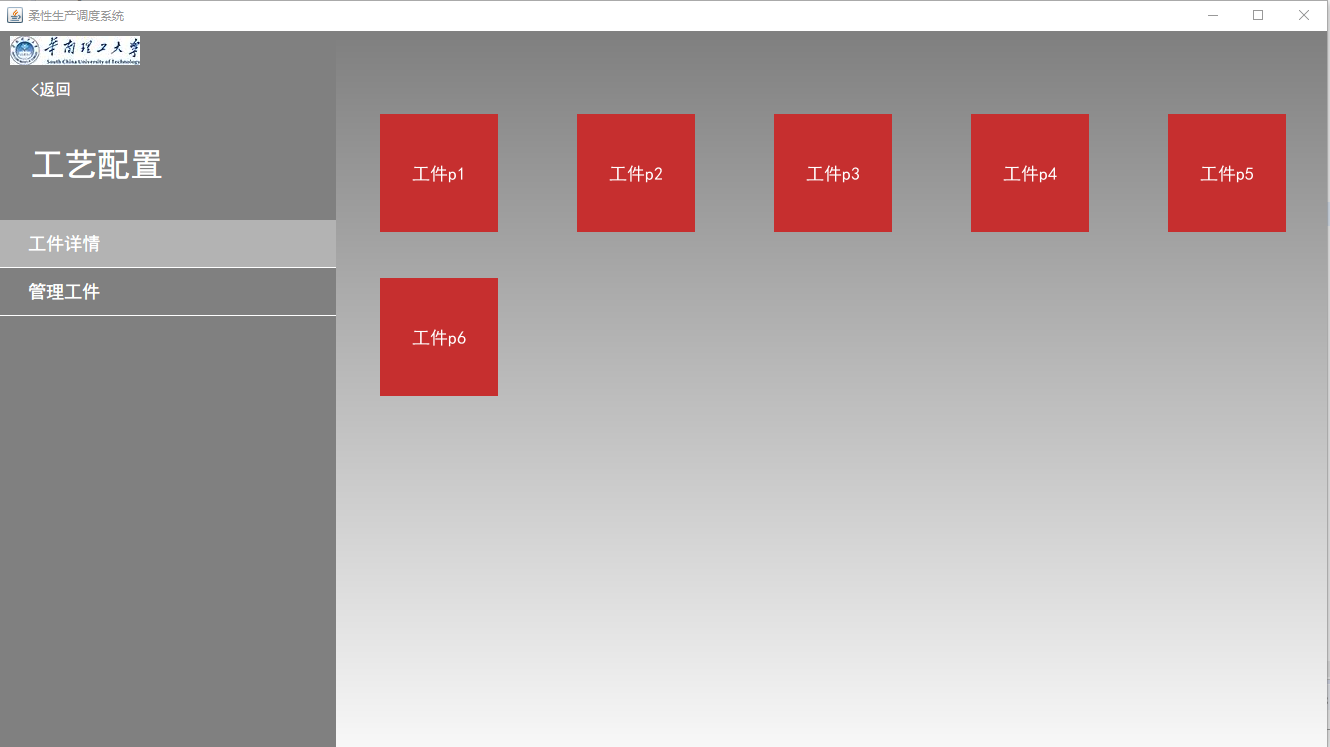


图4-8 工艺管理界面

如图4-9是点击了“管理工件”选项后进入的管理模式，此时可对工件信息进行增删查改。在非管理模式下，选择工件p1后将呈现工件的加工工艺信息，如图4-10所示，从中可看出工件p1包含的工序以及工序的机器选择和加工时间。

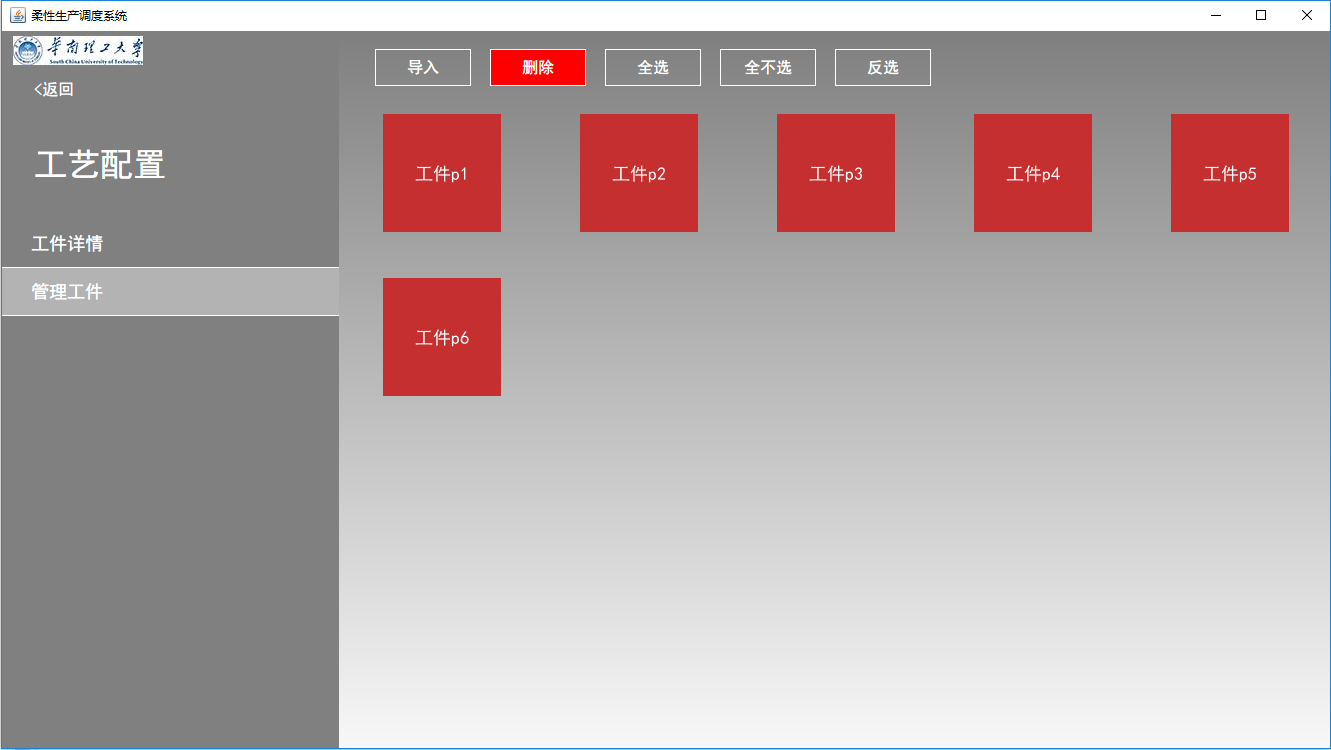
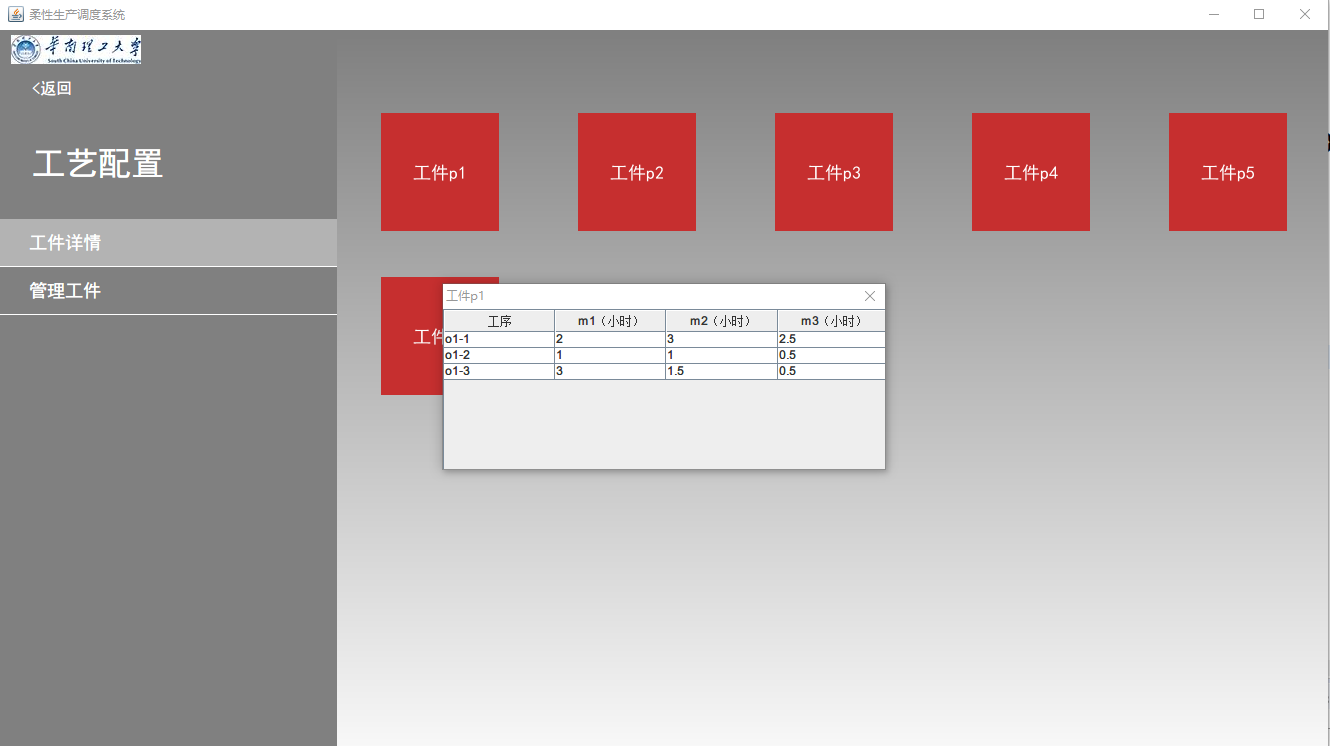


图4-9 工艺信息管理界面



4-10 工件p1的加工工艺信息

4.4.5 生产调度模块

生产调度模块为系统管理员提供生产计划的制定，如图4-10为生产调度的界面，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4-10 生产调度主界面

选择生产调度主界面的参数配置按钮，可以对本文的调度算法的参数进行设置，下图4-11呈现了关于蚁群算法的参数，同时还可设置设备故障触发概率用于测试系统对生产过程中的由于设备故障引起的异常调度。

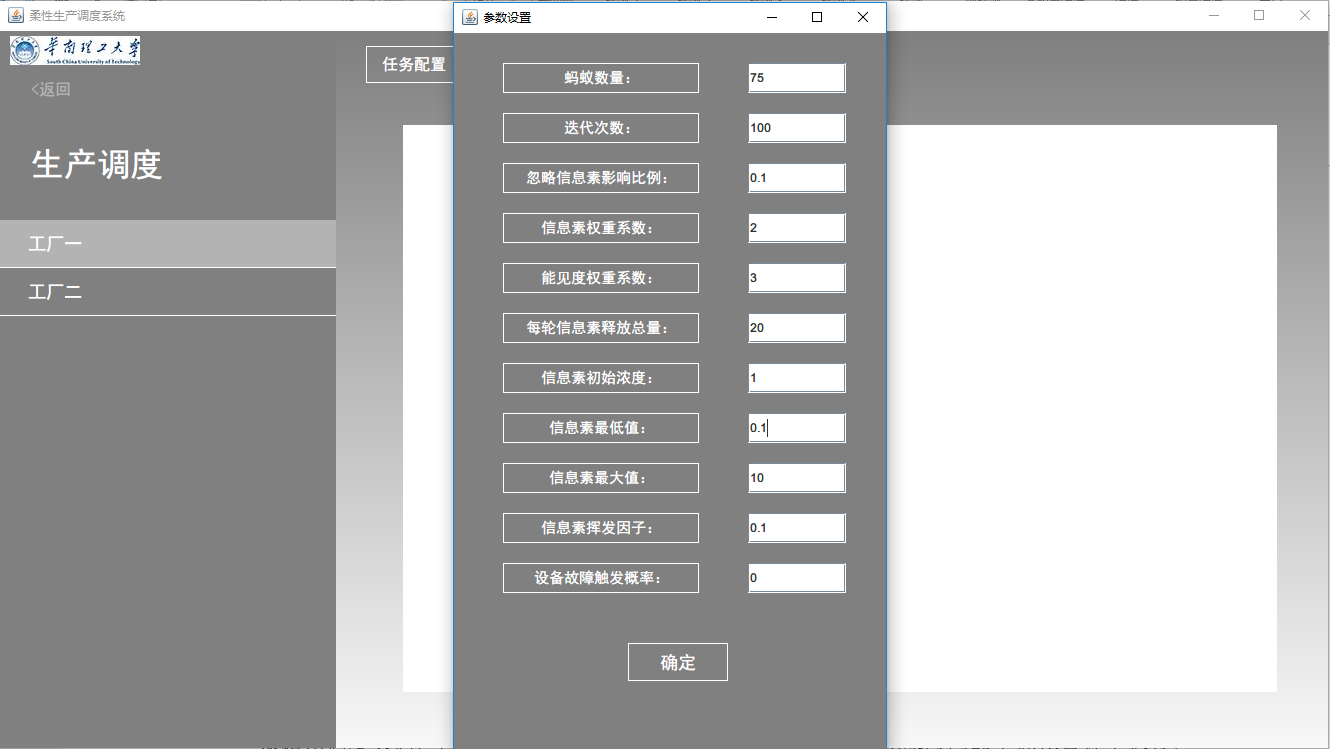


图4-11 算法参数设置

当选择生产调度主界面的任务配置按钮时，可以选择参与调度的工厂以及工件种类和数量，如图4-12所示，管理员选择了工厂一接收生产计划，生产计划包括工件p1到工件p6等六类工件，每类工件数量均为一个。完成参数设置和任务配置后，点击启动按钮，系统最终得到的调度方案如图

4-13所示，该结果呈现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开始加工时间。

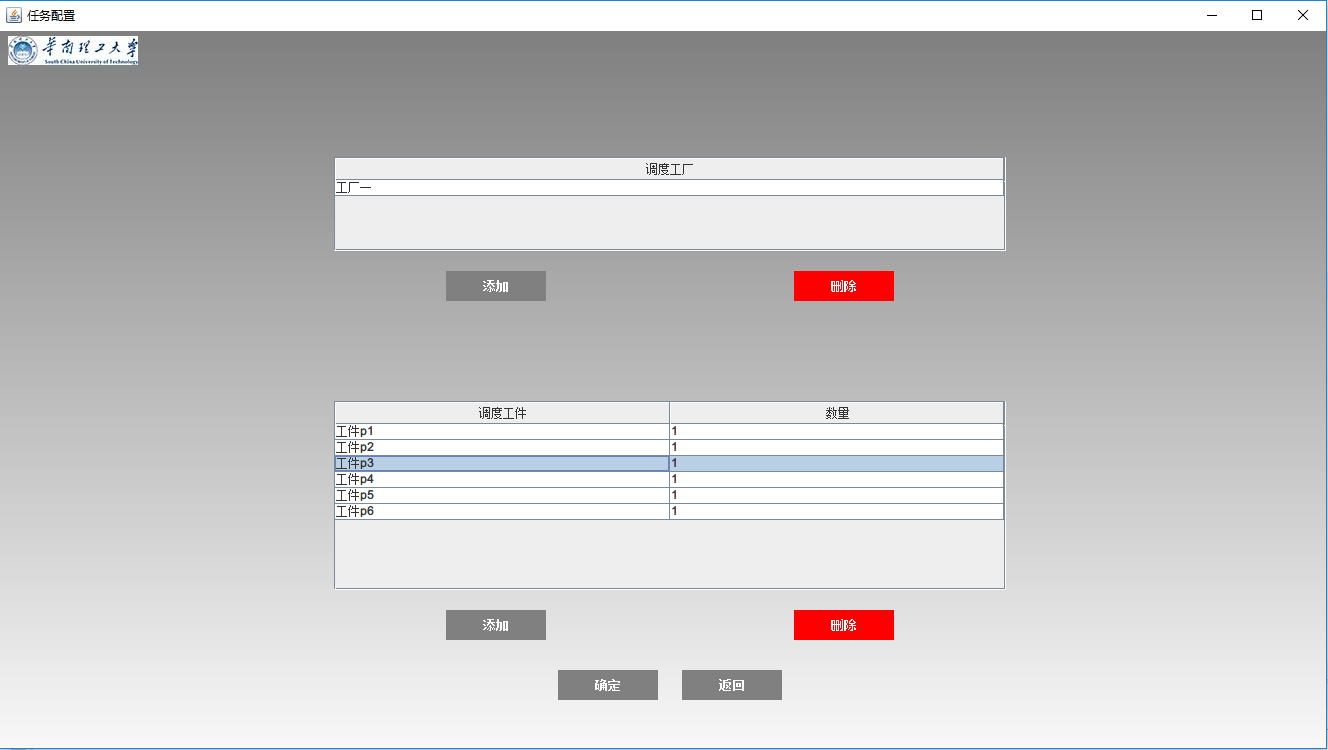


图4-12 任务配置

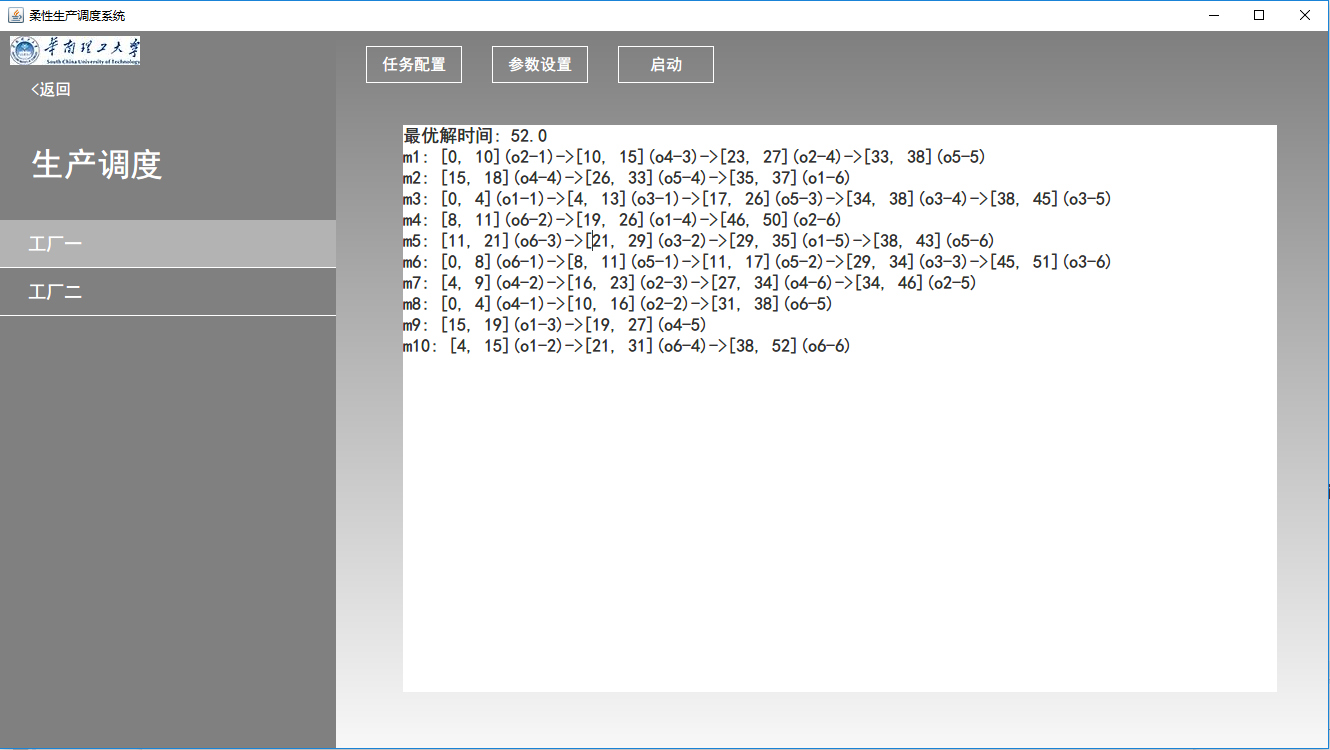


图4-13 调度结果

**文献引用**

[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] 郭克莎. 中国制造业发展与世界制造业中心问题研究[J]. 开放导报, 2006(2):29-32.

[5] 刘琳. 动态不确定环境下生产调度算法研究[D]. 上海交通大学, 2007.

[6] 王艳红等. "制造系统控制结构研究的新进展." 信息与控制 30.5(2001):427-433.

[7] 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.

[8] 孙志勇. 多Agent系统体系结构及建模方法研究[D].合肥工业大学,2004.

[9]李莉. 柔性作业车间调度中的群智能优化算法研究[D].东北林业大学,2011.

[10] 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.

[11]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.

[12]王晓娟. 多目标柔性作业车间调度方法研究[D].华中科技大学,2011.

[13]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.

[14] 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.

[15]陈成,邢立宁.求解柔性作业车间调度问题的遗传—蚁群算法[J].计算机集成制造系统,2011,17(03):615-621.

[16] 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.

[17] 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.

[18] Wiendahl H P, Ahrens V. Agent-Based Control of Self-Organized Production Systems[J]. CIRP Annals - Manufacturing Technology, 1997, 46(1):365-368.

[19] Duffie N A, Prabhu V V. Real-time distributed scheduling of heterarchical manufacturing systems[J]. Journal of Manufacturing Systems, 1994, 13(2):94-107.

[20] Gozzi A, Paolucci M, Boccalatte A. A multi-agent approach to support dynamic scheduling decisions[J]. 2002:983-988.

[21] 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.

[22]Sycara K P, Roth S F, Sadeh N, et al. Resource allocation in distributed factory scheduling[J]. IEEE Expert Intelligent Systems & Their Applications, 1991, 6(1):29-40.

[23] 肖晴. 多智能体系统用于企业集成[A]. 中国自动化学会控制理论专业委员会.1998年中国控制会议论文集[C].中国自动化学会控制理论专业委员会:,1998:5.

[24]郭文胜,殷国富,胡晓兵.基于Multi-Agent的虚拟车间调度作业原理与算法[J].中国机械工程,2001(12):109-112+9.

[25]乔兵,孙志俊,朱剑英.基于Agent的分布式动态作业车间调度[J].信息与控制,2001(04):292-296.

[26]廖强,周凯,张伯鹏.基于现场总线的多Agent作业车间动态调度问题的研究[J].中国机械工程,2000(07):45-47+5.

[27]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.

[28]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.

[29]Betul Yagmahan,Mehmet Mutlu Yenisey. A multi-objective ant colony system algorithm for flow shop scheduling problem[J]. Expert Systems With Applications,2009,37(2).

[30]Xue H, Wei S, Zhang P, et al. Flexible job-shop scheduling based on multiple ant colony algorithm[J]. Computer Engineering & Applications, 2013, 49(24):243-228.

[31]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.

[24] D’Inverno M, Luck M. Understanding Agent Systems[J]. Springer, 2001.

[25] 张健. Agent角色模型与多agent系统构造方法研究[D].山东大学,2012.