**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

**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**

在全球化和科学技术的飞速发展的背景下，制造业在生产调度过程中面临着市场需求动态多变、产品种类众多、工艺复杂、具备生产柔性等严峻挑战，制造业传统的生产管理模式已越来越难以适应。同时企业所处的生产环境充满不确定性[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。因此如何保证企业生产管理的稳定高效进行，提高企业对动态生产环境的响应能力，提升企业的核心竞争力，是广大研究人员需要深入研究的重要课题。

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，更能满足市场丰富的产品需求，但同时意味着企业需要投入更多的设备，增加了生产成本。调度系统作为车间调度的核心，保证了生产调度的平稳有序进行的同时，实现了企业自动化管理、智能化监控，因此一直是制造领域的研究热点。尽管有很多研究人员投入了大量的时间精力，并且有了众多的研究成果，车间调度系统依旧是制造领域的难点，这是由于其涉及的方面广，问题规模庞大，异常因素多且解决难度大，因此至今仍未有十分成熟通用的调度系统能够广泛应用到制造业中。

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。现代制造业使用较为广泛的调度系统为递阶控制式的调度系统，属于智能控制和动态调度理论的早期成果，该形式的调度系统的特点是分层管理，决策单元位于上层，对下层单元进行管辖和监控，信息传递通常由上层经过多层迭代传输至目标单元。对比集中控制式的调度系统，强化了管理控制职能和信息反馈速度，但是对于规模庞大的复杂系统而言，递阶控制式意味着系统的层次会过深，各层次间单元的通讯需要经过多次中转处理，灵活性较低，效率不高，这将导致系统在应对环境的动态异常因素时不够灵敏。为克服以上不足，同时还需要考虑到当今大多数企业制造资源的分布式特点[7]，即一家企业在世界各地设有工厂车间，如何综合利用不同地理位置的制造资源和生产技术，对提高制造业的生产效率和资源利用率是非常关键的。由此，制造业的生产调度系统更趋向于向智能化、信息化的方向发展，同时应强调系统的分布式协调和控制能力，提高系统的自适应能力和稳定性，从而保证系统的容错能力。

近年来，与Agent相关的理论在制造领域一直备受关注[8]，主要原因在于单个的Agent能够自主地获取来自外界的信息，感知周边环境的变化，从而根据自身需求作出反应。每个Agent都被赋予一定的学习能力，这取决于Agent的计算能力和配置的决策规则，在与外界环境交换信息的同时，能够收集自身需要的数据，强化自主学习能力。考虑把多Agent理论应用到车间调度系统上，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，能够快速灵活地完成任务，Agent的这些特点，能够很好地满足车间调度系统在灵活性、自适应性、分布式上的需求。

目前多Agent理论已经较为成熟，但能成功应用多Agent理论并正式投入市场使用的车间调度系统仍很少，这主要是由于调度问题涉及方面广，难度大，因此如何把理论与实践相结合，构建满足市场需求以及实际生产需要的调度系统，仍然是广大研究人员急需解决的一个难题。因此本文基于多Agent理论，设计一个分布式的动态车间调度系统，根据柔性作业车间调度问题的特点和需求，定义各个Agent的功能和结构模型，通过Agent间灵活高效的协商，实现对市场需求和环境不确定性的及时响应，同时针对基本蚁群算法收敛速度慢、易陷入局部最优的缺点进行改进，提出了应用于FJSP中改进的蚁群算法模型。

**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**

自1954年，Johnson[9]开展对基于两台机床的Flow Shop型调度问题的研究后，便逐渐引起实践人员的广泛研究。但随着时代的变迁以及生产力的发展，车间调度的种类越来越多，从最原始的单机调度、多机调度过渡到广泛的流水车间调度和作业车间调度，对应的调度模型越来越复杂，难点越来越多，因此关于车间调度的大部分研究是针对某个方面展开的，例如调度算法、重调度算法的优化、调度系统结构的调整等等。

在研究方法上，最初是集中在线性规划、动态规划、分支定界法等精确求解算法[10-11]，这些经典的调度方法在应用中存在很大的局限性，如难以建立准确约束条件下的数学模型，并且无法适用于求解最优解时间是随问题规模呈指数成倍增长的NP难问题，因此只能对小规模的系统求解。近年来，随着各种新型科学理论的建立与发展，出现了许多新的优化方法，如模拟退火、禁忌搜索、神经网络、遗传算法和蚁群算法等[12]。这些优化方法通常可以相互结合起来，建立基于单目标或多目标的优化算法模型。

柔性作业车间调度问题一直是广大学者的研究热点。Gen M[13]等提出了一种多阶段的遗传算法求解最小化瓶颈机器负荷、平均机器负荷和最大完工时间的多目标柔性作业车间调度问题。Cochran K.J.[14]等提出一种两阶段的多目标优化方法。T'Kindt V[15]等提出了一种采用蚁群算法求解多目标柔性作业车间调度问题。夏蔚军等[16]提出了一种模拟退火和粒子群优化的混合算法，并求解了多目标柔性作业车间调度问题。Kacem等[17]提出了一种采用模糊逻辑和进化算法的混合算法求解多目标柔性作业车间调度问题，在对多个目标的优化判断以及选择上采用了模糊逻辑方法。Sakawa[18]研究了以最大模糊完工时间，最小满意度和平均满意度为优化目标的多目标作业车间调度问题。

**1.3.2 多Agent调度系统的研究现状**

近年来，随着对DAI和先进制造系统的进一步研究，表明拥有较高的智能性、灵活性和良好的合作能力的Agent之间彼此协作的分布式制造系统可以快速地响应环境变化。近些年来，各国纷纷将Agent理论车间调度问题的研究上，己经取得了一定的研究成果，建立了各种各样基于Agent的调度系统。

国外的研究机构已展开了对基于Agent调度系统的研究。Hadvai[19]等提出的分布式实时调度的多Agent系统结构REDS（Real-time Dynamic Scheduling）;Baker[20]等研究了包括调度在内的工厂控制算法在分布分层结构的多Agent制造环境中的实施问题；Shaw[21]提出用Agent方法进行制造调度和控制，指出制造单元可以通过投标机制，将作业以子合同的形式转包给其它制造单元;Chen[22]等采用Agent技术进行制造调度与供应链协调研究;Butler等提出了用于分布式动态调度的多Agent体系结构，将调度分为两个层次，第一层通过Agent的协商机制，以一种分布式的方法制造单元分配给作业，第二层对共享制造资源进行动态分配。

国内的对Agent的研究近年来也相继开展开来，尤其是大学己展开这方面的研究。华中理工大学成功地以数控加工系统为背景，开发了一个基于Agent分布式网络化IMS调度原型系统，实现了制造单元的柔性智能化与基于网络的制造系统柔性智能化集成。此外，沈阳自动化所在多智能控制器研究与设计中，把Agent及多Agent模式应用于多机械手、多机器人的协调与协作设计中，开发了基于多Agent的多机器人协调调度系统。这表明国内越来越多的研究人员开始进驻Agent技术和应用制造系统调度问题的研究。

随着研究的深入和调度系统的发展，特别是柔性作业车间调度系统，由于实际生产系统的复杂性和动态性增加，促使人们进一步研究能够处理生产环境的不确定性和意外时间的生产调度技术，使得调度目标变成柔性调度。可见,面对现代制造的复杂要求,如果依然遵循传统的调度研究方法和思路,单纯通过研究某些算法的改进和扩展等努力来解决,是达不到预期的效果的,需要从系统的角度出发,采用新的思路研究理想的调度系统模型。

**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 基于多Agent的柔性作业车间调度系统结构的设计**

实用的车间调度系统要求能够集成企业在生产管理中的各个环节，并在没有或很少人为干预的情况，完成对生产环境中各种异常因素的响应，这要求系统功能的划分上符合生产需求，在结构的设计上兼顾灵活性和稳定性。本章将主要阐述Agent和多Agent的基本理论，同时根据调度系统在生产计划制定、生产调度、环境监控、重调度等多个方面对系统的结构进行分析，最终应用多Agent理论设计分布式的动态调度系统。

**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比较主流的定义是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，能够分析信息的种类和价值，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不仅具有学习能力，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系统的概念以及特征**

多Agent系统是由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。

多Agent系统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，最终获取对复杂问题的完整求解结果。与传统集中控制式、递阶控制式的系统而言，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。

**2.1.3 多Agent系统的结构**

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进行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，最顶层和底层之间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存取和共享操作的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占用过多的通信带宽，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，可能导致通信风暴，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。本文的调度系统的结构设计采用的是混合型。

**2.2** **动态车间调度系统功能与结构分析**

**2.2.1 系统功能分析**

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异常因素，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，基于各设备原有的加工任务计算各工序的分配方案；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，对原有调度方案进行调整。根据以上功能，本文把柔性车间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、产品工艺信息的管理、订单任务的分解分配、计算和筛选调度方案、异常检测和开启异常调度。对应以上流程，本文基于多Agent理论，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，负责模拟设备的工作进度和状况，车间Agent负责管理分布其中的设备Agent；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是对系统调度策略的封装，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，各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。

**2.2.2 系统的结构分析**

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，本文对基于多Agent的柔性作业车间动态调度系统的体系结构设计如下。由于现代制造业的资源具有分布式的特点，在世界各地可能均设有加工工厂，各工厂根据生产技术特点有着各自的车间和设备配置，形成了相对独立的生产系统。为了能够联合各个地理位置的制造资源和生产技术，设置了车间Agent对其中的设备Agent进行管理，同时分配子管理Agent协调工厂内的各个车间Agent，因此子管理Agent、若干车间Agent以及对应的设备Agent形成了一个具有层次型结构的资源Agent组。由图可知，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。



系统机构图



资源Agent组结构

**2.3 各Agent的功能分析**

本文的动态调度系统中各Agent相互配合协调的过程为：全局管理Agent接收来自客户的订单任务，通过把任务中产品对原材料、精度、加工设备等要求与工艺Agent提供的数据进行对比来判定任务的合法性，若合法性通过，把任务进行封装后分发到各个资源Agent组，由对应的子管理Agent负责任务分解，得到若干任务分配的可行解，最终交由算法Agent计算每一个可行解需要的完成时间，从中筛选出最优解作为调度方案。当发生设备故障时，由监控Agent通知全局管理Agent，根据故障类型以及影响执行重调度，调整调度方案。个Agent具体的功能如下：

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。每个Agent在启动运行后需在全局管理Agent注册自身的信息，该信息是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，协调各个资源Agent组进行任务的分解分配工作。全局管理Agent还需负责接收来自监控Agent的故障诊断信息，根据故障类型和影响协调各资源Agent组进行重调度。子管理Agent是系统分配给资源Agent组的管理员，作为Agent组与全局管理Agent通信的主要接口。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和设备Agent的配备结构和加工能力来计算生成若干任务的分配可行解。

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，负责对设备的工作状况进行模拟仿真，仿真的数据作为算法Agent执行调度策略和重调度策略的重要依据，包括对应设备当前工序的加工进度、待加工工序的开始时间和结束时间等。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，在订单任务分解过程中，车间Agent根据其中设备的加工能力来生成对应的任务分配方案。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。

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、产品中各零部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；工艺Agent还负责为算法Agent提供关于工序在设备上的加工选择以及加工时间。

算法Agent: 封装了调度系统内的算法逻辑，为其他Agent提供计算服务。本文的算法Agent封装了两种算法策略，分别是基于蚁群算法的车间调度策略以及重调度策略。当全局管理Agent和各资源Agent组经过任务分解过程，并得到若干工序任务在设备上的分配可行解后，由算法Agent执行调度策略为每个可行解计算完成时间，从中筛选最优的调度方案；当系统发生设备故障时，由算法Agent执行重调度策略，调整调度方案。

监控Agent:代表系统内的检测设备。一方面，监控Agent根据监控设备采集的加工设备工作状态信息，判定故障类型以及故障的影响，并通知全局管理Agent开启重调度过程。故障种类分为两种，分别是工作效率下降和中止运行，前者导致故障设备上工序加工耗时延长，后者导致加工设备停工等待维修。

Agent间信息传递方向示意图

**2.4 各Agent的决策库设计**

Agent属于人工智能领域，具有自主性、能动性、社交性等，这些特点保证了Agent的灵活性，能够相互配合解决复杂问题，同时强调了Agent的智能性，即Agent的学习能力，能够根据自身的目标主动探索周边环境，调整自身状态，这提供了Agent对外界信息的快速响应能力，根据学习能力的强弱可以一定程度上减少人为的干预，提高系统的稳定性。

Agent的学习能力主要由决策库提供，库中配备的决策规则的完善程度决定了Agent学习能力的强弱，并由此可分为三种类型的Agent：

慎思型：需要人工智能领域发展到较高阶段，赋予Agent以较高的智能，对应的Agent具备最强的学习能力，拥有对自身和环境的认知能力。该认知能力关键在于能够根据所接受的信息、行动导致的结果与目标的一致性来调整所配备算法、规则的相关参数，从而改变后续的行为模式，提高目标的完成度以及效率。因此慎思型Agent需要由强大的推理决策能力，无需人工干预。

反应型：不具备或者具备很低的学习能力，在功能上更偏向于是一个程序编程接口，内部封装了提供各种服务的函数，简单地根据决策库中定义的规则来决定自身的行为，运行过程中不会涉及对原有规则的优化，较为被动地等待接收外界的信息请求。

混合型：拥有最强的学习能力，但运行过程中需要结合多方面的信息和规则来进行推理，从而决定自身行为，因此需要更多的计算时间；反应型学习能力低，但是由于行为模式已经事先确定，因此处理速度快。混合型就是结合了两种Agent的优点，兼顾了计算能力以及处理速度。

本文的基于多Agent的车间调度系统并未统一定义Agent的类型，而是根据各个Agent的功能来进行决策库的设计，从而决定Agent的类型。各Agent的决策库设计如下：

（1）管理Agent

全局管理Agent属于混合型Agent,决策规则包括：生产订单合法性审核，根据订单任务对产品的精度、规格、精度、加工设备的要求与工艺Agent提供的数据进行对比；维护任务队列，根据生产订单交货期进行任务优先级计算、排序，出现紧急订单时，调整任务队列；开启并执行订单任务分解分配的全过程，协调各资源Agent组之间的配合；开启并执行重调度过程，当接收到设备故障信息，基于正常设备调整原调度方案；Agent运行状态监测，每隔一段时间，向系统内的Agent发起探询，若连续十次探询均未收到回复，判定Agent运行故障，通知车间管理人员进行检测。

子管理Agent属于混合型Agent。决策规则包括：执行任务分解策略，协调其下的车间Agent任务分解分配的工作，从各车间Agent返回的调度结果中选择用时最少的方案；监控对应资源Agent组Agent的运行状况，如上层Agent与下层Agent间通信是否可达，设备Agent的实时仿真结果与设备实际工作进度的误差等。

（2）资源Agent

车间Agent属于混合型Agent,决策规则包括：接收子管理Agent传递的任务，根据车间内设备的配置和加工能力判定能否完成任务；对车间内满足任务要求的设备所仿真的数据进行封装，提交给算法Agent计算完成时间。

设备Agent属于反应型Agent,决策规则包括：记录对应设备的工作历史数据，如设备的工作状态、已完成的工序种类和数量、读取对应设备的工作进度信息，并与自身仿真的设备提工作进度数据进行同步，控制误差；

（3）算法Agent

算法Agent属于混合型Agent。决策规则包括：根据车间Agent传递的零件工序任务集合以及设备的仿真数据，执行调度策略，计算任务的完成时间；根据全局管理Agent传递的零件工序任务集合执行重调度策略；当算法Agent处于空闲状态时，根据调度方案计算的历史数据，调整算法参数以得到更优的调度方案。

（4）工艺Agent

工艺Agent属于反应型Agent。决策规则包括：向其他Agent提供产品、零件以及工序的信息，如产品的种类以及对应的规格、精度等，零件的加工流程、工序的机器选择和对应的加工时间等；响应系统管理人员对产品工艺的增删查改操作。

（4）监控Agent

监控Agent属于反应型Agent。决策规则包括读取监控设备如机械状态监测设备、机械故障诊断仪的监控数据，判定故障类型和对应的影响，并封装数据返回到全局管理Agent;当设备故障修复后，通知全局管理Agent.

**3 基于多Agent的任务协商策略**

本文的柔性车间动态调度系统是基于多Agent的分布式系统，其内各Agent相对独立，并行处理各项子任务的同时，通过信息交换和数据共享，彼此间相互配合，共同完成大型复杂的柔性车间调度问题。自主性、社会性、能动性等特点赋予了Agent对外界信息的快速响应能力，移动性等特点决定了系统的高可扩展性。本文利用系统中Agent的以上特性，定义信息在各个Agent之间传递的方向，提出一种基于多Agent的订单任务跨区域分解策略以及异常因素发生时的重调度策略。

**3.1 基于多Agent的订单任务跨区域分解策略**

对比传统作业车间调度，柔性作业车间调度问题在生产过程中具备更多的柔性，系统中的产品通常由若干工件组成，一个工件从投入生产到加工完成需要按照一定的顺序经历若干设备，由于工件在加工过程中存在工序的顺序约束，工序加工也存在多个设备选择，对应不同的完成时间，因此零件的加工工艺并不唯一，当零件的工序数量增多、顺序约束更复杂时，零件的工艺路线的数量将呈指数型增长，属于典型的NP-hard问题，对系统在任务分解和工序任务分配中有更高的要求。目前对车间调度理论的研究成果大部分是面向处于在固定地理位置的工厂车间的调度问题，根据工件的加工工艺以及工序的机器选择等约束，执行进化算法、神经网络算法、禁忌搜索算法等来进行任务的分解分配，这对于资源集中的制造系统通常是有效的，但是在全球化背景下，制造企业在生产管理方式上具有社会性，通常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，对资源分布在不同地区的企业而言，如果仍使用传统的任务分解分配策略，则一个工件的各个加工工序可能会被分配到各个地区的设备上进行加工，这显然是不符合实际的。本文根据制造业资源的分布式特点，设计了一种基于多Agent的订单任务跨区域分解策略，实现分布式资源和技术的综合利用。

本文通过全局管理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流程，最终可得到若干加工设备集合，每个集合均能满足完整生产任务对设备的要求；算法Agent执行其中封装的调度策略，基于每个设备组合计算对应的调度时间，全局管理Agent从中选择用时最短的结果最为最终的调度方案，该基本流程可概括为：调度系统接收到来自客户的生产订单后，根据订单对生产对象的要求封装为若干生产任务。系统把生产任务传递到各个地区的工厂，各工厂根据生产任务跟设备和原材料的需求，对比自身的设备配置和原材料库存来判定是否有能力完成任务。若各工厂均无法独立完成任务，则系统将对任务进一步分解为一组子任务，协调各个工厂的任务匹配过程，直至得到多个满足任务要求的设备集

**3.2 生产任务模型设计**

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CPT（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的模型表示如下：

MT={MTI,CST,CSN,R}

MTI(Manufacturing Task Infomation)描述了任务的基本信息,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、CPT或WPT。DESCR（Description）是对生产对象的详细描述，如CPT中DESCR描述了零件的精度和规格等信息，CPT中描述了工序加工的注意事项等。

CST（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对加工时间，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，如零件在规格、精度上的要求。由于CPT和WPT作为PRT的子任务，因此TIME为空，不指定。

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

CSN={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指定了零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。

R（Resource）描述了生产任务MT对资源的要求，表示如下：

R={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的设备集合；T（Time）与E一一对应，表示每一个机器选择对应的加工时间。

**3.3 基于多Agent的订单任务跨区域分解策略的设计**

传统的任务分解分配根据客户的生产订单对设备、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，对比使用枚举法的传统算法，更适合用在问题规模较大的车间调度问题上，但随着生产设备的数量规模的增长，满足任务生产需求的设备组合数目将呈指数增长，所需的计算量也会大大增加，这不利于调度系统响应生产需求中的动态变化以及生产环境中异常状况的发生。

本文把任务的分解分配过程进行一定程度的分离，首先根据生产任务MT对资源的需求、任务分解得到的子任务集合以及制造资源的跨地区分布特点，单独进行任务分解过程，即执行基于多Agent的订单任务的跨区域分配策略，得到能够满足任务生产条件的若干设备组合；而后进行任务分配流程，由算法Agent对每个设备组合执行调度策略，生成对应的调度方案最后由全局管理Agent从中选出用时最少的方案。把任务分解单独分离出来，能够大幅减少满足生产条件的设备组合，使调度策略执行时面向的设备规模降低从而减少所需计算量。

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、零件级任务CPT、工序级任务WPT从上往下逐层分解。

（2）生产任务MT分解过程中的传递顺序需符合资源Agent的结构，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、CPT、WPT分别与SMA、JSRA、MRA对应。当TM粒度大于自身级别时，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；当两者级别相同时，判断自身是否有能力完成TM，判断的依据是原材料和设备是否满足TM的要求。

（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CPT所包含的若干子任务WPT分配到不同地区的生产设备上，在实际生产中，需要把零件在多个地区工厂车间之间进行运输，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的订单任务的跨区域分解策略是由系统内的管理Agent和资源Agent之间的协调和配合来完成的。在生产任务分解过程中，全局管理Agent以得到满足生产任务要求的最优调度方案为全局目的，执行生产任务的跨区域分解策略，为每个资源Agent组指定个体目标，由于全局管理Agent和各资源Agent组之间构成的是层次型结构，上层Agent对下层Agent有控制和指导的作用，便于数据共享、状态同步的同时，能够降低Agent间通信的冗余，通过协调各Agent的决策与执行，从各资源Agent组的计算结果中选取最优的方案来满足全局目标，从而更有效率地完成任务分解。

生产任务跨区域分解协商策略的执行流程如图，该流程可以划分为以下步骤：

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TM。

步骤二：任务发布。资源Agent把TM广播至其下各个子Agent。

步骤三：粒度判定以及任务分解。资源Agent接收到上层Agent发布的TM，若TM粒度与资源Agent级别对应，进行能力判定，即进行步骤四；若TM粒度小于资源Agent的级别，直接把TM发布至其下各个子Agent，由子Agent进行能力判定；若TM粒度大于资源Agent的级别，需进一步分解TM，得到若干子任务，并逐一把子任务发布至其下的子Agent，即执行步骤二。

步骤四：能力判定。当资源Agent级别与TM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TM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TM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TM和设备集发送至算法Agent，由算法Agent基于TM和设备集合生成调度方案，计算所需的时间，并返回调度方案到上层Agent。

步骤六：结果筛选。资源Agent收集到下层子Agent对TM的判定结果，分两种情况进行处理：

（1）所有子Agent均返回无法完成的判定结果。若TM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TM无法再分解（粒度为WPT），则RA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。

（2）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，返回到商机Agent。若该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TM的交货期且TM可进一步分解，分解任务后执行步骤二，若TM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。

**3.4 基于多Agent的重调度协商策略**

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，即开启重调度过程。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GMA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重调度策略进行说明。

**3.4.1 紧急订单处理的重调度策略**

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的重调度策略：

（1）局部最高优先级是指紧急订单对应的生产任务在GMA的任务队列中具备最高优先级。紧急任务被插入到GAM的任务队列的最前端，按照跨区域的生产任务分解、算法Agent执行调度策略计算等流程最终筛选出最优的调度方案，该过程是基于系统内MA原有的加工序列进行的，不对之前完成任务分解分配的生产订单造成影响。

（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，包括之前已完成任务分解分配的生产任务。该情况下系统先进行一次打赌，即按照局部最高优先级来对紧急任务进行分解分配，若此过程汇总筛选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之前已完成分配的生产任务两者的需求。若打赌失败，对系统中所有MA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跨区域生产任务分解并由算法Agent计算、筛选最优的调度方案。最后把对MA原有的生产序列备份的工序任务，按照相对顺序放回到对应的工序序列中。

**3.4.2 设备故障处理的重调度策略**

本文把设备故障分为两种，一是设备由于零部件磨损、断裂等导致设备无法运行；二是由于设备老化或运行环境发生改变，如气压、温度等出现异常导致设备的工作效率出现不同程度的下降。前者故障发生时，设备必须终止运行等待维修，后者根据系统管理人员的判定后若无生产危险，可短时间内继续进行生产加工，但对应的加工序列中工序的加工时间需要根据工作效率下降的程度进行等比例地延长。无论是何种故障，必然导致当前的调度方案已不再适用，必须开启重调度。

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，先查看所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，把工件安排到替换设备上加工，否则交由GMA重新进行工件任务的分解分配流程。

（3）若重调度后调整所得的若干调度方案均无法满足生产任务的交货期，从中选择延迟时间最小的方案，并提交到客户说明原因。

设备故障处理的重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）子管理Agent把对应设备Agent的加工序列中封装的工件任务进行重新封装，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{pt1,pt2,...,ptk}，集合内的工件剩余的后续未加工工序重新封装，得到新的工件任务集合{t1,pt2,...,ptk}；若故障设备只是效率下降，仍可继续运行，。然后子管理Agent根据自身的设备配置与P中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。

（3）若故障设备仍可运行，但工作效率降低，则分配到该设备上的工序的加工时间需等比例延长。而后在资源Agent组中按照生产任务跨区域分解协商策略完成工件任务集P的分解分配。

（4）若故障设备无法继续运行，则所属车间Agent需移除对应的设备Agent，而后子管理Agent根据资源Agent组中的设备配置寻找是否存在可替换设备，若存在则同样由该资源Agent相互协商进行工件任务集合的分解分配。若不存在，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中进行任务集合分解协商，求解新的调度方案。

（5）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。

**3.4.3 订单取消处理的重调度策略**

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，其中的各工序任务已经安排到不同设备Agent的加工序列中，则全局管理Agent需向各个资源Agent的发送撤除请求，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。

**4 基于改进蚁群算法的柔性作业车间调度策略**

在生产任务跨区域分解过程中，当资源Agent组判定自身有能力完成生产任务TM时，RA通过把相关的设备集以及生产任务发送至算法Agent，由算法Agent执行调度策略计算并返回调方案。尽管在TM分解过程中，遵循尽量保持TM粒度的原则，得到的设备组合的数目以及组合内设备的数量将大大减少，降低了算法Agent的计算量，但由于柔性作业车间调度问题具有产品工艺复杂、约束多。充满不确定性等特定，因此是典型的NP完全问题，传统的基于枚举法的精确调度算法无法适用。为此，本文的算法Agent中封装的是基于改进蚁群算法的柔性作业车间调度策略。

**4.1 制造系统中的车间调度问题**

**4.1.1 车间调度问题的描述和分类**

车间调度问题可以描述为在生产资源一定的前提下，如何把这些资源分配给特定时间范围内的不同生产任务，以达到完成一个或多个目标的目的[1]。更具体地，车间调度问题可以表示为把n个工件安排在m个设备上进行加工，其中每个工件从加工到完成需要经历若干道工序的加工，工序间存在顺序约束；每个设备能够对若干道工序进行加工，对应不同的完成时间，而调度目的就是如何定义设备的分配、工件工序的加工顺序以及工序的开始和结束时间，以达到一定的优化目标，因此车间调度问题的实质是全局考虑若干生产任务的约束条件，按照一定的策略制定资源的分配以及任务的生产顺序，最终满足生产任务在一个或多个目标上的期望值。这里的约束条件主要由生产环境以及生产任务的特点所提供，例如生产设备的加工能力，有些设备只能对单一工序进行加工，有些设备能够对若干工序进行加工，对应不同的完成时间；生产任务上的约束主要体现在工艺约束上，存在生产工艺固定的工件，也存在生产工艺可选的工件，后者在工序的前后顺序选择上存在多个可选分支，并且每道工序可以在多个设备上进行加工，对应不同的完成时间，因此具有更多的柔性。

根据制造业生产系统的特点，车间调度可以分为以下几种类型：单机调度、多机调度、流水车间调度、作业车间调度。单机调度是最简单最早期的调度种类，所有工件的生产加工都由一个设备来完成；多机调度是相对于单机调度而言的，区别体现在生产设备的数量上更多，单一工件的加工同样只需要由一个设备来完成，但是由于相同功能的设备更多，因此效率上更高。单机和多机调度灵活性较低，不存在柔性，属于早期生产力较低下的调度方式，已不适用于现代制造业；流水车间调度和作业车间调度细化了生产流程，强调设备的职能分工，对于工件中的每一道工序都会分配特定的设备来进行加工。其中流水车间调度中所有工件都有相同且固定的加工工艺，每个工件有着严格的工序顺序约束，需要按照特定的顺序经过每一个设备，这些设备就组成一条流水线，若存在功能相同的设备可并行运作，则其中的工序可选择任意一个设备，可选择的设备数量越大，则系统具备的柔性越多；而作业车间调度中，每个工件的工艺路径可以是不同的，工件与工件间也不存在排队间的顺序约束，由于可能存在多个可选工艺路线的情况，甚至即使是同一种工件，在设备选择上也存在多种形式，因此当设备规模较为庞大、工件加工工艺复杂时，作业车间调度可看作是柔性作业车间调度。

流水车间调度在上世纪八十年代以来一直是众多车间工厂最主要的调度方式，尤其适用于大批量小品种的长期静态生产，极大地降低了人力成本，使生产力得到高速提升。但由于现代市场商品生命周期更短，需求动态变化，制造业的生产模式逐渐转向面向订单的小批量多品种生产模式，因此具备更高灵活性和更多柔性的作业车间调度已逐渐成为主流。

**4.1.2 柔性作业车间调度问题的特点**

在传统的作业车间调度问题中，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。而在柔性作业车间调度问题中，加工工序具有多个机器选择，对应不同的加工时间。同时由于生产环境充满如随机订单、订单更改、设备故障等动态不确定性，柔性作业车间调度问题在工艺规划时往往会添加更多的柔性，即为工件提供可选的多个加工路线。当制造系统出现异常因素时，工件加工可以灵活替换工艺路线，显著提高系统的动态响应能力。但机器选择、柔性工艺在提高系统稳定性和可适应性的同时，也扩大了调度方案可行解空间，对系统的计算能力有了更高的要求。

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，其中每个工件存在一种或多种工艺选择，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。柔性作业车间调度中工件的工艺规划可以使用网络图表示，网络图通常也成为与或图（AND/OR图），下图使用网络图表示工件的工艺规划。

网络图中首先必定包含一个开始节点S和一个终止节点E，这两种节点仅分别用于表示加工的开始和结束，不包含其他信息。一条完整的工艺加工路线必须始于节点S，终于节点E，一个工件经历一条完整的加工路线后即可完成。其次，网络图中还包含工序节点，该节点用于表示一道工序的加工信息，例如图中节点1、2、3、4等均是工序节点，其中节点6表明对应工序可在设备7、8上进行加工，加工时间分别为23，29个时间单位。节点间的箭头表示顺序约束，例如节点2和节点3之间有箭头相连，表明节点3对应的工序必须在节点2对应的工序加工完成后才能开始。网络图中还有三种节点用于约束工序间的顺序，分别是AND、OR和JOIN节点。AND和OR节点会分出两条以上的加工路线的分支，其中AND节点要求其后的分支都必须选择，而OR节点要求仅能从其后的分支中选择一条。例如节点3为AND节点，存在两条分支：3-4-5和3-6-7，则工序4、5、6、7是工件加工过程中必须经历的；节点9是OR节点，存在两条分支：9-10-11和9-12-13，则仅能从中选择一条分支。此外网络图中的JOIN节点表示分支的合并，每个JOIN节点均与一个AND或OR节点对应。要注意的是，柔性作业车间调度的工艺规划中可以不包含OR节点，此时表明该工件的工艺规划只有一种选择。当所有工件的工艺规划只有一种选择，并且所有工序只有一种设备选择时，该调度问题就是传统的作业车间调度问题。

**4.1.3 柔性作业车间调度问题的数学模型**

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：

n：工件数量

m：设备数量

P={p1,p2,p3,...,pn}：工件集合，pi表示第i个工件

M={ m1,m2,m3,...,mn }：设备集合，mi表示第i个设备

Oi={0i1,0i2,...Oik}:第i个工件的工序集合，其中Oij表示工件i的第j个工序

MCij=mcijk: 工序Oij的机器选择集合，mcijk表示工序Oij能在设备mk上进行加工

TSi=(tsi1,tsi2,...tsik)：设备i的加工序列，tcik表示设备i上的第k个加工工序

Tij=tijk：工序Oij的加工时间集合，与MCij中元素一一对应，tijk表示工序Oij在设备mk上的加工时间。

Sij=sijk:工序Oij开始加工时间，sijk表示工序Oij若安排在设备mk上的最早加工时间

Eij=eijk: 工序Oij结束加工时间，eijk表示工序Oij若安排在设备mk上的结束加工时间。

C=ci:工件的完工时间，ci表示工件i的完工时间

D=di：工件的交货期，di表示的工件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oij+1必须在oij完成后才能开始加工，则有

Sij+1k<=eijk’

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

Eijk<=ei’j’k

（3）工序在加工过程中不可中止：

tijk=eijk-sijk

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=minCmax=min(max(Ci)) 1=<i<=n

**4.2 基本蚁群算法**

**4.2.1 基本蚁群算法的原理**

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。

**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**

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中若所有工件只有一种工艺规划（工艺规划网络图中不存在OR节点），则每个工件的工序都必须加工；若存在工件有可选的工艺规划，由于工艺规划网络图中包含OR节点，因此有可选的分支，存在非必选的工序，因此不要求所有工序都包含在最终的调度方案中。

（4）FJSP中工序存在机器选择，因此在工艺规划网络图中进行状态转移，即从一个节点转移到另一个节点时，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:算法循环次数

Q:每轮循环中蚂蚁洒下的信息素总量

:信息素浓度矩阵, 1≤i,j≤n。τij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

: 蚂蚁x在工序i和工序j的路径间洒下的信息素浓度

: 蚂蚁x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

：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完成工序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1、设备2和设备3。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。表格中的“--”表明工序1不能再设备2上进行加工。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1 | 设备2 | 设备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，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1,2,3，即tabu={o1,o2,o3},则根据工序间的约束关系此时allowed={o4,o5}。

（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，如下：

其中，为上一轮循环中蚁群在工序i和工序j之间洒下的信息素浓度。为循环中蚂蚁在在工序i和工序j之间洒下的信息素。

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的机器选择mcj,k的启发式，是蚂蚁选择mcj.k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图。

（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。当所选节点属于OR节点的某个分支，需要把另一个分支的所有节点均添加到禁忌池中。当且仅当每只蚂蚁均经历了所有工件网络图的起始节点S和终止节点E，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。

**4.2.3 基本蚁群算法在柔性作业车间调度问题中的改进**

根据基本蚁群算法的执行流程克制，该算法具有如下优点：

（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。

基本蚁群算法的优点从另一方面而言也带来了缺点，主要缺点有两个：收敛速度较慢和较容易偏离最优解，陷入局部最优。算法初始由于循环次数少，环境信息素积累少，浓度低，对蚂蚁的指引作用较弱，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，需要经过多次循环累积信息素，当不同工序节点的信息素区分度存在较大差异时，才能使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中各个工件的网络图中工序存在顺序约束，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陷入局部最优解。若算法循环次数较低，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。

本文根据FJSP的特点以及基本蚁群算法收敛速度较慢以及容易陷入局部最优的缺点来对算法进行改进，改进方法如下：

（1）提高较差的设备选择的接受概率。对转移概率高于平均值的设备选择降低为原来的90%，降低的数值平均分配到其他的机器选择中。为防止转移概率过低的机器选择导致结果过于偏离最优解，规定原转移概率低于0.05的机器选择不接受数值分配。该方法主要用于降低蚂蚁的贪婪程度，提高蚂蚁的可见度、扩大选择范围，减少陷入局部最优的几率。

（2）在N次循环的前20%的循环中，均匀分布所有蚂蚁的位置。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，蚁群易陷入局部最优。因此，本文把蚁群平均分配到各个工件网络图的起始工序中。随着循环次数的增加，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，若之后在每轮循环初始化时忽略信息素的影响仍然均匀分布蚂蚁的初始位置，会增加算法的执行时间，降低收敛速度，因此限定旨在前30%的循环中进行蚂蚁初始位置的均匀分布，在后70%的循环中蚂蚁根据状态转移概率计算公式决定初始位置。

（3）在每轮循环中，若当前最佳路径Lk优于全局最佳路径Lbest,提高Lk路径上节点的信息素增量，否则降低信息素增量。信息素更新公式如下：

由上式可知，若当前的最佳路径Lx优于Lbest，则最佳蚂蚁经过的路径洒下的信息素将比其他蚂蚁更多，且Lx优于Lbest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偏离最优解，对应路径上信息素浓度偏高，因此需要降低该路径上的信息素增量，同样能降低蚁群陷入局部最优的概率。

（4）在算法N次循环中，设置两个个监测点，在当前循环次数count累计30%\*N以及60%\*N时分别执行额外的信息素更新规则：若全局最优路径上累计的信息素浓度大于全局信息素浓度的70%时，认为蚁群已陷入局部最优解，过早收敛，因此把Lbest上的信息素浓度降低为原本的70%。

**5 基于多Agent的柔性作业车间调度系统的实现**

**5.1 系统需求分析**

车间调度系统是现代制造业实现生产管理自动化的关键依据，涵盖包括生产计划制定、工艺和库存管理、数据处理、资源分配、质量保证等方方面面，能有效提高制造企业的生产效率和资源利用率，同时信息化和自动化管理能够保证系统的敏捷性和稳定性，保证系统能对订单需求和异常因素的变化及时快速地响应，因此本文对柔性作业车间调度系统提出以下几点应用需求：

（1）系统自动化与信息化

传统的车间调度系统的生产流程高度依赖人力，通常由车间管理员处理生产订单、制定生产计划以及及分配制造资源等，该过程通过管理员的先验知识进行工艺路线规划、原材料和生产设备配置的审核以及生产状况的监控，因此调度方案的确定具有一定的随机性，若设备和工艺约束考虑不周，甚至可能得到错误的生产路线，延误任务的交货期。柔性作业车间调度问题FJSP通常具有产品工艺复杂、设备规模庞大等特点，需要降低系统对人力和人为经验的依赖，建立自动化的任务审核、资源分配、生产监控流程。

（2）系统敏捷性

现代制造领域所处的制造环境时刻动态变化、难以预测，如随机订单、紧急订单、设备故障等，这需要系统具备应变能力，能够及时反馈的同时也要快速采取应对措施，因此调度系统要有良好的调度能力，生成满足优化目标的调度方案；其次，系统要具备灵敏的信息传递机制，能够在各生产流程之间快速传递，应对各种异常因素，保证系统的稳定性。

（3）数据管理能力

车间调度系统运行过程中涉及诸多数据的增删查改，如产品的工艺信息、原材料库存信息、实时监控信息、制造资源分布和结构信息等，生产过程中信息和请求的频繁传递要求系统具备强大的数据处理能力。

（4）生产监测和管理能力

车间调度系统作为生产管理的关键依据，要求能为管理人员实时提供生产数据的形式化展示以及封装管理接口，如管理员需实时掌握各车间设备的运行状态和工作进度、能够人为调整生产计划和产品工艺、能够查看系统历史工作数据、产品合格率等信息。

**5.2 系统设计与实现**

本文开发基于多Agent的调度系统来实现系统需求，根据制造系统的生产管理流程设计设计了包括管理Agent、资源Agent、工艺Agent、算法Agent和监控Agent五类Agent，通过各Agent之前的相互配合完成调度系统的功能，

**5.2.1 开发技术和工具**

本文的调度系统使用Java作为程序开发语言，Eclipse作为开发工具，MySQL作为数据库，Microsoft Win 10作为操作系统。应用Java的多线程和多进程技术来支撑Agent的独立运行，使用Java的并发控制实现Agent间的信息共享和状态同步，应用基于TCP的Socket通信实现Agent间的信息传输，以及Java的序列化和反序列技术进行对象的数据流传输。

**5.2.2 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。每个资源Agent组对应于一个地区的加工工厂，通常可构建局域网来支持其内Agent的相互通信，具有通信带宽高，时延小的优点。同时全局管理Agent作为联系各资源Agent组的通信中枢，需时刻开启监听端口，等待来自子管理Agent的通信请求。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作为各子管理Agent的服务端，协调各地区工厂制造资源和技术的分配；子管理Agent和车间Agent，车间Agent和设备Agent之间同样以“服务端——客户端”的模式进行联结，能够充分利用局域网高带宽，响应速度快的优点，实现生产任务的快速分解。

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：数据层、应用层和表现层。

（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。

**5.3 Agent间的通信机制与实现**

**5.3.1 Agent与Java的多线程技术**

本文的调度系统是基于多Agent的分布式系统，要求各Agent能够独立自主运行于计算机环境中并长期监听来自其他Agent的消息请求，因此系统宏观上需为每一个Agent分配一个独立的进程。同时多Agent理论要求系统具备灵活性，能够快速简易地对Agent节点进行添加或删除，而进程的销毁、创建以及上下文切换需要较多的处理和计算时间，不满足Agent的移动性。相比于进程，线程的粒度更小，线程创建、销毁以及切换所需的开销远远低于进程，因此本系统将每个Agent封装为对应的线程

全局管理Agent、算法Agent、工艺Agent和监控Agent之间以分布式的结构相互联系，因此对应的线程应单独运行在一个进程中。在生产任务跨区域分解过程中，每个资源Agent组中的子管理Agent根据其下各车间Agent的设备配置来判定是否有能力完成任务，该过程涉及大量的数据共享和状态同步操作，对请求的响应速度有很高的要求，若把车间Agent对应的线程运行在独立的进程中，跨进程的通信过程中涉及的连接建立、数据验证、连接释放等操作会降低生产任务分解的效率，因此本系统为每个子管理Agent开辟线程池，其下的车间Agent作为线程池中的线程，由子管理Agent统一管理。Java提供丰富的线程操作和并发机制来兼顾数据的一致性和通信的高效性。

设备Agent对应系统的生产设备，实时模拟设备的工作进度和加工序列，需要对采集的设备数据反应非常灵敏，若把同属一个车间的设备Agent由线程池管理，当设备数量较多而CPU核数较少时，会触发频繁的线程中断、切换、挂起操作，此时设备Agent难以及时读取设备的工作信息，无法做到数据的实时模拟，有可能导致算法Agent的调度方案出现偏差，因此本系统同样是把设备Agent分配到独立的进程中。

**5.3.2 Agent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。

**5.4 Agent的模型设计**

Agent最基本的功能是能够接收外界的信息，根据自身的状态、知识和数据等进行决策，从而发起回应，这是Agent间能够实现协同合作的基础，体现Agent自主性、社会性的一句，因此本系统在进行各类型Agent的开发时设计了如下图所示的基本Agent模型。每个Agent都需要配置通信模块、执行模块、决策库和数据库。

（1）通信模块除了包括5.2.2中Agent的通信模型外，还需为Agent制定唯一标识（AID），AID包含了Agent的IP、监听端口、Agent类型、位置信息等，本系统把AID封装为Java对象，Agent间进行消息传输时需要捎带AID，以帮助接收方进行身份合法性审核。因此通信模块使用Socket通信接口实现Agent双方的信息交互，使用线程池和消息队列保证通信效率和质量。

（2）执行模块和决策库。决策库是Agent自主性的体现，库中决策规则的作用范围和效果决定了Agent学习能力的强弱，是Agent行动的依据，而执行模块是Agent行动的支撑，该模块接收外界信息后，根据决策规则的约束和指引从而采取对应的行为和响应，如更新数据库中数据，向外界提供数据信息等。本系统为各类型Agent分别设计对应的决策库，在程序中每条决策规则对应一个函数，因此决策库在程序中体现为函数封装库，被编译为JAR包为执行模块提供函数查询的接口。调度模块在程序中体现为每个Agent线程运行的入口函数，接收到外界的请求后，根据请求的内容向决策库寻求行动支持，行为过程可能涉及对数据库的操作。

（3）数据库：为Agent提供数据的增删查改操作。不同的Agent数据库信息不同。本系统使用MySQL实现对各Agent数据的操作。

下面针对各类Agent的模型差异进行说明。

全局管理Agent：负责接收、审核生产任务，协调生产任务在各资源Agent组中的分解，协调系统的重调度流程等，因此相对于其他Agent，GMA需要设计一个任务队列，专门用于存储来自客户的生产任务，其次需要设计加工任务队列，用于存储已完成分配的生产任务，当发生订单撤销、订单更改时，能快速定位到对应的生产任务，提高响应速度。同时，GMA是系统的中枢所在，需提供注册注销模块，Agent只有在GMA中注册AID才能与其他Agent进行信息交换，当从系统中移除时，从该模块注销AID。GMA数据库中存储用户信息、工作日志、质检数据、设备故障信息等。

子管理Agent：作为对应资源Agent组与GMA通信的接口，负责协调其下的车间Agent进行生产任务的分解工作，因此需要设计一个应答结果集，专门存储下层Agent对生产任务是否能够完成的应答。数据库中存储对应工厂的产品信息、车间信息等。

车间Agent：根据其下设备Agent的配置来判断是否有能力完成生产任务，Agent模型中无需添加额外的模块，数据库中存储其中的设备信息，如设备加工能力以及数量等。

设备Agent：读取对应设备的工作数据，实时模拟设备的工作状态和加工序列，因此需要设计仿真模块，数据库中存储对应设备的历史工作信息、故障信息等。

算法Agent：接收资源Agent组传递的设备集和生产任务，根据设备集中各设备的工作进度和加工序列进行工序任务的分配，生成对应的调度方案。无需添加额外的模块，数据库中记录历史调度结果，作为调整算法参数的参考依据。

工艺Agent：在功能上相当于数据库，存储系统内的产品信息、工件的工艺规划以及工序的设备选择和对应加工时间等,为其他Agent提供工艺信息的查询接口。同时对管理员对工艺信息的操作进行审核，该Agent无需添加额外的模块。

监控Agent：读取监控设备的监控信息，当设备发生故障时，向对应的子管理Agent发起通知。无需添加额外的模块。

5 仿真实验与结果分析

本文的车间动态调度系统在实验设计上主要从三个方面验证系统的可行性，首先验证基于改进蚁群算法的调度策略在FJSP的调度性能，与基本蚁群算法AOC、遗传算法GA以及相关参考文献的计算结果和效率进行对比并分析。

5.1 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 Win 10 32位 ,开发平台 eclipse， JDK版本 jdk1.8.0\_144

5.2 实验一：基于改进蚁群算法的调度策略验证

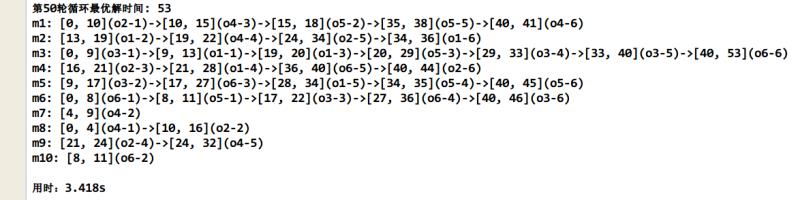
该实验用于验证本文的基于改进蚁群算法的调度策略在FJSP中的可行性，优化目标为最小化最大完工时间。文献[26]提出了一种基于多种群蚁群算法用于FJSP求解中，该文献对从某航空发动机公司提取的6\*10柔性生产调度实例对算法进行了验证，该实例包含6种工件，每种工件各包含6道工序，调度的设备集共有10个设备。各工件的工序组成、工序的设备选择以及对应的加工时间如下表所示：

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 工件 | 工序 | 加工设备集 | | | | | | | | | |
| m1 | m2 | m3 | m4 | m5 | m6 | m7 | M8 | m9 | m10 |
| p1 | o1,1 | - | - | 4 | - | 10 | - | - | 12 | - | 15 |
| o1,2 | - | 6 | - | 8 | - | - | - | - | - | 11 |
| o1,3 | - | - | - | - | - | 9 | - | - | 4 | - |
| o1,4 | - | - | - | 7 | - | - | - | - | - | - |
| o1,5 | - | - | - | - | - | 6 | - | - | - | - |
| o1,6 | - | - | - | - | - | - | 13 | - | - | 14 |
| p2 | o1,1 | 10 | - | - | - | - | 15 | - | - | - | - |
| o1,2 | - | - | 8 | - | - | - | - | 6 | - | - |
| o1,3 | - | - | - | 5 | - | - | 7 | - | - | - |
| o1,4 | - | - | - | 4 | - | 8 | - | - | 3 | - |
| o1,5 | - | 10 | - | - | - | - | 12 | - | - | - |
| o1,6 | - | - | - | 4 | - | - | 7 | - | - | - |
| p3 | o1,1 | - | - | 9 | - | - | - | - | - | - | 14 |
| o1,2 | - | - | - | - | 8 | - | - | - | - |  |
| o1,3 | - | 9 | - | - | - | - | - | 6 | - |  |
| o1,4 | - | - | 4 | - | - | 6 | - | - | - |  |
| o1,5 | - | - | 7 | - | - | - | - | - | - | 9 |
| o1,6 | - | - | - | - | - | 6 | - | - | 9 |  |
| p4 | o1,1 | - | - | - | 5 | - | - | 4 | - | - |  |
| o1,2 | - | - | 6 | - | - | - | 5 | - | - |  |
| o1,3 | 5 | - | - | 8 | - | - | 11 | - | - | 9 |
| o1,4 | - | 3 | - | - | 11 | 12 | - | 5 | - | 10 |
| o1,5 | - | - | - | - | - | 8 | - | - | 8 |  |
| o1,6 | 1 | - | - | 5 | - | - | 7 | - | - | 8 |
| p5 | o1,1 | 6 | - | - | - | - | 3 | - | - | - | - |
| o1,2 | 3 | - | - | - | - | 6 | - | - | - | 8 |
| o1,3 | - | - | 9 | - | - | - | - | - | - | 11 |
| o1,4 | - | 7 | - | - | 1 | - | - | - | - | 9 |
| o1,5 | 5 | - | - | 14 | - | - | - | - | - | - |
| o1,6 | - | - | - | - | 5 | - | - | 8 | - | - |
| p6 | o1,1 | - | - | - | - | 10 | 8 | - | - | - | - |
| o1,2 | - | - | - | 3 | - | - | - | - | - | 3 |
| o1,3 | - | - | - | - | 10 | - | - | - | - | - |
| o1,4 | - | - | - | - | - | 9 | - | - | - | 10 |
| o1,5 | - | - | - | 4 | - | - | - | 7 | - | - |
| o1,6 | - | - | 13 | - | - | 15 | - | - | - | 14 |

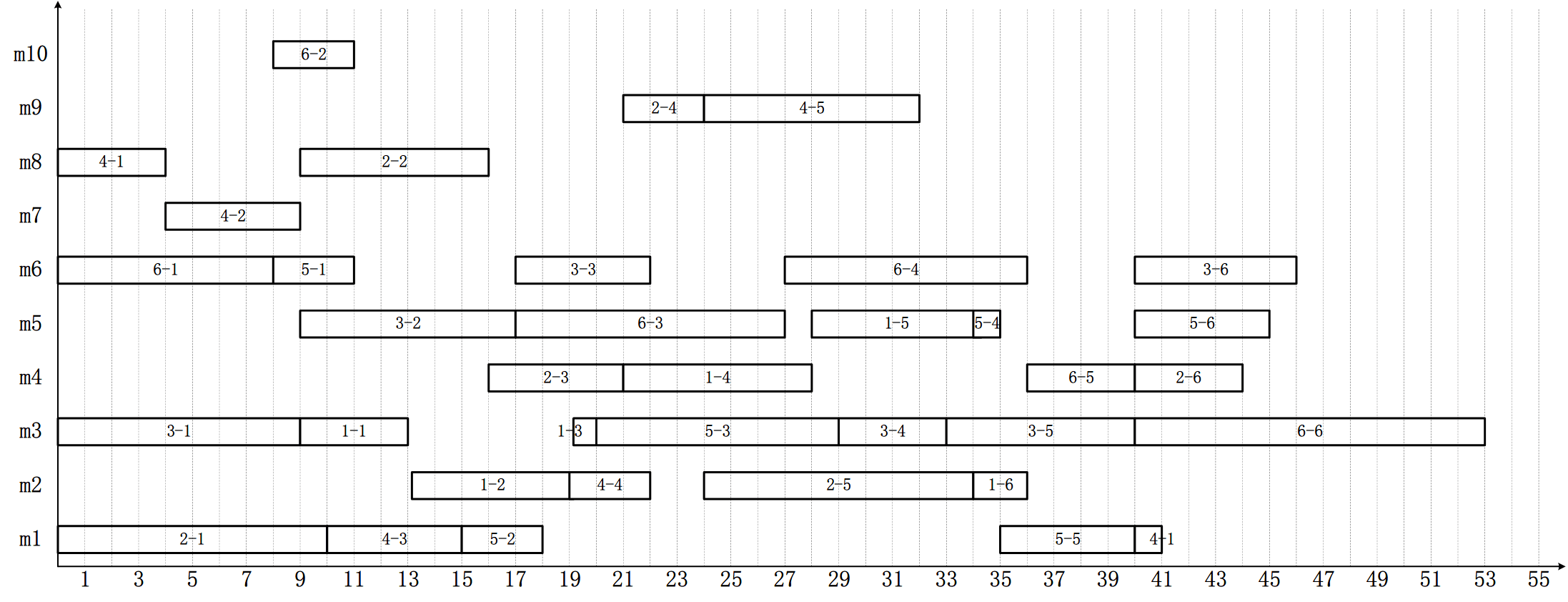
本文的基于改进蚁群算法调度策略的参数设置如下：

|  |  |
| --- | --- |
| 参数名称 | 参数值 |
| 工件数量 | 6 |
| 工序初始设备选择数 | 75 |
| 蚂蚁数量m | 75 |
| 循环次数N | 50 |
| 忽略信息素影响循环次数 | 5 |
| 第一次检查点循环次数c1 | 15 |
| 第二次检查点循环次数c2 | 30 |
| 信息素权重系数 | 2 |
| 能见度权重系数 | 3 |
| 初始信息素浓度 | 0.1 |
| 每轮循环释放信息素总量 | 100 |
| 节点间信息素浓度最小值 | 0.02 |
| 节点间信息素浓度最小值 | 8 |
| 信息素挥发因子 | 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，所得最优解时间为53，在eclipse中的运算结果如下图，转换为对应甘特图后如下图所示。文献[25]得到的最优解时间同样为53，对应甘特图如图所示。



50次循环所得的最优解



设备-时间甘特图

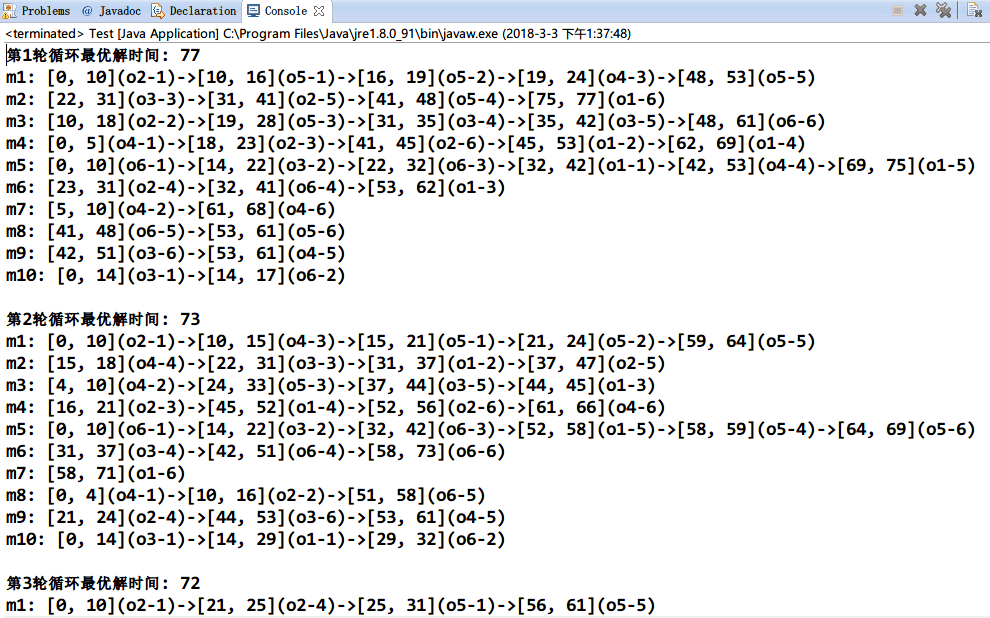


文献最优解甘特图

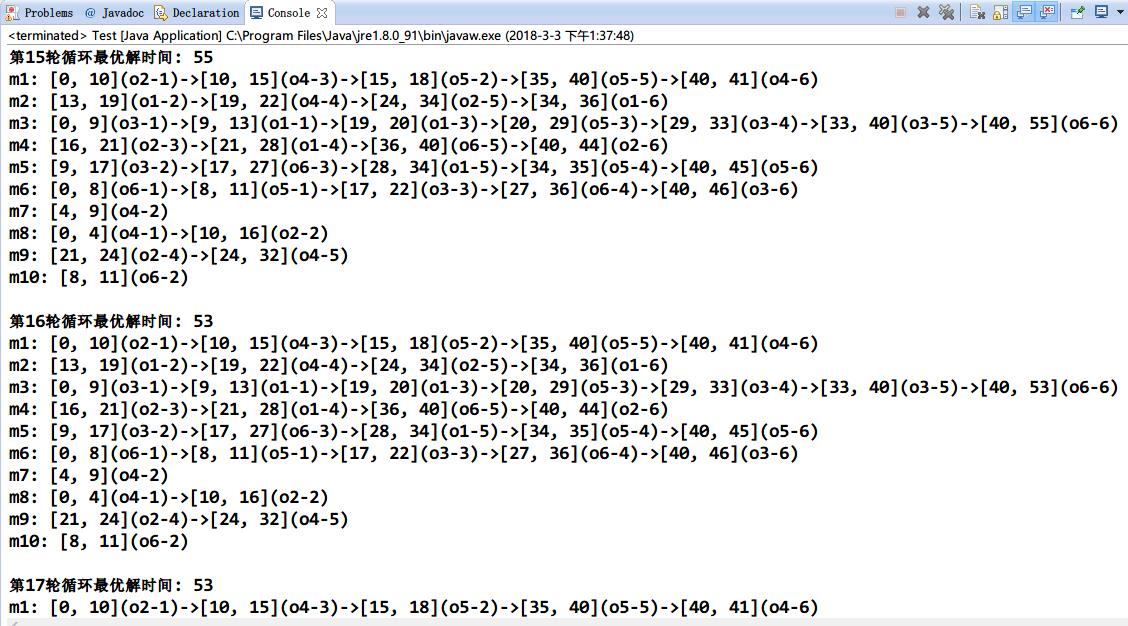
文献[25]使用多种群蚁群算法（MACO）、基本蚁群算法（ACO）、遗传算法（GA）以及两者的混合算法（ACO-GA）在前30次循环中的求解结果如下，从平均解和最差解的角度去衡量，本文所改进的蚁群算法在能够求解最优解的同时，还保持较优的平均调度性能。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 算法 | 最优解时间 | 前30循环的平均解 | 最差解 | 计算时间/s |
| GA | 60 | 63.3 | 65 | 9 |
| ACO | 58 | 60.4 | 61 | 7 |
| GA-ACO | 53 | 54.8 | 55 | 6 |
| 文献[] | 53 | 54.1 | 54.5 | 4 |
| 本文改进算法 | 53 | 53.4 | 55 | 3.263 |

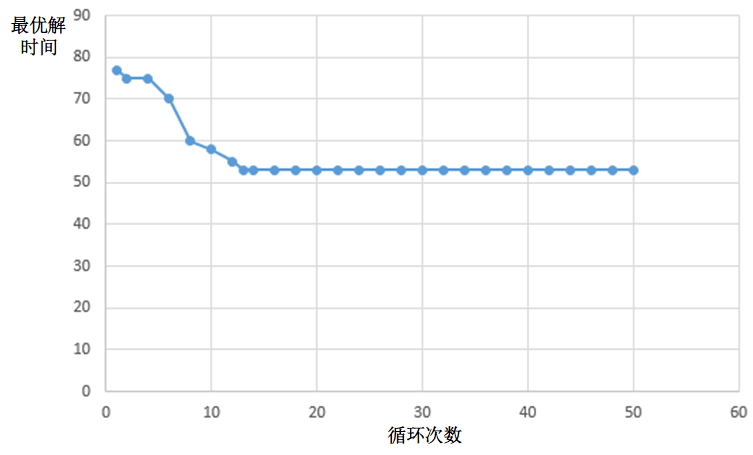
本文在基本蚁群算法的基础上，根据FJSP的柔性生产特点以及蚁群算法收敛慢易早熟的缺点对算法进行了改进，提出了基于改进蚁群算法的柔性作业车间调度算法，在要求算法能够求解最优解的同时，为了应对随时可达的客户订单以及具有复杂工艺规划的产品，还要求算法有较高的求解效率。下图是对算法执行过程中的50次循环的部分最优解，图是该算法的收敛曲线，算法在第22次循环中收敛到了时间为53的最优解，在前十次循环中，算法收敛速度较平缓，这是由于算法在前10%的循环中忽略信息素影响，由能见度内的工序的完工时间计算状态转移概率。在循环次数达到15次时，收敛速度大幅提高，并在22次循环时收敛到了最优解。图是文献[25]提供的几种调度算法的收敛曲线图，其中表现最好的MACO在第28次循环时收敛到最优解，相比之下，本文的算法具有更快的收敛素的。由此可见，本文的基于改进蚁群算法的柔性车间调度算法具有快速得到最优解的性能，能够很好地适用于FJSP的求解中。

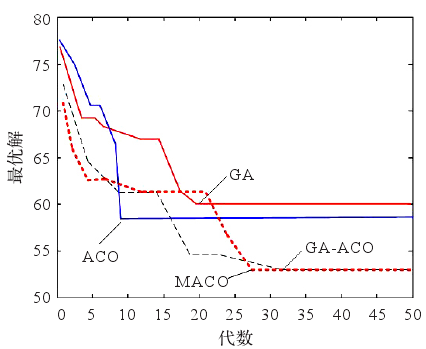


前3次循环的最优解



第13次循环收敛得到最优解



收敛曲线  
  


文献收敛曲线

5.3 实验二：验证基于多Agent的跨区域生产任务分解策略

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[26]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，其中每个工件由3到工序，工序的设备选择及对应的加工时间如下表。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 工件 | 工序 | 设备选择和对应的加工时间 | | |
| m1 | m2 | m3 |
| p1 | o1,1 | 2 | 3 | 2.5 |
| o1,2 | 1 | 1 | 0.5 |
| o1,3 | 3 | 1.5 | 0.5 |
| p2 | O2,1 | 1 | 1.5 | 2 |
| o2,2 | 1 | 2 | 0.5 |
| o2,3 | 3 | 1.5 | 0.5 |
| p3 | o3,1 | 2 | 1 | 1 |
| o3,2 | 1 | 0.5 | 2 |
| o3,3 | 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，如图所示。企业2的设备均为空闲，加工序列为空。



企业1设备原始加工序列甘特图

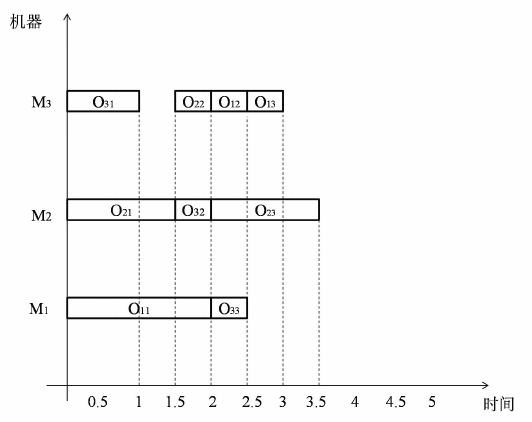
在本实验中，基于以上调度实例把两家企业对应为位于不同地区的两间加工工厂，从而得到的两个资源Agent组的组织结构如图所示。实验初始化时，把三个工件任务封装为一个完整的产品任务，由全局管理Agent广播至两个子管理Agent。由于两间工厂设备配置相同且均有三个设备，因此在跨区域分解过程中两个资源Agent组均会得到设备集，分别为{m1,m2,m3}和{m1’,m2’m3’}，而后把设备集中Agent模拟的设备加工序列以及生产任务TM发送到算法Agent，算法Agent计算调度方案后返回到对应的资源Agent组。算法Agent执行本文改进的蚁群算法后对应的最优解甘特图如图所示。其中是工厂1的最优解，时间为4，图是工厂2的最优解，时间为3.5；由于工厂1的设备存在工序占用，调度方案在完工时间上超过工厂2，因此最终全局管理Agent从中选择工厂2。



基于工厂1资源配置的最优解甘特图



基于工厂2的资源配置的最优解甘特图



文献[27]在云制造环境中筛选的最优解甘特图

文献[27]中列出了文献[28]和文献[29]算法在该调度实例下执行100次计算的统计结果，前者使用基于竞争规则的多种群蚁群算法，后者使用多目标蚁群算法进行局部搜索，结果如下表。根据表中数据可知，本文的跨区域任务分解策略配合改进的算法在能够得到最优解的同时，计算时间更少，这是由于通过减少调度设备集的规模，降低了算法Agent的计算量，从而更加快速地求得最优解。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 文献[27] | 文献[28] | 文献[29] | 本文算法 |
| 工件数 | 3 | 3 | 3 | 3 |
| 工序数 | 9 | 9 | 9 | 9 |
| 工厂数 | 2 | 2 | 2 | 2 |
| 设备数 | 6 | 6 | 6 | 6 |
| 平均解 | 3.7 | 3.9 | 3．7 | 3.723 |
| 最优解 | 3.5 | 3.5 | 3.5 | 3.5 |
| 平均计算时间/s | 1.3 | 1.5 | 1.7 | 0.276 |

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，能够快速高效地得到最优的资源分配方案。

5.4 实验三：验证设备故障条件下的多Agent重调度协商策略

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、重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。

本实验根据第四章中软件架构的设计，把各类型Agent封装为对应的线程，通过线程间的通信来构建基于多Agent的分布式调度系统。该实验基于文献[26]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所示。文献[26]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。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 工件 | 工序 | 设备选择以及对应的加工时间 | | | | |
| m1 | m2 | m3 | m4 | m5 |
| p1 | o1,1 | 3 | 4 | - | - | - |
| o1,2 | - | 8 | - | 9 | - |
| o1,3 | - | - | 9 | - | 7 |
| o1,4 | - | 5 | - | 4 | - |
| o1,5 | - | 3 | - | - | 3 |
| p2 | o2,1 | 6 | - | - | 8 | - |
| o2,2 | - | 8 | - | 6 | - |
| o2,3 | 5 | - | 4 | - | - |
| o2,4 | - | - | 2 | - | 3 |
| o2,5 | - | 3 | - | - | 4 |
| p3 | o3,1 | - | - | 4 | - | 6 |
| o3,2 | 5 | - | 7 | - | - |
| o3,3 | - | 7 | 9 | - | - |
| o3,4 | 5 | 5 | - | - | - |
| o3,5 | - | - | - | 6 | 6 |
| p4 | o4,1 | - | 9 | - | - | 7 |
| o4,2 | - | 3 | 5 | - | - |
| o4,3 | - | 4 | - | 6 | - |
| o4,4 | - | - | - | 3 | - |
| o4,5 | - | 3 | - | - | 7 |
| p5 | o5,1 | 6 | 4 | - | - | - |
| o5,2 | - | - | 7 | 5 | - |
| o5,3 | - | - | - | 7 | 9 |
| o5,4 | 8 | - | 7 | - | - |
| o5,5 | - | - | 5 | 5 | - |
| p6 | o6,1 | - | 3 | 7 | - | - |
| o6,2 | 9 | - | - | 7 | - |
| o6,3 | - | - | 6 | - | 9 |
| o6,4 | - | - | 6 | 7 | - |
| o6,5 | - | 4 | - | 4 | - |

根据该实例，该实验设置两个车间Agent线程,JSAT1和JSAT2，JSAT1内包含3个设备Agent，对应设备m1,m2和m3，JSAT2内包含2个设备Agent，分别对应设备m4,m5，同时监控Agent线程数量为1个，对应为MAT，用于设备故障发生时进行通知。由于紧急订单和订单撤销的调度策略较为简单，本文进行在设备故障条件下的重调度验证实验。

该实验中，设备故障由监控Agent线程MAT触发，而后MAT基于Socket通信机制把故障信息发送到子管理Agent，故障影响为设备需停工，故障设备上工序需转移。子管理Agent进行指令下达，通过资源Agent组间信息传递最终到达设备Agent，进行故障设备上的工序回收，重新封装为新的工件任务在资源Agent组间进行分解分配，最终得到重调度方案。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；新工件任务封装完成到重调度方案生成定义为重调度时间，体现系统调度算法的效率；重调度方案与正常调度方案的完工时间差值用于衡量系统重调度方案的有效性。

根据3.1中对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重调度，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oij，oi’j’为某一设备k加工工序序列TSk中的相邻工序，当时间t满足以下条件时，可触发设备故障:

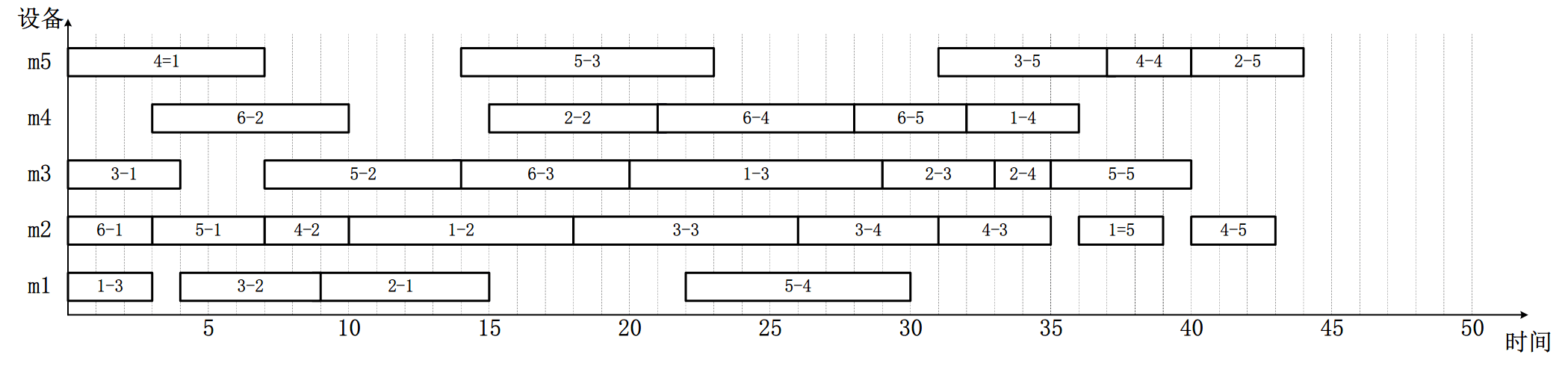
Eijk=<t<=Si’j’k

（2）任意时刻只有一台设备触发设备故障。

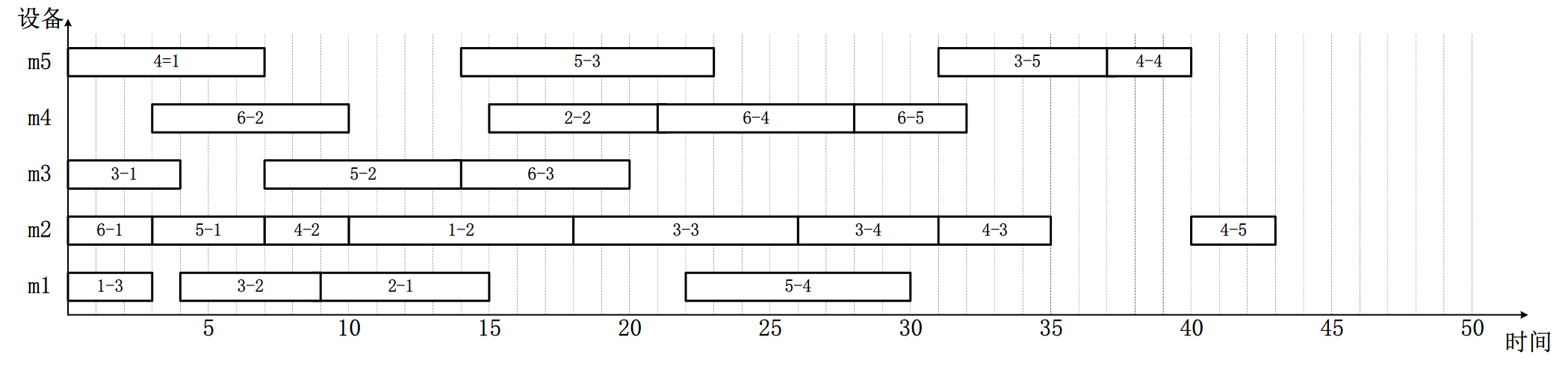
（3）设备故障触发概率设置为50%

（4）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。

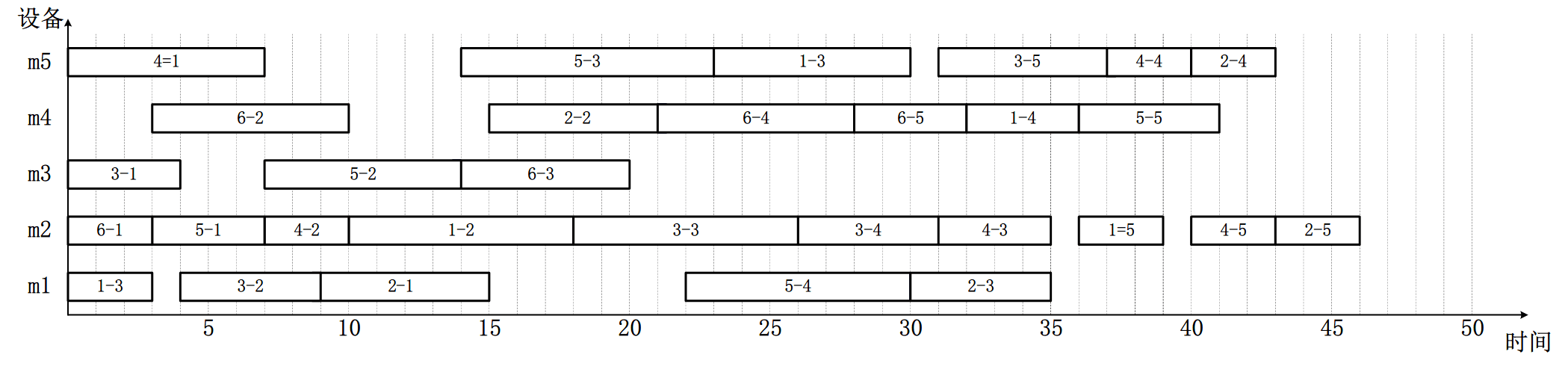
本实验基于甘特图对应的调度方案进行，在指定设备m2在时间t=20故障的条件下，系统反应时间以及重调度方案如下图，对应的甘特图如图。反应时间为0.297s,重调度时间为0.783s,完工时间为46，与正常调度的完工时间44的差异为2。可见在设备故障发生时，各Agent间通过快速灵敏的通信，能够及时开启重调度，减少与原生产计划的误差。



正常调度的设备-时间甘特图



故障发生、工序回收后的设备-时间甘特图



重调度后的设备-时间甘特图

以50%的概率随机触发空闲设备故障，在100次故障中重调度的反应时间、重调度时间，时间差异如下表所示。故障发生时间越早，所需的重调度时间越长，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。而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之间信息的传递决定，依赖Agent之间的通信质量和效率。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 时间段 | 故障次数 | 平均反应时间/s | 平均重调度时间/s | 平均时间差 |
| 0-5 | 11 | 0.638 | 2.764 | 1.32 |
| 5-10 | 15 | 0.569 | 2.863 | 1.80 |
| 10-15 | 12 | 0.471 | 2.368 | 3.51 |
| 15-20 | 5 | 0.587 | 2.472 | 2.36 |
| 20-25 | 18 | 0.483 | 1.806 | 2.43 |
| 25-30 | 16 | 0.431 | 1.342 | 2.26 |
| 30-35 | 11 | 0.472 | 0.879 | 4.71 |
| 35-40 | 12 | 0.433 | 0.203 | 6.25 |
| 40-45 | 0 | - | - | - |

平均反应时间为0.511s,平均重调度时间为1.837s,平均时间差为3.08，最优调度时间差异为1，最差调度时间差异为9。由以上数据可知，当故障时间发生越早，重调度方案与正常调度时间差越小，这是由于时间早，回收的工序多，重调度接近正常调度，因此需要的计算时间更长；若故障发生时间接近调度尾声，由于回收工序少，计算时间大幅降低，但同时工序转移选择少，时间差可能偏大；从中还可看出重调度过程中反应时间与故障发生时间无关，这是由于反应时间主要是由Agent间通信所占用的时间决定，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。最终可得到如下结论，系统在故障设备发生时反应时间和重调度时间短，与正产调度时间差较小，说明系统对异常状况反应灵敏，重调度执行时间短，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的协商策略的有效性。

[1] 基于多Agent的可中断作业车间调度问题研究\_万明谊