申请上海交通大学博士学位论文

基于问题分解与蚁群算法的半导体晶圆制造系统调度方法的研究

学　　校： 上海交通大学

院　　系： 机械与动力工程学院

博 士 生： 郭乘涛

工程领域： 机械工程（工业工程）

导　　师： 江志斌（教授）

上海交通大学机械与动力工程学院

2012 年 5 月

A Dissertation Submitted to Shanghai Jiao Tong University for the Degree of Philosophy Doctor

THE RESEARCH ON SCHEDULING OF WAFER FABRICATION SYSTEM BASED ON DECOMPOSITION METHOD AND ANT COLONY OPTIMIZATION ALGORITHM

Author: GUO Chengtao

Specialty: Mechanical Engineering (Industrial Engineering)

Advisor: Prof. JIANG Zhibin

School of Mechanical Engineering

Shanghai Jiao Tong University

Shanghai, P. R. China

May, 2012

上海交通大学
学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。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结果由本人承担。

学位论文作者签名：郭乘涛

日期：2012 年 5 月 2 日

上海交通大学

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，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在___年解密后适用本授权书。

本学位论文属于

不保密☑。

（请在以上方框内打“√”）

学位论文作者签名：郭乘涛　　指导教师签名：

日期：2012年5月2日　　日期：2012年5月2日

基于问题分解与蚁群算法的半导体晶圆制造系统调度方法的研究

摘 要

半导体晶圆制造系统是半导体制造过程中最为复杂和昂贵的环节，晶圆制造系统的调度问题是一类复杂的车间调度问题。尽管目前晶圆制造系统求解调度问题最常见的方法是启发式调度规则，但是智能算法在未来的晶圆制造系统中应用的潜力非常大，这是因为智能算法具有自组织与自调整能力，可以适应调度问题的变化，并在可接受的时间内获得高质量的解。蚁群算法是一类产生不久但发展迅速的智能算法，它不仅具有一般智能算法的优点，而且具有并发多线程搜索机制与正反馈特点，可以更迅速地获得满意的可行解，因而近年来被逐渐应用于解决生产调度问题。但是到目前为止，应用蚁群算法求解晶圆制造系统调度问题的研究还很不深入，这是因为传统蚁群算法不能很好地应对晶圆制造系统的复杂性，特别是其大规模、多重入、混合型生产的特征。

基于以上的原因，本文提出基于问题分解的蚁群算法，有针对性地对传统蚁群算法进行改造，并结合问题分解方法，以期获得晶圆制造系统调度问题的高质量解。本文所取得的主要工作成果包括：

1）建立应用蚁群算法调度晶圆制造系统的问题分解框架。

元启发算法可解决较大规模的调度问题，但是当其用于解决晶圆制造调度这样的大规模调度问题时，随着问题规模急剧扩大，元启发算法的应用会受到制约。为此，应用问题分解方法将大规模问题分解为小规模的子问题，再用蚁群算法分别求解，就成为一条可行的解决途径。本论文提出两类适合晶圆制造系统特征的问题分解方法，包括基于时间的问题分解方法与基于机器类型的问题分解方法。前者用来应对晶圆制造系统的大规模特征。通过对调度时间的分割，在每个时间区间内只需要处理部分机器与

工件，降低了问题的规模与难度；后者用来应对混合型生产的特征，通过对机器按照类型进行分解，针对每一类型采用相应的蚁群算法进行处理，使得调度问题更易于解决。在此基础上，建立应用蚁群算法调度晶圆制造系统的问题分解框架，以有效处理大规模晶圆制造系统调度问题。

2）提出调度问题分解协调机制。

问题分解是一个将各子问题解耦的过程，在各子问题单独求解完成后，这些解合成原问题的解又面临如何协调的问题，这两个问题如何处理是问题分解方法最需要考虑的问题。本文提出三类协调机制，首先针对基于时间的问题分解方法，提出基于重叠区间的协调机制，保证了原问题调度方案的可行性；其次提出了基于工件上下游的协调机制，通过上下游机器的加工信息，决定当前机器的调度策略，以提供调度方案的精度；最后采用蚁群算法的信息素更新机制作为隐性协调机制，针对基于机器的分解方法，采用蚁群算法的信息素保证分解后的机器间信息的沟通。

3）针对晶圆制造系统与问题分解方法特征对现有蚁群算法进行改进。

为了应对晶圆制造系统的复杂性特征，以及适应前述问题分解方法，对现有蚁群算法进行了改进：首先，针对晶圆制造系统的多重入特征，开发基于重入的蚁群算法，针对加工目标对不同加工阶段的工件赋以不同的信息权重，使其可以获得更好的调度方案。其次，应对基于类型的分解方法，当将子问题分解为若干单机调度问题后，针对其中较难调度且经常为系统瓶颈的批处理设备开发专门的蚁群算法，以保证整个调度问题的顺利求解。最后，提出分类蚁群算法，使得多种类型设备的调度问题可以在一个算法构架下共存。

最后，采用仿真实验对所提出的问题分解方法、协调机制与蚁群算法进行验证，实验分别针对单台设备调度问题、子问题调度、大规模调度问题而进行，实验结果证明了本文所提出方法的可以有效地处理晶圆制造系统的复杂特征，并获得高质量的调度方案。

关键词：蚁群算法，调度，半导体晶圆制造系统，问题分解方法

THE RESEARCH ON SCHEDULING WAFER FABRICATION SYSTEM WITH DECOMPOSITION METHOD AND ANT COLONY OPTIMIZATION ALGORITHM

ABSTRACT

Semiconductor wafer fabrication system (SWFS) is the most complicated and expensive part of the semiconductor manufacture system and the scheduling problem of wafer fabrication system is a type of complex job shop scheduling problem. The most extensive used scheduling method in SWFS is the heuristic rules. However, the intelligent algorithms are more potential to broadly apply in SWFS in the future, because they have the ability of self-organization and self-adjustment, which can adapt the change of the scheduling problem and obtain the high quality solution in reasonable calculation time. The Ant colony optimization (ACO) method is a new but developed very fast intelligent algorithm. It has the features of concurrent multi-thread calculation ability, positive feedback and self-organization, which can help the algorithm obtain the solution much faster. Due to the above reasons, the ACO algorithm is applied to solve the scheduling problem in many fields recently. However, it is not successful applied to solve the scheduling problem of SWFS up to now because the traditional ACO algorithm cannot cope with the complexity of SWFS, especially the features of large scale, multi-reentrant and mixed production.

This paper proposed the decomposition based ant colony optimization method to handle the problem stated above. The proposed method modified the tradition ACO algorithm and combined with the decomposition method to find high quality solution for the scheduling problem of SWFS. The major

contributions are stated as follows.

1. Construct the decomposition framework on scheduling wafer fabrication system with ACO algorithm

The meta-heuristic algorithm can be used to solve the large scale scheduling problem. However, its application will be restricted when the size of problem increased rapidly. As a result, the large scale problem needs to be decomposed into several sub-problems with smaller size and then be scheduled by ACO algorithms. In the thesis, two type of decomposition are presented to cope with the features of wafer fabrication system, including the time-based and machine-type-based decomposition methods. The former method is used to cope with the feature of large scale of SWFS. It divided the scheduling period, and only a part of machines and jobs need to be scheduled in a small period, which decreases the size and difficulty of scheduling problem. The latter one is used to cope with the feature of mixed production. It decomposes the scheduling problem according to the machine-type, and every type of machine is solved by the according ACO algorithm, which make the scheduling problem is solved easily. The decomposition framework is constructed based on these two decomposition methods and ACO algorithms.

2. Develop the coordination mechanisms for the decomposed sub-problems

After the sub-problems have been decomposed and scheduled, the schedules of the sub-problems need to be combined to the solution of original problems, and the coordination mechanisms are needed to adjust these sub-problem schedules. In this thesis, three type of coordination mechanism are presented. The first type was based on overlapping area which is constructed between two neighbor sub-problems and a part of operations were belongs to these two sub-problems. This coordination mechanism can ensure the continuity of the schedule obtained. The second one considered the up and down stream of the process. After the scheduling problem was decomposed into several single

machine scheduling problems by the machine-type decomposition method, the information of the up and down stream of the job should be considered when the current operation was processed. The third coordination mechanism is based on the pheromone updating rule. Different type of machine can run in the same scheduling framework according to the delivery of pheromone.

3. Improve the traditional ACO algorithm to cope with the features of wafer fabrication system and decomposition method

In order to cope with the complicated features of wafer fabrication system and the decomposition method stated above, the traditional ACO algorithm need to be modified. This paper proposed three type of modified ACO algorithm. First, the ACO algorithm based on re-entrant feature is proposed to cope with the multi re-entrant feature of SWFS. The algorithm places different weights to the job at different processing stage to find better solution. Second, two new hybrid-ACO algorithms were proposed to schedule the batch machine, which is hard to schedule and is always the bottleneck of the system. Finally, the classified ACO algorithm is proposed to schedule different type of machine in the same framework.

In the end, the simulation experiments are conducted to testify the proposed decomposition methods, the coordination mechanisms and the modified ACO algorithms. The experiments are executed on single machine scheduling problem, the sub-problem scheduling and the large scale scheduling problem respectively. The results show that the proposed methods can handle the complicated features of the wafer fabrication system and obtain the high quality solution.

Keywords: Ant colony optimization (ACO), scheduling, semiconductor wafer fabrication system (SWFS), decomposition method

目 录

第一章 绪论

1.1 课题的研究背景及意义

近二十年来，随着超大规模集成电路（Very Large Scale Integration, VLSI）和甚大规模集成电路（Ultra Large Scale Integration, ULSI）技术的广泛应用，以及相关信息、通讯电子产业的高速进步，半导体产业蓬勃发展。作为新兴的战略性工业，半导体产业的技术水平和发展规模已经成为衡量一个国家经济发展和科技进步的重要标志，对于国家的综合实力起着巨大的影响和推动作用。

半导体晶圆制造系统（Semiconductor Wafer Fabrication System, SWFS）是半导体制造过程中最为复杂和昂贵的环节，先进的 SWFS 生产调度技术对于半导体制造业所产生的经济效益尤为重大，在制品水平每降低 1%将减少数百万元的成本，而制造周期每缩短 1%将增加数千万元的产出。我国目前半导体芯片产业发展迅速，但生产调度还是以简单的调度规则为主，技术相对落后，亟待利用先进的生产调度理论来提升其调度水平。

晶圆制造系统的调度问题是一类复杂的车间调度问题。车间调度问题从 20 世纪 50 年代开始被广泛研究，产生了大量的求解方法，可被归纳为三类：数学规划方法、调度规则方法与元启发式算法。数学规划法有严格的数学推导过程，结果相对精确，在制造单一或少量品种的车间调度问题中取得良好的理论和实践效果，但是很多实际的生产调度问题难以量化表示，因而其使用受到很大限制。调度规则方法简单易行，计算量小，因而在实际生产调度过程中得到广泛的应用。但是调度规则大多基于局部的信息来进行决策，有可能出现所产生的解比全局最优解差很多的情况，而且差的程度难以衡量；另一方面，调度规则对所应用问题的针对性较强，如果想达到较好的调度效果，必须根据具体调度问题的特征选择合适的规则并进行修改及参数优化，而当调度问题出现变化时又需要及时地调整，因而对规则制定者的经验与理论水平的要求很高。元启发式算法是一类邻域搜索方法，它是以数据为基础，通过训练来建立各个关联体间的联系。当车间调度问题的元启发式算法搜索框架建立后，算法在运行过程中，会通过反复的迭代搜索，逐步接近最优解。当问题在一定的范围内发生变化时，算法可以自行调整，重新寻找新问题的近优解。元启发式算法还能以一定的概率接受劣解，从而逃离局部最优。随着晶圆制造自动化程度的提高，人工干预程度的降低，可以预见基于元启发式算法的调度方法必然在未来的晶圆制造系统中得到越来越广泛的应用。

蚁群算法是一类产生不久但发展迅速的元启发式算法，它具有并发多线程计算、正反馈与自组织的特征。并发多线程计算可以提高搜索的效率，正反馈可以较快地收敛到较优解，自组织可以在一个搜索框架下适应不同的问题。由于蚁群算法这样的特点，近年来被逐渐应用于解决生产调度问题，取得了良好的效果。但是到目前为止，应用蚁群算法求解晶圆制造系统调度问题的研究还很不深入，这是因为传统蚁群算法由于自身的局限，无法处理晶圆制造系统的复杂性特征，具体表现在以下三个方面：：

1. 难以应对晶圆制造系统的大规模特征。晶圆制造系统在同一时刻存在上百台机器运作，加工上千个工件，其调度问题规模庞大，蚁群算法相对来讲运算时间较长，因而无法在可接受的时间内处理这样庞大的问题；
2. 难以应对晶圆制造系统的复杂多重入特征。由于晶圆制造机器的昂贵以及制造过程中多个工序的相似性，因而一个工件在加工过程中会多次进入同一个机器。对该机器而言，除了要加工不同类型的工件外，还需要加工同一类型处于不同阶段的工件，这也增加了调度的困难度。传统的蚁群算法没有很好的考虑晶圆制造系统的这一特征；
3. 难以应对晶圆制造系统的混合型生产特征。晶圆制造过程中有多种类型的机器同时加工，这些不同类型的加工机器具有不同的加工特性，因而造成了调度的困难。目前蚁群算法应用于生产调度主要用来求解传统的 Job shop 和 Flow shop 问题，或是处理单台机器的调度问题，对混合型生产的研究尚未充分开展。

针对上述特征，本文提出基于问题分解的蚁群算法，有针对性地处理现有的问题，主要是从以下三个方面进行：

1. 针对晶圆制造系统的特点，相应地提出基于时间与基于机器两类问题分解方法，将大规模调度问题分解为较小、易于解决的子问题，使其可以应用蚁群算法进行求解。基于时间的问题分解方法用来应对晶圆制造系统的大规模特征，通过对调度时间的分割，在每个时间区间内只需要处理部分的机器与工件，降低了问题的规模与难度；基于机器的问题分解方法则用来应对混合型生产的特征，通过对机器按照类型进行分解，针对每一类型采用相应的蚁群算法进行处理，使得调度问题更易于解决。
2. 与问题分解方法相对应，采用三类协调机制，将子问题的解还原成原问题的解。第一类是基于重叠区间的协调，即当按时间对问题进行分解时，在相邻子问题间设置一定的重叠区间，使得一部分操作属于两个子问题所共有，从而保证调度结果的连续性。第二类协调机制是基于工件加工上下游的协调。即当按机器类型将问题分解为若干单机调度子问题时，调度一台机器的同时还需要考虑上

下游机器的加工状况，并以之做为本机调度的参数，保证上下游调度结果的一致性。第三类协调机制是基于蚁群算法自身的信息素更新机制进行的，通过信息素的传递，保证各种类型的机器可以在同一个调度框架下协调运作。

3. 对现有蚁群算法进行修改，使其能够更好地适应问题分解与晶圆制造系统的特征。首先，针对晶圆制造系统的多重入特征，开发基于重入的蚁群算法，针对加工目标对不同加工阶段的工件赋以不同的信息素权重，使其可以获得更好的调度方案。其次，当将子问题分解为若干单机调度问题后，针对其中较难调度且经常为系统瓶颈的批处理设备开发专门的蚁群算法，以保证整个调度问题的顺利求解。最后，开发分类蚁群算法，使得多种类型设备的调度问题可以在一个算法构架下共存，利用蚁群算法的信息素更新机制保持不同设备间信息的沟通。

本文试图通过上述的研究为晶圆制造系统建立一个良好的基于问题分解的蚁群算法框架，使得可以良好地处理晶圆制造系统的各类特征，从而进一步探索建立一整套运转良好的基于元启发式算法的晶圆制造系统调度方法，以提升我国半导体晶圆制造企业的生产控制水平。

1.1.1 半导体产业概况

半导体集成电路产业作为信息产业的基础和核心，是关系国民经济和社会发展全局的基础性、先导性和战略性产业，是提高产品质量，改造和提升传统产业的核心技术。集成电路应用领域覆盖了几乎所有的电子设备，并且由于其具有推动作用强、倍增效应大的特点，对于诸如计算机、家用电器、数码电子、自动化、通信、航天等产业的发展意义重大。2006-2010 年，我国集成电路行业工业产值基本保持两位数的高速增长，集成电路行业工业产值占 GDP 的比重总体上呈上升态势。2010 年，随着全球经济的复苏以及国家政策上的支持，我国集成电路行业的工业产值达到 2378.6 亿元，同比增长了 35.5%，在 GDP 中的比重为 0.6%[1]。而目前发达国家信息产业产值已占国民经济总产值的 40 % - 60 %，国民经济总产值增长部分的 65 %与集成电路有关。因此，抓住了集成电路产业发展，就能抓住国民经济发展的主动权[2]。

晶圆制造是半导体集成电路产业的核心和基础，在一个相当长的时间内都处于增长趋势。根据 iSuppli 公司的统计，2010 年全球纯晶圆代工营业收入上升为 268.6 亿美元，比 2009 年增长了 34%，超过了整体半导体行业 13.8%的增长率。而根据野村半导体的产业分析，全球晶圆代工从 2011 年到 2012 年的营收年复合增长将大升 22%，是半导体

产业增长速度的 3 倍。在未来数年，晶圆代工业的成长仍然极为迅速，由于制程技术开发成本及 12 英寸晶圆厂建厂成本的快速上扬，许多集成器件生产商 (Integrated Device Manufacturer, IDM)积极向轻晶圆厂（Fab-lite）转移，而把订单放出委托晶圆代工企业进行生产，图 1-1 为 2009-2020 年全球 IDM 委外代工产能需求的预测。受到需求的刺激，晶圆代工企业也在逐步扩大规模，积极走向超级工厂（MegaFab）的模式。所谓 MegaFab，根据业界共识，是指月产能 10-15 万片，投资金额 70-80 亿美元的 12 英寸晶圆工厂。

图 1-1 2009-2020 年全球 IDM 委外代工产能需求

Fig. 1-1 The global requirements of capacity released from IDM to foundry in 2009-2020

在全球半导体产业前景向好的大环境下，中国的半导体产业也在飞速发展。在 2001-2010 年“十五”规划与“十一五”规划期间，半导体产业被列为重点扶植产业之一。包括中芯国际、和舰科技、台积电松江厂、宏力半导体等主要晶圆制造厂商都于“十五”规划期间相继设立，并快速产能，我国晶圆代工业产值从 2001 年的 36 亿元人民币成长至 2010 年的 221 亿元人民币，年复合成长率达 21%，并已培育出一批有较强竞争力的企业，其中中芯国际、华虹 NEC 与宏力半导体已进入全球晶圆代工企业前 15 名（见表 1-1）。

表 1-1 2010 年全球晶圆代工企业排名

排名	公司	国家（地区）	业务收入（亿美元）	同比增长
1	台积电	中国台湾	133.07	48%
2	联电	中国台湾	39.65	41%
3	Global Foundries	美国	35.1	219%
4	中芯国际	中国	15.55	45%
5	TowerJazz	美国	5.1	70%
6	世界先进	中国台湾	5.08	33%
7	Dongbu	韩国	4.95	25%
8	IBM	美国	4.3	28%
9	MagnaChip	韩国	4.2	60%
10	三星	韩国	4	38%
11	SSMC	新加坡	3.3	18%
12	X-Fab	德国	3.2	51%
13	华虹 NEC	中国	2.95	23%
14	德州仪器	美国	2.85	14%
15	宏力半导体	中国	2.6	44%

数据来源：IC Insight

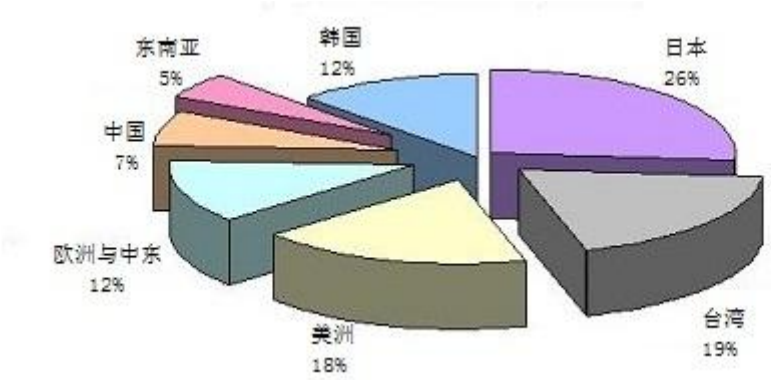
在地区分布上，我国半导体产业呈现群聚效应，表 1-2 显示了我国 2006 年到 2010 年间半导体行业企业数量区域的分布情况，半导体企业主要集中在以长江三角洲、京津环渤海以及珠江三角洲为主体的华东、华北和中南地区。特别是以上海为龙头的长江三角洲地区，拥有全国一半左右的晶圆制造企业，并已经形成了以华力、Intel、中芯国际、台积电等国际半导体生产集团为首的芯片制造加工密集区。目前，上海已初步形成了所谓“一带二区”的芯片加工区，即以张江高科技园区为核心、连接金桥出口工业区和外高桥保税区的面积达二十二平方公里的浦东微电子产业带，并连带辐射漕河泾新兴技术区与松江出口加工区。同时，作为上海制造业战略升级的重点，浦东、松江、青浦和漕河泾地区将逐步形成上海八大产业群之一的微电子及信息产业群，并将建设成具有国际先进水平的半导体芯片制造基地，成为上海市重点建设的六个世界级制造基地之一。根据“上海制造业战略升级行动纲要”，到 2020 年，上海将力争拥有 30 条集成电路生产线，成为世界最大的、技术领先的微电子制造基地。

表 1-2 2006-2010 年间半导体行业企业数量区域分布

单位：个，%

	2006 年		2007 年		2008 年		2009 年		2010 年	
	数量	累计比重	数量	累计比重	数量	累计比重	数量	累计比重	数量	累计比重
全国	385		420		452		491		493	
华北	30	7.79	34	8.10	32	7.08	38	7.74	26	5.27
东北	8	9.87	10	10.48	10	9.29	15	10.79	11	7.51
华东	202	62.34	218	62.38	231	60.40	246	60.9	261	60.45
中南	126	95.06	129	93.10	148	93.14	161	93.69	147	90.26
西南	13	98.44	20	97.86	22	98.01	24	98.57	24	95.13
西北	6	100.00	9	100.00	9	100.00	7	100.00	24	100.00

数据来源：国家统计局



数据来源：World Fab Forecast, SEM

图 1-2 2010 年全球晶圆厂产能分布

Fig. 1-2 Worldwide Installed Capacity in 2010

但从另一方面来看，我国半导体产业产值虽然大幅攀升，其整体竞争力并不强。2010 年产能仅占全球产能的 7%（图 1-2），而在制程技术上，国内技术最领先的中芯国际与领导厂商台积电、联电相比，仍然有 2 年左右的技术落差。所以 2011-2015 年的“十二五”规划期间，我国半导体产业政策发展方向将从追求产能与产值的成长，转变为先进技术与先进产能研发能力的提升，逐步培育出一批具技术创新能力且有相当全球市场占有率的半导体企业，这对我国半导体生产技术改进提出了更高的要求。

1.1.2 半导体晶圆制造业面临挑战

在过去的二十年中，半导体产业发生了巨大的变化，并在可预见的未来二十年内继续面对挑战和变革，具体表现在以下三个方面：

1. 技术变革速度加快：根据国际半导体技术路线图（International Technology Roadmap for Semiconductor, ITRS）的分析预测，国际晶圆制造业在 2007 年完善 70nm 的加工精度，在 2010 年达到 32nm 的工艺水平，2012 年将进入 28nm 制程大量投产阶段，而在 2015 年则将进入 22nm 制程阶段。技术的快速变革使得新产品以更快的速度不断涌现、产品寿命周期不断缩短。因此，以摩尔速度缩短产品制造周期、降低生产成本，在日益缩短的产品生命周期内获得最大的利润是半导体制造企业所面临的首要挑战。
2. 半导体芯片制造产业规模亟待扩大：由于开发和实施下一代制程的成本都在迅速上升。要想成为业内的领导厂商就必须要在半导体工艺研发方面走在前列，而只有规模够大的公司才能够承担这些费用。基于此，晶圆代工产业的并购成为 2008 年以来的行业焦点。Tower Semiconductor Ltd.在 2008 年收购了 Jazz Semiconductor Inc.，GlobalFoundries 在 2009 年收购了新加坡特许半导体，一跃成为第二大晶圆代工企业，日本主要的芯片制造商 NEC 和 Renesas 也在 2010 年宣布合并。这一并购浪潮同样影响了中国，2010 年，华虹 NEC 与宏力半导体合并，这将极大地改变中国的晶圆代工产业。但是，我国目前的半导体产业规模仅能覆盖 20%左右的国内市场份额，因此，提升我国半导体芯片制造产业规模是亟待解决的一个重要问题。
3. 市场需求多元化：现代社会对电子数字化产品的依赖和需求程度无论从数量和质量、还是在花样品种方面都将保持持续快速上升态势。技术上的革新使得半导体产品的生命周期越来越短，而激烈的市场竞争使得半导体芯片的单位利润越来越低，企业只有迎合市场多元化的动态需求，在最短的时间内生产出大量的满足不同顾客要求的高质量的新产品并快速抢占市场才能赚取最大利润。因此，有效集成企业内外资源，以少品种大批量生产的成本满足制造多品种大批量产品的需求，也是半导体制造企业所面临的重大挑战。

1.1.3 问题的提出

1.1.3.1 晶圆制造系统的特征

整个半导体芯片制造过程可以分为晶圆制备、晶圆制造、晶圆拣选测试、芯片装配与封装、以及芯片终测共五个主要阶段。其中，晶圆制造与晶圆拣选测试被合并称为“前端操作”（Front-end Operations），而装配封装和最终性能测试被总称为“后端操作”（Back-end Operations）。半导体晶圆制造系统（Semiconductor Wafer Fabrication System, SWFS）是将几十个甚至上百个特定的集成电路芯片制作在以半导体材料硅为衬底的晶圆片上的复杂过程。它所涉及的运作风险主要来自繁复的多重入型加工过程、大量贵重的加工设备、以及为数众多的价值快速增长的晶圆在制品。SWFS 的生产管理与控制问题非常复杂，归纳起来，其复杂性主要表现在以下几个方面：

1. 规模庞大，设备繁多，大批量生产，在制品数量巨大

以一座月产能为 3.5 万片的 8 英寸（200mm）晶圆制造厂为例，该系统共包含了 300～400 余台各类晶圆加工设备，及若干条大型物料传送系统和大量辅助生产设备[2]，总造价约为 100 亿人民币。由于半导体晶圆制造的复杂性、时效性和投资巨大性，迫使业界将不同的制造阶段根据市场和资源优化的需要配置在跨国或跨地区的不同区域，造就了一大批颇具实力的半导体晶圆代工厂（Semiconductor Wafer Foundry FAB）。这些晶圆代工厂直接面对来自全球范围内的客户定制产品（ASIC）市场，面向订单组织生产(Make-to-Order, MTO)。产品品种变化大，产品设计寿命短甚至是一次性设计制造，且批量巨大，是典型的大批量、超多品种混合生产模式。此外，在一座月产能为 3.5 万片的晶圆制造厂内，晶圆在制品多达 7 万片，约合 3000 个产品批次。因此，合理控制在制品水平，是在充分利用加工设备的前提下降低在制品库存成本和成品库存成本的重要手段之一，也是减少杂尘玷污、保证产品质量的有效方法。

2. 典型的高度重入型复杂制造过程

晶圆制造是要在以半导体材料硅或砷化镓为衬底的圆片上制造出几十甚至上百个特定的集成电路芯片，需要经过数百道工序逐层加工形成，制造过程精密复杂，工序繁多，周期漫长。根据集成电路逐层印制的工艺要求，整个晶圆制造过程由一定数量的重复的工艺阶段（Stages）构成，而每一阶段又包括若干个加工工序（Steps），不同阶段的工序类型相同，但具体的加工工艺参数不同。晶圆厂根据成组技术原理，把不同阶段但类型相同的工序安排到同一个设备群组（由一台或多台功能相同或相似的设备组成）来进行

加工。因此，晶圆在整个制造过程中要多次访问相同的设备（群组），这种产品在制造过程中多次访问同一设备群组甚至相同设备的特征称为重入型（Re-entrant）制造过程[3, 4]。

3. 机器加工特性复杂，多种机器混合加工

半导体生产线上不同的加工机器具有不同的加工方式，一般将其分为单件加工机器和批次加工机器。另外这些机器特性复杂，操作方式和运行模式各不相同。而且，各种精密设备还容易出现无法预测的故障并导致停机，且修复时间难以确定。处于同一阶段的不同产品以及处在不同阶段的同一产品在同一台机器加工，其工艺参数也不尽相同，这又形成了复杂的依赖于加工顺序的机器设置问题，更增加了调度的复杂性。晶圆制造系统的调度问题属于大规模复杂动态调度问题。在半导体晶圆生产线上，流动着不同工艺流程的几十种甚至上百种产品，这些工艺流程存着着多重入现象，而且执行这些流程的机器具有类型繁多、特性复杂，高度动态不确定等特征，使得生产线上的在制品对机器设备使用权的竞争非常激烈。

1.1.3.2 晶圆制造系统的调度现状

面对来自晶圆制造过程的大规模复杂性、多重入性、混合加工等因素，建立既能精确描述制造系统特性，又易于求解的调度模型非常困难，即使经过简化的模型其计算复杂性也是 NP-hard 的。受到实际需求的激励，人们对晶圆制造系统的调度与控制这一富有挑战性的问题进行了大量的研究，但是尚未形成一套行之有效的方法和理论，以实现在合理的时间里得到系统的最优解。

在晶圆制造系统的调度领域，方法的实用性和解的最优性似乎是一对很难调和的矛盾。目前被广泛使用的方法主要有三大类：数学规划方法、启发式规则和元启发式算法，这些算法具有不同的特点，它们在各种条件下的性能也各不相同，对它们的比较分析如表 1-3 所示：

表 1-3 晶圆制造系统中的主要调度方法

调度方法	数学规划方法	启发式规则	元启发式算法
描述	即运筹学方法，它将生产调度问题简化为数学规划模型，采用整数规划、动态规划以及决策分析等方法来解决调度最优化或近似优化问题	是一类基于直观或经验构造的算法，在可接受的花费（计算时间、占用空间）下给出待解决组合优化问题每一个实例的一个可行解，一般依靠与任务无关信息来简化搜索过程	指一类通用型的启发式算法，其优化机理不过分依赖待解问题的结构信息，可以应用到众多类别的组合优化或函数优化中。应用最多的元启发式算法包括人工神经网络、模拟退火、遗传算法和蚁群算法等
优点	任务分配和排序的全局性比较好，所有的选择同时进行，因此可以保证求解凸和非凸问题的全局优化	利用了面向特定问题的知识和经验，因而可以产生好的解决方案，求解时间也相对较短	具有自学习能力，容错性好，可以突破局域搜索的限制，获得全局性的优化解。具有分布式存储与计算能力，可以加快收敛速度，如果选择参数得当，会在很快的时间内收敛
缺点	需要对调度问题进行统一的建模，任何参数的变化会使得算法的重用性很差，对于复杂多变的生产调度来说，单一的数学规划模型不能覆盖所有的因素，存在求解空间大和计算困难等问题	算法精确度不高，而且不一定保证解的可行性，容易陷入局部最优解。此外用来评估解决方案的质量手段还较少，所获得的可行解与最优解的偏离程度也难以衡量	理论基础目前较为薄弱，其计算复杂性一般是通过实验获得的，理论上的分析不多见。参数选择较困难
在 SWFS 中的应用	由于晶圆制造系统问题复杂，因而只应用于小规模的调度问题	由于其算法简单，易于理解，目前是晶圆制造系统中应用最为广泛的调度算法	在晶圆制造系统调度中的应用刚起步，目前大多限于理论研究。但随着晶圆制造系统自动化程度及对调度精度要求的提高，元启发式算法将是未来晶圆制造系统调度的主流方法

数学规划方法在理论上能够求出问题的最优解，它们的有效性当然是最高的，适应范围也比较广，几乎能适应任何一种调度问题，但是计算复杂性很高（为指数函数），这使得它们的计算规模容易受问题规模的限制，即使用来验证其他算法时，也只能在小规模问题上验证，因而难以应用于实际问题。启发式调度规则的特点是算法简单，计算

复杂度低，精确性不高，但是其计算时间短，简单直观，易于理解，易于应用。通过对基本调度规则进行适当的组合和变形，可以提高调度效果。到目前为止，调度规则仍然是晶圆制造系统中应用最为广泛的调度算法。元启发式算法的特点介于数学规划方法和调度规则之间。相比调度规则，其问题求解精度较高，而所需计算时间相对数学规划方法要短的多，可用来求解较大规模的问题。元启发式算法是目前研究的热点，应用较多的的元启发式算法包括人工神经网络、模拟退火、遗传算法和蚁群算法。相比前三种算法，蚁群算法为群体智能算法，其特有的并发多线程搜索机制可使得算法较快收敛，并获得更好的调度结果。

随着晶圆制造系统向着更为复杂、对自动化要求更高的趋势发展，对生产调度算法的自学习、自适应要求更高。因而，可以预见元启发式算法将是未来晶圆制造系统调度的主流方法。但是目前元启发式算法还未能很好地应用于晶圆制造系统调度的实际工作，这主要是因为其不能很好地处理晶圆制造系统的特点。基于这个原因，本文对传统蚁群算法进行改造，并结合问题分解方法，试图提出一个基于蚁群算法和问题分解方法，并应对晶圆制造系统大规模、多重入、混合型制造特点的调度框架。

1.1.3.3 本文拟解决的关键问题

晶圆制造系统的特征决定了其是一个规模庞大的、制造资源和加工产品之间具有高度复杂逻辑关系的先进制造系统，同时，上述复杂特性也使得当前针对该系统的生产管理与先进控制方法的研究相对于晶圆加工技术设备发展水平的滞后。目前晶圆制造系统的调度方法还是以调度规则为主，辅以人工判断选择合适的调度规则，因而迫切需要引入先进有效的算法，以匹配晶圆制造过程中的高自动化现状。蚁群算法由于可在较短的时间内获得较优解的特性，且其具有自组织、分布式、协作性、鲁棒性与实现简单方便的特点，可以在没有全局信息的情况下，寻找复杂问题的解决方案，因而近年来受到了很多学者的关注与研究。但是传统的蚁群算法无法处理晶圆制造系统的大规模、多重入、混合生产等特征，为此，本论文提出了基于问题分解与蚁群算法的半导体晶圆制造系统调度方法的研究这一科学问题，以此入手，研究和探索适用于晶圆制造系统的科学生产管理方法。这一科学问题主要研究两部分内容，其一是面向半导体晶圆制造系统特征的问题分解方法的研究，其二是基于问题分解方法与半导体晶圆制造系统特征的改进蚁群算法研究（图 1-3），分述如下：

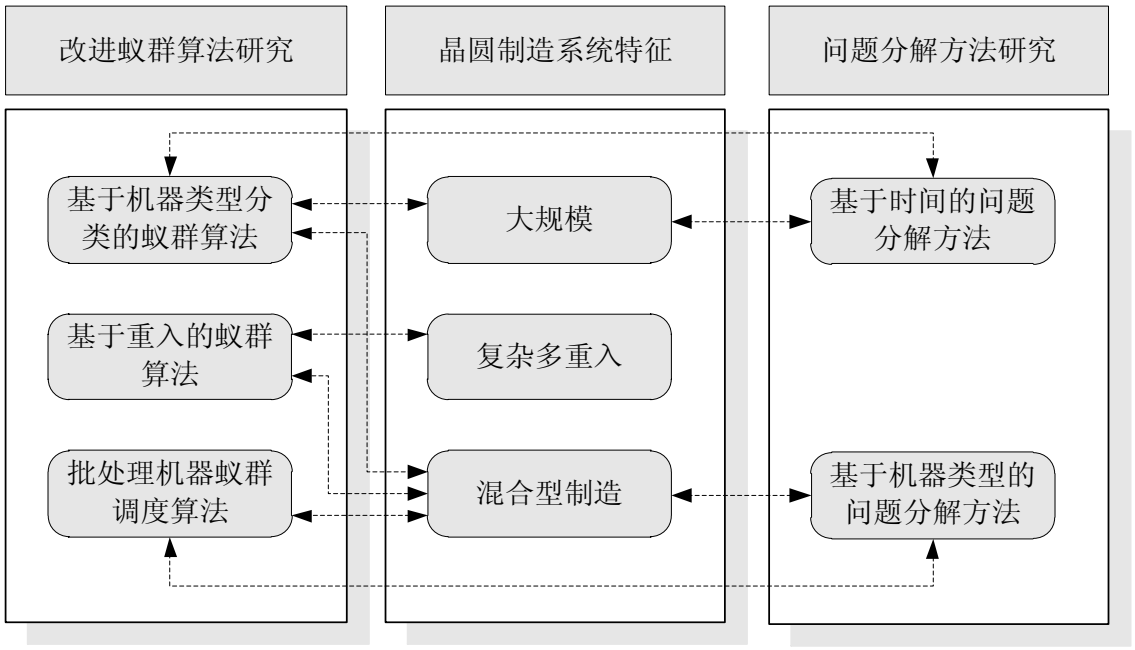


图 1-3 基于问题分解与蚁群算法的半导体晶圆制造系统调度方法的研究

Fig. 1-3 The research on scheduling wafer fabrication system with decomposition method and ant colony optimization algorithm

（1）面向半导体晶圆制造系统特征的问题分解方法的研究

大规模调度问题的关键是对问题进行分解。问题分解不仅降低了大规模调度问题求解的复杂度，而且也提高了优化调度的精度及运算速度。目前主要有基于机器、工件和时间三个决定调度问题规模的分解方法。但是半导体芯片制造工艺复杂、产品众多、混合型生产、生产周期长，而且有明显的重入性特征，使得调度问题更为复杂。如何根据晶圆制造系统的特点选择合适的问题分解方法，是本文研究的重点。

此外，子问题之间还需要进行协调，以使得子问题的解可以合成为原问题的解。按照子问题之间进行协调时的通信方式，协调机制可分为两类：隐式协调（Implicit Coordination）和显式协调（Explicit Coordination）。隐式协调是指各个子问题可以依据自己的目标和知识，遵照多子问题系统的自然或社会规则、标准和惯例来选择适当的行为，在选择时根据自身或者是附近部分子问题的求解情况进行优化，使各子问题的行为就象事先经过通信协调一样，此时子问题是根据局部的非全局信息做出决策的。隐式协调在子问题的智能算法的求解过程中进行。而显式协调是指子问题之间具有明显的协调机制，用于对子问题潜在的交互进行推理，必要时可以借助全局控制器来交换各自的目标，通过协商进行冲突消解，直至子问题的目标一致或不能达成协议（协调失败）。当

智能算法求解得到子问题的局部解后，可以采用显式协调的方法进行全局优化。根据晶圆制造系统的特点，以及所采取的问题分解方法的特点，选择与开发合适的子问题协调机制也是本论文研究的重点。

（2）基于问题分解方法与半导体晶圆制造系统特征的改进蚁群算法研究

目前，大多数晶圆厂仍在沿用简单的诸如 CR+FIFO（临界比＋先到先加工）或 EDD（最早交货期）等调度规则，生产管理水平有待提高；而现有算法多是针对晶圆制造过程中的局部小系统、且多是在诸多假设和约束条件下完成的研究成果，因而迫切需要先进的调度算法。蚁群算法曾经解决过很多不同类型的调度问题，在其中的某些问题，如单机器总权重延迟问题、开放车间问题和资源约束项目调度问题上，蚁群算法是性能最好的求解算法之一。但是当蚁群算法应用于晶圆制造系统的时候，却遇到很大的困难，这种困难在单类设备调度问题到系统调度问题中都有所体现。在单类设备调度问题中，由于晶圆制造系统的高度多重入特征，使得不仅要考虑不同类型工件的调度问题，还要考虑同一类型但属于不同加工阶段的工件调度问题，这增加了问题的复杂度。为此需要研究相应的蚁群算法以应对多重入特征。另一方面，批处理设备由于加工时间长而成为制造过程中的瓶颈设备，对批处理设备的调度非常重要，目前已有一些应用蚁群算法求解批处理设备的研究，但基本集中在批次调度过程，对更重要的组批过程涉及较少。因而对批处理设备加工过程进行深入研究，并对蚁群算法进行改进，使其能够用来处理批处理设备的组批过程，不仅能够获得质量更优的调度方案，而且能够扩大蚁群算法的应用范围。在系统调度问题中，重点要解决晶圆制造系统混合调度所带来的调度困难，因而需要对传统蚁群算法进行改造，使其既能用于求解单个机器的调度问题，又能在同一架构下获得混合制造系统的调度解。

1.1.4 课题研究的意义

作为半导体芯片制造过程中最为复杂和昂贵的关键环节，晶圆制造（Wafer Fabrication）企业所面临的挑战极为严峻。面对前所未有的来自技术和市场两方面的压力，半导体晶圆制造企业只有一方面密切跟踪世界科技前沿，适时引进先进的制造装备，提升企业的硬件水平和制造能力；另一方面要紧密结合自身特点，不断改善企业的生产管理和控制方法，降低生产成本、提高生产效率、满足不同客户的个性化需求，才能在激烈的国际市场竞争中占得先机、赢得挑战，才能求得企业的生存和长远发展。广大晶圆制造厂商虽然已经意识到先进的生产管理对于提高企业效益和竞争力的重要性，但在过去的十余年间，他们还是主要通过技术装备上的革新来提高企业的制造能力和绩效水

平。归纳起来，主要包括以下三个方面的努力和探索[5]：

为了提高系统的绩效水平，晶圆制造企业主要采取了如下的一些措施以降低成本，减少损耗，增加利润：

1. 引进和开发高度可控的生产设施，如群集制造设备（Cluster Tools），提高晶圆制造系统的自动化程度，在提高设备产能的同时减少晶圆在制品消耗在非生产环节上的时间，尽可能地避免由于晶圆表面灰尘污染而造成的损失，从而提高产品良率；
2. 应用生产执行系统（Manufacturing Execution System, MES）等生产管理信息系统（例如：HP 公司的 Promis 系统和 Applied Material 公司的 FAB300 系统），实现生产过程的动态监控，使生产管理者和设备操作人员能够实时掌握现场状态，以便及时做出调整和部署；
3. 减小单位芯片设计尺寸，增加晶圆面积（1965 年：50mm → 1975 年：100mm → 1981 年：125mm → 1987 年：150mm → 1992 年：200mm → 2000 年：300mm→2015 年：450mm），即通过不断增加每片晶圆上所能制作的单位 IC 数量，降低制造成本并提高整个制造系统的产量。

通过上述措施的实行，晶圆制造系统的自动化程度提高，晶圆尺寸增大，产品成本下降。但与此同时，相应的调度工作变得更加困难。中国芯片制造的设备及工艺技术水平已与国际先进水平接近。但是我国绝大多数芯片制造企业甚至包括一些著名的大型企业，对于晶圆制造最为核心、最为复杂但最重要的调度控制技术还十分落后，绝大多数企业在工厂的有效集成方面、生产过程的优化控制方面、以及对未来生产的准确预测方面仍然停留在“借助现有 IT 系统、依靠管理经验为主”的半人工计算水平，仍然存在不足和很大的提升空间。因此，开展面向 SWFS 的调度方法的研究意义重大，其成果也具有广阔的应用前景。以一座每月最大产量为 30,000 片晶圆，制程为 0.15 微米、7 层金属双嵌刻铜逻辑组件的 12 英寸晶圆厂为例[6]：

1. 制程设备的采购和安装架设需要 102 亿元人民币（16 亿美元），如果分 5 年折旧每天按 24 小时连续运行计算，则每小时的固定资产成本达到 23 万元人民币，如果将其中每台设备利用率都提高 1%，则每年可节约设备成本高达 2,000 万元人民币；
2. 一片工艺为 0.15μm 的 12 英寸晶圆，其出厂平均价值约为 1.6 万元（2500 美元），则月产 3 万片的总产值将达到 4.8 亿元，因此提高制造系统月产能的 1%即可获得每月 480 万元的直接收益；
3. 对于一座每月投产 30,000 片 12 英寸晶圆的晶圆厂，其每年的生产成本约 43 亿

元人民币（6.78 亿美元），这不包括制程设备或其他设施的运转成本，主要是晶圆的投入成本，因此合理控制晶圆在制品 WIP 水平，缩短制造周期所能产生的效益将以千万元计。

先进的调度控制技术可以极大地缩短芯片加工周期，提高制造能力，降低制造成本。例如，美国加州大学 Berkeley 分校工业工程系 Leachman 教授等于 1996 至 2000 年间为韩国三星电子实施了 SLIM (Short cycle time and Low inventory In Manufacturing)项目，旨在实证研究在保持晶圆制造生产率和成品率的情况下缩短制造程周期。该项目在不增加任何硬件投入的情况下，在 DRAM 市场价格 5 年内大幅降低的情况下，获得了 10 亿美元的收益，其 DRAM 产品在全球市场的份额也由 18%增加到 22%[7]。因此，改善晶圆制造系统的生产管理方法对于提高企业绩效和综合竞争力具有巨大的推动作用。

综上所述，本论文所进行的基于问题分解和蚁群优化的晶圆制造系统调度方法的研究，不仅可以在相关学术研究领域有所创新和贡献，更可以将研究成果直接应用到实际的晶圆制造系统中，改善企业的科学生产管理和控制水平，提高我国半导体制造企业的生产绩效和综合竞争力，推进我国以半导体制造为龙头的现代高科技制造业的健康发展。

1.2 论文主要内容及贡献

元启发式调度方法由于可在较短的时间内获得较优解的特性，近年来受到了很多学者的关注与研究，但是晶圆制造系统的大规模、多重入、混合生产等特征阻碍了元启发算法在其中的应用。针对这个问题，本文从半导体晶圆制造企业的实际应用出发，提出基于问题分解的蚁群算法，力图探索出一套能够有效解决晶圆制造系统复杂生产控制问题的科学方法，以期能提高我国半导体晶圆制造企业的生产控制水平。

本文针对晶圆制造系统调度困难的现状，提出基于问题分解的分类蚁群算法，提出统一的调度框架，并对批处理设备的蚁群调度算法做了更深一步研究，其具体研究内容与贡献如下：

1. 提出适用于晶圆制造特征的大规模问题分解方法

元启发算法可解决较大规模的调度问题，但当问题规模急剧扩大，类似晶圆制造这类调度问题时，元启发算法的应用会受到制约。为此，应用问题分解方法将大规模问题分解为小规模的子问题，再用蚁群算法分别求解，就成为一条可行的解决途径。本课题试图在现有问题分解方法的基础上，提出适合晶圆制造系统特征和蚁群算法应用的问题分解方法，并结合蚁群算法，以有效处理大规模晶圆制造系统调度问题。

针对晶圆制造系统的典型特征，采用不同的问题分解方法。对于晶圆制造系统大规模调度的特征，按照时间进行分解，在一个有限的时间区间（一天，一个班次等）内，所涉及的机器数量与工件数量相对可控，从而将大规模的调度问题分解为若干较小规模的子问题。在这些较小规模的子问题中，仍然存在着混合加工的问题，即单件加工机器（Unit-processing machine）与批处理加工机器（Batch-processing machine）存在于同一个调度问题中，针对这一特征，采用基于机器类型的问题分解方法，将子问题分解为以批处理机为核心，单件加工机器为上下游机器的更小的调度问题。

问题分解方法将大问题分解为若干小问题，使其更易求解，但是其所付出的代价是需要使用协调机制将小问题的解整合为原问题的解。在本文采用了三种协调机制来整合子问题的解。首先，应用于协调基于时间分解的子问题的是一类显式协调机制，即基于重叠区间的协调机制。该机制在相邻子问题间设立重叠区间，共有一部分操作，从而使得二者所获得的解可以保持连贯性。其次，基于机器类型的分解方法，采用一类隐式协调机制，即根据上下游的信息来调整调度算法。此外，还有另一类隐式协调机制应用于重入与分类蚁群算法，即利用蚁群算法本身的信息素更新机制进行协调。

2. 改造传统蚁群算法，使其适应问题分解与晶圆制造特征

为使得蚁群算法可以应用于晶圆制造系统调度问题，需要对其改造，其研究内容如下：

（1）基于问题分解的蚁群算法研究

元启发算法可解决较大规模的调度问题，但当问题规模急剧扩大，类似晶圆制造这类调度问题时，元启发算法的应用会受到制约。为此，应用问题分解方法将大规模问题分解为小规模的子问题，再用蚁群算法分别求解，就成为一条可行的解决途径。本论文试图在问题分解的基础上，通过对蚁群算法更新机制的研究，使其既可以用以求解各类子问题，又可以整合为原问题的解，扩大了蚁群算法的应用范围。

（2）基于重入的蚁群算法研究

为应对晶圆制造系统多重入的特征，开发基于重入的蚁群算法，将重入信息整合进信息素更新过程中，从而使得算法能自动识别工件的不同状态，并结合调度目标采取相应的调度策略。

（3）批处理设备蚁群调度算法研究

晶圆加工设备种类多样，但是可分为两大类：单件加工设备与批处理加工设备，将问题分解后需要分别对这两类设备进行调度。应用蚁群算法调度单件加工设备问题较为成熟，但调度批处理加工的研究刚刚起步。批处理调度可分为两个步骤：组批与批次分配，目前蚁群算法多应用于后一步骤，而对组批方法却涉及不多，这限制了蚁群算法应

用的范围和调度结果的质量。为此，本论文重点研究蚁群算法组批调度方法，并结合批次分配算法，提出完整的批处理设备调度蚁群算法。

（4）混合生产类型的蚁群调度算法研究

蚁群算法在生产调度中获得了较好的应用，但大多局限于单一设备类型的调度问题，因而难以应用于实际晶圆制造系统中。为此，本课题研究混合生产类型的蚁群算法统一框架，提出分类蚁群算法，可将蚁群算法成功应用于晶圆制造系统。

本博士学位论文的研究工作得到了以下科研基金和合作项目的资助和支持，在此表示衷心的感谢。

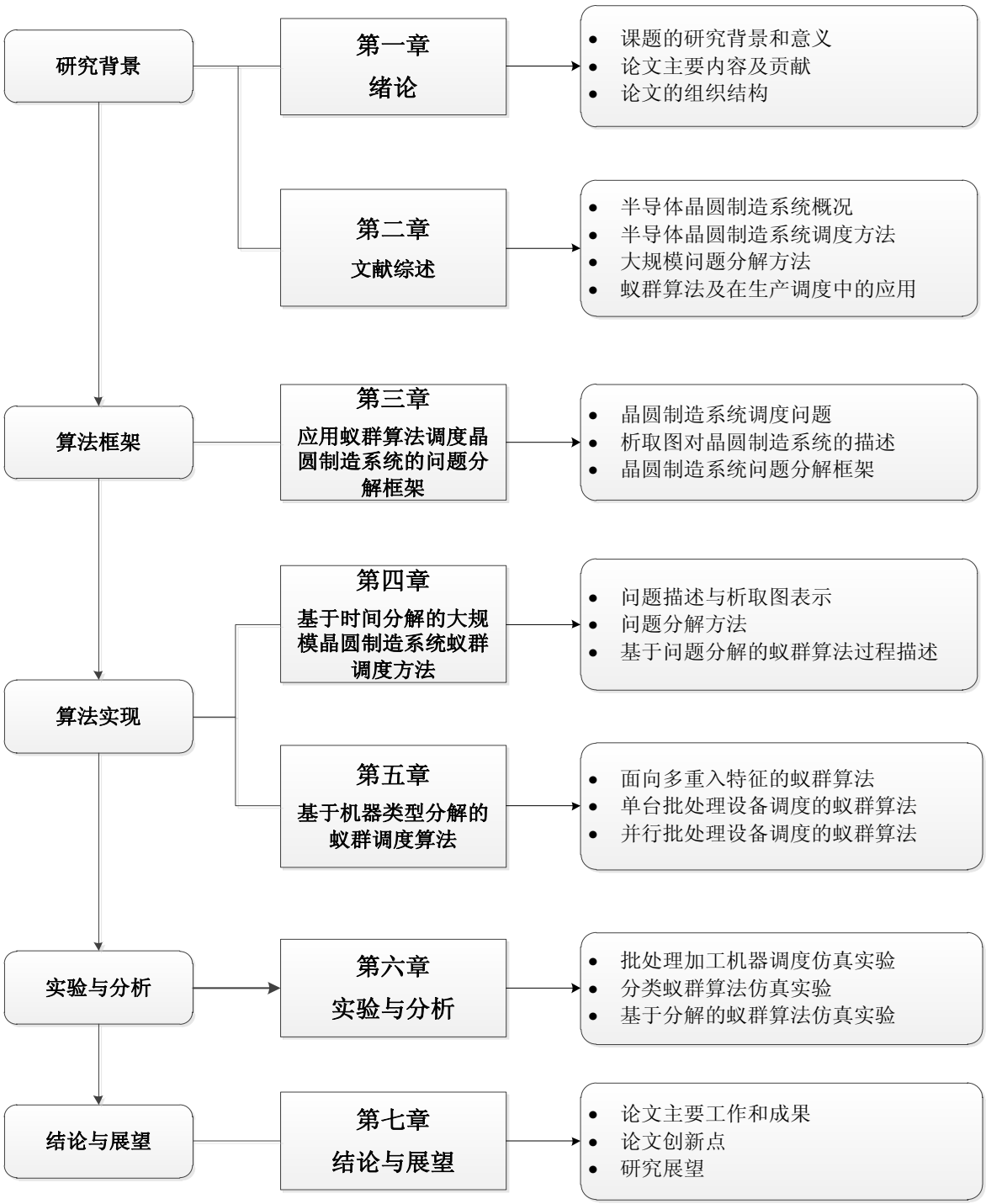
(1) 国家自然科学基金项目（No. 50475027）——大规模复杂重入型制造系统的建模方法的研究，2005.1～2007.12

(2) 国家高技术研究发展计划项目（863 项目）（No. 2006AA04Z128）——半导体芯片制造的大规模自适应优化调度方法及关键技术研究，2006.12～2008.12

(3) 高等学校博士学科点专项科研基金资助课题（No. 20040248052）——重入型复杂制造系统时变多目标生产控制方法的研究，2004.9～2006.9

(4) 国家科技重大专项课题（No. 2011ZX02501-005）——国产设备和材料考核评价，2011.1～2013.12

1.3 论文的组织结构


研究背景
第一章
绪论
课题的研究背景和意义
论文主要内容及贡献
论文的组织结构
第二章
文献综述
半导体晶圆制造系统概况
半导体晶圆制造系统调度方法
大规模问题分解方法
蚁群算法及在生产调度中的应用
算法框架
第三章
应用蚁群算法调度晶圆制造系统的问题分解框架
晶圆制造系统调度问题
析取图对晶圆制造系统的描述
晶圆制造系统问题分解框架
算法实现
第四章
基于时间分解的大规模晶圆制造系统蚁群调度方法
问题描述与析取图表示
问题分解方法
基于问题分解的蚁群算法过程描述
第五章
基于机器类型分解的蚁群调度算法
面向多重入特征的蚁群算法
单台批处理设备调度的蚁群算法
并行批处理设备调度的蚁群算法
实验与分析
第六章
实验与分析
批处理加工机器调度仿真实验
分类蚁群算法仿真实验
基于分解的蚁群算法仿真实验
结论与展望
第七章
结论与展望
论文主要工作和成果
论文创新点
研究展望

第二章 文献综述

2.1 引言

半导体晶圆制造系统是一个具有大规模资源构成、反复重入型制造、设备类型多样和多品种大批量混合生产等复杂特征的先进制造系统。由于半导体晶圆制造系统本身所具有的多种复杂特性，该系统已经成为目前世界上最为复杂的制造系统之一。本章首先介绍半导体晶圆的制造过程与其加工设备特征，然后总结自上世纪 60 年代以来针对晶圆制造系统调度的研究，根据所使用的方法分为数学规划方法、启发式规则、人工智能与元启发式算法四类分别介绍。

2.2 半导体晶圆制造过程简介

整个半导体芯片制造过程可以分为晶圆制备、晶圆制造、晶圆拣选测试、芯片装配与封装、以及芯片终测共五个主要阶段[8, 9]。其中，晶圆制造与晶圆拣选测试被合并称为“前端操作”（Front-end Operations），而装配封装和最终性能测试被总称为“后端操作”（Back-end Operations）。

晶圆制造的过程如同盖房子，一层一层地往上盖，层与层之间由光刻制程进行区分，每一层包括了薄膜、扩散、离子植入、微影、蚀刻、化学机械研磨等多道不同制程组合的循环。制程步骤的数量，随着产品的不同而有所差异，但是一般来说制作一个产品至少需要上百道制程。

如图 2-1 所示，半导体晶圆制造的基本工艺步骤是：晶圆在首先经过清洗后，送到热炉管，在含氧的环境中，以加热氧化的方式在晶圆表面形成一层二氧化硅（SiO_2），紧接着采用化学气相沉积或物理气相沉积的方式在二氧化硅表面形成一层薄膜。然后整个晶圆将进行曝光显影制程，即先在晶圆表面涂上一层光阻，再将光罩上的图案曝光光刻到光阻上面。接着利用蚀刻技术，将部分未被光阻保护的氧化硅层除去，留下的就是所需要的集成电路图。随后，以磷为离子源对整个晶圆进行离子植入，并将光阻剂去除。最后接着进行电镀制程，制作金属导线，以便将各个晶体管与组件加以连接。这样一个层次（Layer）的集成电路便制造完成。此时必须进行一些电性或物理特性量测，以检验加工结果是否在规格内，即第一道晶圆级电学在线测试（WET）。如此反复以上步骤，通过重入型加工过程完成集成电路的剩余各层的电路。

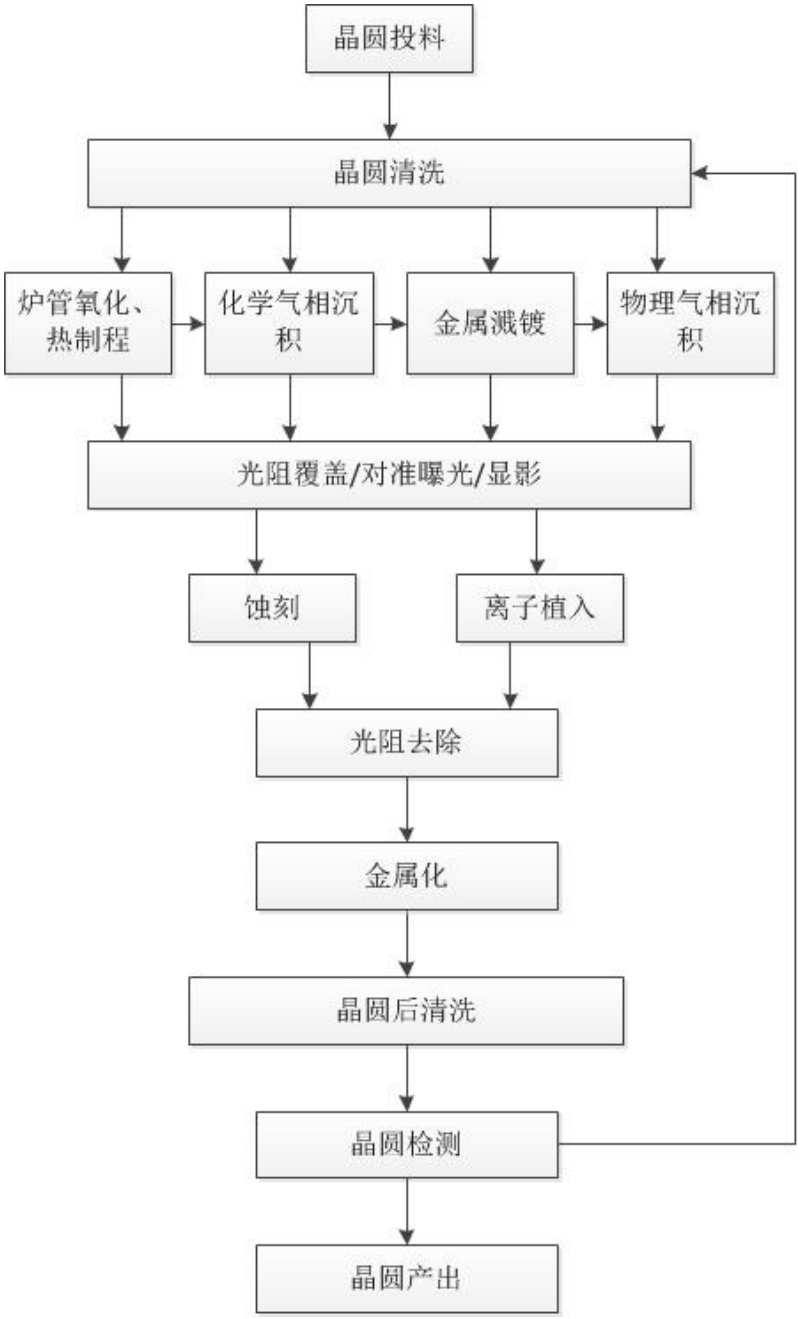


图 2-1 晶圆制造流程

Fig. 2-1 The flowchart of wafer fabrication

晶圆制造过程中的几个主要制程包括晶圆制备、沉积、光刻、蚀刻、离子植入、金属化与切割封装，如图 2-2 所示：

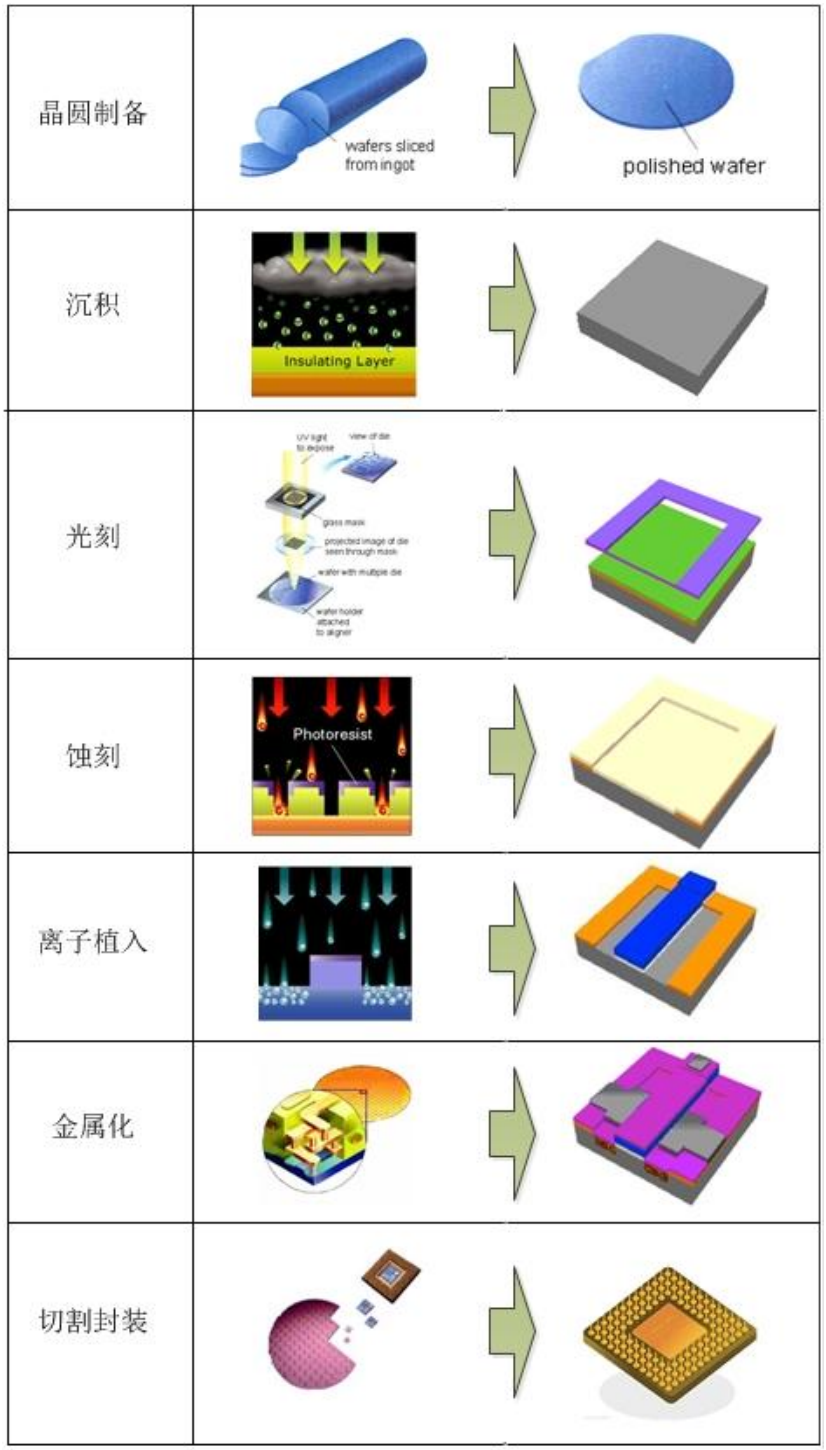


图 2-2 晶圆制造的主要步骤

Fig. 2-2 The main stages of wafer fabrication

（1）晶圆制备（Wafer Preparation）

硅是用来制造集成电路芯片的主要半导体材料，也是半导体产业中最重要的材料。用来做芯片的高纯硅被称为半导体级硅（Semiconductor-Grade Silicon, SGS），有时也被称作电子级硅。晶圆制备的基础是将半导体级硅的多晶硅块转换成一块大的单晶硅，即硅锭。制备晶圆的时候，首先将单晶硅棒的头部和尾部切掉，再用机械对其进行修整至合适直径，之后得到的是一个有合适直径和一定长度的“硅棒”。由于硅很硬，需要用金刚石锯来把“硅棒”切成一片片薄薄的圆片，圆片每一处的厚度必须是近似相等的。将单晶硅棒切晶圆片后，需要对晶圆片进行研磨，目的是减少晶圆片正面和背面的锯痕和表面损伤，同时打薄晶圆片。研磨后需对晶圆片进行刻蚀和清洗，即使用氢氧化钠、乙酸和硝酸的混合物以减轻磨片过程中产生的损伤和裂纹。之后，需利用倒角工艺将晶圆片的边缘磨圆，彻底消除将来电路制作过程中破损的可能性。倒角之后需要对晶圆片的边缘进行抛光，以提高整体清洁度以进一步减少破损。这些步骤完成后就形成了一片片的晶圆。

（2）沉积（Deposition）

制作晶片的第一步是在晶圆上沉积一层不导电的二氧化硅薄膜。在晶片的后续制作过程中，二氧化硅层的成长、沉积会进行很多次。二氧化硅薄膜成形的技术主要可分为物理气相沉积（Physical Vapor Deposition, PVD）与化学气相沉积（Chemical Vapor Deposition, CVD）。物理气相沉积技术是对欲沉积薄膜的材料源施加热能或动能，使之分解为原子或原子的集合体，并结合或凝聚在晶圆表面形成薄膜。物理气相沉积技术分为电阻加热蒸镀法、电子枪蒸镀法与溅镀法三种。化学气相沉积则是将反应气体导入高温炉，用气态的化学原料在晶圆表面产生某种化学作用，并在晶圆表面沉积一层薄膜。

（3）光刻（Photolithography）

光刻技术是在一片平整的晶圆上构建半导体 MOS 管和电路的基础，这其中包含有很多步骤与流程。首先要在硅片上涂上一层耐腐蚀的光刻胶，随后让强光通过一块刻有电路图案的镂空掩模板照射在晶圆上。被照射到的部分（如源区和漏区）光刻胶会发生变质，而构筑栅区的地方不会被照射到，所以光刻胶会仍旧粘连在上面。接下来就是用腐蚀性液体清洗晶圆，变质的光刻胶会被除去，露出下面的晶圆，而栅区在光刻胶的保护下不会受到影响。

（4）蚀刻（Etching）

蚀刻是将材料使用化学反应或物理撞击作用而移除的技术。蚀刻技术可以分为“湿蚀刻”（wet etching）及“干蚀刻”（dry etching）两类。在湿蚀刻中是使用化学溶液，经由化学反应以达到蚀刻的目的，而干蚀刻通常是一种电浆蚀刻（plasma etching），电浆蚀刻中的蚀刻作用，可能是电浆中离子撞击芯片表面的物理作用，或者可能是电浆中

活性自由基（Radical）与芯片表面原子间的化学反应，甚至也可能是这两者的复合作用。蚀刻结束后利用另一种称为去胶机的等离子体装置，用离化的氧气将晶圆表面的光刻胶去除，紧接着用一种化学试剂彻底清洗晶圆。

（5）离子植入（Ion Implantation）

离子植入技术是将掺质以离子型态植入半导体组件的特定区域上，以获得精确的电子特性。这些离子必须先被加速至具有足够能量与速度，以穿透（植入）薄膜，到达预定的植入深度。离子植入制程可对植入区内的掺质浓度加以精密控制。基本上，该掺质浓度（剂量）是由离子束电流（离子束内的总离子数）与扫描率（晶圆通过离子束的次数）来控制的，而离子植入的深度则由离子束能量的大小来决定。

（6）电镀（Electroplating）

芯片的功能是靠其间各晶体管的连接而实现的。通过电镀工艺，使得导电金属（通常是铝）沉积在晶圆表面。使用光刻和蚀刻制程去除没有用的金属，留下连接晶体管间的电路金属。现在复杂的晶片都需要很多层绝缘体。一个正常运作的晶片需要连接数以百万计的传导线路，包括同一层上的水平连接和各层之间的垂直连接。

（7）切割封装与测试（Assembly & Testing）

在晶圆制造厂完成所有制程工艺后，通过电学测试的晶圆可以进行单个芯片的切割、装配与封装，即将每一个芯片组装到用于保护和粘附到更高级装配的管壳中。传统的装配与封装主要分为以下几个步骤：晶圆测试和拣选 → 分片 → 贴片 → 引线键合 → 塑料封装 → 最终封装与测试。对于所有的集成电路，芯片封装有四个重要功能，即：a.保护芯片以免由环境和传递引起损坏、b.为芯片的信号输入和输出提供互连、c.芯片的物理支撑、d.散热。

2.3 晶圆制造设备特征

半导体晶圆制造车间（FAB）被划分为若干个主要加工区域（Work area），如光刻区和蚀刻区等，每个区域负责完成一定数量的功能性工艺操作。每个工作区域又可细分为一定数量的工作站（Work Station），或称其为设备群组（Equipment Group）或隔间（Bay）。每个工作站内分布有一定数量的加工机器、在制品缓冲区、传送机构和相应的设备操作员。工作站内的晶圆加工机器具有基本相同的功能，或相互组合在一起能够实现某一工艺步骤。整个晶圆制造系统可以看作由晶圆厂（FAB）、加工区域（Work Area）、工作站（Work Station）和生产设备（Equipment）组成的四层体系结构。

晶圆在 FAB 中的运输和加工单位为 lot，一个 lot 约包含 25 片晶圆（wafer），通常

由统一的晶盒（Unified Pod）进行承载，以避免晶圆在生产环节转运过程中受到灰尘污染。因此，lot 在生产系统中可看作一个独立的工件，在本论文中，工件和 lot 代表相同的含义。

半导体晶圆制造系统中的生产设备大体可以分为三类，即晶圆加工机器（Processing Machine）、物料传送系统（Material Handling System, MHS）和辅助设备（Auxiliary Equipment）。其中，晶圆加工机器最为复杂，根据加工操作方式的不同可以将它们划分为五大类，即单件加工设备（Single-type Processing Machine, SPM），批处理加工设备（Batch-type Processing Machine, BPM），管道型加工设备（Piping-type Processing Machine, PPM），群集加工设备（包括多相同处理室和多不同处理室）（Multi-Chamber Processing Machine with Same Chambers / Different Chambers, MPM-SC / DC），和 WET-BENCH 型加工设备。这五种类型的加工设备又可分为单件加工设备与批处理加工设备两大类，其中 SPM，MPM 和 PPM 三种设备属于单件加工，BPM 和 WET-BENCH 两种设备属于批处理加工。五种加工设备具体介绍如下：

（1）单件加工设备

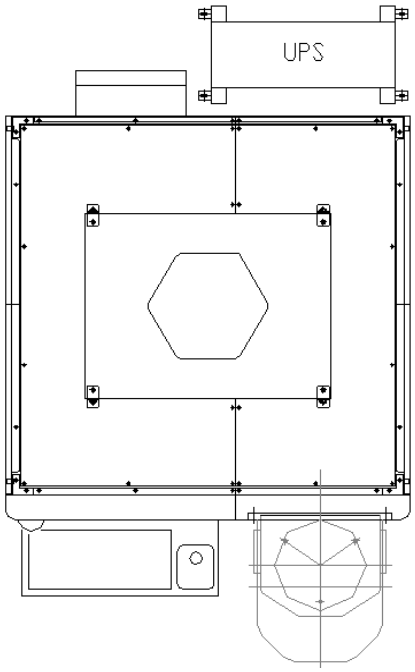


图 2-3 单件加工设备

Fig. 2-3 Single-type processing machine

单件加工设备一次只能载入一个 lot 的晶圆，逐片加工，待所有晶圆均完成处理后

再一起载出。

（2）批处理加工设备

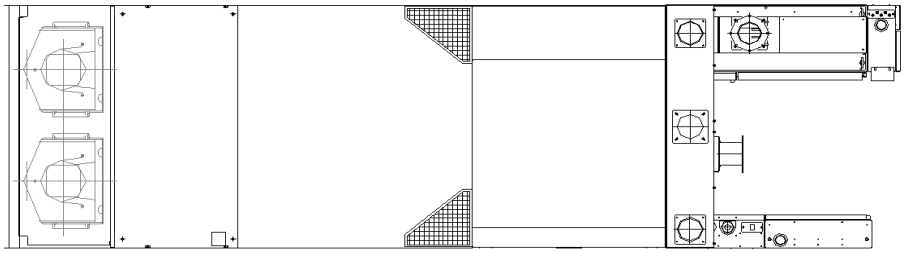


图 2-4 批处理加工设备

Fig. 2-4 Batch-type processing machine

批处理加工设备每次可载入多个 lot 的晶圆片，且载入的所有晶圆 lot 具有本工序相同的加工程序菜单（Processing Recipe），它们将同时进入机器且同时完成加工并载出。

（3）管道型加工设备

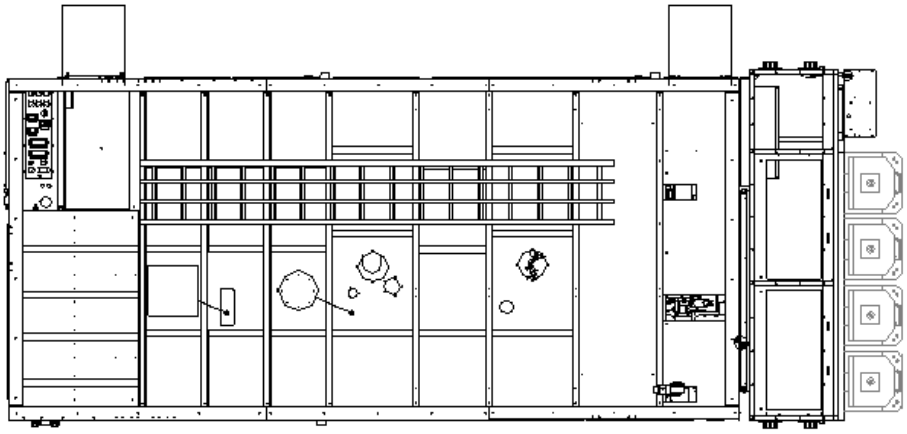


图 2-5 管道型加工设备

Fig. 2-5 Piping-type processing machine

管道型加工设备拥有若干个不同的处理室（槽），依次串行排列。该型机器一次仅可载入一个 lot 的晶圆，要依次进入全部处理室进行加工。当一个 lot 的晶圆结束在第一个处理室内的加工后，在其导出第一处理室并进入第二处理室的同时，一个新的晶圆 lot 将被载入刚刚空闲的第一处理室，如此呈管道型依次进行加工。

（4）群集加工设备

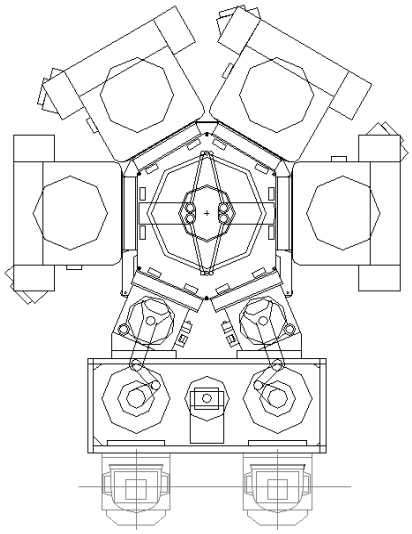


图 2-6 群集加工设备

Fig. 2-6 Multi-Chamber processing machine

群集加工设备又可细分为两种类型：一种是 MPM_SC 型，该型机器拥有若干个相同的晶圆处理室（with Same Chambers）；另一种为 MPM_DC 型，该型设备的多处理室功能各不相同（with Different Chambers）。两种 MPM 型设备一次均只可载入一个 lot 的晶圆，所有晶圆全部完成处理后共同载出。在 MPM_SC 型机器中，晶圆可以逐片的在任何一个空闲和可利用的处理室内进行加工；而在 MPM_DC 中，晶圆以 lot 为单位载入，要按照其加工程序菜单以某种顺序在某个或几个处理室内依次完成加工。

（5）WET-BENCH 型加工设备

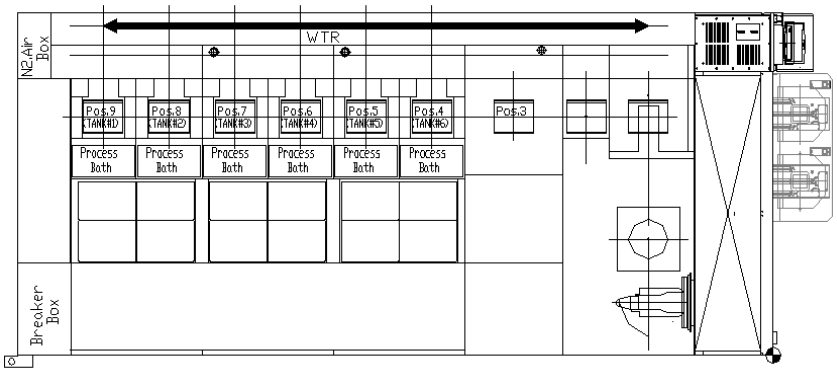


图 2-7 WET-BENCH 型加工设备

Fig. 2-7 WET-BENCH processing machine

2.4 晶圆制造系统调度方法研究综述

2.4.1 生产调度问题与调度算法分类

生产调度问题是一类在实际生产过程中决定工件加工顺序的调度问题。传统上人们主要从应用数学的角度来研究调度问题，调度问题通常被定义为“分配一组资源来执行一组任务”，也就是“排序(sequencing) ”。与之相对应，生产调度中的调度问题可以描述如下：“在某一时间期限内分配一组机器来执行生产订单任务”[10]。但是，针对当今先进制造模式，可以把生产调度定义得更详细一些：“生产调度是针对一项可分解的生产任务，探讨在尽可能满足约束条件(如交货期、工艺路线和资源情况等）的前提下，通过下达生产指令，安排其组成部分(操作）使用哪些资源、其加工时间及加工顺序，以获得生产任务执行时间或成本的最优化。[11]”。

生产调度问题的分类方法很多，主要有以下几种：

（1)根据加工系统的复杂度，生产调度可以分为单机调度、Job-shop 调度、Flow-shop 调度、Open-shop 调度、多机器并行加工（K-machine in parallel）调度等几个基本类型[12]。单机调度是指所有的操作任务都在一台机器上完成，需要对任务进行优化排队;Job-shop 调度是最一般的调度类型，它是指由 m 个不同的机器加工 n 个有特定加工路线(顺序)的工件，不同工件的工序间没有顺序约束，工序加工不能中断;Flow-shop 调度假设所有工件都在同样的设备上加工，并有一致的加工操作和加工顺序;多机器并行加工调度是指多台机器并行加工工件，而且并行加工的机器和工件都是类似的。实际的调度问题通常是上述几种调度类型的组合，晶圆制造系统则与传统的 Job-shop 调度或 Flow-shop 调度都不相同，被认为是第三种类型的复杂调度问题[3]。

（2）根据优化准则，可以分为基于代价和性能的调度两大类。代价包括为了实现调度方案所消耗的各种费用和所造成的损失，如运行费用、运输费用、存储费用和延期交货损失等。性能主要包括设备利用率、最大完成时间、拖延加工任务的百分比等。虽然在理论分析上，大部分只注意调度的性能，但在实际生产中，通常要综合考虑代价和性能两方面因素。

（3）根据生产环境的特点，可将调度分为确定性调度和随机性调度。前者是指加工时间和其他参数是已知的、确定的量;而后者的加工时间和有关参数是随机的变量。

（4）根据加工任务或被加工工件的特征，可将调度分为静态调度和动态调度。静态调度是指所有待安排加工的工件均处于待加工状态，因而进行一次调度后，各作业的加工被确定，在以后的加工过程中就不再改变;动态调度是指作业依次进入待加工状态，

各种作业不断进入系统接受加工，同时完成加工的作业又不断离开，还要考虑作业环境中不断出现的不可预测的动态扰动，如作业的加工超时、设备的损坏等。

晶圆制造系统调度问题是一类带有复杂特征的调度问题，在该领域的研究主要可以分为四类：启发式规则、数学规划方法、人工智能方法和元启发式算法（见表 2-1）。从研究的历史角度，学者们最开始关注的调度算法是启发式规则，接着采用数学规划方法提供调度的新的方向。最近研究的重点转向了人工智能方法和元启发算法，特别是后者，是晶圆制造系统调度的未来方向。

表 2-1 晶圆制造系统中的调度方法

序号	调度类型	调度技术
1	启发式调度规则	
2	数学规划方法	分支定界法
		拉格朗日松弛法
		过滤束搜索法
		排队论模型
3	人工智能方法	专家系统/基于知识的系统
		人工神经网络
		模糊逻辑
		基于 Petri 网的方法
4	元启发式算法	禁忌搜索法
		模拟退火法
		遗传算法
		蚁群算法

2.4.2 启发式调度规则

调度规则在晶圆制造系统中应用极广，主要用来为复杂系统提供实时的调度方案。在加工车间，每当一个机器空闲，则调度规则检视所有正在等待的工件，并在其中选择优先级最高的工件进行加工。由于晶圆制造系统的复杂性，研究初期多使用的是简单规则。如先进先出规则（FIFO）[13-15]；最短加工时间规则（SPT）[15, 16]；最早交货期规则（EDD）[17, 18]；最短剩余加工时间规则（SRPT）[13, 19]；到达瓶颈机器时间最长规则（LTNV）[13, 18]；到达瓶颈机器时间最短规则（STNV）[13, 18]等。

在简单调度规则的基础上，学者们研究了一系列复合型规则，即把多个调度规则组合使用，以弥补单个规则的缺点。如 Lee 等[15]提出了关键比例规则（CR），每个等待工件的 CR 值=（交货期-当前时间）/剩余加工时间，然后从所有工件中选择最紧迫的工件优先加工；Wein[13]在 FIFO 规则的基础上提出了 FIFO+调度规则，即对于当前机器组前的待加工工件来说，如果其中存在某些工件的下一加工工步所在的机器组的等待队列长度小于或等于 4 的情况下，这些工件将被优先挑选，并利用 FIFO 进行派工；如果不存在，则选择 FIFO 对所有工件进行派工；类似的也提出了 SRPT+调度规则[13]，即对于当前机器组前的待加工工件来说，如果其中存在某些工件的下一加工工步所在的机器组的等待队列长度小于或等于 4 的情况下，这些工件将被优先挑选，并利用 SRPT 进行派工；如果不存在，则选择 FIFO 对所有工件进行派工。Kumar. P. R[3]提出了基于缓冲区优先级策略，即缓冲区编号小的优先级高策略（FBFS），和缓冲区标号大的优先级高策略（LBFS）；从缩短制造周期方面考虑，给出了一类最小松弛策略，即波动光滑策略；从考虑产品切换时间出发提出了一类“清空”策略 CAFI 等。Lee 和 Tang[20]等学者还提出了一种降低波动和平衡生产策略 MIVS（Minimum Inventory Variability Scheduling），在不降低生产率的情况下，减少制造周期的均值和方差。

Uzsoy 等[21]在半导体测试生产线上对一系列与交货期和生产周期相关的调度规则进行了计算实验。这些被测试的调度规则包括 FIFO，SPT，SSPT，SRPT，Job-EDD，Operation-EDD，COVERT 和 ATCS，测试指标是平均生产周期、平均拖期、拖期工件数和最大拖期。他们的实验结果表明没有一个规则能满足所有的测试指标。Holthaus 和 Rajendran[22]提出了五个新的调度规则，是加工时间、下一工序等待工件数、工件到达时间和工件松弛期的组合。通过仿真实验，他们同样观察到任何一个单一规则可以优化所有的指标。

启发式规则易于理解、易于应用，并且所需计算时间较少，因而在晶圆制造系统中应用较广。其主要的缺点是对实际的动态调度问题其调度方案精度较差，容易陷入局部最优。

2.4.3 数学规划方法

数学规划方法将生产调度问题简化为数学规划模型，采用整数规划、动态规划以及决策分析等方法来解决调度最优化或近似优化问题，也称为优化调度方法。生产调度中广泛使用的是混合整数线性规划（mixed integer linear programming, MILP）和混合整数非线性规划（mixed integer non-linear programming, MINLP）方法。Dessouky[23]曾提出了

两种新型的整数规划公式来描述半导体生产线的特性：第一种模式是限定了开始加工时间；第二种模式是对加工时间相同的操作采用一种特定的时间网格来表示，在每一时间网格点上开始调度，与传统的整数规划方法相比，该方法更容易获得解析解。Kumar[24]等学者也曾利用线性规划的方法解决了一些问题，如给出了生产过程中保证生产稳定进行的 WIP 水平界限和开环带重入生产线最小延迟的界限，对实际生产起到一定的指导作用。但总体而言，由于调度问题的复杂度为 NP-hard，所以数学规划方法的应用受到了限制。为此，提出了若干问题分解的方法以降低问题的难度。

（1）分支定界法

两类较为常用的求解整数规划问题的方法是分支定界法和拉格朗日松弛法。Morton 和 Pentico[25]总结了分支定界方法：分支的主要想法是把问题概念化为决策树，每一个决策点对应了一部分解决方案。从每一个决策点出发，会生长出大量新的分支，每个分支对应于一个可能的决策。这个分支的过程持续进行直到不能继续分支的树叶节点。这些树叶节点即该问题的解决方案。

分支定界法有两个基本的步骤：计算优化问题调度方案的下界与分支的过程。前者指导着如何进行分支，后者将问题分解为两个穷尽但互不重叠的小问题，这些小问题可以进一步被分解。分支定界法的一个主要的缺点是缺乏足够好的下界，以保证在问题的初始阶段就除去一部分分支，简化运算。分支定界法在半导体晶圆实际调度过程中使用较少，Sung 和 Choung[26]曾使用分支定界法解决半导体生产工厂中的一类批处理设备—炉管的动态调度问题，其优化目标是最大化所有工件的完成时间（makespan）。

（2）拉格朗日松弛法

在拉格朗日松弛技术中，采用另一种方法简化整数规划问题，即忽略掉一些特定的整数值约束，代之以在目标函数上添加由于这种忽略所造成的成本。和分支定界法一样，对于大规模调度问题拉格朗日松弛法的计算量也非常大。拉格朗日松弛法的主要思想是将多工件调度问题简化为小的工件数量的问题，这通过使用拉格朗日乘子，松弛数量约束而实现。

在拉格朗日松弛法在晶圆制造系统的应用上，Chen 等[27]将集成电路测试车间建模为整数规划模型，其调度目标是使得工件的提前期与拖期均最小，其所考虑的约束包括资源约束、加工优先约束和加工时间需求。通过应用拉格朗日松弛法，松弛了资源约束和加工优先约束而获得调度方案。Chen 和 Hsia[28, 29]同样使用拉格朗日松弛法调度集成电路测试设施，并与启发式规则做对比。他们的结论是启发式规则寻找到可行解所需要的时间更少，但是拉格朗日松弛法可以在可接受的时间范围内获得好的调度解。Kaskavelies 和 Caramanis[30, 31]使用拉格朗日松弛算法对超过 10,000 个资源约束的半导体

测试工厂进行调度。他们介绍了两类拉格朗日乘子更新过程的新特征，以帮助运算的进行。Sun 等[32]使用拉格朗日松弛法处理带顺序依赖设置时间的单机调度问题，他们将顺序依赖设置时间表示为容量约束，并通过拉格朗日算子进行松弛。

（3）过滤束搜索

过滤束搜索是分支定界法的引申。这类方法试图通过一类智能的方法消除掉一些分支，使得不必检查所有的分支，降低了问题的计算量。因此相比分支定界法而言需要较少的计算时间，但也不能保证获得如分支定界法的优化方案。使用束搜索方法，只有最有希望的决策点被选择作为下一分支的节点，其余的决策点都被永久地放弃。

De 和 Lee[33]使用过滤束搜索方法，构建一个基于知识的调度系统，以调度半导体测试生产线。其调度目标是最小化总生产周期。在调度过程的搜索树上，所需要搜索的节点的数量取决于过滤宽度和束的宽度。

（4）排队论模型

对许多制造系统建模的一个方法是建构成多级排队网络。在这样的网络中，工件流沿着预先设定好的路径通过系统。

Connors 等[34]提出了一个新的方法，基于确定流网络模型，通过确定下一个工件的方法来调度半导体制造生产线，以产生动态调度方案。Neuts[35]通过对 Poisson 输入流系统的研究给出一般性的策略，即最小批量法：MBS（K, L）——对满负载为 K 的系统，当 lot 数大于或等于最小批量 L 时，可以开始加工。Medhi[36]将该策略扩展到 $M / M^{a,b} / 1$ 排队模型，其工件到达服从 Poisson 分布，机器加工时间服从指数分布。Hopp 等[37]提出了一类优化排队网络模型，以支持半导体制造设备的特性，包括批处理、重入流、多种产品类型等。Huang 等[38]针对炉管加工的特点提出了排队模型（$M^y / M^{x,y} / c$），考虑到了一个制造操作有多个加工菜单的情况。研究以 Poisson 流输入、指数型服务时间的调度系统，用数学推导的方法近似解出批加工的数量。并与仿真实验结果相比较，在加工菜单较多、中负载、达到速率几乎相同的情况下排队模型的近似解优于仿真结果。Li 等[39]针对晶圆批的到达时间和交货期限的可获性，以减少延期时间和延期晶圆批数为目标，从研究批调度问题的复杂性出发，提出了动态规划模型。Sung 等[40]也采用了动态规划的方法，在已知晶圆批到达的时间和所属类别的条件下，以所有晶圆批总完成时间最短为优化目标。

2.4.4 人工智能调度方法

（1）专家系统

专家系统是人工智能中最重要的也是最活跃的一个应用领域，它实现了人工智能从理论研究走向实际应用、从一般推理策略探讨转向运用专门知识的重大突破。20 世纪 60 年代初，出现了运用逻辑学和模拟心理活动的一些通用问题求解程序，它们可以证明定理和进行逻辑推理。但是这些通用方法无法解决大的实际问题，很难把实际问题改造成适合于计算机解决的形式，并且对于解题所需的巨大的搜索空间也难于处理。

De 和 Lee[33]提出了一类基于知识的调度系统，用来处理半导体测试设备的调度问题。这个系统有两个非常明显的组件：一个基于知识表达的框架和基于过滤束搜索的问题解决策略。Burdick[41]则研发一个面向半导体制造的专家系统：ALECS（ACS LPCVD Expert Control System）。Aytug 等[42]对调度过程中的机器学习文献进行了总结。

（2）人工神经网络

人工神经网络是一种应用类似于大脑神经突触联接的结构进行信息处理的数学模型。神经网络是一种运算模型，由大量的节点（或称神经元）和之间相互联接构成。每个节点代表一种特定的输出函数，称为激励函数（activation function）。每两个节点间的连接都代表一个对于通过该连接信号的加权值，称之为权重，这相当于人工神经网络的记忆。网络的输出则依网络的连接方式，权重值和激励函数的不同而不同。而网络自身通常都是对自然界某种算法或者函数的逼近，也可能是对一种逻辑策略的表达。

Huang 等[43]建立了一个人工神经网络模型以预测半导体制造车间的产品绩效；Liao 和 Wang[44]采用神经网络对于自动物料运输时间进行估计，实验证明该方法具有很好的正确性和有效性；Zhang 等[45]基于模糊神经网络提出了半导体生产线重调度优化方法。

（3）模糊逻辑

模糊逻辑是模仿人脑的不确定性概念判断、推理思维方式，对于模型未知或不能确定的描述系统，以及强非线性、大滞后的控制对象，应用模糊集合和模糊规则进行推理，表达过渡性界限或定性知识经验，模拟人脑方式，实行模糊综合判断，推理解决常规方法难于对付的规则型模糊信息问题。模糊逻辑善于表达界限不清晰的定性知识与经验，它借助于隶属度函数概念，区分模糊集合，处理模糊关系，模拟人脑实施规则型推理，解决因“排中律”的逻辑破缺产生的种种不确定问题 。

由于车间调度具有许多模糊特征，比如不确定的加工次数、不确定的约束数量以及不确定的加工时间等，因此模糊逻辑理论对于求解 job-shop 调度问题非常有效。Azzaro-Pantel[46]采用模糊方法对半导体晶圆制造系统中的批处理设备进行绩效建模，通过一个离散事件仿真模型对绩效进行评估。

（4）基于 Petri 网的方法

Petri 网最早是由德国的 Carl A. Petri 博士提出，用于描述计算机系统事件之间的因

果关系。早期 Petri 网主要用于计算机信息处理领域，后来具有工程背景的研究人员将 Petri 网用在工程系统尤其是自动制造系统的研究。Petri 网采用可视化的图形描述，但却可被形式化的数学方法所支持，可以用来准确表达离散事件动态系统的静态结构和动态变化。

Zhou[47]于 1994 年系统的提出了用基本 Petri 网为半导体制造系统进行建模、分析、仿真和调度控制的方法。作者系统的阐述了 Petri 网的定义、性质和主要的分析方法，针对半导体制造系统中的正常加工、设备周期维护、资源争夺、加工任务优先级及任务的重做等特征一一建立了 BPN 模型，并以 SWFS 中的黄光区为案例进行了详细研究。Xiong 和 Zhou[48]提出了两个基于 Petri 网络的混合启发式搜索策略，同时考虑 79 个资源和 30 个工件的约束。

人工智能技术相比启发式调度规则与数学规划方法，能够在较短的计算时间内提供优化调度方案。其主要的缺点是这些方法容易陷入局部最优，并且使用这些技术时对其机理不是非常的了解。

2.4.5 元启发式调度算法

芯片制造系统调度问题是典型的大规模组合优化问题，也是一类典型的 NP-Hard 问题，即在多项式的计算时间内无法获得最优解。所以人们不得不采取近似的方法以求在一个相对较短的时间内获得近似最优解。这类近似算法被称为启发式算法（heuristics），通常这种算法利用针对具体问题的相关知识来建立或者改进所求得的解。智能算法是一类类似于启发式算法的新型算法，也可被称为元启发式算法（metaheuristics）。这类算法在学术研究和实际应用等领域都非常成功地解决了多种组合优化问题。

元启发式算法是一类算法概念的集合，可以用来定义应用于一个大范围内不同问题的各种启发式方法。即元启发式算法可以看作一种多用途的启发式算法，用来指导潜在与问题有关的启发式方法朝着可能含有高质量解的搜索空间进行搜索。

有关元启发式算法的例子包括模拟退火（simulated annealing），禁忌搜索（tabu search），进化计算（evolutionary computation）和蚁群优化等，前三者简要介绍如下，蚁群算法将在后续章节详细介绍：

（1）模拟退火算法

模拟退火（simulated annealing, SA）[49]是模仿固体（晶体）的物理退火过程。退火是一种物理过程，首先把一个固体加热到一定的温度，此时其所有的分子保持活跃状态，然后逐渐冷却，随着温度的降低，固体的分子逐渐稳定下来，重新以一定的结构排列。

通过一段比较长的时间，分子获得一个处于最小能量状态的完美结构。模拟退火算法把这个物理过程转换为针对组合优化问题的局部搜索算法的一个扩展，它把问题的解集关联到物理系统的状态上，把目标函数对应固体的物理能量，而最优解对应最小能量状态。

Yim 和 Lee[50]提出了一种基于模拟退火调度半导体生产系统中群集设备的方法，其优化指标是使总生产周期（Makespan）最小。群集设备包含一组加工单片晶圆的加工室。由于每种类型晶圆的路径受到约束，且机器没有缓冲区，因而难以获得优化解。模拟退火方法被用来搜索近优解。Erramilli 和 Mason[51]将不同的晶圆序列组合为工件，采用了混合整数规划模型和基于模拟退火的启发式算法，获得这些序列在单台批处理机器上加工的最小总加权拖期。

（2）禁忌搜索

禁忌搜索（tabu search, TS）[52, 53]使用记忆存储来指引搜索的过程，是局部搜索算法的一种扩展。禁忌搜索算法标记已得到的局部最优解或者求解的过程，并在此后的局部搜索过程中避开这些局部最优解和过程。禁忌搜索所采用的记忆存储包括短期记忆和长期记忆，短期记忆是把当前解的邻域限制到它的一个子集中，而长期记忆则可引入额外的解来扩充邻域。只依赖短期记忆的禁忌搜索算法被称为简单禁忌搜索算法[52]。

Geiger 等[54]提出有效的启发式规则和一类禁忌搜索过程，以保证在合理计算时间内获得高质量的解，其应用于半导体制造设备中的湿蚀刻过程。他们把该问题建模为流水车间的调度模型，其目标是最小化 makespan。

（3）进化计算

进化计算（evolutionary compution, EC）进化策略[55]、进化编程[56]和遗传算法[57, 58]的统称。这些算法的原理都是由物种自然进化模型的启发得到的，其共同点是它们都是基于群体的算法。这些算法所使用的算子（如选择、交叉和变异算子）都来源于种群演化。算法中的每一个个体都直接或者间接地代表所考虑问题的一个解。选择算子指的是选择一些个体可以生存下来进入下一次迭代中的过程；交叉算子把两个或多个个体的部分合并起来生成新的个体，这些新的个体又称为后代；变异算子对某个个体进行随机修改。不同的进化计算算法的主要区别在于个体的表现方式和所使用算子的执行方式的不同，但是这些差异已经越来越模糊了。

Liu 及 Wu[59]提出面向机器的基于遗传算法（GA）的启发式调度方法来解决半导体生产系统的多目标控制优化问题。Wang 和 Uzsoy[60]结合基于 random key 的遗传算法与动态规划方法，处理带有工件动态到达特征的单批处理机调度问题。Melouk 等[61]提出了一种模拟退火算法, 用来解决具有不同加工时间和尺寸的工件在单机批处理设备上的调度问题。Mönch 等[62]研究了多产品、工件动态到达的并行批处理设备的调度问题, 其

优化目标为最小化总加权拖期，文章提出了两种混合遗传算法，获得了良好的效果；Koh等[63]研究了带有不兼容工件族和非相同尺寸工件特征的单批处理机加工系统，提出了一些启发式算法和一个基于 random key 的遗传算法，其优化目标是减小 $C_{\max}$ 和$\sum w_i C_i$。

2.5 研究现状总结

晶圆制造是半导体制造系统中最为复杂、昂贵和关键的部分。由于晶圆制造系统内部资源繁多、重入型制造过程复杂冗长，以及混合型加工模式，使得晶圆制造系统成为现代高科技大规模复杂生产系统的典型代表。由于半导体制造产业在我国快速发展，因而就晶圆制造系统的调度方法进行深入研究，对于提高我国现有半导体制造企业的生产管理水平，促进相关产业的健康发展，提高企业综合竞争力都具有十分重大和深远的意义。

现有的晶圆制造系统主要集中在投料策略与派工策略两类方法，从实际的实验与实践效果来看，派工策略对系统调度的影响更大。现有的派工策略方法可分为数学规划、调度规则与元启发调度方法。其中数学规划方法可以获得较为精确的解，但是只能应用于较小的简化问题，当问题变复杂后即失效。调度规则方法是目前晶圆制造系统调度的主流策略，其计算时间短，易于应用，但是随着问题的复杂以及自动化程度的提高，其调度结果鲁棒性差的特点将制约进一步的应用。元启发调度方法是近来兴起的一类计算智能调度方法，其以一定概率接受劣解的机制可以跳出局部最优，而其分布式计算可以提高搜索的速度。元启发式调度方法是目前发展很快的一类方法，也必将成为未来晶圆制造系统调度的主流方法。但是元启发式调度方法计算时间较长，对越来越庞大的晶圆制造系统调度问题难以支持，因而必须对传统元启发是调度方法进行改造，才能适应晶圆制造的大规模复杂特征。

第三章 晶圆制造系统的问题分解框架

3.1 引言

大规模晶圆制造系统调度问题难以被常用的调度方法处理，因而需要将问题进行分解，然后再分别求解。分解方法的思路是将一个调度问题分解为若干个小的子问题，逐个求解，然后将单个子问题的解合并成原问题的调度方案。一旦一个子问题得到求解，则将对其他未求解子问题施加额外的约束。这些额外约束可能会导致未求解子问题无法找到可行解。所以一个分解方法的关键部分是要找到一个有效表达子问题间关系的模型。在本论文中我们使用析取图模型来表示大规模晶圆制造系统。析取图模型是 Roy 和 Sussmann 在 1964 年提出的，并被许多学者有效应用在精确算法和启发式算法中，以解决经典的 job shop 调度问题。但是，传统的析取图模型无法处理晶圆制造系统的混合加工模式等特征，因而本文在分析传统模型的基础上，对析取图模型进行了扩展，使之可更好地表达晶圆制造系统的特征。在对析取图模型扩展的基础上，提出了应用蚁群算法对晶圆制造系统调度问题进行分解的框架，提出了相应的调度方法、解决算法以及协调机制。

本章首先描述晶圆制造系统的调度问题及特点，然后概述现有的大规模调度问题的分解方法，接着分析析取图模型存在的问题及对其进行的扩展，最后提出应用蚁群算法求解晶圆制造系统调度问题的分解框架。

3.2 晶圆制造系统的调度问题

3.2.1 问题描述

晶圆制造系统有 H 个机器组，分别用 G_i 表示，i=1,2,…,H，第 i 个机器组有 A_i 台机器。系统共加工 C 类产品，其中第 j 类产品的生产需要经过 w_j 道工序的顺次加工，用 p_{jk} 表示 j 类产品的第 k 道工序，t_{jk} 表示第 k 道工序需要的加工时间，j=1, 2, …, C; k=1, 2, …, w_j，假定每类产品各工序所需要的加工时间是确定的。调度目标是为每道工序选择最合适的机器，以及确定每个机器组前所有待加工任务的加工次序，得到一种具有最优性能指标的调度方案。

在实际调度问题中，经常会遇到多项性能指标要求，在晶圆制造系统中，平均生产

周期、在制品水平、产出率、及时交货率等是比较重要的性能指标

3.2.2 调度约束

半导体晶圆制造系统的调度约束主要为工艺约束和设备约束。

（1）工艺约束

工艺流程规定了在制品的详细加工工序，同时规定了在制品要访问的设备群，因为任何一个确定的加工工序都只能在某个确定的设备群内进行加工。

（2）设备约束

半导体晶圆制造系统中，使用的设备数量多、加工类型多，设备使用条件要求苛刻，设备的维修维护频繁。这些与设备有关的约束条件进一步提高了半导体制造系统调度问题的复杂程度。

除了上述主要约束外，加工过程还需满足以下约束条件：

- 每个工件至少访问某些特定机器至少两次。
- 连续的两个工序不能在同一个机器上加工。
- 不允许任务的强行插入。即只有当一个操作结束时，另一个操作才可以开始。
- 每个机器组所包含的机器具有相同的属性。
- 每个工件（lot）包含 25 片 wafer。
- 缓冲区的空间无限大。
- 运输时间采用平均值。
- 机器的当机和修复时间服从指数分布。

3.3 基于析取图模型的晶圆制造系统描述

3.3.1 析取图模型

一个析取图 $G(N, A, E)$由一组节点 N，一组合取弧 A，以及一组析取弧 E 组成。每一个节点 ij 代表工件 j 上的操作 i。起始节点 0 代表加工开始，终止节点*代表所有工件加工结束。一个析取弧由一对方向相反的弧线组成，任何一个确定的路径都只能包含这两条弧线中的一条。一个合取弧则是一条简单的单向弧。

一组合取弧代表同一个工件中操作间的优先关系，这个优先关系是由预先设定的工件加工路径确定的。对于一对操作 mj 和 nj，如果 mj 是 nj 的紧前操作，则在两个操作间设置一个合取弧(mj, nj)，以表示其优先约束。对每一个工件 j 添加一个合取弧$(0, 1j)$，

表示起始节点是所有工件的前置操作。对每一个工件 *j* 添加一个合取弧(*nj*, *)，表示终止节点是所有工件的后置操作。弧的长度代表了该弧源出操作的加工时间。从起始节点到终止节点一共有 *n* 条路径，每一条路径代表了一个工件，其长度是该工件的总加工时间。

对于在同一台机器上加工的两个操作 *ij* 和 *kl*，设置一对析取弧进行连接。由于一台机器同一时刻只能加工一个操作，因而一个可行调度只能选择两个方向弧线中的一条。在同一台机器上加工的操作间的析取弧组成了一个一个团（一个完全的子图)。所有节点属于一个团的操作必须在同一台机器上加工，所以其加工时间不能重叠。一个调度就是对应于确定析取图中各析取弧的方向。通过选择并确定每一对析取弧中某个方向的弧线，就获得了一个可行调度。从起始节点到终止节点的最长路径长度给出了该调度的生产周期（makespan）。

图 3-1 表示了一个具有两台机器、三个工件调度问题的析取图。操作 11，22 和 13 在第一台机器上加工，操作 21，12 和 23 在第二台机器上加工。节点 0 代表调度开始，节点*代表所有工件加工结束。图 3-2 代表该调度问题的一个可行调度。粗实线代表析取弧中被确定的那一个方向的弧线。由图可知，第一台机器的加工序列为 13-11-22，第二台机器的加工序列为 12-23-21.

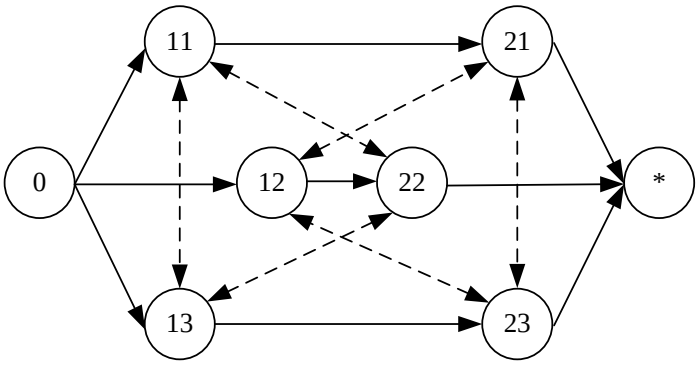


图 3-1 两台机器三个工件调度问题的析取图模型

Fig. 3-1 Disjunctive graph representation of a shop with two work-centers and three jobs

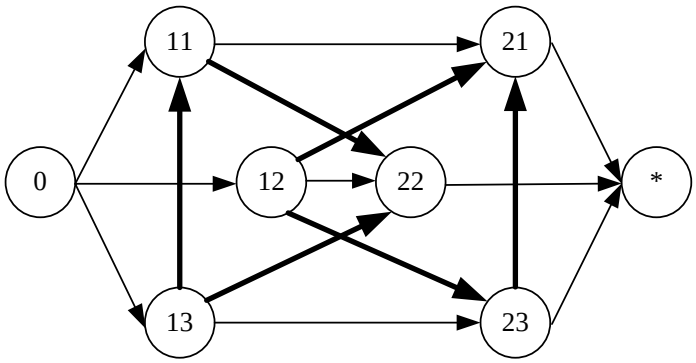


图 3-2 问题的一个可行调度

Fig. 3-2 Representation of a schedule for the example

一旦对应于一台机器的析取弧都确定了方向，则在该台机器上加工的操作间的优先关系就建立了起来。一台机器的调度方案事实上便是在该台机器上加工的所有操作的优先序列。对于表示一个调度，只有那些代表操作间紧前关系的弧是必不可少的，其他的弧则是冗余弧，可以被消除以减小弧集合的规模。通过删除那些不表示操作间紧前约束关系的弧可以简化对应于这个部分调度的析取图。余下的弧必须满足三角不等式，即对于所有的节点 i, j 和 k，必须满足 $c_{ik} \leq c_{ij} + c_{jk}$。因而，图 3-2 被简化为图 3-3，其中弧(13, 22)和(12, 21)被删除。这样除了起始节点和终止节点外，一个节点只有最多两个引入弧，同样最多有两个引出弧。两个引入弧中的一个是从其紧前操作中引出的合取弧，另一个是从与该节点属于同一个团中节点引出的析取弧。同样的，两个引出弧中的一个是指向其紧后操作的合取弧，另一个是指向与该节点属于同一个团中节点的析取弧。

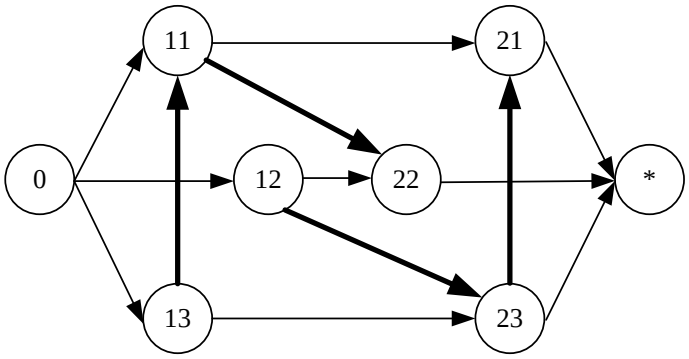


图 3-3 从可行调度中消除冗余弧

Fig. 3-3 The Schedule with redundant arcs eliminated

一个操作的投料时间和交货期可以在图中通过直观的方式计算获得。一个操作的投料时间是从起始节点到代表该操作节点的最长路径长度。令 $L(p,q)$代表节点 p 到节点 q

的最长路径，操作 ij 的投料时间 r_{ij} 和交货期 d_{ij} 可以通过下式计算：

$r_{ij} = L\ (0, ij)$

$d_{ij} = L\ (0, *) - L\ (ij, *) + p_{ij}$

3.3.2 现有析取图存在的问题

3.3.2.1 延迟优先约束

Adams 等使用上述析取图模型表示经典 job shop 模型，并提出了 shifting bottleneck 调度方法。该方法仅对每个子问题施加投料时间和交货期约束。这样的过程并不能保证获得可行解。Balas 等指出对于一台机器的调度可能会对在其他机器上加工操作开始时间施加约束。因此，他们提出增强型子问题，将这些约束包含其中，并称之为延迟优先约束(delayed precedence constraints, DPC)。

延迟优先约束通过下面的例子进行说明。对于一个有两台机器和三个工件的调度问题，操作 11，22 和 13 在机器 1 上加工，操作 21，12 和 23 在机器 2 上加工。我们假定先调度机器 2。图 3-4 表示了一个可行解的析取图。合取弧上的数字表示这些弧的成本。在机器 2 上加工的操作的顺序为 21，12，23。机器 1 的一个可行解是 13，22，11.图 3-5 表示包含机器 1 调度方案的析取图。虽然该解对于机器 1 是可行的，但是由于存在弧(11-21-12-22)，因而这个解从总体上说是不可行的。通过对图 2-4 的检查，我们发现机器 2 的调度建立了一条从操作 11 到 12 的路径，意味着操作 11 必须在 12 开始前加工完成。然而，对于单一机器来说，加在其上的约束只有同一个工件中各操作的优先关系。由于新增加的调度方案导致的新约束被忽略了。这导致了调度方案的不可行，并需要用额外的计算时间去检测和更正。

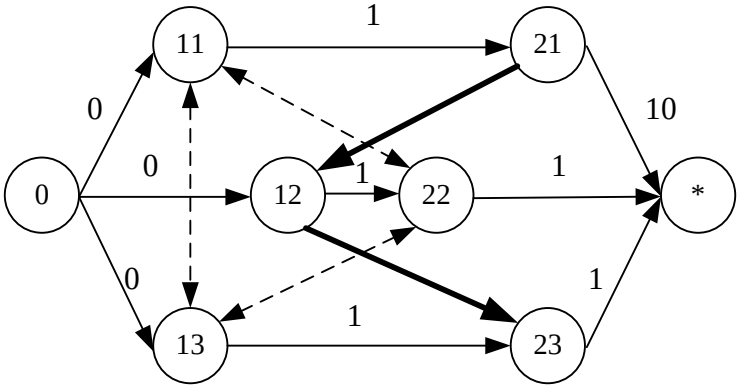


图 3-4 仅对机器 2 进行调度后的析取图

Fig. 3-4 Disjunctive Graph Representation with only machine 2 scheduled

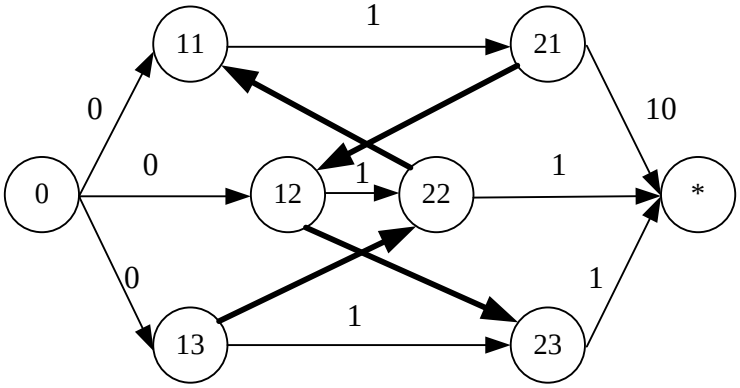


图 3-5 机器 1 调度后的析取图

Fig. 3-5 Directed graph after Machine 1 is scheduled

延迟优先约束的存在导致需要建立一个新的子问题模型，包含投料时间，交货期和延迟优先约束。与此同时，批处理机器和带有设置时间需求的单机处理机模型也需要进一步考虑。

3.3.2.2 批处理机器加工

批处理机可将多个工件作为一个批次进行加工。在半导体加工中 burn-in 流程和扩散、沉积流程中存在这样的加工过程。一个批次中的工件同时开始加工，同时加工完成。为代表同时加工的一组操作，我们添加节点 c，代表在批处理机上的批次加工。对应于一个批次中操作的节点通过合取弧连接到节点 c，节点 c 作为批次的中心。并从节点 c

再连接到该批次中各操作的下一紧后操作。从一个节点到中心节点的弧的长度设置为 0. 所以一个中心节点的准备时间是批次中操作的最后到达时间。从中心点出发的弧的长度设置为批处理的加工时间，代表批次中所有的操作有同样的开始加工时间和完成时间。图 3-6 表示一个带有批处理加工机器的调度问题。

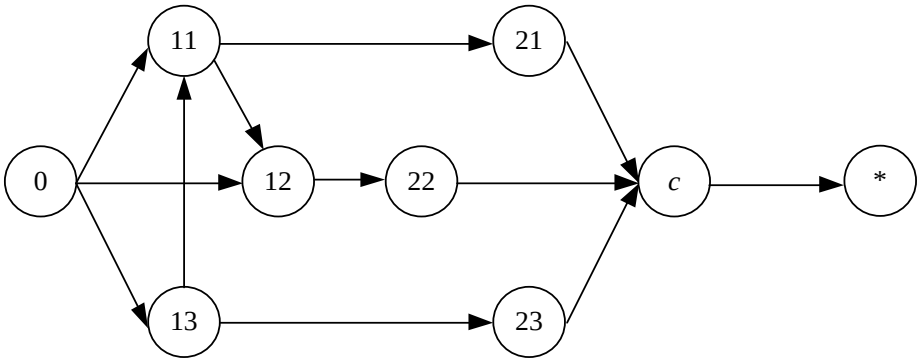


图 3-6 带有批处理机器的调度图

Fig. 3-6 Schedule in a shop with batch processing machine

在图 3-6 中，操作 21，22，23 作为一个批次同时加工，节点 c 代表该批操作，弧(21, c), (22, c)和(23, c)的值为 0，弧(c, *)的值为批处理机的加工时间。

3.3.2.3 顺序依赖设置时间

半导体晶圆制造系统中有一些机器在加工前需要进行设置，如光刻机。在光刻机操作中，多个模板被印刷到晶圆上，并逐一进行加工。一旦一台机器被设定为加工一个特定的模板，它可以持续加工同一模板的晶圆而不需要进行设置。如果下一个待加工晶圆的模板与当前晶圆的不一样，则需要一定的设置时间以重新更换模板。

3.3.3 对析取图的扩展

为了更好地应对晶圆制造系统的典型特征，需要对析取图进行扩展，Masan 等[175]在这方面做了大量的工作，通过扩展吸取图更好地表达晶圆制造系统的一些特征。以一个带有 4 台机器和 3 个工件的调度问题为例。其析取图如图 3-7 所示。

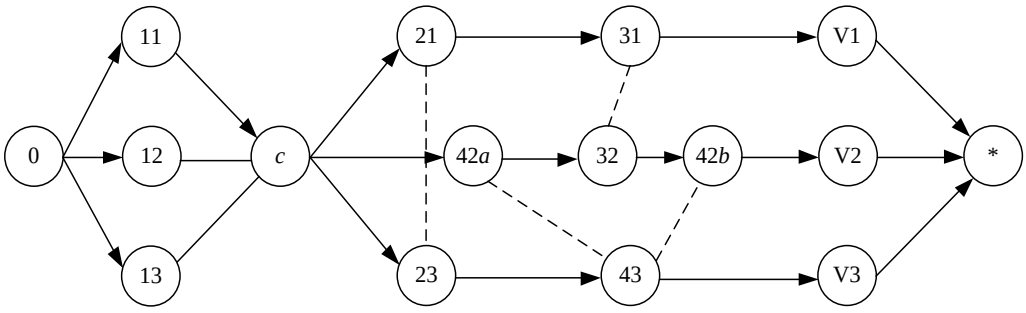


图 3-7 带有 3 个工件和 4 台机器的调度问题的析取图

Fig. 3-7 Disjunctive graph for scheduling problem with 3 jobs and 4 machines

析取图 3-7 与传统析取图有一些重要的不同之处。

3.3.3.1 虚拟工件结束节点

对于调度目标为总加权延迟最小的调度问题而言，我们需要知道每个工件的完成时间，以便计算每个工件的延迟。为此，为每一个工件添加一个虚拟工件结束节点（*V*1,*V*2 和 *V*3）。每个工件的虚拟结束节点的准备时间事实上等于该工件的完成时间。如果起始节点 0 和终止节点*代表所有工件的虚拟开始和结束时间，则 $L\ (Vj, *) = \max_j\ (C_j) - C_j$。单个节点的准备时间和交货期通过最长路径法计算。

3.3.3.2 批处理加工机器

如图 3-7 所描述，该调度问题中的机器 1 为批处理加工机器。它包含两台同样的机器，其最大批量为 3，即可同时加工 3 个工件。一个批处理操作的加工时间并不直接关联到从该节点引出的弧上，而是关联到从代表批处理操作的虚拟节点 *c* 引出的弧上。

有许多不同的方式对工件进行组批。如果总的待组批工件的数量为 *n*，而批处理机器的最大批次为 *b* 个工件则总共有 $\binom{n}{b} = \frac{n!}{b!(n-b)!}$ 种可能组成大小为 *b* 的批次，有 $\binom{n}{b-1}$ 种可能组成大小为(*b*-1)的批次，以此类推。所有潜在的可组批的数量为 $O(n^b)$。为简明起见，在图 3-7 中仅表示了由 3 个工件组成批次的方式。

关于批处理机器组加工的另一个问题是，当批处理机器组未被调度时，如何在析取图中表示批处理步骤间的关系。考虑 3 个工件，(*A, B, C*)，每个工件都需要两个加工步骤，(1, 2)。假定步骤 1 在批处理机 *i* 上加工，该批处理机的最大加工批次 *b*=2。图 3-8 描绘了对步骤 1 的 6 种可能的组批方式。

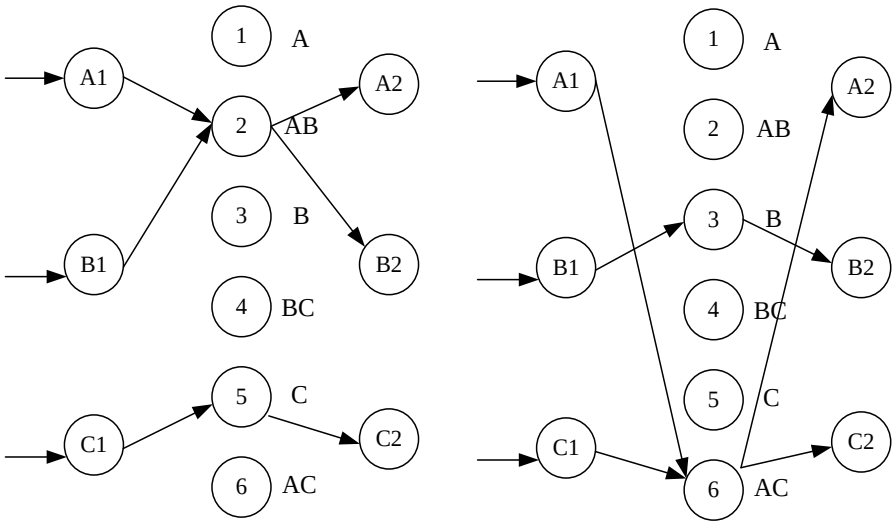


图 3-8 对批处理机器 1 组批的两种可能配置

Fig. 3-8 Two potential batching configurations for batching batch machine 1

每个操作的准备时间和交货期通过计算析取图中的最长路径长度而获得。然而，在批处理机未被调度前，批次中的组成成分并不知道。因而，析取图并没有被完全连接。如图 3-8 所示，节点 *A*2, *B*2, *C*2 并没有和 6 个可能的组批节点（节点 1-6）完全连接。由于相应的析取图并没有被完全连接，节点 1-6 的准备时间和交货期并不能确定。

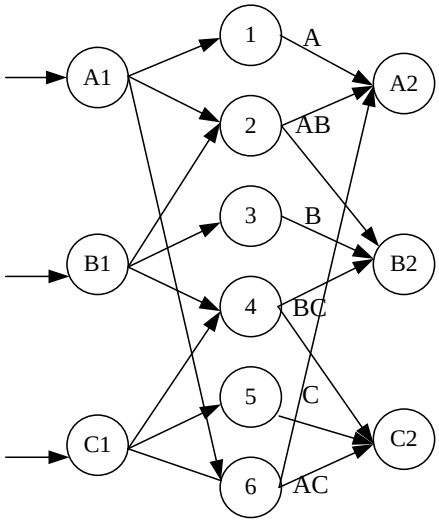


图 3-9 析取图中潜在组批节点全连接

Fig. 3-9 Initial batching node connections in disjunctive graph

为了获得对每一个操作准备时间和交货期的估计，我们为析取图中的每个潜在组批

配置添加合取弧。如图 3-9 所示。通过连接所有的潜在组批配置，我们可以计算所有的可能性。一些潜在的较差的组批方式可以通过估算后排除。

3.3.3.3 顺序依赖设置时间

假定机器 i 具有顺序依赖设置时间的特性，析取图在初始情况下并不包含对这些设置的任何表示，直到机器 i 被调度，方才可以定义顺序依赖时间。在移动瓶颈调度算法中，一旦一台机器 i 被确定为关键机器，其析取图中机器 i 的路径长度等于相应的操作加工时间加所需的机器设置时间。

3.3.3.4 并行机

假定机器组 i 有 m_i 个相同的机器组成，在初始状态下，析取图中不指明并行机的存在。直到机器组 i 被调度，如何将工件分配到这 m_i 个机器上仍然不确定。一旦机器组 i 被定义为关键机器并进行调度，析取图则会添加若干合取弧以定义在 m_i 台不同机器上的加工顺序。

3.3.3.5 重入流

在机器 i 上加工的所有操作彼此间用析取弧进行连接，所以，对应机器 i 的析取弧的集合组成一个团。如果调度问题中有 m 台机器，则存在 m 个团，对应于相应的图。然而，在如半导体晶圆制造系统的多重入流调度问题中，一个工件可能包含多个在同一机器 i 上加工的操作。因而，对应于机器 i 的析取弧集合在面临重入流的时候就无法形成一个团。

图 3-7 中表示了针对重入流的析取图表示方法。对于工件 2，其第 2 个和第 4 个操作都需要在机器 4 上加工。所以在节点对(42a,43)和(42b,43)中通过析取弧相连，而节点对(42a,42b)之间则不连接。每个工件预定的加工流程需要通过合取弧进行连接。在图 3-7 中，按照工件加工顺序，操作 42a 在 42b 之前进行加工，因此(42a,42b)之间不通过析取弧相连。由于重入流的存在，排除了节点(42a)，(42b)和(43)之间形成团。

3.4 大规模调度问题分解方法介绍

分解方法是处理大规模优化问题的一种常用方法。它将一个复杂的大规模问题分解为若干相对简单的小规模子问题，对每个子问题进行独立求解。然后根据子问题之间的关联项将子问题的解集成原问题的解。因此，有效地分解方法取决于问题的分解方式和

协调机制，包括子问题的建模、子问题优化求解，以及子问题之间的协调。不同的分解和协调方式对应不同的分解方法。

3.4.1 基于数学规划方程的分解

在调度问题研究的早期，多采用数学规划方法对问题进行建模与求解，因此调度问题往往可以用一个混合整数线性规划模型来描述。其中加工工艺要求、机器加工能力限制等各种物理约束可以用多个数学等式或不等式描述，如果用传统的割平面方法或分支定界法对其进行求解，则问题规模会急剧增加，导致求解时间的急剧上升，甚至无法在可行时间内获得解。从这个角度出发，学者们开发了一些问题分解的方法，将原问题分解为若干较小的子问题，从而使得原问题更易于求解。这些方法包括 Dantzig-Wolfe 方法，拉格朗日松弛/分解方法与 Bender 分解方法。

Dantzig-Wolfe 方法也称列生成法(column generation)[64]。其核心思想是将可分解约束对应的可行解集描述为顶点集，每个变量可以描述为顶点的凸组合。用顶点重新描述耦合约束和目标函数，得到一个新的规划模型，该模型中约束(行)个数很少但变量(列)个数非常多。由于优化解中很多变量为 0，因此执行该算法时只需包含列的部分子集得到严格主规划问题，通过求解子问题(代价问题)来判断是否需要添加变量，如果添加新的变量则重新求解主规划问题。这样原问题的优化可以分解为一个主规划问题和若干子问题。该方法可以降低内存需求和计算时间。D-W 分解方法一般应用于线性规划问题的求解，它可结合分支代价(branch and price)求解大规模整数规划问题[65-67]。

拉格朗日松弛(Lagrangian relaxation) 法的基本思想是将优化问题中的强约束进行松弛，利用拉格朗日乘子将其转移到目标函数中，构成求解起来相对简单的拉格朗日问题。通过对拉格朗日问题的求解，可得原问题的一个界(对最大化问题是上界，对最小化问题是下界)；再利用对偶问题的求解(通过次梯度法(sub gradient)不断更新拉格朗日乘子，改善原问题的界)，当对偶问题达到最优值时，便可得到原问题的最优解[68, 69]。拉格朗日松弛法最初由 Held 和 Karp 提出用于解决旅行商问题[70, 71]。Fisher 将其成功地应用于求解调度问题[72, 73]。随后被推广应用求解不同类型的生产调度问题[74-76]。拉格朗日松弛法是求解整数规划问题的一种有效的优化算法[77]。拉格朗日分解法(Lagrangian decomposition)是拉格朗日松弛法的一种推广[78]。利用某些问题约束和目标函数的可加性，将一部分变量实现硬拷贝，再将硬拷贝松弛，利用可分性将模型分解成若干个独立的子问题，分别求解各个子问题。最后利用子问题的解构成原问题的一个界，或利用启发式算法构成原问题的一个近优解。吴清烈和徐南荣[79]提出利用各个子问题的有序近优

解构成主问题，通过求解主问题，得到原问题的一个近优解，通过迭代，不断缩小上下界之间的差距，直到满足需要。使用拉格朗日分解的一个局限性是，调度问题的目标函数必须具有可加性。因此，许多大规模生产调度问题必须将目标转换后才能使用拉格朗日分解法。

Bender 方法是针对数学规划方程结构中的复杂变量（complicating variables）而设计的。复杂变量是指那些当其值被暂时固定后的，则优化问题更易求解的那些变量。采用 Bender 方法进行问题分解时，将原优化问题分解为一个主优化问题及若干子问题。在子问题中固定复杂变量，求解子问题得到原问题的下界。随后通过增加 Bender 割使主优化问题的性能指标不断逼近原问题的最优解。主优化问题用于求解添加 Benders 割约束后的复杂变量[80]。Benders 算法的优点在于分解后的子问题一般很容易求解，而且这种分解算法可大大减少计算机内存消耗。Benders 分解方法主要用于求解混合整数规划问题。

以上三种分解方法适用于具有结构可分解特性的数学规划模型。它们有严格的数学保障，最终的解往往是原问题的近似解。然而这些方法受模型结构的限制，物理意义往往不明确。

3.4.2 基于时间的问题分解方法

基于时间的分解是一种直观的分解方法，其基本思路为通过把研究问题的范围限制在一段时间区域内，从而减小问题的规模。Basset 等[81, 82]提出一种分层模型，在计划层，总的调度时间被分为若干时间段，每一时间段构成独立的子问题，利用启发式算法进行求解，这些解在调度层进行协调。滚动时域分解方法是另一种应用更为普遍的时间分解方法，该方法应用广泛，被用于单机调度问题[83]，并行机调度问题[84]，并扩展应用到晶圆制造系统调度问题[85, 86]。该方法每次只考虑一个时间窗内的调度子问题，应用分支定界法或智能调度方法进行求解，求解完成后滚动到下一时间窗，求解新的子问题，如此反复以完成所有子问题的调度。各子问题的解之间需要通过协调算法以保持一致性。

3.4.3 基于工件的分解方法

基于工件的分解方法是将工件按照其属性分类，每次调度只考虑一小批工件，从而降低了调度问题的难度。Sannomiya 等[87]将工件首先按照一定的属性进行排序，然后划分为若干批，逐批加入遗传算法参与运算。Roslöf 等[88, 89]提出一种排序更新算法，首先对一批工件进行排序，然后将剩余工件按照一定的插入算法逐个插入已排序工件中，直

到所有工件调度完成。Fox 等提出的 ISIS 调度系统[90]也采用了基于工件的分解方法。该方法每次选择一个工件，只考虑该工件所有工序的加工安排，应用此方法可以减少库存。基于机器能力松弛的拉格朗日方法也可以看作是一种基于工件层的分解方法。该方法松弛机器加工能力约束，将其乘以拉格朗日乘子添加至目标函数中。松弛后的问题自然分解为一系列子问题，每个子问题对应一个工件。通过固定拉格朗日乘子可独立优化求解子问题，随后根据子问题的解可以修正拉格朗日乘子，协调各子问题。Peter Luh 对此做了大量的研究工作[74-76]。

生产车间常用的调度规则也可看作基于工件的分解方法。该方法每次只考虑机器调度时刻到达缓冲区的工件，从而降低了调度问题的规模。该方法对这些已到达工件进行优先排序，优先级最高的工件最先加工。

3.4.4 基于机器的分解方法

基于机器层的分解方法将一个车间问题分解为若干子问题，每个子问题对应一个单机或并行机调度问题。对这些子问题采取不同的调度方法，子问题的解最终合成原问题的解。

Chu 等[91]将车间中的机器划分为若干单元，分单元进行调度，但是并未考虑单元之间的耦合问题。在此研究基础上，Sun 等[92]提出了代理操作和虚拟机器的概念，使各单元解耦成独立子问题。

移动瓶颈方法是一种应用最广泛的基于机器的分解方法。该方法最初是由 Adams 等[93]提出，并经过修改和扩展，应用在晶圆制造系统调度问题上[94]。该方法每次确定一台瓶颈设备，通过松弛其约束，使车间调度问题转化为单机调度问题，将该瓶颈设备调度完成后，再在剩余设备中寻找一台瓶颈设备，如此方法直到所有设备调度完毕。每调度完成一台设备后，其调度方案需要和先前的调度结果进行协调，以消除冲突，保持调度结果的一致性。Dauzere-Peres 和 Lasserre 改进移动瓶颈算法，对子问题考虑迟延顺序约束(delay precedence constraints, DPC)[95]。随后 Balas 等提出了一种分枝定界算法优化求解考虑 DPC 的子问题[96]。Demirkol 等对不同加工类型(Flow shop, Job shop 和 2-set Job shop)、不同性能指标($C_{\max}$ 和 $L_{\max}$)的调度问题进行了大量的仿真测试，仿真结果显示 SB 得到的调度性能很好，但计算时间很长[97]。松弛工件顺序约束的拉格朗日方法也可看作是另一种基于机器层的分解方法[98]。

3.5 应用蚁群算法求解晶圆制造系统调度问题的分解框架

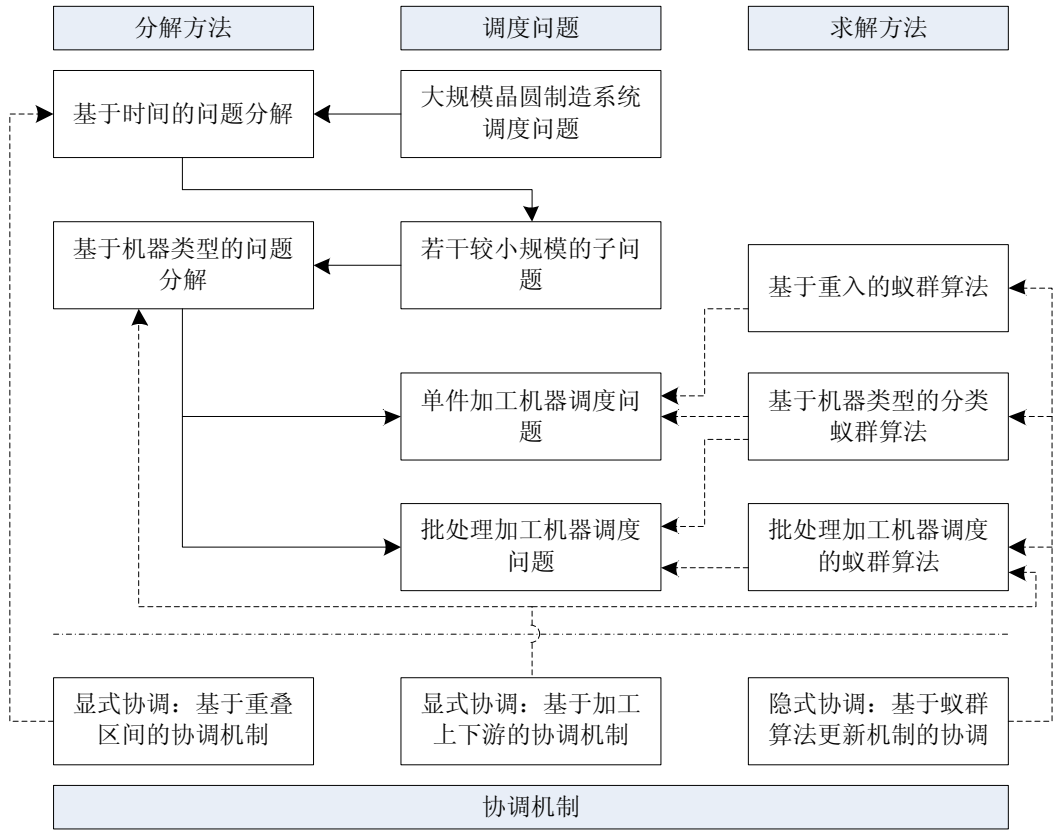


图 3-10 应用蚁群算法求解大规模晶圆制造系统调度问题分解框架

Fig. 3-10 The framework of decomposition method for scheduling large-scale wafer fabrication system by ACO algorithm

对晶圆制造系统进行调度，必须较好地处理其超大规模、复杂多重入及混合型制造的特征。在采用析取图对其进行描述的基础上，还需要应用问题分解方法，将大规模的问题分解为若干较小的子问题，将复杂的问题分解为较简单的问题，然后根据问题的类型采用相应的蚁群算法求解，再将子问题与简单问题的求解结果组合为原问题的解。基于以上思路，提出应用蚁群算法求解晶圆制造系统调度问题的分解框架如图 3-10 所示：

针对晶圆制造系统复杂多重入的特征，在求解过程中，针对批处理机器开发相应的蚁群算法进行处理，并整合进基于加工上下游的协调机制，考虑上下游机器的加工信息。此外，分类蚁群算法将单件加工机器与批处理加工机器的信息素更新放到一个框架下进行，保证了信息素更新的一致性。

3.5.1 调度问题分解方法

在上述的问题分解框架中，针对晶圆制造系统的典型特征，采用不同的方法进行处理。对于晶圆制造系统大规模调度的特征，按照时间进行分解，在一个有限的时间区间（一天，一个班次等）内，所涉及的机器数量与工件数量相对可控，从而将大规模的调度问题分解为若干较小规模的子问题。在这些较小规模的子问题中，仍然存在着混合加工的问题，即单件加工机器（Unit-processing machine）与批处理加工机器（Batch-processing machine）存在于同一个调度问题中，针对这一特征，采用基于机器类型的问题分解方法，将子问题分解为以批处理机为核心，单件加工机器为上下游机器的更小的调度问题。

3.5.2 求解算法

为处理分解所得到的若干子问题，需要修改蚁群算法，使其可以适应晶圆调度系统的特征。对蚁群算法的修改从几个方面进行。首先，为应对晶圆制造系统多重入的特征，开发基于重入的蚁群算法，将重入信息整合进信息素更新过程中，从而使得算法能自动识别工件的不同状态，采取相应的调度策略。其次，针对晶圆制造系统中最难调度的批处理机，开发有针对性的基于批处理机调度的蚁群算法，本文所提出的批处理设备蚁群调度算法与之前相关文献有所不同，以前的文献大多着眼于将蚁群算法应用于批次调度，本文则将蚁群算法应用于组批，扩展了蚁群算法的应用范围。最后，考虑在同一个蚁群算法框架下混合调度批处理机和单件加工机器的问题，提出分类蚁群算法，整合两类加工机器的蚁群算法，使他们在同一个信息素更新机制下运作。

3.5.3 协调机制

问题分解方法将大问题分解为若干小问题，使其更易求解，但是其所付出的代价是需要使用协调机制将小问题的解整合为原问题的解。在本文采用了三种协调机制来整合子问题的解。首先，应用于协调基于时间分解的子问题的是一类显式协调机制，即基于重叠区间的协调机制。该机制在相邻子问题间设立重叠区间，共有一部分操作，从而使得二者所获得的解可以保持连贯性。其次，基于机器类型的分解方法，采用一类显式协调机制，即根据上下游的信息来调整调度算法。此外，还有另一类隐式协调机制应用于重入与分类蚁群算法，即利用蚁群算法本身的信息素更新机制进行协调，具体过程可见第四章对分类算法的描述，通过将针对不同类型机器的蚁群算法信息素放置于同一个更

新框架下，实现不同机器间信息的沟通与协调。

3.6 本章小结

应用蚁群算法调度晶圆制造系统必须要和问题分解方法相结合，而同样的问题分解方法必须要针对晶圆制造系统的特征以及蚁群算法的特性进行。本章通过对晶圆制造系统调度问题描述，介绍了采用析取图对其进行表示的方法，并在此基础上提出了应用蚁群算法求解晶圆制造系统调度问题的分解框架，为具体的问题分解与蚁群算法的应用奠定了基础。

第四章 基于时间分解的大规模晶圆制造系统蚁群调度方法

4.1 引言

如第二章所述，半导体晶圆制造系统与传统的 job shop 和 flow shop 生产方式不同，其既有相对固定的加工流程（类似于 flow shop），又存在大量的重入流（类似于 job shop），为此被一些学者看作第三种制造系统[3]，这增加了其生产流程的复杂性。加之晶圆制造系统的大规模生产特性，对其进行调度非常困难，传统基于数学规划的调度方法无法获得应用。在实际生产中大多采用基于规则排序的启发式调度方法，这类方法简单有效，但是通常仅利用机器的局部信息，因而难以获得全局最优解。基于全局信息的启发式智能算法可以获得更佳的全局优化解，但是受制于计算时间的庞大，难以应用于大规模调度问题。

蚁群优化方法（ACO, Ant Colony Optimization）作为一种有潜力的群体智能算法，其主要优点是并发多线程搜索机制，可以获得更优的搜索解。因而近年来被成功应用于车间调度问题的研究[99, 100]，但是在晶圆制造系统调度中，蚁群算法调度主要应用在某些特殊问题上，如梁静等[101]设计了一种用来解决多品种晶圆批连续到达环境下非等效平行多机器的批调度问题的双层蚂蚁算法；Song 等[102]用蚁群算法解决半导体装配测试生产中瓶颈机器的调度问题。到目前为止，面向全局调度优化的蚁群算法应用研究尚为空白，这是因为蚁群算法具有如下缺点：当问题复杂，要素众多时，需要较长的搜索时间；当问题规模增大到一定程度时，蚁群算法面临失效的困境。

为应对上述困难，需要对晶圆制造系统的大规模调度问题进行分解，分解从两个维度进行，基于时间的分解与基于机器类型的分解。本章的侧重点为前者，重点提出基于时间的分解方法以及子问题间的协调机制。同时也兼顾基于机器类型的分解，提出一类分类蚁群算法，针对不同的机器类型采用相应的蚁群算法求解。这种基于分解的蚁群优化求解方法降低了所求解问题的规模，并可有效应对多重入以及多生产类型所带来的复杂性。

4.2 蚁群算法及在生产调度中的应用

近年来，人们提出了各种智能算法，如模拟退火(SA)[49]，遗传算法(GA)[55-58]，禁忌搜索算法(TS)[52, 53]等来解决调度问题，虽然不能保证获得最优解，但在问题维数较大时

也能够在可行时间内找到问题的满意解。这些智能方法无论是理论研究还是应用研究都空前活跃[103]。同时，一些新的元启发式方法(Meta-heuristie)也逐渐发展起来。意大利学者 Dorigo 等[104]受蚁群觅食行为(Foraging behavior)中的基于信息素(Pheromone)的间接通讯机制的启发，提出了一种蚂蚁算法(Ant algorithm)，并应用该算法求解旅行商问题(TSP)获得了很好的效果。在上世纪 90 年代后期，这种算法逐渐引起了很多研究者的注意，并对算法作了各种改进或应用于其他更为广泛的领域，取得了一些令人鼓舞的成果。为了给这些算法提供一个统一的描述框架，Dorigo 等研究者[105-107]提出了称为蚁群优化(Ant Colony Optimization, ACO)的算法框架，所有符合蚁群优化描述框架的蚂蚁算法都可称之为蚁群优化算法(ACO Algorithin)，或简称为蚁群算法。研究发现，蚁群优化方法在解决离散组合优化问题方面有着良好的性能。具有 NP-hard 属性的生产调度问题作为组合优化领域的一个研究热点，也是蚁群算法的一个重要研究方向。

4.2.1 蚁群优化领域的主要算法及其应用的领域

Dorigo 等提出了第一个蚁群算法，AS 算法(Ant System)[104, 108]，AS 算法首先应用于解决旅行商问题(TSP)，然后 Maniezzo 和 Colorni[109]把 AS 算法应用于解决二次分配问题(QAP)。近年来，AS 算法得到了一定的改进和扩展。Gambardella 和 Dorigo[110, 111]提出的 Ant-Q 算法在 AS 算法的随机比例规则基础上，在解构造过程中提出了伪随机比例状态迁移规则(pseudo-random-proportional state transition rule)，从而能够实现解构造过程中知识探索(exploration)和知识利用(exploition)的平衡，并引入信息素局部更新过程，信息素局部更新规则引入了强化学习理论[112]中的 Q-学习机制[113]，此外，在信息素的全局更新中采用了精英策略。Dorigo 和 Gambardella[114]在 Ant-Q 算法基础上又提出了 ACS 算法(Ant Colony System)，用于解决 TSP 问题，该算法可看作 Ant-Q 算法的特例。Stutzle 和 Hoos[115]提出了一种称为 MAX-MIN AS 的算法，并与邻域搜索算法相结合来解决 TSP 问题，算法采用了用当前找到的最好解更新信息素来指引蚂蚁向更高质量的解空间搜索的贪婪策略，并对信息素设立上下限来避免算法的早熟。Stutzle[116]又用同样的方法对 Flowshop 问题进行求解，获得了比较满意的结果。Stutzle 和 Hoos[117]应用 MAX-MIN AS 算法对 TSP 问题，ATSP 问题(不对称 TSP)以及 QAP 问题作了仿真和对比研究，取得了很好的结果。Bullnheimer 等[118]提出了一种称为基于蚂蚁等级的 ASrank 算法。Gutjahr[119]提出了一个基于图的蚂蚁算法 GBAS (Graph-based Ant System)，并证明了算法将以概率 1-ε 收敛于最优解，其中 ε 可以通过选择足够大的蚂蚁数量 Y 或足够小的信息素挥发系数 ρ 来获得任意小的值。Blum 等[120]针对可转换为 0-1 整数规划问题的组合优化问题，

提出了一种 HC-ACO 算法。这类组合优化问题的解可描述为一个 0-1 值的二进制向量，解构成元素可定义为二进制向量中的每一位上的二进制数。HC-ACO 算法的信息素更新规则中解构成元素上的信息素增量是每个用于更新信息素的解的质量的加权平均，此外，对每个解构成元素定义了全局期望度(global desirability)和全局频度(global frequency)。全局期望度定义为包含该解构成元素的所有已找到的解中的最好解的质量，全局频度定义为该解构成元素出现的频度，也就是所有找到的解中包含该解构成元素的解个数。HC-ACO 算法在利用解构成元素上的信息素概率性的构造解之外，还有两个附加的解构造过程，其中一个根据解构成元素的全局期望度采用随机比例规则选择解构成元素来装配成解，另一个则根据全局频度的倒数来选择解构成元素。这两个过程构造出的解将用来更新信息素。这种机制既能保证高质量解的解构成元素具有较高的信息素，同时当算法陷入局部最优的停滞状态时，后一个附加的解构造过程倾向于构造新的解来更新信息素以跳出局部最优点。此外，国内一些研究者也对蚁群算法作了一些研究。覃刚力和杨家木[121]提出了一种基于自适应调整信息素的改进蚁群算法。该算法根据人工蚂蚁所获得解的情况，动态地调整路径上的信息素，从而使得算法跳离局部最优解。谢剑英和王颖[122]通过自适应地改变算法的挥发度等系数以克服蚂蚁算法易限于局部最优点的缺陷，并能够在保证收敛速度的条件下提高解的全局性。吴斌和史忠植[123]首先在蚁群算法的基础上提出了相遇算法，提高了蚁群算法蚂蚁一次周游的质量，然后将相遇算法与采用并行策略的分段算法相结合提出一种基于蚁群算法的 TSP 问题分段求解算法。

上述不同版本的蚁群算法主要运用于解决各种组合优化问题，比如 TSP 问题[110, 111, 114, 115, 117, 123-125]，QAP 问题[109, 126]，网络路由问题[127]，最大约束满足问题[128, 129]，资源约束的项目调度问题[130]，智能交通，图形染色问题，车辆路由问题，生产调度问题。一些研究者也提出了一些用于连续空间优化的蚂蚁算法[131-133]，但这些算法并不符合蚁群优化算法框架，这些算法在一定程度上可以看成是一种具有局部搜索过程的混合遗传算法。

4.2.2 蚁群算法在生产调度问题上的研究

蚁群算法在生产调度问题上的研究主要有最小化总拖期的单机调度问题 SMTTp[134-136]；最小化加权总拖期的单机调度问题 SMTWTP[99, 137]；单阶段并行机调度问题[138]；Flowshop 调度问题[139]；Jobshop 调度问题[140, 141]。

Bauer 等[134]修改了解决 TSP 问题的 AS 算法并应用于最小化总拖期的单机调度问题，并利用了一种称为改进的交期规则的启发式信息，并在解构造过程中在线进行局部信息

素更新。此外与分解启发式算法(decomposition heuristies)，交换启发式算法(interchange heuristics)和模拟退火算法的比较试验发现，他们提出的算法在更多的测试问题上获得了最优解。Merkle 和 Middendorf[136]研究了 SMTWTP 问题，其提出的蚁群算法的解构造过程不是按产品的排序来进行的，而是随机选择一个产品加工序列的一个位置，再按照信息素为此位置安排所要加工的产品，这样就能够公平地利用信息素，试验结果表明了这种方法的有效性。此外，还提出了几种不同的启发示信息并作了试验对比分析。Merkle 和 Middendorf[99]提出了一种新的全局信息素评价方式，称为信息素累加规则(pheromone summation rule)，在解构造概率规则公式中用解构成元素上的信息素局部累加值代替单个解构成元素上的信息素。

Stutzle[139]运用 MAX-MIN 算法，并与基于插入式邻域结构的邻域搜索算法相结合来解决排列 Flowshop 调度问题，算法采用了用当前找到的最好解更新信息素来指引蚂蚁向更高质量的解空间搜索的贪婪策略，并对信息素设立上下限来避免算法的早熟，算法获得了相对来说比较好的结果，但与目前学术界获得的最好结果还有一定的差距，其主要原因在于未对 Flowshop 问题进行深入研究，邻域结构的设计使得搜索空间过于庞大，此外，算法的解构造过程完全靠信息素的指引而没有利用问题的启发式信息，上文提到过，蚁群算法的一个主要优点就在于能在解构造过程中的解空间随机搜索时方便地利用基于问题的启发式信息，如果不利用该机制，算法性能有时就不如遗传算法等智能方法。此外 Stutzle[139]的算法与邻域搜索相结合而未能给出单纯的蚁群算法的计算结果。

Jobshop 调度问题中每个作业有多个工序，每个工序只能在某个指定的机器上加工(工序可在多个机器上加工的 Jobshop 称为柔性 Jobshop)，问题的目标是在满足作业的各个工序间的顺序约束前提下，决定在同一机器上加工的各工序的加工顺序，这是一种具有复杂约束的多队列排序问题，传统的方法是把 Jobshop 调度问题描述为析取图的形式，图中的结点为工序，同一作业的工序间以实线有向弧连接，弧的方向代表工序的加工顺序(硬约束)，在同一机器上加工的工序以虚线相连，调度的方法就是给虚线边一个方向，成为有向弧，弧的方向表示工序的加工顺序。问题的一个可性解就是为每个机器指定在此机器上加工的工序间的虚线方向，但不能形成回路[142, 143]。这种方法的一个主要缺点是难以保证解的可行性。

Colorni 等[140]首先提出了把 Jobshop 问题映射为解构造图的方法，并提出了基于此解构造图的解决 Jobshop 问题的 AS 蚁群算法。der Zwaan 和 Marques[141]采用了类似的方法，还在算法中引入了类似于遗传算法变异算子的机制，并在一组 Benchmark 问题上进行仿真测试，还对算法参数的不同组合进行统计分析。Blum 和 Sampels[100, 144, 145]提出了一种能处理 Jobshop 与 Open shop 混合形式的调度问题的蚁群算法，他把这种问题称为

Group shop 问题，该问题把 Jobshop 问题中作业的工序扩展为工序组，即多个要处理的子工序，同一工序组中的子工序间没有加工顺序约束。Jobshop 问题可看为 Group shop 问题的一种特例。

所有这些研究者在利用蚁群算法求解 Jobshop 问题时，都利用了一种类似于 Jobshop 问题析取图的解构造图来确定在同一机器上加工的各工序顺序，同时又能保证解的可行性。解构造图上的结点代表工序，所有结点两两相连组成完全连接图(另有一虚拟起始结点，该结点只与代表各作业第一个工序的结点两连，弧的方向为从虚拟起始结点出发)，此外，代表同一作业的工序以有向弧连接，作为作业加工顺序的硬约束，而其他结点间的弧无方向。解构造过程中，蚂蚁从虚拟起始结点出发，此时，其可行的邻域结点为各作业第一个工序，蚂蚁按照分布在弧上的信息素指引，移动到其邻域中的某个结点。在解构造的每一步，蚂蚁访问过的结点从其邻域中删除，同时，如果蚂蚁当前所处的位置不是作业的最后一个工序，则把该作业的下一个工序加入蚂蚁的可行邻域中。在蚂蚁访问过所有结点后，解构造过程结束，蚂蚁所经过的弧按其移动的路径标明方向，这就相当于确定了 Jobshop 问题析取图上的虚线弧方向，同时上述解构造过程能够保证析取图上不会形成回路，即保证了解的可行性。这样，解决 Jobshop 问题的蚁群算法解构造过程转变为蚂蚁在 Jobshop 解构造图上的从虚拟结点出发的结点遍历问题。

然而，上述对 Jobshop 蚁群算法的研究发现，算法性能与最好的算法比较而言还有一定的差距，Blum 和 sampels[145]研究了其中的原因，发现作为解空间参数化概率分布模型的信息素模型的不恰当导致了很强的解空间搜索时的方向偏差，即蚂蚁在基于 Jobshop 析取图上旅行时，总是倾向于在访问其他作业的工序前访问同一作业的后续工序，这样构造出的解往往质量不高，分布在图上的信息素虽然能够记忆蚂蚁在解空间探索的经历，但作为模型参数的信息素与解质量的相关性不强。

4.3 问题描述与析取图表示

本章所考虑的问题的描述如下：共有 L 个工件，需要在 K 台机器上加工。这些工件属于同一个工件族，每个工件的加工由 J 个操作组成。符号 i, j 和 k 分别代表了工件的序号、工件某一操作的序号和机器的序号。令 $C_{max} = \max\{C_1, C_2, \ldots, C_L\}$，为调度的最大完成时间，其中 C_i 代表工件 i 最后一个操作的完成时间，并有 $1 \le i \le L$。晶圆制造过程可以描述为带重入流的 Job shop 模型，其评价指标通常为工件的生产周期（Cycle time），对于全局问题而言，其优化目标是最小化最大完成时间 $C_{\max}$（Makespan）。由文献[85]可

知，获得优化 $C_{\max}$ 的同时可以获得优化的 $L_{\max}$。

上述问题可以被表示为一个数学规划问题，如下所示：

参数：

p_{ijk}：工件 i 的第 j 个操作的加工时间，该操作在机器 k 上加工；

b_k：机器 k 的容量，如果 k 是单件价格机器，则 b_k 等于 1；如果 k 是批处理加工机器，则 b_k 等于机器 k 一个批次中所能加工的最大工件数；

M：一个足够大的正数

决策变量：

r_{ijk}：工件 i 的第 j 个操作的开始时间，该操作在机器 k 上加工；

σ_{ikt}：二进制变量，用来表示在时刻 t 工件 i 是否在机器 k 上加工，σ_{ikt} 的值定义如下：

$$\sigma_{ikt}=\begin{cases}1, & \text{如果工件}i\text{在时刻}t\text{在机器}k\text{上加工},\\ 0, & \text{其他}\end{cases}$$

$\delta^k_{ij,mn}$：二进制变量，用来定义两个操作的顺序，$\delta^k_{ij,mn}$ 的值定义如下：

$$\delta^k_{ij,mn}=\begin{cases}1, & \text{如果由机器}k\text{加工的工件}i\text{的第}j\text{个操作比工件}m\text{的第}n\text{个操作先加工}\\ 0, & \text{其他}\end{cases}$$

根据以上的定义，调度问题表示如下：

Objective: Minimize C_{max}

s.t.

$$r_{i,j+1,k(j+1)}-r_{i,j,k(j)}\geq p_{i,j,k(j)} \qquad \forall\, i,j \tag{4-1}$$

$$\begin{aligned}\delta^k_{mn,ij}\cdot p_{ijk}-M\cdot\delta^k_{ij,mn}\cdot(1-\delta^k_{mn,ij})&\leq r_{mnk}-r_{ijk}\\ &\leq M\cdot\delta^k_{mn,ij}\cdot(1-\delta^k_{ij,mn})-\delta^k_{ij,mn}\cdot p_{mnk} \qquad \forall\, k,i,j,m\neq i,n\neq j\end{aligned} \tag{4-2}$$

$$\sum_i \sigma_{ikt}\leq b_k \qquad \forall\, k,t \tag{4-3}$$

$$\sum_k \sigma_{ikt}=1 \qquad \forall\, i,t \tag{4-4}$$

$$C_{\max}-r_{ijk}\geq p_{ijk} \qquad \forall i,j,k \tag{4-5}$$

$$r_{i,1,k}\geq 0 \qquad \forall i,k \tag{4-6}$$

约束(4-1)表示工件 i 的第 j 个操作完成后才能开始第(j+1)个操作。约束(4-2)确定了要在同一台机器上加工的操作间的加工顺序。约束(4-2)也同时表明同一批次中的操作应该在同一时间开始加工，在这种情况下，$\delta_{ij,mn}$ 和 $\delta_{mn,ij}$ 都等于 0，所以约束可以简化为 $r_{mnk} - r_{ijk} = 0$。约束(4-3)确保了在一台机器上加工的操作数量不能超过该机器的容量 b_k。约束(4-4)确保每个工件的每个操作都只被加工一次。约束(4-5)指出每个工件最终都必须加工完成。约束(4-6)指出没有工件可以在开始时间前进行加工。该问题的每一个可行解决方案被称为一个调度，而求解该调度问题的难点在于约束(4-2)，(4-3)与(4-4)。

我们使用析取图方法来分析该调度问题。如图 4-1 所示，析取图由节点、合取弧（实线）与析取弧（虚线）组成。节点 $O_{i,j,k}$ 表示工件 i 的第 j 个工件在机器 k 上加工。合取弧表示两个操作确定的加工顺序约束。必须在同一台机器上加工的操作由无方向的析取弧两两连接。一个可行的调度就是确定图中所有析取弧的方向，从而获得一个非循环的方向图[146, 147]。

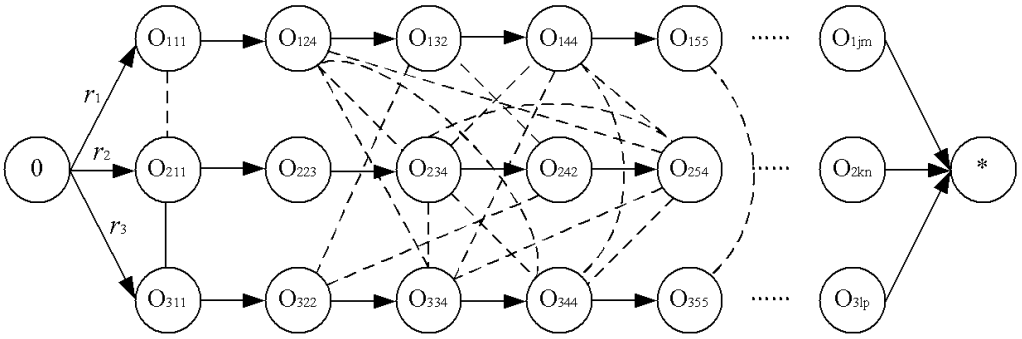


图 4-1 晶圆制造系统的析取图表示

Fig. 4-1 The disjunctive graph of semiconductor manufacturing system

4.4 基于时间的调度问题分解方法

4.4.1 问题分解流程

导致晶圆制造系统调度问题的规模庞大的一个主要因素是其加工周期长，为此，基于时间维度对其进行分解，每次只处理一个较小时间段内的操作，可以有效地降低问题的难度，易于处理。同时也要考虑混合加工对调度问题带来的困难，为此沿着时间轴将原调度问题分解为若干较小的子问题后，对于每一个子问题针对机器的类型进行分解，从而可以较好地处理混合机器类型的问题。这种针对机器类型进行的分解是将在同一台

机器上加工的操作看作一个组，采用分类蚁群算法对每个组进行调度，以避免其他组的工件所造成的干扰。所有操作组的调度在同一个框架下进行。本章主要考虑基于时间分解的方法，同时也兼顾基于机器类型的分解，并提出分类蚁群算法以保证调度工作的进行。

问题分解方法的主要困难在于处理子问题之间的耦合。耦合度越高，分解的困难就越大，调度的困难度也相应增大。一般来讲，基于时间的问题分解方法可以分为两类：并行分解方法与串行分解方法。基于并行分解的调度方法由两个相分离的阶段组成：分解阶段与调度阶段。在分解阶段，原问题被分解为若干个规模相当的子问题，这些子问题在调度阶段被各自独立地求解，其解决方案被合并为原问题的解。并行分解方法可以同时处理多个子问题，因而效率较高，但是子问题的解之间耦合度较高，合并为原问题的解较为困难，因为需要进行大量的协调工作，如果子问题调度方案间的冲突无法解决，所涉及的子问题必须重新求解。串行分解方法每次迭代过程只从原调度问题上分解出一个子问题，等该子问题调度结束后再考虑其他的子问题。串行分解方法的优点是较容易由单个子问题的解合成原问题的解。在串行分解过程中，前一个问题的解可以作为后一个调度问题的约束存在，由此可保证操作调度顺序的可行性。相比并行分解方法而言，串行分解方法的计算效率较低，但是在协调上所需要的时间也较少。为避免问题的复杂性，本章采用“单向解耦”[148]的方式进行问题分解，即一类串行分解方法。所谓“单向解耦”是指子问题之间只在一个方向上存在耦合关系，即子问题 β 只依赖于子问题 α，在此情况下，可以先对问题 α 求解，然后再对 β 求解，即可获得原问题的解。在析取图所描述的生产过程模型中，调度要素不是机器或工件，而是以“操作（operation）”的形式存在。一个操作相当于一个工件在一台机器上的一次加工。对于一个工件而言，其所有操作形成一个按严格时间顺序排列的单向序列，整个系统从而可以看作由若干单向序列所组成的序列集。将整个系统的调度问题按照时间轴分解为若干子问题，每个子问题仅包含当前区域内的若干操作，问题的规模可以用时间区间来衡量，也可以用所包含的操作数量来衡量。由于子问题之间的时间顺序依赖关系，因而按照时间轴对各子问题依次求解可以保证原问题解的合理性。其分解流程如图 4-2 所示。

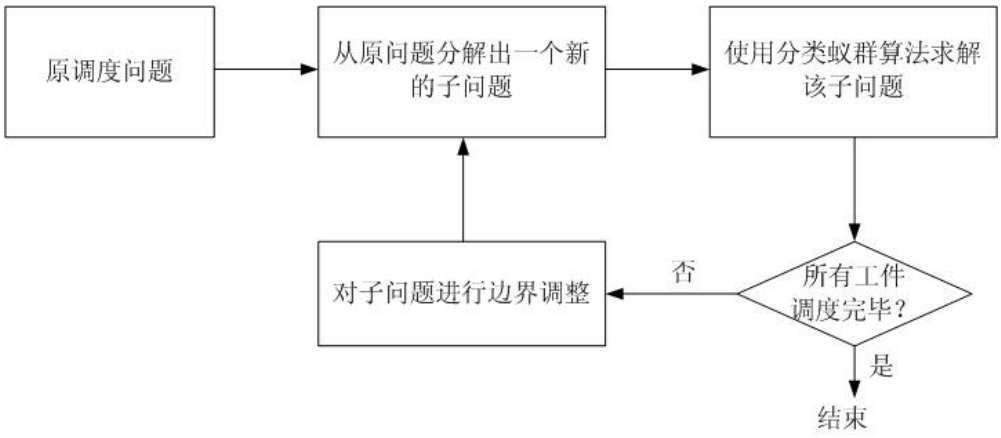


图 4-2 问题分解流程

Fig. 4-2 The procedure of decomposition method

4.4.2 子问题构建

基于时间的问题分解方法通常使用两个参数来控制子问题的规模：分解周期 T 与子问题包含的操作数 N_s。本章使用 N_s 作为主要分解控制参数，这是因为蚁群算法求解时间对所需要调度的操作数非常敏感。基于时间的问题分解方法如下所示：

算法 4-1：基于时间的问题分解过程

步骤 1：（初始化）

$\Omega = \Omega \cup V$

$\omega = \emptyset$

步骤 2：（生成子问题）

While $|\omega| \leq N_s$-1,

在 Ω 中寻找投料时间 $r_{i,j,k}$ 最小的操作 $O_{i,j,k}$

$\omega = \omega \cup \{O_{i,j,k}\}$

$\Omega = \Omega \setminus \{O_{i,j,k}\}$

End While

步骤 3：（调度子问题）

使用分类蚁群算法调度子问题 ω

步骤 4：（子问题边界调整）

运行算法 4-2

在算法 4-1 中，Ω 是原调度问题中所有未被调度的操作的集合，ω 是分配到当前子问题的操作集。V 包含了上一个子问题与当前子问题间的重叠操作，这些操作被合并到当前子问题再次进行调度求解，以确保相邻子问题调度方案的连续性。

在步骤 1 中，将上一个子问题的重叠操作移除，并作为未调度操作并入 Ω，此时当前子问题中未包含任何操作，所以 ω 应设置为空集。步骤 2 首先计算子问题集合 ω 的秩，以确定其是否超过了子问题的操作数限制 N_s，如果没有超过，则意味着该子问题尚未选择足够的操作，则选中 Ω 中带有最小到达时间 $r_{i,j,k}$ 的工件 $O_{i,j,k}$，将其从 Ω 中移除，并入当前子问题 ω。如果超过了 N_s，则中止选择操作的过程，并执行下一步骤。在步骤 3，对已获得的子问题进行调度，其具体步骤见 4.4 节的分类蚁群算法。在步骤 4，对已经调度完成，获得调度方案的子问题执行协调机制，即对当前子问题进行边界调节，这一步骤在下一节详述。

4.4.3 子问题协调机制

对于已调度完成的子问题需要进行协调，即调整问题的边界。通过这一步骤，一些操作将从当前子问题移除并放入下一子问题，这些操作被称为重叠操作，因为它们属于两个相邻的子问题。之所以要对调度完成的子问题进行边界调整，有如下几个原因：首先，通过边界调整可以获得更好的调度结果，因为考虑了更多的操作，从而使得一些操作间的优先关系约束得到了松弛。例如，要调度一个含有 80 个操作的子问题，有两种方法。第一种是通过算法 4-1 的步骤 1-3 选择 80 个操作组成子问题，对其进行调度。另一种方法是选择 100 个操作组成子问题并进行调度，但是只保留调度方案中的前 80 个操作。相比前一种方法，第二种方法考虑了更多的操作，从而可以获得较好的调度方案。进行边界调整的第二个原因是可以保证下一个子问题中的各工件的开始时间分布在一个相近的时间范围内。在上一个子问题调度完成后，将在同一台机器操作之间被加入了优先约束，并且在某些操作前添加了等待时间，这将导致该子问题中不同工件的完成时间差异很大，所以下一子问题中的不同工件的开始时间也会有很大的差异。边界调整过程将调度结果中最后的若干操作操作移到下一个子问题，以保证下一子问题中的工件在近似相同的时间开始。进行边界调整的最后一个原因是可以帮助批处理机器找到更好的调度方案，因为可以使得这类机器克服操作数量的约束。在调度过程中，批处理机的最后一个批次可能由于没有足够的操作数而不能获得较优的调度结果。例如，一个批处理机的最大组批数量为 6，但是在子问题中对最后一个批次组批时只有 2 个操作可选，其结果是最后一个批次只能将这剩余的 2 个操作组批加工。通过边界调整过程，则可以将

这最后的 2 个操作从当前子问题移出，放到下一个子问题中，这样批处理机就可以在下一个子问题里考虑更多的操作，从而获得更优的调度方案。

对已调度完成的子问题进行调整的算法 4-2 如下：

算法 4-2：对调度完成的子问题进行边界调整的过程

步骤 1：（初始化）

$V = \emptyset$

步骤 2：（选择重叠操作）

While $|V| \le N_s \cdot f_a - 1$

在 ω 中选择具有最大投料时间 $r_{i,j,k}$ 的操作 $O_{i,j,k}$

$V = V \cup \{O_{i,j,k}\}$

$\omega = \omega \setminus \{O_{i,j,k}\}$

End While

步骤 3：（处理批处理机操作）

对所有操作包含在 V 中的机器 k 重复下列步骤

If $O_{i,j,k} \in \omega$, $O_{l,m,k} \in V$, $O_{i,j,k}$ 和 $O_{l,m,k}$ 属于同一个个批次，则

$V=V \cup \{O_{i,j,k}\}$

$\omega=\omega \setminus \{O_{i,j,k}\}$

End If

在算法 4-2 中，V 是包含重叠操作的集合，步骤 1 对算法初始化，定义 V 的初始状态为空集。步骤 2 逐步选择操作添加进集合 V 中，每次从已调度好的子问题 ω 中选择具有最大投料时间 $r_{i,j,k}$ 的操作 $O_{i,j,k}$，将其从 ω 中移到集合 V 中，该操作即被定义为重叠操作。这个操作是子问题 ω 的调度方案中最后一个进行加工的操作。令 f_a 为调整因子，用来确定重叠操作数量占整个子问题中操作数量的比率。例如，若 V 中应该包含 10 个操作，而子问题的规模 N_s 是 100，则 f_a 等于 10%。通过一个判断式来确定集合 V 是否包含足够的重叠操作，若操作数量不足，则反复执行步骤 2，直到满足 V 所需的操作数。步骤 3 协调批处理机的调度，对存在集合 V 中的每一个批处理操作所在的机器 k 进行检测，如果在步骤 2 的调整过程中把属于同一批次的操作分成了两部分，一部分属于子问题 ω，另一部分属于集合 V，则将属于子问题 ω 的那部分操作转移到集合 V 中。

4.5 分类蚁群算法

调度问题可以看作对析取图中的每一个团寻找一个不循环路径的过程。一个团是由在同一台机器上加工的操作所组成。为了应对晶圆制造系统的混合加工特征，我们使用

基于机器类型的分解方法以避免不同类型机器的干扰。不同的机器类型所使用的调度方法也不同，所以应用基于机器类型的分解方法是一种自然的选择。

虽然蚁群算法已经被应用于多种调度问题，但是还没有其应用于混合机器类型的记录。为此，我们提出分类蚁群算法以应对晶圆制造系统的混合加工特征，其所使用的变量定义如下：

令 s_{ib} 表示迭代最优调度；s_{gb} 表示全局最优调度；f_{con} 为收敛因子，其用来表示算法收敛的程度。G_k 是在机器 k 上加工的所有操作的集合。M'是在时刻 t_0 空闲的机器集合。G_k'是 G_k 的子集，其包含时刻 t_0 是等待在机器 k 上加工的操作。na 是人工蚂蚁的数量，s_a 是人工蚂蚁 a 所构建的子问题调度方案。t_{next} 表示时刻 t_0 后下一个即将到达操作的到达时间。N_{rs} 代表算法重启的次数。当 N_{rs} 超过了一个预定的最大重启次数 $N_{rs,\max}$，则算法终止。S 是所有单件加工机器的集合，B 是所有批处理加工机器的集合。分类蚁群算法 4-3 如下所示：

算法 4-3：分类蚁群算法

步骤 1：（初始化）

$s_{ib} = \text{Ø}$, $s_{gb} = \text{Ø}$, $f_{\text{con}} = 0$

为每个机器 k 构建 G_k

信息素初始化

步骤 2：（为人工蚂蚁构建路径）

For $a = 1$ to na, do

$s_a = \text{Ø}$

While (操作未全部调度完成) do

生成空闲机器集合 M'

为 M'中的每一个机器 k 构建 G_k'

For $k = 1$ to $|M'|$, do

If $k \in S$

通过迁移概率公式(4-7)在 G_k'中选择一个操作 $O_{i,j,k}$

将操作 $O_{i,j,k}$ 添加到路径 s_a 中

Else If $k \in B$

If $|G_k'| = b_k$

将 G_k'中的所有操作构建为一个批次

Else If $|G_k'| > b_k$

$q = 0$

While $q < b_k$ do

通过迁移概率(4-8)在 G_k'中选择一个操作 $O_{i,j,k}$
将 $O_{i,j,k}$ 添加到当前批次中
$q = q + 1$
$G_k' = G_k' \setminus \{O_{i,j,k}\}$
End While
Else If $|G_k'| < b_k$
If (通过公式(4-9)确定要等待下一个到达工件 O_{next})
$G_k' = G_k' \cup \{O_{next}\}$
Else If
将 G_k'中的所有操作构建为一个批次
End If
End If
将该批次加入路径 s_a
End If
End for
$t_0 = t_{next}$
End while
End For
Step 3. (信息素更新)
$s_{ib} = \arg\min (C_{\max}(s_1), \ldots, C_{\max}(s_{na}))$
对 s_{ib} 采用局部搜索算法，以获得一个更好的解
If $C_{\max}(s_{ib}) < C_{\max}(s_{gb})$
$s_{gb} = s_{ib}$
End If
使用公式(4-14)计算 f_{con}
If $f_{\text{con}} < 0.9$
使用 s_{ib} 更新信息素
Else If $0.9 \le f_{\text{con}} < 0.99$
使用 s_{gb} 更新信息素
Else If $f_{\text{con}} \ge 0.99$
$N_{rs} = N_{rs} + 1$
If $N_{rs} \le N_{rs,\max}$
$s_{ib} = \emptyset$
初始化信息素
Else If
终止算法
End If
Goto Step 2
End If

算法 4-3 扩展了传统最大最小蚁群算法（MMAS, Max-Min Ant System）[117]的使用范围。与传统 MMAS 算法不同，分类蚁群算法考虑了混合加工的特征。针对不同类型的机器采用不同的迁移概率公式。针对批处理机器，考虑了三种不同的情况，即当等待的工件数小于、等于或大于机器最大容量。针对上述的每一种情况，采用相应的批处理调度策略。此外，算法 4-3 还使用了一种新的局部搜索方法，可以成功地应用于混合制造系统。

步骤 1 对算法所需的参数进行初始化，并将操作按照其所需加工的机器进行了组合，另外对信息素值也进行了初始化。对信息素值的初始化遵循两个规则。首先，信息素值只施加于在同一个机器上加工的操作之间，这是因为在一台机器上要加工的操作是从所有需要在该机器上进行加工的操作集合中选择的；其次，信息素的初始值被设定为信息素值的上限 $\tau_{\max}$，结合一个较小的蒸发系数 ρ，这样可以在算法最开始的若干迭代过程中增加寻找更优路径的概率，因为这样的信息素初始设定可以延长信息素蒸发过程，从而使得不同操作间的信息素差异缓慢增加。

步骤 2 为每一个人工蚂蚁 a 构建寻优路径。M'包含时刻 t_0 所有的空闲机器。在每次迭代的开始，M'的值都会被更新，并为每一个机器 k 创建在其上加工操作的集合 G_k'。在当前迭代步骤中，算法从集合 G_k'中选择一个操作，接着在新的迭代步骤中，新的 M'和 G_k'被创建，迭代过程持续进行直到子问题中的所有操作都被调度完毕。

对不同类型的机器，采用不同的选择待加工操作的方法。如果机器 k 是单件加工机器，则采用下式从 G_k'中选择一个操作进行加工：

$$P(O)=\begin{cases}\dfrac{(\min_{O'\in G_k',O'\neq O}\tau_{O,O'})^{\alpha}\cdot(\eta_O)^{\beta}}{\sum_{O''\in G_k}(\min_{O'\in G_k',O'\neq O''}\tau_{O'',O'})^{\alpha}\cdot(\eta_{O''})^{\beta}}, & \text{如果 } O\in G_k' \\ 0, & \text{其他}\end{cases} \tag{4-7}$$

在公式(4-7)中，信息素值 $\tau_{O,O'}$越大，则操作 O 在 O'之前加工的可能性越高。操作 O 所在的工件剩余加工时间被选择作为启发式信息 η_O。式(4-7)中的指数参数 α 和 β 的值表征了信息素和启发式信息的相对重要性。如果 $\alpha = 0$，人工蚂蚁则不通过信息素进行交流，而变成了基于调度规则的随机贪婪搜索算法。如果 $\beta = 0$，则人工蚂蚁不再利用启发式信息。所以，信息素浓度与启发式信息之间的平衡需要进行细致的衡量。

如果机器 k 是一个批处理加工设备，则需要考虑三种可能的情况：如果在当前时刻 G_k'中的操作数量等于机器 k 的容量 b_k，则将 G_k'中的所有操作组合成一个批次进行加工；如果在当前时刻 G_k'中的操作数量大于机器 k 的容量 b_k，则需要从 G_k'中选择 b_k 个工件进行组批加工，工件的选择通过公式(4-8)进行：

$$P(O)=\begin{cases}\dfrac{\left(\sum\limits_{O'\in G_k',O\neq O'}\tau_{O,O'}\right)^{\alpha}\cdot(\eta_O)^{\beta}}{\sum\limits_{O''\in G_k'}\left(\sum\limits_{O'\in G_k',O''\neq O'}\tau_{O'',O'}\right)^{\alpha}\cdot(\eta_{O''})^{\beta}}, & \text{如果 } O\in G_k' \\ 0, & \text{其他}\end{cases} \tag{4-8}$$

最后，如果在当前时刻 G_k'中的操作数量小于机器 k 的容量 b_k，则通过公式(4-9)来确定是否要等待下一个即将到达的操作 O_{next}。如果确定要等待下一个工件，则当前时刻 G_k'中的工件不进行组批，而是等到 O_{next} 到达后将其包含进 G_k'，并将 G_k'的到达时间设定为新的当前时刻，重复利用公式(4-9)进行判断。如果(4-9)确定不等待工件 O_{next}，则当前在 G_k'中的操作立即组批加工。

$$P(O_{next})=\begin{cases}\dfrac{(\min_{O\in G_k'}\tau_{O,O_{next}})^{\alpha}\cdot(\eta_{O_{next}})^{\beta}}{[(\min_{O\in G_k'}\tau_{O,O_{next}})^{\alpha}\cdot(\eta_{O_{next}})^{\beta}]+[(\min_{O,O'\in G_k'}\tau_{O,O'})^{\alpha}\cdot(\eta_{O}')^{\beta}]}, & \text{如果 } t_{next}<t_0+p_k \\ 0, & \text{其他}\end{cases} \tag{4-9}$$

其中：

$$\eta_{O_{next}}=t_0+p_k-t_{next} \tag{4-10}$$

$$\eta_O'=|G_k'|\cdot(t_{next}-t_0) \tag{4-11}$$

在公式(4-8)中，信息素值 $\tau_{O,O'}$代表将操作 O 与 O'放入同一个批次的期望，操作 O 所在的工件剩余加工时间被选择作为启发式信息 η_O。

在公式(4-9)中，未来到达工件 O_{next} 的信息，包括其到达时间 t_{next}，都可以通过晶圆制造系统中的制造执行系统（Manufacturing Execution System, MES）获得。t_0 是当前时刻，p_k 是机器 k 的加工时间。信息素值 $\tau_{O,O'}$代表将操作 O 与 O'放入同一个批次的期望，操作的等待时间作为启发式信息。

如果下一到达工件的到达时间大于当前时间加上机器的加工时间，即 $t_{next}\geq t_0+p_k$，意味着下一个工件将在一个加工循环之后到达，则等待下一到达工件的概率 $P(O_{next})$为 0，即 G_k'中的工件不必等待下一工件的到来，而是应该在当前时刻便组批加工。如果下一到达工件的到达时间小于当前时间加上机器的加工时间，即 $t_{next}<t_0+p_k$，则 O_{next} 将在一个加工循环内到达，如图 4-3 所示：

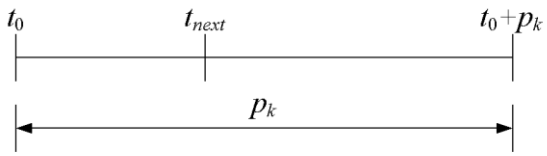


图 4-3 组批启发式信息的图示

Fig. 4-3 An illustration for heuristic information of batching.

在这种情况下，如果 G_k'中的工件在当前时刻进行组批加工，则工件 O_{next} 必须要等待($t_0 + p_k - t_{next}$)长的时间才能进行调度。相反的，如果等待 O_{next} 到达，则所有现在存在于 G_k'中的操作必须要等待($t_{next} - t_0$)长的时间，即总体等待时间为$|G_k'|\cdot(t_{next}-t_0)$，如公式(4-11)所示。因此，是否等待工件 O_{next} 的概率受这两个等待时间的影响，如公式(4-9)所示。

等当前迭代完成后，将 t_{next} 设置为 t_0，从而开始一轮新的迭代。

步骤 3 对信息素进行更新。通过步骤 2 对每一只人工蚂蚁寻找到了一条路径，其最好的路径被定义为迭代最优路径 s_{ib}，并通过局部搜索方法进行进一步寻优。本章所采用的是 Nowicki 和 Smutnicki 所提出的局部搜索方法[149]的改进版本。局部搜索方法在迭代最优路径 s_{ib} 的邻域中进行搜索，以获得更好的路径。s_{ib} 的邻域定义如下：s_{ib} 的一条关键路径 C 被分割为若干子序列 $B_1, \ldots, B_r$，这些子序列被称为块（block）。B_k 包含在机器 k 上加工的一组连续操作，两个相邻块中的操作在不同的机器上加工。如果将 s_{ib} 的关键路径 C 中的两个操作的位置按照公式(4-12)所示的规则进行交换，所获得的新的操作序列 s_{ib}'位于 s_{ib} 的邻域中。

$$V(s_{ib})=\begin{cases}\left\{\left(O_{k,1},O_{k,2}\right),\left(O_{k,n_k-1},O_{k,n_k}\right)\right\}, & \text{如果 } k\in S\\ \left\{\left(O_{k,m},O_{k+1,n}\right)\right\}, & \text{如果 } k,k+1\in B, m(k)=m(k+1)\\ \varnothing, & \text{其他}\end{cases} \tag{4-12}$$

B_k 包含 n_k 个操作，依序为 $O_{k,1},O_{k,2},\ldots,O_{k,n_k-1},O_{k,n_k}$，$n_k \geq 2$。如果机器 k 是单件加工机器，则操作序列的头两个操作或最后两个操作位置互换，以获得一个新的操作序列。如果两个连续的块 B_k 和 B_{k+1} 在同一台机器上加工，即 $m(k) = m(k+1)$，则从每一个块中随意选择一个操作进行互换。

如果局部搜索方法获得了一个更好的路径 s_a*，有 $C_{\max}(s_a^*) < C_{\max}(s_a)$，则用 s_a*代替 s_a 作为迭代最优路径，并继续对 s_a*进行新的局部搜索，这个过程一直持续直到找不

到更好的路径为止。当 s_{ib} 优化结束后，如果 $C_{\max}(s_{ib}) < C_{\max}(s_{gb})$，则用迭代最优路径 s_{ib} 代替 s_{gb} 成为新的全局最优路径。

信息素的更新规则如公式(4-13)所示，

$$\tau_{ij} = (1-\rho)\tau_{ij} + \Delta\tau_{ij}^{best}, \tag{4-13}$$

其中： $\Delta\tau_{ij}^{best} = 1/C^{best}$.

如果机器 k 是单件加工机器，则调度方案中连接相邻工件的信息素给更新。如果机器 k 是批处理加工机器，则只有属于同一批次的操作间的信息素被更新。

信息素更新过程中所采用的最优路径可以是全局最优路径 s_{gb}，即 $C^{best} = C^{gb}$；也可以是迭代最优路径 s_{ib}，即 $C^{best} = C^{ib}$。选择不同的最优路径会导致不同的结果。当使用全局最优路径 s_{gb} 时，搜索过程会很快地收敛，寻找更好路径的可能性较低，很有可能会使得搜索收敛于较差的路径。使用迭代最优路径 s_{ib} 可以获得更好的路径，可以搜索更大的空间，但是算法收敛时间较长。两种路径各有有点，为此，我们选择混合搜索策略。迭代最优路径 s_{ib} 被用于前期迭代，以寻找更多的路径，增加获得更好路径的可能。全局最优路径 s_{gb} 则用于后期迭代，以加快算法的收敛过程。收敛因子 f_{con} 被用来确定什么时候使用哪一个最优路径更新信息素，其计算公式如下：

$$f_{\text{con}} = \frac{\sum_{k\in M_{\text{sub}}, O_i \neq O_j} \max(\tau_{\max} - \tau_{O_i,O_j}^{k}, \tau_{O_i,O_j}^{k} - \tau_{\min})}{|M_{\text{sub}}| \cdot (\tau_{\max} - \tau_{\min})} \tag{4-14}$$

其中：M_{sub} 是该子问题所涉及的机器的集合。

从公式(4-14)可知，当所有的信息素值都等于最大信息素 $\tau_{\max}$ 或最小信息素 $\tau_{\min}$ 时，f_{con} 等于 1。如果 $f_{\text{con}} < 0.9$，则采用迭代最优路径 s_{ib} 更新信息素。如果 $0.9 \le f_{\text{con}} < 0.99$，则采用全局最优路径 s_{gb} 更新信息素。在这两种情况下，算法更新完信息素后将返回步骤 2 以在新的信息素环境下为人工蚂蚁寻找新的路径。如果 f_{con} 等于或大于 0.99，则算法陷入了停滞，此时需要重新初始化信息素后重启算法。如果算法重启次数 N_{rs} 大于最大重启次数 $N_{rs,\max}$，则算法将被终止，此时的全局最优路径 s_{gb} 即子问题的解。

信息素的上界 $\tau_{\max}$ 设置为 $1/\rho C^{gb}$。如果寻找到了一个更好的路径，则 $\tau_{\max}$ 也要同步进行更新。信息素下界 $\tau_{\min}$ 设置为 $\tau_{\max}/\gamma$，其中参数 γ 是信息素上界与下界间的比例，我们设置 $\gamma = 2N_s$，N_s 为子问题包含的操作数。当信息素按照上面所属的更新规则更新完毕后，将对其数值进行检查，如果超过了最大信息素值或小于最小信息素值，则需要对其校正，使之不超出规定范围之外。

4.6 本章小结

基于时间分解的蚁群算法突破了传统蚁群算法受制于问题规模的求解能力，可以用来解决大规模复杂的调度问题，并取得良好的结果。其缺点是运算时间依然较长，只适合解决静态调度的问题。但是蚁群算法较适合采用并行计算，因而下一步研究的目标是增强蚁群算法的并行计算能力，研究其并行计算机制，并在静态调度的基础上，增强即时重调度的能力，以实现在一定范围内进行动态调度。

第五章 基于机器类型分解的蚁群调度算法

5.1 引言

晶圆制造系统的多种加工方式混合加工加剧了调度问题的复杂程度。如前所述，根据加工操作方式的不同可以将晶圆加工机器划分为五大类，即单件处理设备（Single-type Processing Machine, SPM），多件处理设备（Batch-type Processing Machine, BPM），管道型处理设备（Piping-type Processing Machine, PPM），群集处理设备（包括多相同处理室和多不同处理室）（Multi-Chamber Processing Machine with Same Chambers / Different Chambers, MPM-SC / DC），和 WET-BENCH 型处理设备（BPM 与 PPM 的结合体）。其中，SPM 型设备和 MPM-SC / DC 型设备每次只能载入一个 lot 的晶圆进行加工；WET-BENCH 型设备每次可以载入两个 lot 的晶圆进行加工；BPM 型设备和 PPM 型设备则可同时加工多个 lot 的晶圆。为简便起见，上述五种晶圆加工设备可分为单件加工与批处理加工两类设备，SPM、MPM-SC / DC 与 PPM 三种设备为单件加工设备，WET-BENCH 与 BPM 设备为批处理加工设备，其中 WET-BENCH 型设备的最大加工容量为 2。

在晶圆制造系统中，单件加工设备的数量较多，但是批处理设备由于加工周期长，每次可加工多件设备，因而常常是制造系统中的瓶颈设备，所以其重要性也较高。问题分解的一个原则是将一个大问题分解为若干小问题，选择其中最重要的问题优先解决。因此本章的侧重点在于将调度问题按照类型分类后，重点研究批处理机器加工的蚁群算法，这部分研究目前在学术界刚刚展开。对于单件加工设备，则重点考虑其面对多重入特征时的处理方法。第四章重点是讨论基于时间分解的蚁群算法，但同时也提出了基于机器类型的分类蚁群算法，不过对于其处理的单个机器的蚁群算法采用了简化处理，本章则讨论在基于机器类型分解的架构下，调度过程考虑上下游机器的协调机制。

本章首先讨论基于机器类型分解的架构，提出考虑加工上下游的协调机制；然后讨论面向多重入特征的单件加工机器的蚁群算法应用，提出面向多重入的蚁群算法；接着重点研究批处理加工设备的蚁群算法应用，针对单台批处理设备与并行多台批处理设备提出两类混合蚁群算法；最后通过实验对所提出的算法进行验证。

5.2 基于机器类型的调度问题分解与协调

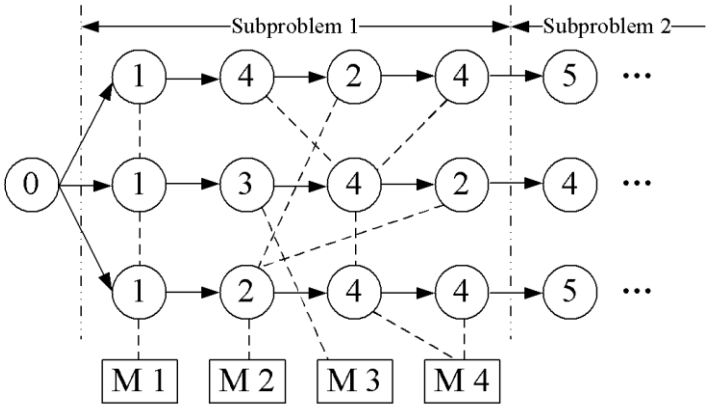


图 5-1 基于时间与基于机器类型的混合分解过程

Fig. 5-1 Time-based and machine-type-based hybrid decomposition procedure

图 4-1 所示的析取图被简化为图 5-1，其中节点上的数字代表加工该操作所需要的机器。图中的每一个节点都有时间与机器类型两个属性，在调度过程中二者都应该被考虑。

现代晶圆制造系统自动化程度很高，生产执行系统（MES）中存在大量的制造信息可被利用。从这些信息中不仅可以获得当前加工机器的信息与状态，而且可以获得其上下游机器的状态信息，这为我们更好地配置工件加工顺序，以保证生产线平衡提供了条件。

对于每一个机器，我们可以对其生产状态进行定义，取出若干关键指标，以衡量该机器的运行情况是否健康。这些指标通常包括在制品水平、瓶颈设备载荷和库存容量等，使用指标集 $MI = \{MI_1, MI_2, \ldots, MI_n\}$。我们为每个指标设定一个合理的范围，当其目前指标值在该范围内时，则其相应的指标 MI_j 设定为 1，当其指标值偏离该范围时，根据其偏离方向对机器所造成的影响进行加权。若该指标的偏离表明机器当前需要加工的工件过多，库存接近满荷，则该指标 $MI_j < 1$，以传达信息，当前不要把更多的工件分配到该机器。反之，若该指标的偏离表明机器当前需要加工的工件较少，呈现饥饿状态，则该指标 $MI_j > 1$，以将信息传递给调度系统，使之尽快将工件传送来。

同时，对于每一个工件，也定义若干关键指标，以表示其当前状态。这些状态包括其当前工序需加工时间、已等待时间、加工紧急程度、下一工序所在机器的状态与重要度等，使用指标集 $JI = \{JI_1, JI_2, \ldots, JI_n\}$表示。这些指标同样被预先设定一个范围，当其

当前值落在这个范围内时，该指标值设定为 1。若指标有所偏离，若其偏离方向表明该工件迫切需要进行加工，则该指标 $JI_i < 1$；反之则令该指标 $JI_i > 1$。

对当前正在加工的机器及工件指标确定后，可以将其信息放入蚁群算法中，以影响调度过程。

5.3 面向多重入特征的单件加工设备调度蚁群算法

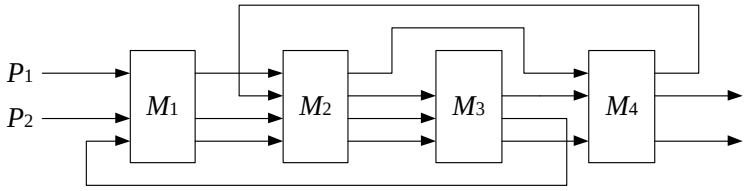


图 5-2 晶圆制造系统典型重入制造过程

Fig. 5-2 The re-entrant process of semiconductor manufacturing system

晶圆制造系统是典型的多重入制造系统，“重入”（Re-entrant）是指产品在制造过程中访问同一台机器 2 次及其以上的特征。如图 5-2 所示，两类产品（P_1, P_2）在四台机器（M_1, M_2, M_3, M_4）上进行加工，其反复进入其中的若干机器，以完成加工过程。一个制造系统，重入的机器数量越多、重入越频繁，则越复杂[3]。多重入特征导致在制造系统中，不仅不同类产品之间要竞争同一机器，而且即使属于同一类但是处于不同阶段的产品也要竞争同一台机器；这增加了问题的复杂度。对于不同的调度目标，处于不同重入阶段的工件应该采用不同的调度策略。例如，对于与产出率相关的调度目标，处于后期重入阶段的工件应该获得占用机器的较高优先度，这样可以尽量快地提升产出产品的数量；而对于与加工周期相关的调度目标，则应该根据生产线状况，实时调整位于不同加工阶段工件的优先级，以保证整个生产的平衡（如图 5-3 所示）。

基于重入的蚁群算法即在保证整体生产线平衡的基础上，对等待在该机器前的工件，根据其所处的不同重入阶段，分配相应的权重，以保证获得较好的调度方案。

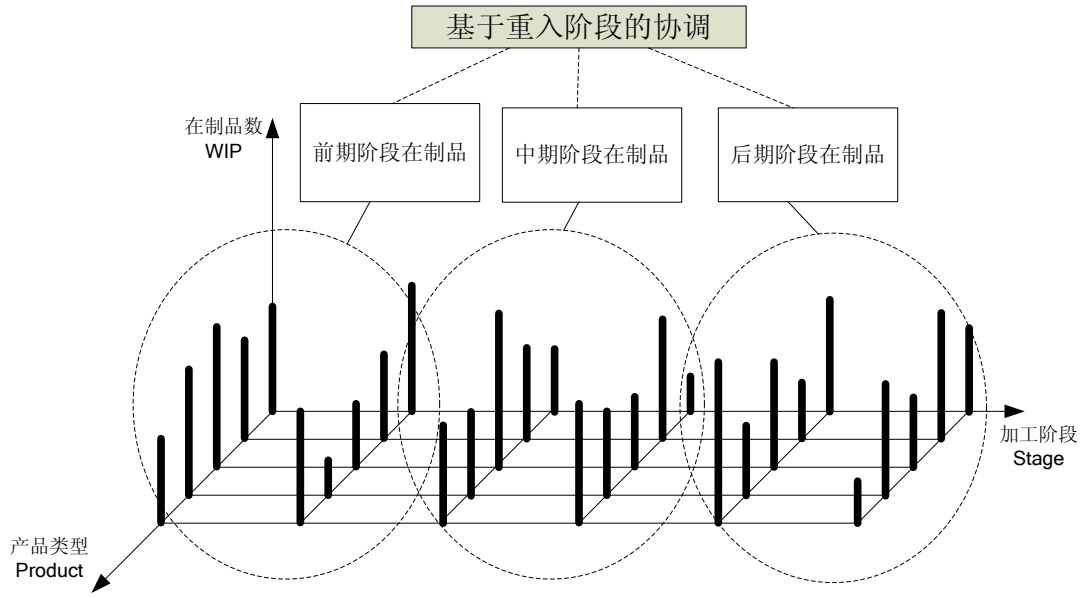


图 5-3 基于重入阶段的协调

Fig. 5-3 The coordination based on re-entrant stage

RT_{ik} 是工件 i 在所调度的机器 k 上重入的总次数，RC_{ik} 是工件 i 当前在机器 k 上重入的次数，RS_i 是工件 i 当前所处的重入阶段，$RS_i = RC_{ik} / RT_{ik}$。OC_k 是由当前调度目标与整体生产线状态所确定的机器 k 应采用的调度策略，OC_k 是二进制数，其值定义如下：

$$OC_k = \begin{cases} 1, & \text{如果当前状态下位于前期重入阶段的工件应获得较高优先级} \\ 0, & \text{如果当前状态下位于后期重入阶段的工件应获得较高优先级} \end{cases} \tag{5-1}$$

RH_i 是工件 i 在当前状态下调度的启发式信息

$$RH_i = \begin{cases} RS_i, & \text{如果 } OC_k = 0 \\ 1 - RS_i, & \text{如果 } OC_k = 1 \end{cases} \tag{5-2}$$

将工件的重入状态信息加入蚁群算法中，则机器 k 在当前等待工件中进行选择的迁移概率为：

$$P(O) = \begin{cases} \dfrac{(\min_{l \in G_k', l \neq i} \tau_{i,l})^{\alpha} \cdot (\eta_i)^{\beta} \cdot (RH_i)^{\gamma}}{\sum_{m \in G_k} (\min_{l \in G_k', l \neq m} \tau_{m,l})^{\alpha} \cdot (\eta_m)^{\beta} \cdot (RH_m)^{\gamma}}, & \text{如果 } i \in G_k' \\ 0, & \text{其他} \end{cases} \tag{5-3}$$

其中 G_k'是在机器 k 前等待工件的集合，i, l, m 是 G_k'中的工件，$\tau_{i,l}$ 是启发式信息，信息素值 $\tau_{i,l}$ 越大，则操作 i 在 l 之前加工的可能性越高；η_i 是机器 i 的启发式信息。

5.4 批处理加工设备调度蚁群算法

批处理加工设备主要存在于化学气相沉积(Chemical vapor deposition)、扩散(Diffusion)和氧化(oxidation)等制程中。相对于单件加工工序 1-2 小时的平均加工时间，批处理加工所需时间更长(10 小时以上)。批处理加工的工件属于不同的种类(family)，具有不同的权重(weight)、交货期(due date)和到达时间(arrival time)，一个批次(batch)由数个相同种类的工件(lot)组成，工件的数量不能超过机器的容量(capacity)。加工一旦开始则不能中断，批次中的工件也不可增减。此外，生产调度工作还必须满足不同产品的交货期(due date)，否则会造成工件的拖期(Tardiness)，从而降低客户满意度。

批处理调度可分解为两个阶段：组批与调度。前者将工件按照一定的规则组成批次，后者将这些批次分配到机器上进行加工。当考虑工件动态到达的情况时，如果在某一时刻当前等待的工件数量小于批处理机的容量时，就面临两种选择：将当前工件组成一个批次进行加工以提高机器利用率；或者等待更多工件到达后再组批以提高满批率。这种选择的不确定性给组批工作带来了困难。近年来有一些学者逐步将蚁群算法应用于批处理机的调度研究，但大部分研究着眼于将蚁群算法用于后一阶段，即批次的调度，而组批过程则采用传统的调度规则方法获得，这无疑限制了蚁群算法的应用和所取得调度的精度。为此，本章提出两种基于组批的混合蚁群算法，分别用来解决单台批处理设备加工问题与多台并行批处理设备加工问题，以扩展蚁群算法在批处理机调度问题中的应用范围。

5.4.1 批处理设备调度研究概述

批处理设备调度问题可以描述如下：在单机调度的情况下，有 n 个工件$\{J_1, J_2, \ldots, J_n\}$需要加工，每个工件 J_j 的加工时间为 p_j，到达时间为 r_j，交货期为 d_j（可能 $r_j \equiv 0$ 或/和 $d_j \equiv \infty$）。加工最大批量为 B。某一批次中加工时间最长的工件决定了该批次的加工时间。需要研究 n 个工件应该如何组批，各批次如何排序，以使指定的目标函数值最小。在并行机调度的情况下，假设有 m (≥ 2)台机器，每个工件可自由选择在其中某台机器上加工。工件 J_j $(1 \leq j \leq n)$在机器 M_i $(1 \leq i \leq m)$上加工时所需加工时间为 p_{ij}，工件的其他参数与单机调度情况相同。需要为每个工件选择一台加工机器，及对同一台机器上的工件进行合理的组批与排序，使得给定的目标函数值最小。并行机又有以下三种类别：（1）每台机器完全相同，即对每一个工件均有 J_j $(1 \leq j \leq n)$均有 $p_{1j} = p_{2j} = \ldots = p_{mj}$，成为并行相同机（identical parallel machines）；（2）每台机器 M_i 有一个固定的加工时间比例 s_i，使得 $p_{ij} =$

p_j / s_i $(1 \le j \le n, 1 \le i \le m)$，称为并行同类机（uniform parallel machines）；（3）上述两种关系都没有时，称为不相关并行机（unrelated parallel machines）。

批处理调度打破了经典调度问题中一台机器在同一时间只能处理一个工件的约束，是经典调度问题的一种推广（令 $B = 1$），因而其难度不低于经典调度问题，即如果某个调度问题是 NP-Hard 的，则其相应的组批调度问题也是 NP-Hard 的，即使其最大批量 B 为常数。事实上，学者们已经证明了大多数组批调度问题都是 NP-Hard 问题，因而没有多项式时间的精确算法。

从所调度工件种类的角度出发，批处理调度可分为对单一品种工件的调度与对多品种工件的调度。其中多品种工件还可分为多种不兼容工件与多种兼容工件两种情况。不兼容工件不能放入同一个批次加工，而兼容工件可以组批加工，其批次加工时间等于最长的那类工件加工时间。另外，在某些情况下，不同类型工件的尺寸也存在差异，这时要保证一个批次中所有工件的体积和不能超过批处理机的容量。本文从以上几个方面总结相关的研究情况。

（1）单一品种工件的调度研究

批处理机调度领域最早的一篇文献是 Ikura 和 Gimple 在 1986 年的论文[150]，该文章考虑单台批处理加工设备加工单一品种工件的情况，在投料时间与交货期一致（即投料期早的交货期也早）的假设下，提供了确定是否存在满足所有工件交货期的调度的方法。Lee 等[151]针对这个问题提出了一个可选的动态规划方法，其调度目标是最小化最大延迟 $L_{\max}$。同时他们也提出了一个算法以最小化拖期工件的数量 $\sum U_i$。Li 和 Lee[39]证明了当到达时间和交货期不一致时，这些问题都是强 NP-hard 问题。他们也扩展了 Lee 提出的算法，以求解到达时间、交货期和加工时间都一致的情况。Ahmadi 等[152]研究一个包含批处理机和单件处理机的流水线调度问题，优化目标是最小化平均完成时间和 $C_{\max}$，他们提出了一个多项式时间算法求解一部分问题，而对其他问题采取启发式方法。Glassey 和 Weng[19]研究了面向工件动态到达的单批处理机单产品调度问题，优化目标是最小化工件的平均等待时间。他们假定未来的 L 个到达工件信息已知，提出了动态组批规则（Dynamic Batching Heuristic）。该法以加工周期作为计划期，当机器空闲时，若等待晶圆的个数大于或等于机器的最大加工批量，则立即开始加工；否则，计算立即开始加工或等待其后若干个工件到达后加工的情况下，所有工件的总等待时间，选择总等待时间最小的时刻开始加工。

（2）多品种不兼容工件族的调度研究

Fowler 等[153]扩展了 Glassey 和 Weng[19]的模型，用来解决多个不兼容工件族和多个机器的问题，他们提出了 NACH 调度规则（Next Arrival Control Heuristic），在每一个新

工件到来时，则重新产生调度方案。NACH 有两类策略：工件到达时采用“PUSH”策略、机器空闲时采用“PULL”策略。在多个不兼容工件批次间，选择加工时间最短的优先加工。Ham 和 Fowler[154]拓展了 Fowler 等[153]所提出的 NACH(Next arrival control heuristic)启发式规则，提出了 NACH+调度系统，使得待加工的批次能够准时到达，以减少等待时间。Weng 与 Leachman[155]提出最小成本率(Minimum Cost Rate . MCR) 规则，通过降低等候队列内等候加工工件的数目，使工件单位时间的等待成本最小，该方法先根据所有可能加工点的最小成本率为每个产品找到最优加工时间，然后利用总最小成本率来选择产品。Chandra 和 Gupta[156]研究批处理调度问题中在机器容量约束与不兼容工件族的约束下最大化产出。他们把这个问题构建为一个多约束箱子包装问题并提供了启发式方法。在所有工件都只需要占用一个单位容积的情况下，Uzsoy 和 Yang[157]提供了一种分支定界法和启发式方法以最小化总加权完成时间$\sum w_i C_i$。Lee 和 Uzsoy[158]研究了最小化 $C_{\max}$ 的这类批处理问题，并考虑到工件动态到达。他们对一些特殊情况提供了精确算法，并提出了一些启发式方法，通过计算实验进行了验证。Mönch 等[62]研究了多产品、工件动态到达的并行批处理设备的调度问题，其优化目标为最小化总加权拖期，文章提出了两种混合遗传算法，获得了良好的效果。梁静等[101]利用启发式信息对工件组批，应用蚁群算法对这些批次进行调度，解决了多品种晶圆连续到达动态环境下非等效并行多批处理机的调度问题。

（3）多品种兼容工件族的调度研究

Lee 等[151]针对并行相同批处理设备，提供了最小化 $C_{\max}$ 问题的一个较差误差界限（tight worst-case error bound）的启发式方法。Chandru 等[159]针对并行相同批处理机器组成的系统，给出精确算法和启发式方法，以最小化总完成时间$\sum C_i$。在另一篇文章[160]中，他们针对该问题的一个特殊情况，即有限的加工时间种类，给出了一个多项式时间的动态规划算法。Brucker 等[161]在机器容量无限的假设下给出了几个精确算法和复杂性结果，对于有限的情况，他们证明基于交货期的调度准则会导致 NP-hard 问题，并且提供了一个复杂度为 $O(n^{B(B-1)})$的精确算法以最小化$\sum C_j$。Wang 和 Uzsoy[60] 考虑动态工件到达情况下的单批处理机调度问题，其优化目标是最小化最大延迟。他们选择动态规范算法来确定是否存在一个交货期可行的组批方案，然后结合基于随机 key 的遗传算法对问题进行求解。Li 等[162]提出一种基于蚁群算法的多目标调度方法，同时考虑工件在并行批处理机上加工的生产周期与总加权拖期最优化。

（4）多品种不相同尺寸工件族的调度研究

Dobson 和 Nambinadom[163]研究了单台批处理机，不兼容工件族，且每一个工件需要不同的机器空间的调度问题，优化目标是最小化总加权完成时间$\sum w_i C_i$。他们提供了

一个整数规划方程，用来获得优化值的一个下界。他们同时证明了这个问题是 NP-hard，并提供了一些启发式方法。Hochbaum 和 Landy[164]提供了一个更有效的动态规划方法，同时提出了一种启发式方法对带有任意加工时间的问题求解最小化$\sum C_i$指标。Uzsoy[165]针对单台批处理设备，工件具有不同尺寸的调度问题，提出了一种分支定界法和启发式方法，使得$C_{\max}$和$\sum C_i$为最小。Azizoglu 和 Webster[166]考虑调度带有不兼容工件的单台批处理设备，工件具有任意尺寸和权重，调度目标是最小化总加权完成时间。他们所采用的方法是分支定界法，这个算法可在合理时间内获得最多 25 个 job 问题的优化解。Sevaux 和 Peres[167]首先引入了遗传算法调度带有不同尺寸工件的批处理机，其优化指标是最小化加权延迟工件数$\sum U_i$。Koksalan[168]利用遗传算法对该问题进行多目标优化，所优化的目标函数为加权延迟工件数$\sum U_i$和最大提前期。Melouk 等[61]开发了一种模拟退火方法以最小化带有不同加工时间和尺寸的工件在单机批处理设备上的$C_{\max}$。Koh 等[169]提出了一些启发式算法和一个基于 Random key 的 GA 方法，以求解带有不兼容工件和不相等工件尺寸的批处理生产问题，其优化目标是$C_{\max}$和$\sum w_j C_j$。Cheng 等[170]提出一种混沌蚁群算法，用来解决工件具有不同尺寸和加工时间的单机批处理设备调度问题，以最小化生产周期。

5.4.2 单台批处理设备调度的混合蚁群算法

5.4.2.1 问题描述

在本部分，我们构建了扩散操作中的单台批处理设备的调度问题，并考虑不兼容工件族（即在该台设备上有多个加工菜单）和工件动态到达的问题。调度目标是最小化总加权拖期时间。问题描述如下：

1）本文考虑在一个调度区间（例如一个班次、一天或若干天）上的调度问题。在此调度区间上调度计划已确定，在各个时间段所需要调度的工件也已确定。在实际工厂运营过程中，所需要调度的工件信息可以通过历史数据和制造执行系统（Manufacturing Execution System, MES）的运行数据获知。

2）使用同一个加工菜单的可以同时加工。但是同时加工的工件数不能超过批处理加工设备容量的上限，即要满足最大批次约束。

3）一个批次的加工时间与该批次所包含的工件数无关，只与该批次的加工菜单有关。

4）一旦一个批次开始加工，在加工完成前其中的工件不能移除，也不能增加新的工件

5）在更换不同工件族工件的时候存在顺序依赖随机设置时间，但同一工件族工件更换不需要设置时间。

5.4.2.2 算法描述

（1）参数初始化

蚁群算法的搜索空间由所有工件组成，即所有的工件作为析取图的点，而每对工件间的联系作为析取图的弧。在由析取图所组成的蚁群算法构建图中，每个弧上的初始化信息素被设定为一个小的正数，如下所示：

$$\tau_{ij}=\begin{cases}\varepsilon, & \text{如果} i\ j\text{属于同一个工件族}\\ \varepsilon/2, & \text{如果} i\ j\text{属于不同的工件族}\end{cases} \tag{5-4}$$

信息素 τ_{ij} 代表两个工件 i 与 j 之间的关系。同一个工件族中的工件相对于不同工件族的工件关系更为密切。信息素也同时代表了将工件 j 与工件 i 组为同一批的期望。

（2）构建单只人工蚂蚁搜索路径

从一个空的调度序列出发，蚁群算法利用析取弧上的信息素和启发式信息逐步将工件加入当前批次中。如果工件 i 是最新加入当前批次的工件，则增加另一个工件 j 进入当前批次的概率由公式(5-5)计算：

$$P_{ij}=\frac{(\tau_{ij})^{\alpha}\cdot\eta_j^{\beta}}{\sum_{k\in N_p}[(\tau_{ik})^{\alpha}\cdot\eta_k^{\beta}]} \tag{5-5}$$

其中 N_p 是未加入已调度批次的工件集；η_j 是工件 j 的启发式信息，它代表了当前加工工件 j 的迫切程度。η_j 由 CR 调度规则确定，其计算公式如(5-6)所示；α 和 β 分别代表了信息素与启发式信息的相对重要程度。

$$\eta_j=\frac{t_{R,j}}{d_j-t} \tag{5-6}$$

其中 d_j 是工件 j 的交货期，t 是当前时间，$t_{R,j}$ 是工件 j 的剩余加工时间。

通过式(5-5)选择工件 j 后，仍然需要通过若干规则来决定该工件是否应该加入当前批次还是需要建立一个新的批次。第一个规则是满批规则，即如果当前批次已经达到了最大所能容纳的工件数，则工件 j 应该加入一个新建立的批次。第二个规则是相容性规则，如果工件 j 和工件 i 不属于同一个工件族，则工件 j 不能加入当前批次。第三个规

则是从 NACH（Next Arrival Control Heuristic）规则引导出来的。NACH 提出了在当前时刻存在未满批次的情况下，是继续等待下一个到达工件还是加工此未满批次的判断规则。对每一个即将到达的工件 j，计算等待其到达的净拖延时间，如下所示：

$$S_1 = q \times (A_j - A_i) \tag{5-7}$$

$$S_2 = T + A_i - A_j \tag{5-8}$$

$$K = S_1 - S_2 \tag{5-9}$$

公式(5-7)描述了由于等待工件 j 所造成的拖延时间，其中 q 是当前批次中的工件数量，A_i 和 A_j 分别是工件 i 和工件 j 的到达时刻。公式(5-8)描述了工件 j 所节省的时间。K 是由于等待工件 j 所造成的净拖延时间。如果 $K > 0$，则意味着等待工件 j 会造成增加总的拖延时间，所以不应该等待工件 j，当工件 j 到达后应该重新建立一个新的批次。上述的描述可以通过图 5-4 来表示。

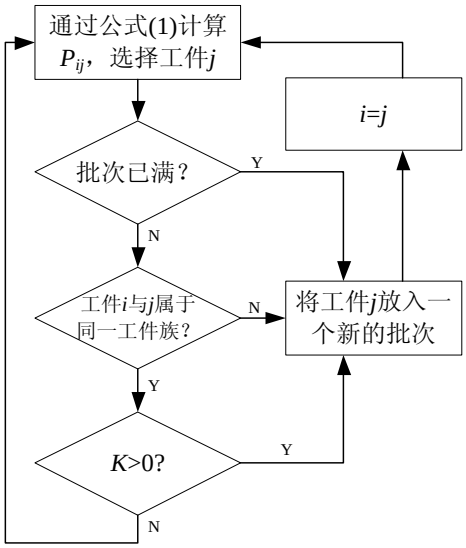


图 5-4 组批逻辑流程图

Fig. 5-4 Flow chart of batching logic

（3）更新信息素

计算每只人工蚂蚁所获得调度结果的目标值，目标值最小的调度结果设定为 S_{best}。如果当前最佳调度结果的目标值与现存最佳调度结果目标值之差不大于一个小的正数（记为 δ），则终止搜索过程。接下来根据最优调度结果更新相应的信息素，如下所示：

$$\Delta\tau_{xy}=\begin{cases}1/\min\limits_{k}\sum_i w_iT_i, & \text{如果 } (x,y)\in s_{best}\\ 0, & \text{其它}\end{cases} \tag{5-10}$$

$$\tau_{xy}(t+1)=(1-\rho)\tau_{xy}(t)+\Delta\tau_{xy},\quad 0<\rho<1 \tag{5-11}$$

其中 ρ 是析取弧上信息素的蒸发系数。

5.4.3 并行批处理设备调度的混合蚁群算法

5.4.3.1 问题描述

本文研究工件在多个相同批处理机组成的并行制造系统中的加工调度问题，其特征描述如下：

（1）有 n 个工件，属于 f 种产品，属于产品 j 的工件数量为 n_j，$\sum_{j=1}^{f} n_j = n$；

（2）有 m 个相同的批处理机器，可加工上述 f 种产品，对应 f 种工艺菜单;

（3）只有同一种类的工件可以组批加工，一个批次中工件的数量不能超过机器的容量 B;

（4）一台机器同一时刻只能加工一批工件，加工时间与该批工件的数量无关，只与工件种类所对应的工艺菜单有关;

（5）工件 i 的权重为 w_i，交货期为 d_i;

（6）工件动态到达，工件 i 的到达时间为 r_i，可以通过晶圆制造系统中的 MES 系统(Manufacturing Execution System)获知;

（7）机器一旦开始加工，则不能被中断，即不允许抢占。

调度分为两个阶段，首先将同一类型的工件组成合适的批次，然后将这些批次分配到合适的机器上加工，并确定加工顺序和开始时间，以使得所关注的性能指标最小。本文以总加权拖期时间最小为调度目标，即：

$$\min\ F=\sum_{i=1}^{n} w_iT_i$$

其中工件拖期 $T_i = w_i\,(C_i - d_i)^+$，C_i 为工件 i 的完工时间，$(C_i - d_i)^+ = \max\,[(C_i - d_i),\quad 0]$。

$\alpha\,|\,\beta\,|\,\gamma$ 表示法常用来描述调度问题，其中 α 表示机器加工环境，β 表示工件的加工特性和数据，γ 表示加工性能指标，使用该表示法本文所研究的问题可描述为

$Pm|r_i,batch,incompatible|\sum w_iT_i$，其中 *incompatible* 表示不同种类的工件不能组批加工。

5.4.3.2 算法描述

1. 最大最小蚁群算法

蚁群算法[108]模拟蚁群寻找通往食物源最短路径的信息交换机制，被广泛用来解决组合优化问题。在这种机制下，每只蚂蚁在寻找路径的过程中，释放一定数量的信息素(pheromone)，影响其后蚂蚁的寻径行为。这种影响通过路径评价得到强化，形成一个正反馈环，通过蚂蚁之间的协同而获得最佳路径。本文应用蚁群算法的并行搜索机制进行工件组批，采用最大最小蚁群算法(MAX-MIN Ant System, MMAS)[117]，该方法有以下几个特点：

（1）信息素的值限制在区间[$\tau_{\min}$, $\tau_{\max}$]中，以防止因信息素过大或过小使算法陷入停滞;

（2）只有获得最优路径的蚂蚁可以释放信息素;

（3）信息素的初始值被设定为最大值 $\tau_{\max}$，并配合一个较小的信息素蒸发系数，这使得在初始阶段各操作之间的信息素差异不大，可以探索更多的搜索路径。

2. 蚁群组批算法

组批算法采用事件驱动模式，在任一批处理机器空闲的时刻 t，对各类产品分别进行组批操作。对于任一类产品 j，此时有 n_j^t 个工件在等待加工，这些工件组成集合 M_j^t。按以下方法对产品 j 进行组批操作：

(1) 如果 $n_j^t = B$，即当前等待的工件数量等于机器加工的最大批量，则将此 n_j^t 个工件组成一个批次;

(2) 如果 $n_j^t > B$，即当前等待的工件数量大于机器加工的最大批量，则在其中选择 B 个工件组成一个批次。其中工件 k 被选中组批的概率由下式确定：

$$P(k)=\begin{cases}\dfrac{\left(\sum\limits_{l\in M_j^t,l\neq k}\tau_{kl}\right)^{\alpha}\cdot(\eta_k)^{\beta}}{\sum\limits_{q\in M_j^t}\left(\sum\limits_{l\in M_j^t,l\neq q}\tau_{lq}\right)^{\alpha}\cdot(\eta_q)^{\beta}}, & \text{如果}k\in M_j^t\\ 0 & \text{其他}\end{cases} \tag{5-12}$$

上式中，τ_{kl}为信息素，表示工件 k 与工件 l 组为一批的期望，$\tau_{kl}=\tau_{lk}$，信息素表征了 k 与其他工件关系的密切程度，密切程度越高的两个工件被放在一批加工的可能性越大；与其他工件密切程度较高的工件有较大的概率被选中。η_k为启发式信息，表示该工件优先加工的紧迫程度，本文选用该工件的权重作为启发式信息，即$\eta_k=w_k$；α 与 β 是参数，用来确定信息素和启发式信息在组批概率中的影响力。

(3) 如果$n_j^t<B$，即当前等待的工件数量小于机器加工的最大批量，则面临两种情况的选择：在当前时刻将n_j^t个工件组成一批进行加工，或等待更多工件到达后再组批加工。前者可提高机器利用率，但由于批次中工件较少，因而工件平均加工成本较高；后者可降低工件平均加工成本，但是机器会有一定的等待时间。机器选择等待下一个工件 k 到达后再组批的概率为：

$$P(k)=\frac{\left(\sum\limits_{l\in M_j^t}\tau_{kl}\right)^{\alpha}\cdot(w_k)^{\beta}\cdot\left[n_j^t\cdot\exp(r_k-t)^{+}\right]^{-\gamma}}{\left(\sum\limits_{l\in M_j^t}\tau_{kl}\right)^{\alpha}\cdot(w_k)^{\beta}\cdot\left[n_j^t\cdot\exp(r_k-t)^{+}\right]^{-\gamma}+\left(\sum\limits_{l\in M_j^t}\sum\limits_{q\in M_j^{t'},q\neq l}\tau_{lq}\Big/n_j^t\right)^{\alpha}\cdot\left(\sum\limits_{l\in M_j^t}w_l\Big/n_j^t\right)^{\beta}} \tag{5-13}$$

其中：$M_j^{t'}=M_j^t\bigcup\{k\}$，$\gamma$ 是表示工件等待时间所造成影响的系数。

上式的分母由两部分组成，第一部分表示下一个到达工件 k 对当前等待工件的影响力，第二部分表示当前等待工件的平均影响力；分子则表示工件 k 的影响力。由上式可知，工件 k 对当前等待工件的信息素与启发式信息值越高、等待时间越短、当前等待工件越少，则被选中组批的概率越大。

如果 k 未被选中加入当前工件批，则将n_j^t个工件组批；如果 k 被选中，且$n_j^t+1=B$，则将当前n_j^t个工件和 k 一起组批；如果 k 被选中，且$n_j^t+1<B$，则令$M_j^t=M_j^{t'}$，$n_j^t=n_j^t+1$；$t=r_k$，继续利用公式(5-13)判断是否加入下一到达工件。

通过上述过程为每类产品组批，共获得 f'个不同类工件的待加工批次，其中 $f' \le f$。

3. 加工批次选择

当所有类工件组批完成后，从中选择一个批次进行调度。在时刻 t，计算 f'个不同种类工件批次的优先级，对于产品 j 的批次 B_{bj}，其优先级 I_{bj} 为：

$$I_{bj}(t)=\sum_{i=1}^{n_{bj}}\left(\frac{w_i}{p_j}\right)\cdot\exp\left(-\frac{\left(d_{bj}-p_j-t+\left(r_{bj}-t\right)^+\right)^+}{K\cdot\bar{p}}\right)\cdot\frac{n_j^t}{B} \tag{5-14}$$

其中： n_{bj} 是批次 B_{bj} 的工件个数;

$d_{bj}=\min_{i\in B_{bj}}(d_i)$，即取批次中最紧迫工件的交货期作为整个批次的交货期;

$r_{bj}=\max_{i\in B_{bj}}(r_i)$，取批次中最后到达工件的到达时间作为整个批次的到达时间;

K 是比例参数， $\bar{p}$ 是其余批次的平均加工时间。

具有最大优先级的产品批次被选择进行加工。

4. 混合蚁群算法描述

综合以上两节，进行批处理调度的混合蚁群算法描述如下：

（1）初始化算法，确定蚂蚁数量 n_a，终止条件，以及初始信息素，设定迭代最优路径 S_{ib} 和全局最优路径 S_{gb} 为空，收敛因子 f_{con} 设定为 0;

（2）每一只蚂蚁 a ($1 \le a \le n_a$)按照图 5-5 所示流程构建路径 S_a;

（3）从所有蚂蚁构建的路径中选择本次迭代最优路径 S_{ib}，将 S_{ib} 与全局最优路径 S_{gb} 比较，如果 S_{ib} 比 S_{gb} 更优，则令 $S_{gb}=S_{ib}$;

（4）根据收敛因子 f_{con} 的值选择 S_{ib} 或 S_{gb} 更新信息素，并更新 f_{con}，如果新计算的 f_{con} 超过一个极限值，则回到(1)重启算法;

（5）如果满足算法的终止条件，停止计算并输出所得到的最优解；否则回到(2)进行下一次迭代。

每次算法重启后，随着迭代次数的增加 f_{con} 从 0 逐渐趋向于 1，当超过一个极限值后，算法重启，f_{con} 复归于 0。终止条件设定为算法重启次数不超过 N_{rs}。

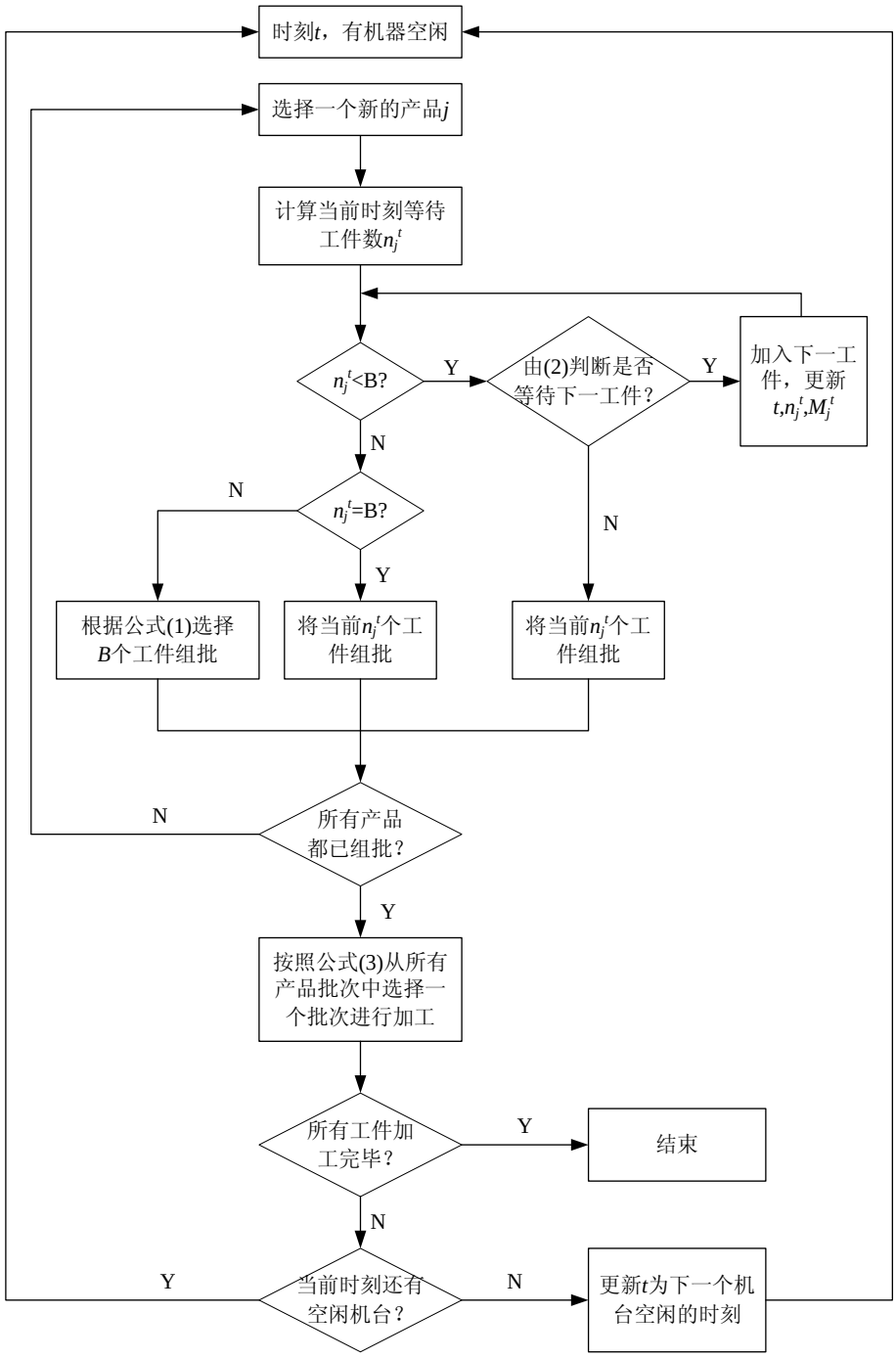


图 5-5 单只人工蚂蚁路径构建流程图

Fig. 5-5 Flowchart of solution construction by a single artificial ant

步骤(4)中的信息素更新规则为：

$$\tau_{ij}=\begin{cases}(1-\rho)\tau_{ij}+\Delta\tau_{ij}^{\text{best}}, & \text{如果}i,j\text{属于同一批次}\\ (1-\rho)\tau_{ij}, & \text{其它}\end{cases} \tag{5-15}$$

其中 ρ 为信息素蒸发系数，$\Delta\tau_{ij}^{\text{best}}=1/C^{\text{best}}$，信息素更新只应用在最优路径上。$C^{\text{best}}$ 是最优路径 S^{best} 的加权总拖期值，S^{best} 可以选择本次迭代最优路径 S_{ib}，也可以选择全局最优路径 S_{gb}。采用 S_{ib} 进行信息素更新可使算法有更好的搜索性，在算法早期配合大的信息素初始值和较小的蒸发系数，可以搜索更多的路径。采用全局路径 s_{gb} 对信息素进行更新，可使算法尽快收敛到最优全局路径，减少运算时间，所以可在算法后期使用。何时采用 s_{ib} 或 s_{gb} 由收敛因子 f_{con} 确定，收敛因子表征了信息素的分布情况，如果工件与同一批中工件间的信息素值很高，而与其他批次中工件间的信息素值很低，则探索新的路径的可能性较小，算法会趋于停滞。收敛因子由下式计算：

$$f_{\text{con}}=\frac{\sum_{i=1}^{n}\tau_i^{\text{diff}}}{n\cdot\left(\tau_{\max}-\tau_{\min}\right)} \tag{5-16}$$

其中：
$$\tau_i^{\text{diff}}=\frac{\sum_{k\in M_{b(i)},k\neq i}\tau_{ik}}{|M_{b(i)}|}-\frac{\sum_{l\in M_j,l\notin M_{b(i)}}\tau_{il}}{n_j-|M_{b(i)}|},\quad \forall i\in M_j \tag{5-17}$$

M_j 是产品 j 所有工件组成的集合，$M_{b(i)}$是工件 i 所在批次所有工件的集合。当 $0<f_{\text{con}}<0.9$ 时，选择本次迭代最优路径 S_{ib} 进行信息素更新，当 $0.9\le f_{\text{con}}<0.99$ 时，选择全局最优路径 S_{gb} 进行信息素更新。

信息素更新完毕后，要对其进行上下限检查。如果某个信息素值大于最大信息素 $\tau_{\max}$，则令其等于 $\tau_{\max}$；同理，如果某个信息素值小于最小信息素 $\tau_{\min}$，则令其等于 $\tau_{\min}$。其中 $\tau_{\max}=1/\rho C^{gb}$，如果在蚁群搜索过程中找到了更好的全局最优路径 S_{gb}，则更新 $\tau_{\max}$；$\tau_{\min}=\tau_{\max}/\varepsilon$，$\varepsilon$ 设定为 $2n$。

5.5 本章小结

基于机器类型分解的晶圆制造系统调度问题需要同时考虑各类加工设备的调度问题，其中最重要的是如何对其中的批处理设备进行调度。这类设备加工时间长，通常是制造系统中的瓶颈机器，对其进行调度是处理晶圆制造系统大规模调度问题的关键环节。其次，还需要应对由于多重入所带来的调度问题的复杂性。本章在分类蚁群算法的框架

下提供基于重入的蚁群算法与批处理机器蚁群算法，并探讨了不同类型设备调度的协调机制。实验证明，所提出来的算法与协调机制可以获得好的调度解。

第六章 实验与分析

6.1 引言

为了应对晶圆制造系统的复杂性特征，本文提出了基于问题分解的蚁群算法，从时间与机器类型两个维度对晶圆制造系统的大规模复杂调度问题进行分解。分解与求解的整体思路是首先利用基于时间的问题分解方法将大规模问题分解为若干较小的子问题，然后再用基于机器类型的问题分解方法将子问题分解为若干单机调度问题，对每一类机器采用相应的蚁群算法进行求解，特别是针对难以调度的批处理机器开发了新的基于蚁群组批的调度算法，提高了问题求解的精度。为了将各单机调度结果合成为子问题的调度，开发了分类蚁群算法，将不同类型的机器调度问题放在同一个调度框架下，通过蚁群算法的信息素进行信息的传递与整合，并结合基于工件加工上下游的协调机制，保证子问题调度结果的质量。由子问题合成原问题的解则是通过单向解耦，并依靠基于重叠区间的协调机制，保证原问题解的获得。为了验证所提出的各类分解方法与蚁群算法，而进行仿真实验，仿真实验的顺序与问题分解的顺序相反。首先验证所提出的批处理设备调度方法的有效性，其次针对一个小的混合机器加工调度问题验证基于机器类型的问题分解方法与分类蚁群算法，最后针对一个大规模调度问题验证基于时间的问题分解方法的有效性。

6.2 基于批处理加工机器调度的仿真实验

6.2.1 单台批处理设备调度的混合蚁群算法仿真实验

针对单台批处理设备调度的混合蚁群算法，采用仿真实验的方法验证其有效性。从真实的晶圆加工厂扩散区选择一台批处理设备，其批次最大工件容量为 4。此外定义五个相关参数：工件数量$|J|$，工件 j 到达时间 A_j，工件 j 交货期 d_j，人工蚂蚁的数量 n_a。参数值如表 6-1 所示：

表 6-1 仿真实验相关参数

参数	参数值
$\|J\|$	20, 40, 60, 80
A_j	Uniform[1,500]
d_j	Uniform[A_j+100, A_j+150]
n_a	10, 20, 30
工件权重	Uniform[0,1]

传统批处理调度规则 MCR 被用作蚁群算法的对比参照算法。蚁群算法的参数设置如下：α=1，β=2.5，ρ=0.2，ε=0.01；调度结果目标值比较的最小差异数 δ 设置为 0.001。最大迭代次数 $t_{\max}$ 设置为 100，人工蚂蚁的数量分别设置为 10,20 和 30。

仿真实验平台用 Visual Basic .NET 开发，运行硬件环境为 Pentium 4 2.4GHz CPU，1024MB 内存。仿真结果见表 6-2，图 6-1 和 6-2 所示。

表 6-2 由仿真实验所获调度方案的总加权拖期值（TWT）

$\|J\|$	MCR	Hybrid ACO		
		n_a=10	n_a=20	n_a=30
20	47.69	52.16	48.27	46.35
40	102.15	98.26	90.31	88.25
60	156.78	136.82	122.43	116.28
80	187.29	155.31	146.10	137.47

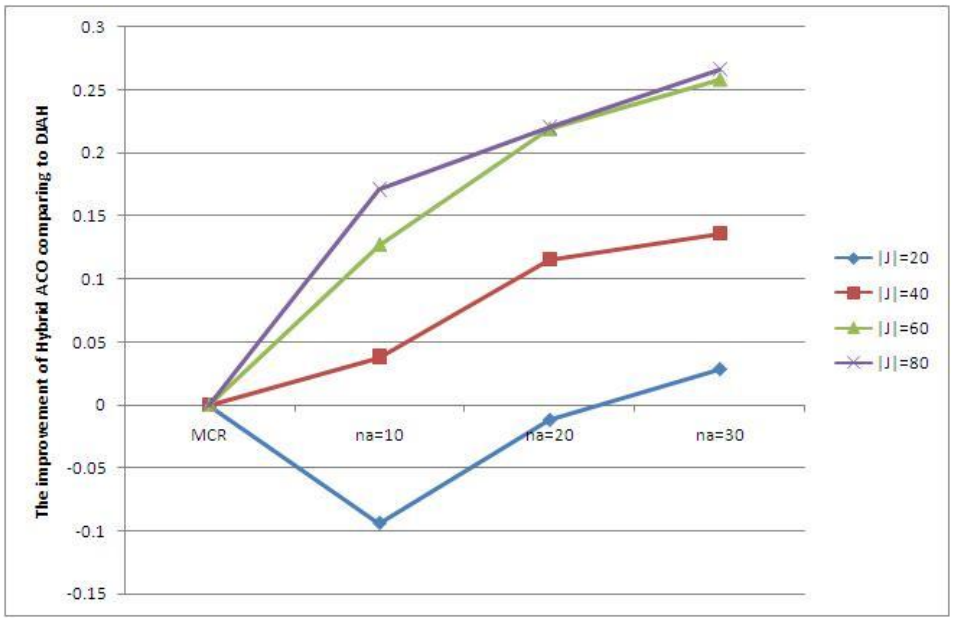


图 6-1 相比 MCR 调度规则混合蚁群算法所得到的性能提升

Fig. 6-1 The improvement of Hybrid ACO method comparing to MCR

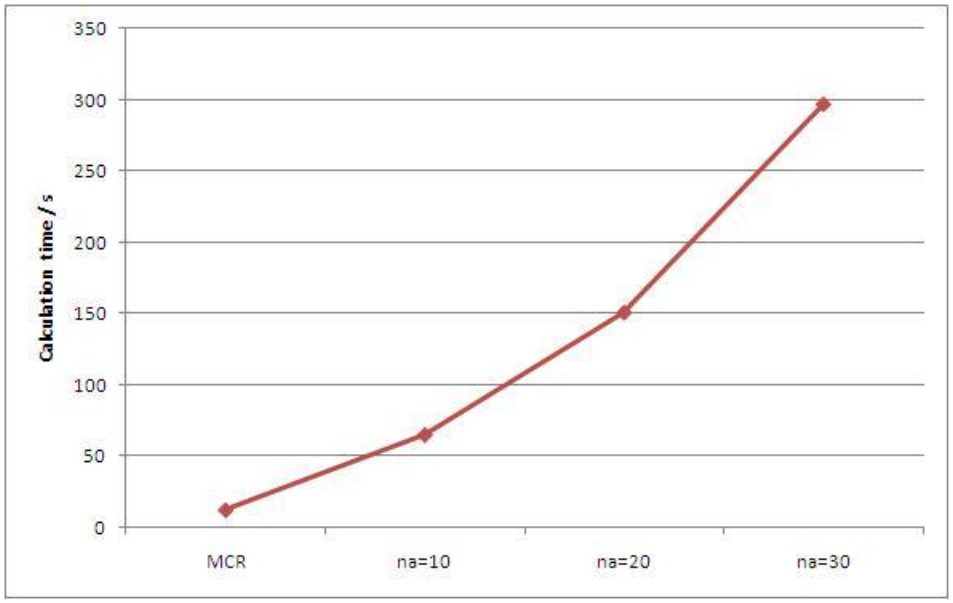


图 6-2 MCR 调度规则与混合蚁群算法所需的平均调度时间

Fig. 6-2 Average scheduling time of MCR and Hybrid ACO

仿真结果显示相比 MCR 调度规则混合蚁群算法在很大程度上提升了调度目标（总加权拖期）的水平。并且随着调度工件数的增加，水平提升更多。当工件总量较少时，

混合蚁群算法相比 MCR 规则没有太大的又是。当工件数较多时，混合蚁群算法所得到的调度方案要优于 MCR。举例来说，当需要调度的工件数分别为 20,40,60 和 80，人工蚂蚁数为 10 的时候，对总加权拖期的提升分别为-9.37%，3.81%，12.73%，和 17.07%。

如果蚁群算法使用更多的人工蚂蚁，则调度结果还能得到进一步的提升。如上面的这个例子，当人工蚂蚁数增加到 30 时，对 20,40,60 和 80 个总调度工件而言，其总加权拖期的提升分别为 2.81%，13.61%，25.83%和 26.60%。然而，人工蚂蚁的数量对计算时间影响很大，如图 5-3 所示，随着人工蚂蚁数量的增加，所需的计算时间也显著增加。但是计算时间仍然在可接受的范围之内，特别是相对于批处理设备本身加工时间的漫长，有足够的计算时间以保证获得更好的调度结果。

6.2.2 并行批处理设备调度的混合蚁群算法实验

针对并行批处理设备调度，采用仿真实验的方法验证所提出的混合蚁群组批算法，并与 ATC-BATC 算法[62]进行比较。ATC-BATC 算法在批处理调度中应用广泛，可以获得高质量的解。算法测试数据按照 Akturk 和 Ozdemir[171]所提出的随机生成方法产生。机器的数量考虑两种情况，分别包含 3 台和 5 台相同批处理机；工件加工时间考虑两种情况，分别满足均匀分布[50, 100]和[100, 200]；工件权重 w_{ij} 满足均匀分布(0, 1)。工件的投料期 r_i 满足均匀分布 $\left(0,\varphi\middle/\left(mB\right)\cdot\sum_{i=k}^{n}p_k\right)$，工件的交货期 d_i 满足均匀分布 $\left(0,r_i+p_i+\psi\middle/\left(mB\right)\cdot\sum_{k=1}^{n}p_k\right)$。各参数的值见表 6-3：

表 6-3 仿真实验参数

问题参数	参数数量	参数值
机器数量 m	2	3, 5
每种工件的数量 n_j	3	25, 50, 100
机器最大批量 B	1	6
工件种类 j	1	3
加工时间 p_i	2	Uniform [50, 100], Uniform [100, 200]
工件权重 w_i	1	(0, 1)
投料期参数 φ	3	0.25, 0.50, 0.75
交货期参数 ψ	3	0.25, 0.50, 0.75

上表的实验参数组合数为 108，针对每一组合产生 10 个算例进行仿真实验，共产生

1080 个仿真算例。算法参数设置为：$\alpha = 0.8$，$\beta = 1.2$，$\gamma = 1$，$\rho = 0.1$，$N_{rs} = 20$，$K = 0.5$，$n_a = 10$。信息素初始值设定为 $\tau_{\max} = 1 / C^{gb}$，其中初始 C^{gb} 应用 ATC-BATC 算法获得。

对比混合蚁群算法与 ATC-BATC 的仿真实验结果如表 6-4 所示

表 6-4 混合 ACO 算法与 ATC-BATC 算法对比仿真实验结果

	$m = 3$			$m = 5$		
	$n_j = 25$	$n_j = 50$	$n_j = 100$	$n_j = 25$	$n_j = 50$	$n_j = 100$
$\varphi = 0.25$ $\psi = 0.25$	0.758	0.763	0.723	0.862	0.846	0.821
$\varphi = 0.25$ $\psi = 0.50$	0.782	0.752	0.747	0.853	0.852	0.845
$\varphi = 0.25$ $\psi = 0.75$	0.795	0.781	0.752	0.872	0.861	0.839
$\varphi = 0.50$ $\psi = 0.25$	0.857	0.811	0.798	0.881	0.874	0.856
$\varphi = 0.50$ $\psi = 0.50$	0.838	0.827	0.805	0.893	0.890	0.854
$\varphi = 0.50$ $\psi = 0.75$	0.869	0.843	0.812	0.876	0.893	0.883
$\varphi = 0.75$ $\psi = 0.25$	0.891	0.856	0.863	0.899	0.911	0.905
$\varphi = 0.75$ $\psi = 0.50$	0.927	0.872	0.849	0.920	0.917	0.922
$\varphi = 0.75$ $\psi = 0.75$	0.952	0.894	0.880	0.919	0.943	0.931

表中数据是按照混合 ACO 算法所获得的目标值与 ATC-BATC 算法所得目标值之比。由表 6-4 可知，蚁群算法在总体上性能比 ATC-BATC 算法要好。其中，当 φ 较小时 ACO 获得的解质量较高(如图 6-3 所示)，这是因为 φ 越小则说明工件到达越密集，组成满批的可能性越大，因而可以较好地发挥蚁群算法在一个固定搜索空间中的寻优能力。较小的 ψ 也对解质量的提升有帮助，但效果并不明显(如图 6-4 所示)，这是因为当 ψ 较大时，工件交货期较松，加工滞后的情况会得到缓解，因而会抵消一部分算法的有效性。工件数量的增加对绩效也有一定程度的提升(如图 6-5 所示)，因为蚁群算法是一种全局算法，问题规模越大其优越性越突出，但同时也需要更多的计算求解时间。机器数量的增加使得算法绩效略有下降(如图 6-6 所示)，其原因是单台机器的负荷减轻抵消了一定的算法有效性。

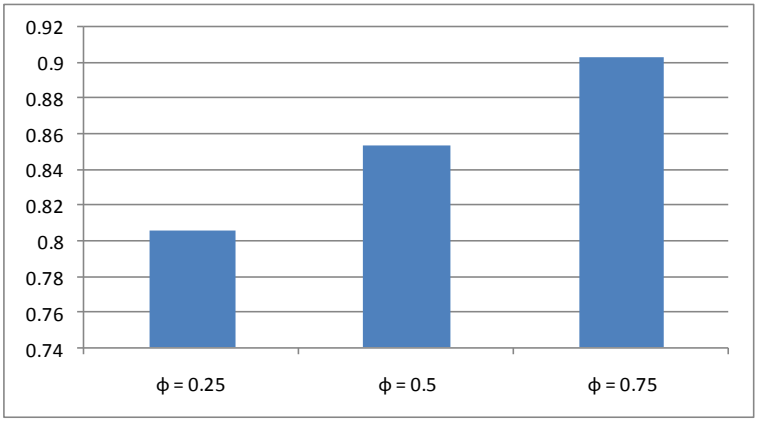


图 6-3 工件到达时间分布参数对绩效的影响

Fig. 6-3 The improvements by parameters of jobs arrival time distribution

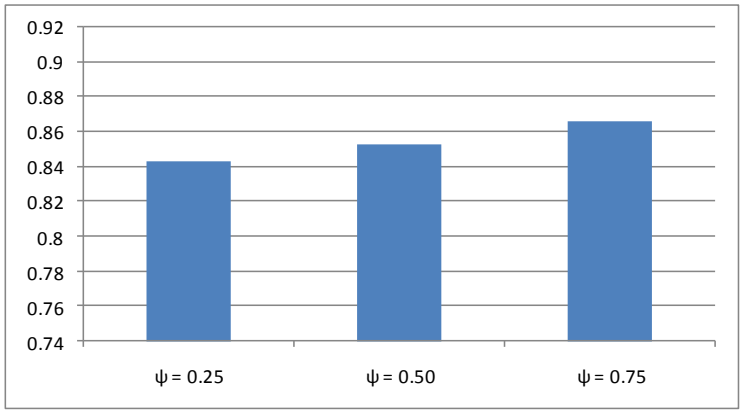


图 6-4 工件交货期分布参数对绩效的影响

Fig. 6-4 The improvements by parameters of jobs due dates distribution

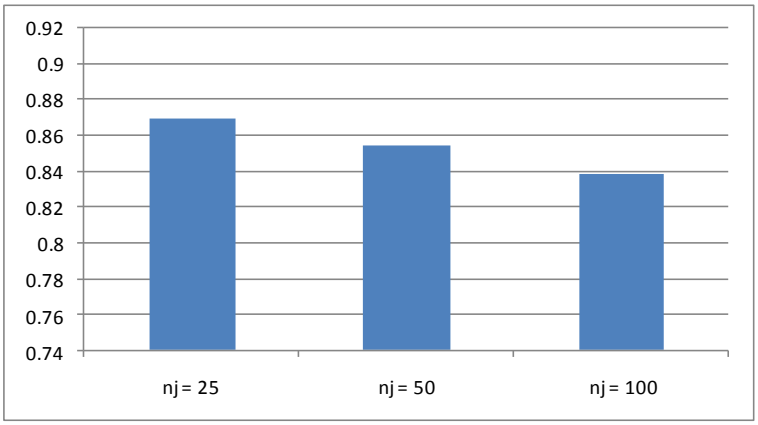


图 6-5 调度工件的数量对绩效的影响

Fig. 6-5 The improvements by numbers of jobs to be scheduled

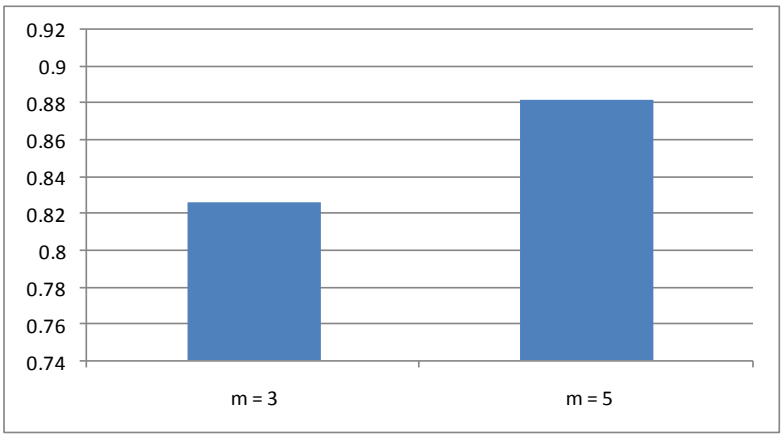


图 6-6 机器数量对绩效的影响

Fig. 6-6 The improvements by numbers of parallel machines

6.3 基于分类蚁群算法的仿真实验

为了验证基于机器类型分解的分类蚁群算法，我们使用 mini-fab 模型[172]进行仿真实验。该模型包括 5 台机器，6 个加工步骤。五台机器为 *A*, *B*, *C*, *D* 和 *E*，分别属于三个不同的机器群。工件在加工过程中分别重入每台机器两次，加工流如图 6-7 所示，其中连接弧上面的数字代表加工顺序。

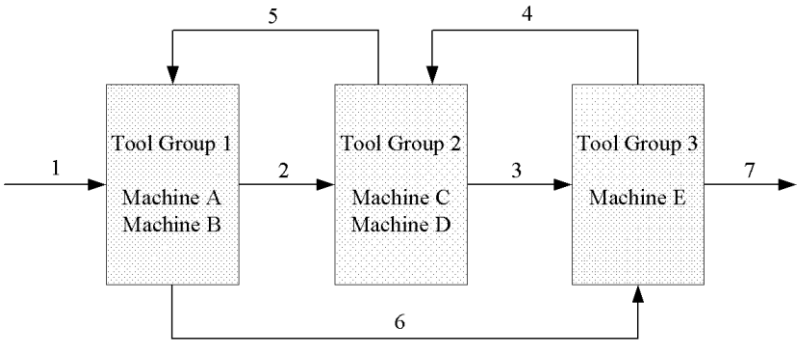


图 6-7 基于机器组的 mini-fab 加工流示意图

Fig. 6-7 Tool group based diagram of mini-fab process flow

mini-fab 加工流中的每一工步的加工时间、组批规模以及所需要的机器组如表 6-5 所示。其中机器组 1 与机器组 2 包含两台相同的机器，并行运行。机器组 1 中的机器是批处理机器，其最大组批数 $b_k = 3$。

表 6-5 mini-fab 中各工步的加工时间与组批规模

工步	机器组（所包含的机器）	加工时间（分钟）	组批规模（lot 数）
1	1 (A, B)	225	3
2	2 (C, D)	30	1
3	3 (E)	55	1
4	2 (C, D)	50	1
5	1 (A, B)	255	3
6	3 (E)	10	1

首先通过实验选择合适的参数 α 与 β，为此在 mini-fab 模型上进行双因子分析。仿真实验假设以固定投料策略向 mini-fab 中投入 100 个工件，假定参数 $\rho = 0.1$，对每一对 (α, β)数值的组合，执行 5 次分类蚁群算法，记录下平均加工周期和收敛时间，结果如表 6-6 所示：

表 6-6 参数 α 与 β 对分类蚁群算法的影响

参数	平均加工周期（分钟）	平均收敛时间（分钟）
α=1, β=0	16084.31	58.7
α=1, β=1	16028.04	38.5
α=1, β=2	16041.59	32.2
α=1, β=3	16064.51	20.1
α=1, β=4	16100.98	15.3
α=2, β=0	16040.55	62.3
α=2, β=1	15983.24	41.8
α=2, β=2	15926.97	36.9
α=2, β=3	15931.14	20.7
α=2, β=4	16045.76	18.5
α=3, β=0	16210.39	82.8
α=3, β=1	16173.92	64.4
α=3, β=2	16129.12	52.1
α=3, β=3	16060.35	45.3
α=3, β=4	16075.98	42.2
α=4, β=0	16593.85	158.9
α=4, β=1	16310.43	92.2
α=4, β=2	16169.76	76.3
α=4, β=3	16248.95	61.9
α=4, β=4	16278.12	50.3

通过仿真结果可知，当两个参数分别设定为 $\alpha = 2$，$\beta = 3$ 时，绩效指标与计算时间都可以获得满意的结果。

为了研究信息素更新规则对算法的影响，继续在 mini-fab 模型上对蒸发系数 ρ 进行实验。表 6-7 展示了不同蒸发系数下所获得的平均加工周期与收敛时间。其中，算法参数设定为 $\alpha = 2$，$\beta = 3$。

表 6-7 蒸发系数 ρ 对分类蚁群算法的影响

ρ	平均加工周期（分钟）	平均收敛时间（分钟）
0.02	15862.37	88.2
0.05	15892.58	45.7
0.1	15931.14	20.7
0.15	16103.04	18.3
0.2	16170.77	16.2
0.25	16272.88	15.1
0.3	16319.77	14.7
0.35	16348.95	14.2
0.4	16368.75	13.6

从仿真结果可知，当 $\rho < 0.1$ 时，随着 ρ 值的减少，算法收敛时间增加得非常快，但是绩效指标的改进非常有限。当 $0.1 < \rho \leq 0.4$ 时，随着 ρ 值的增加，算法的绩效指标越来越差，而算法收敛时间减少得非常缓慢。所以，我们将算法的蒸发系数设定为 $\rho = 0.1$。

为了评价所提出的算法，我们采用 ILOG CPLEX 软件和'CR+FIFO'调度规则作为比较的对象。ILOG CPLEX 集成了分支定界法和一些启发式算法。虽然 CPLEX 可以获得优化解，但是仅能处理一些小规模的问题。CR+FIFO 规则是目前在实际晶圆厂中广泛应用的调度规则，在 CR+FIFO 规则下，等待工件按照其 CR 值进行排队等待，如果两个工件具有相同的 CR 值，则按照其到达时间的先后排队，先到者先加工。

在实验的过程中，特定数量的 lot 按照固定投料规则投入到虚拟工厂中，投料速率是每小时一个 lot。分类蚁群算法和 CR+FIFO 规则在仿真平台上运行 10 次，记录下其所获得调度的平均加工周期。同样的模型也通过 CPLEX 进行运算，三个算法的调度结果如表 6-8 所示：

表 6-8 调度仿真结果

工件数量	平均加工周期（分钟）		
	CR+FIFO	CPLEX	分类蚁群算法
20	4121.71	3992.26	3998.15
30	5468.52	5127.83	5132.16
40	6672.32	6308.19	6325.31
50	8032.47	unknown	7613.22
60	9583.67	unknown	9021.52

实验调度的结果是系统加工完成所有工件所需要的平均加工时间，即 makespan。由

实验结果可知，分类蚁群算法所获得的调度结果比 CPLEX 较差，但非常近似，而远远优于 CR+FIFO 调度规则所获得的结果。但是当工件数量逐渐增加到超过 50 个时，CPLEX 失去了效用，而分类蚁群算法依然能够提供较好的调度结果。

6.4 基于时间分解与机器分解的仿真实验

为验证针对大规模调度问题分解及相应算法的有效性，参考 Wein[13]所提供的晶圆制造生产模型和来自于实际晶圆制造工厂的简化生产模型，并根据 Liu[173]所提供的依据静态产能计算机器数量的方法，构建虚拟晶圆制造系统，包含 42 个机器组和 80 个机器，如表 6-9 所示。

表 6-9 虚拟晶圆制造系统的构成

区域	机器组数量	机器类型			
		单件加工	批处理	并行加工	合计
清洗	1	1			1
扩散	9	4	7		11
光刻	6	6		10	16
刻蚀	7	16			16
淀积	10	12	6		18
物理研磨	6	11			11
离子注入	3	7			7
合计	42	57	13	10	80

仿真实验基于先前开发的仿真平台[174]进行适当的修改，系统采用 Visual Basic .Net 进行开发，面向对象、支持多线程。仿真平台采用三个线程分别处理界面操作、调度算法以及仿真运算，防止相互影响。

所采用的生产流程共 183 道加工步骤（Processing Step），净加工时间合计 421.8 小时。采用固定时间间隔投料策略，投料速率为 50 片/天。

此虚拟晶圆制造系统中同一时刻存在 150~300 个 lot，在一个制造周期（500~800 小时）内要对 30000~60000 个操作进行排序，对于这样大规模的调度问题，传统蚁群算法难以应用，而基于分解的蚁群算法则可以将原问题分解为若干小规模调度问题分别求解并最终获得初始问题的调度。

表 6-10 基于分解的蚁群算法仿真实验结果

调度策略			Ave. T	Ave. WIP	TP	Ave.Time	Percentage Improvement (%)			
			(*h*)	(*P*)	(P/D)	(s)	CT	WIP	TP	TIME
	CR+FIFO		821.72	7,211.18	211.37	512.29				
D-ACO	*Ns*=100	*fa*=10%	817.54	7,024.65	218.72	772.37	0.51	2.59	3.48	-50.77
		fa=20%	812.10	6,943.25	217.23	812.52	1.17	3.72	2.77	-58.61
		fa=30%	802.32	6,724.32	219.28	842.18	2.36	6.75	3.74	-64.40
		Average	810.65	6,897.41	218.41	809.02	1.35	4.35	3.33	-57.92
	Ns=200	*fa*=10%	782.33	6,638.27	218.37	1,876.38	4.79	7.94	3.31	-266.27
		fa=20%	775.25	6,528.16	220.21	2,031.29	5.66	9.47	4.18	-296.51
		fa=30%	772.20	6,502.17	219.37	2,219.76	6.03	9.83	3.78	-333.30
		Average	776.59	6,556.20	219.32	2,042.48	5.49	9.08	3.76	-298.70
	Ns=300	*fa*=10%	754.29	6,421.28	222.54	3,620.65	8.21	10.95	5.28	-606.76
		fa=20%	747.30	6,386.49	218.39	4,162.27	9.06	11.44	3.32	-712.48
		fa=30%	742.28	6,329.05	225.30	4,629.38	9.67	12.23	6.59	-803.66
		Average	747.96	6,378.94	222.08	4,137.43	8.98	11.54	5.07	-707.63
	Ns=400	*fa*=10%	723.21	6,253.92	226.21	7,920.23	11.99	13.27	7.02	-1,446.04
		fa=20%	721.20	6,194.78	228.36	8,592.46	12.23	14.09	8.04	-1,577.26
		fa=30%	716.29	6,083.26	230.17	9,123.20	12.83	15.64	8.89	-1,680.87
		Average	720.23	6,177.32	228.25	8,545.30	12.35	14.34	7.98	-1,568.06
	Ns=500	*fa*=10%	709.32	6,021.28	235.18	19,350.23	13.68	16.50	11.26	-3,677.20
		fa=20%	707.83	5,942.17	239.20	22,652.13	13.86	17.60	13.17	-4,321.74
		fa=30%	706.26	5,921.24	238.63	24,879.35	14.05	17.89	12.90	-4,756.50
		Average	707.80	5,961.56	237.67	22,293.90	13.86	17.33	12.44	-4,251.81
	Ns=600	*fa*=10%	701.25	5,902.36	237.18	49,687.63	14.66	18.15	12.21	-9,599.12
		fa=20%	698.20	5,898.33	239.27	55,329.18	15.03	18.21	13.20	-10,700.36
		fa=30%	695.36	5,876.29	241.57	63,267.59	15.38	18.51	14.29	-12,249.96
		Average	698.27	5,892.33	239.34	56,094.80	15.02	18.29	13.23	-10,849.81

基于分解的蚁群算法（D-ACO）需要设定多个参数，包括蚁群算法框架参数和子问题分解参数（子问题规模 N_s，重叠因子，差异系数）。我们把重点放在子问题分解规模

上，因而设定其他各参数设置如下：α=1，β=2.5，γ=0.1，ρ=0.2，f_{con}=30%。

仿真过程采用 CR+FIFO 调度规则作为基于分解蚁群算法（D-ACO）的对照组，这是目前晶圆厂实际广泛采用的调度规则，其基本思想是首先通过 CR（Critical Ratio，临界比）规则对等待加工工件进行排序，在 CR 值相同的情况下按照 FIFO（先入先出）规则进行选择。以此在实际生产中有效应用的调度方法与蚁群算法进行对比，用晶圆制造系统中几个关键绩效指标（Key Performance Index, KPI）进行衡量：即生产周期（Cycle Time, CT），在制品数（WIP），最大产品交期率（On-Time Delivery, OTD）和最大产出率（Throughput Rate, TR）。蚁群算法按照子问题的规模大小分三种情况进行，子问题所包含的操作数分别设定为：N_s=100，200，300，400，500，600，选择蚂蚁数量 k=10。仿真实验结果如表 6-10 所示。

通过仿真实验可知，与 CR+FIFO 调度规则相比，D-ACO 算法在多个指标上都有不同程度的提高。随着子问题规模的增大，生产周期（CT）与在制品（WIP）水平都有较大的提高，而产出率（TP）由于受到固定投料率的影响，提升幅度略小。计算时间是调度整个原问题所需要的总时间而不是调度单个子问题所需的时间，相对 CR+FIFO 规则，则有大幅度的提升。

分析上述仿真实验结果，重叠度（fa）的提高可以改善绩效指标，但不是非常明显，这是因为重叠操作主要的作用是保证相邻子问题调度方案的连续性，其目的并不是用来提高计算的精度。

相比重叠度而言，子问题规模的增大，可以显著地改善绩效指标。这是由于 CR+FIFO 规则仅仅利用了每个机器缓冲区中的产品信息，而蚁群算法则可以充分利用制造系统全局信息。随着子问题规模的增大，求解过程所利用的信息也相应，因而绩效指标也逐步提高。当子问题规模增大到相当于整个原问题时，可以充分利用所有工件的信息，因而理论上可以获得最优调度结果。但是随着子问题规模的增大，调度求解的时间也急剧上升，因而对于如晶圆制造这样的大规模制造系统而言，全局蚁群算法不可行。利用 SPSS 软件分析表 6-10 仿真结果中求解时间与子问题规模之间的关系，可拟合为指数曲线和三次曲线，两条拟合曲线的参数如表 6-11 所示，拟合曲线如图 6-8 所示。

表 6-11 计算时间随子问题规模增长的拟合曲线模型与参数估计

Equation	Model Summary					Parameter Estimates			
	R Square	F	df1	df2	Sig.	Constant	b1	b2	b3
Cubic	0.999	490.843	3.000	2.000	0.002	-10905.381	179.309	-0.764	0.001
Exponential	0.997	1363.740	1.000	4.000	0.000	352.554	0.008		

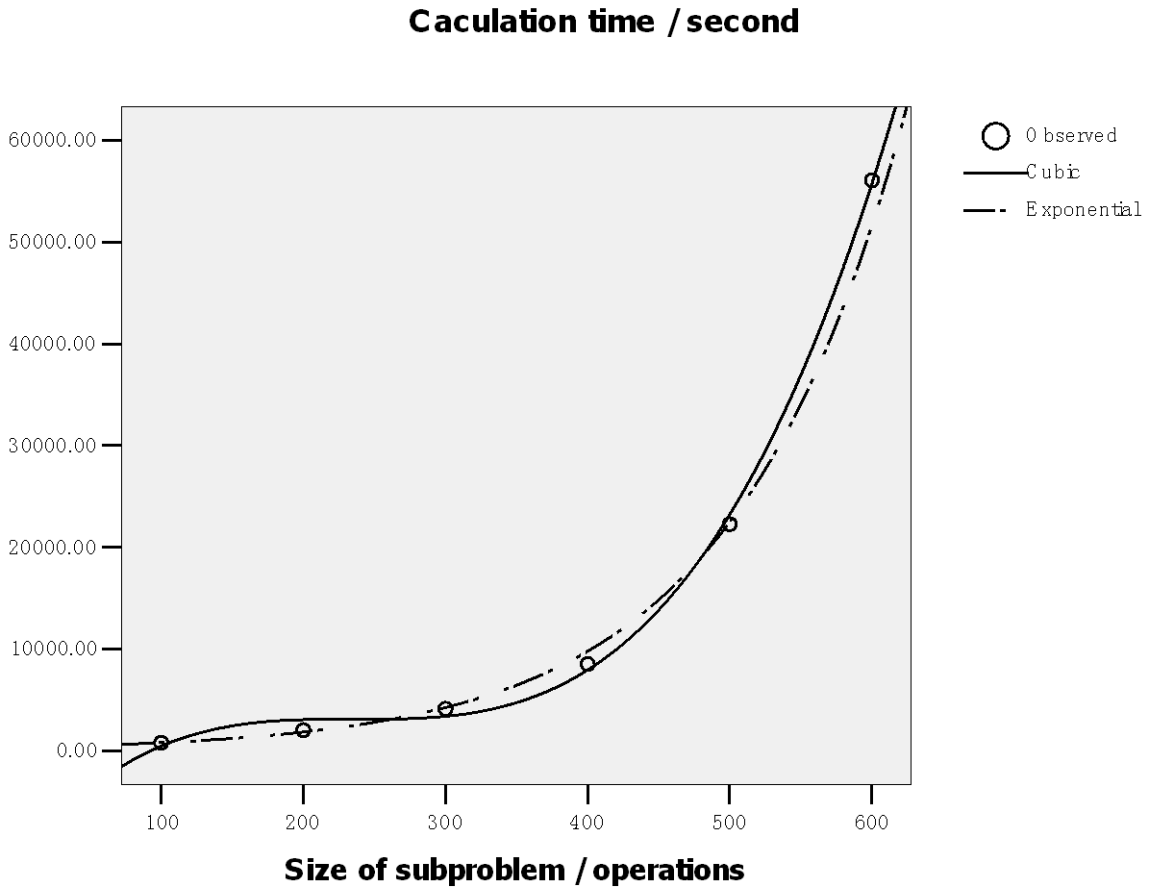


图 6-8 调度子问题规模与求解时间关系图

Fig. 6-8 The relationship between the size of subproblem and the calculating time

由表 6-11 可知，用三次函数和指数函数都可以对仿真结果进行较好的拟合，三次函数的相关系数（R 平方）更好一些（0.999），说明其拟合曲线更符合仿真实验结果，但是指数函数的相关系数（0.997）也足够精确。相对三次函数而言，指数函数犯错误的

概率更小一些（σ 值为 0.000）。所拟合的两个回归方程为：

三次函数： $Y = -10905.381 + 179.309X - 0.764X^2 + 0.001X^3$

指数函数： $Y = 352.554 \times e^{0.008X}$

上述回归方程中自变量 X 表示子问题规模的大小，其单位为操作的个数，因变量 Y 表示当子问题规模为 X 时所需要的计算时间，单位为秒。从两个回归方程可以看出，随着子问题规模的逐渐变大，求解所需的时间急剧上升。如果不进行问题分解，子问题规模的增大会限制求解的能力，因而对问题进行分解是将蚁群算法应用于求解大规模调度问题的关键。

如上所述，D-ACO 算法通过将蚁群算法与问题分解方法相结合，一方面成功将蚁群算法应用于求解晶圆制造系统的调度问题，另一方面有效提升了系统的绩效指标。晶圆制造系统由于投资巨大，因而效率的微小提升就可以产生可观的经济效果。通常对于一座 8 寸晶圆厂来说，在制品水平每降低 1%将可以减少数百万元的成本，制造周期每缩短 1%将增加数千万元的产出。因而 D-ACO 算法的应用有较大的学术和经济价值。

6.5 本章小结

本章针对所提出的基于时间与基于机器的问题分解方法，以及批处理蚁群调度算法及分类蚁群算法，分别进行了仿真实验。从机器调度、子问题调度、原问题调度三个层次分别构建虚拟仿真模型，进行实验验证。结果证明本文所提出的各类分解方法和蚁群算法可以很好地应对晶圆制造系统的特征，与对照的算法相比，可以获得令人满意的调度结果。

第七章 结论与展望

近二十年来，随着超大规模和甚大规模集成电路技术的广泛应用及相关信息、通讯电子产业的高速进步，半导体制造产业蓬勃发展。虽然与传统的机械制造业相比，半导体制造业诞生和发展的历史并不长，但它已经成为当今世界上成长最快、竞争最为激烈的产业之一。作为新兴的战略性工业，半导体制造业的技术水平和发展规模已经成为衡量一个国家经济发展和科技进步的重要标志，做为半导体制造过程中最为复杂和昂贵的部分，半导体晶圆制造系统的科学生产管理水平必将直接影响整个半导体制造业的快速和可持续发展。因此，针对该复杂系统生产管理与控制的科学问题已经引起世界各国专家学者的高度重视，其相关研究正在世界范围内积极展开。

然而，在半导体晶圆制造系统内实施有效和高效生产管理和控制面临极大的挑战，其困难性和复杂性主要源于该系统所具有的以下几个主要典型特征：

（1）高度复杂的反复重入型加工过程；

（2）大规模的系统构成，制造资源繁多，造价极其昂贵；

（3）在制品数量众多，价值昂贵且快速增值；

（4）大批量、超多品种的混合生产模式；

（5）面向多种绩效指标的系统整体优化目标。

因此，半导体晶圆制造系统已经成为当代以高科技为支撑的大规模复杂重入型制造系统的典型代表，而也正是由于上面所述的五个主要复杂特征，才使得当前关于 SWFS 先进生产管理和控制方法的相关研究及成果严重滞后于该领域先进制造装备发展的技术水平。

面对半导体制造产业快速发展及相关课题研究的机遇和挑战，本论文从当前晶圆制造业的实际生产状况和水平出发，在充分吸收过去十余年间针对半导体制造系统的建模、控制及调度等相关研究成果基础上，针对半导体晶圆制造系统生产管理与控制的几个主要科学问题，提出分类蚁群算法，并与问题分解方法相结合，力图探索出一套能够有效解决晶圆制造系统复杂生产控制问题的科学方法，以期能提高我国半导体晶圆制造企业的生产控制水平。本课题研究的开展不仅可以在相关学术研究领域上有所贡献，更可以将研究成果直接应用到实际的半导体晶圆制造系统中，提高企业的生产控制和管理水平，为增强我国半导体制造企业的经济效益和综合市场竞争力起到积极的推动作用。

7.1 本论文的主要贡献

在国家自然科学基金项目（No. 50475027）、国家高技术研究发展计划（863）项目（No. 2006AA04Z128）、高等学校博士学科点专项科研基金资助课题（No. 20040248052）与国家科技重大专项课题（No. 2011ZX02501-005）的资助下，针对晶圆制造系统调度困难的现状，提出两类问题分解方法及相应的协调机制，提出应用问题分解方法与蚁群算法的统一调度框架，并对传统蚁群算法进行了改进，提出了基于重入的蚁群算法，基于问题分解的分类蚁群算法与应用于组批的批处理设备的蚁群调度，其论文的具体贡献如下：

（1）提出基于重入的蚁群调度算法

多重入是晶圆制造系统的典型特征，也是处理调度问题的困难点，为此本文提出基于重入的蚁群调度算法，有针对性地根据工件不同的重入阶段进行处理，以保证系统调度的平衡。

（2）提出基于组批的批处理设备蚁群调度算法

晶圆加工设备可分为两大类：单件加工设备与批处理加工设备。应用蚁群算法调度单件加工设备问题较为成熟，但调度批处理加工的研究刚刚起步。批处理调度可分为两个步骤：组批与批次分配，目前蚁群算法多应用于后一步骤，而对组批方法却涉及不多，这限制了蚁群算法应用的范围和调度结果的质量。为此，本课题试图重点研究蚁群算法组批调度方法，并结合批次分配算法，提出完整的批处理设备调度蚁群算法。

（3）提出基于混合生产类型的分类蚁群调度算法

蚁群算法在生产调度中获得了较好的应用，但大多局限于单一设备类型的调度问题，因而难以应用于实际晶圆制造系统中。为此，本课题试图研究混合生产类型的蚁群算法统一框架，提出分类蚁群算法，可将蚁群算法成功应用于晶圆制造系统。

（4）提出两类应用于晶圆制造系统的问题分解方法

元启发算法可解决较大规模的调度问题，但当问题规模急剧扩大，类似晶圆制造这类调度问题时，元启发算法的应用会受到制约。为此，应用问题分解方法将大规模问题分解为小规模的子问题，再用蚁群算法分别求解，就成为一条可行的解决途径。本论文在现有问题分解方法的基础上，提出两类适合晶圆制造系统特征的问题分解方法，并结合蚁群算法，以有效处理大规模晶圆制造系统调度问题。

（5）提出问题分解调度协调机制

问题分解是一个将各子问题解耦的过程，在各子问题单独求解完成后，这些解合成

原问题的解又面临一个协调方法的问题，这两个问题如何处理是问题分解方法最需要考虑的问题。本文提出三类协调机制，首先针对基于时间的问题分解方法，提出基于重叠区间的协调机制，保证了原问题调度方案的可行性；其次提出了基于工件上下游的协调机制，通过上下游机器的加工信息，决定当前机器的调度策略，以提供调度方案的精度；最后采用蚁群算法的信息素更新机制作为隐性协调机制，针对基于机器的分解方法，采用蚁群算法的信息素保证分解后的机器间信息的沟通。

7.2 本论文的主要创新点

（1）提出针对晶圆制造系统调度问题特征的两类分解方法

为了发挥蚁群算法具有的并行搜索机制与自组织、自适应的性能优势，克服其迭代求解时间较长，难以处理晶圆制造系统的大规模调度的局限，本论文提出了两类问题分解方法以应对晶圆制造系统调度问题的特征。首先，针对晶圆制造系统的大规模特征，提出**基于时间的问题分解方法**，将原问题按照时间维度分解为若干区间，从而将原问题分解为若干较小的子问题，依次进行求解。通过控制时间区间的长度可以获得合适的子问题规模，以求在运算效率与调度精度间取得平衡。其次，针对晶圆制造系统的混合加工特征，提出**基于机器类型的问题分解方法**，将不同类型的机器用相应的调度算法进行处理，从而避免了彼此间的干扰。两类问题分解方法的应用有一定的顺序，首先使用基于时间的问题分解方法，将大规模调度问题分解为若干较小的子问题，然后对这些子问题按照串行的方式逐个求解。这些子问题本身仍然具有混合加工的特征，因而采用基于机器类型的问题分解方法对子问题进行分解，获得若干单机调度子问题，按照机器的类型采用相应的蚁群算法对其进行求解。基于问题分解的方法研究较多，但大多是单一类型的分解方法，与其相结合的调度算法通常为数学规划方法。本文采用复合式的问题分解方法以有效处理晶圆制造系统的特征，并结合蚁群算法对子问题进行调度。

（2）提出适应问题分解方法的三类协调机制

问题分解为子问题并求解后，如何合成为原问题的解是问题分解方法成功与否的关键。为此，本文提出三类协调机制，以适应所提出的两类问题分解方法。这三类协调机制中有两类属于显性协调机制，分别是基于重叠区间的协调机制与基于工件加工上下游信息的协调机制；另一类属于隐性协调机制，即借助于蚁群算法的信息素，通过其更新机制传递机器间的加工状态与信息。

针对基于时间的问题分解方法，提出了**基于重叠区间的协调机制**，这类机制是在分解后的相邻子问题间设置一定的重叠区间，以保证原问题调度结果的可行性。本文提出

的基于时间的问题分解方法采用单向解耦方式，即按照时间顺序每次只求解一个子问题，当前子问题求解完毕后再求解下一子问题，依次进行直到将所有子问题求解完毕。在单向解耦的问题分解方法下，相应的子问题间设置重叠区间，可以保证子问题间调度结果的一致性，并最终保证原问题调度结果的可行性。针对基于机器类型的问题分解方法，**采用蚁群算法特有的信息素更新机制作为不同类型机器间调度方案的协调机制**，在同一个调度架构即分类蚁群算法下，通过信息素的传递、变换与更新，从而保证各机器信息的交流与协调。此外，通过**基于加工上下游信息的协调机制**，机器在调度过程中将该工件与其上下游机器的状态考虑在内，从而保证了调度的精度。本论文所提出的三类协调机制有效处理了问题分解所带来的子问题解合并的困难，是本论文的第二个创新点。

（3）提出面向问题分解与晶圆制造系统特征的三类改进蚁群算法

晶圆制造系统的调度问题非常复杂，除了大规模与混合型加工外，多重入特征也是造成调度困难的一大原因。由于多重入的存在，机器不仅要处理不同类型的工件，也要处理同一类型但处于不同加工阶段的工件。另一方面，针对不同的调度指标，处于不同加工阶段的工件的加工优先级也不同。为此，本文提出**基于重入的蚁群算法**，将工件的重入状态考虑在内，结合制造系统状态与调度指标，以提高调度结果的质量。此外，晶圆制造系统中的批处理设备由于加工时间长，通常是制造系统的瓶颈设备，其调度方案的质量对整体系统的调度质量影响很大。近年来蚁群算法被应用于求解批处理加工设备，取得了较好的进展，但这些算法基本都是应用于批次调度，而对更为重要的组批过程很少涉及。本论文为此提出**面向组批的批处理调度蚁群算法**，针对单机批处理机与并行批处理机提出了两类组批蚁群算法，取得了良好的效果。最后，为了保证基于类型分解后的各单机调度子问题的解可以顺利合成为原问题的解，提出了**基于机器类型分解的分类蚁群算法**，结合信息素更新机制与基于加工上下游信息的协调机制，将各类机器的调度整合在一个调度框架下，通过信息素的传递保持各机器调度结果的一致性。本文所提出的三类改进蚁群算法面向解决晶圆制造系统的特征与所提出的问题分解方法，在传统蚁群算法的基础上进行了改进，是本论文的第三个创新点。

7.3 研究展望

半导体晶圆制造系统以其大规模构成、复杂设备特性、反复重入型加工过程、和多品种大批量混合生产等典型特征成为当今最为复杂的大型制造系统之一，针对该系统的建模、计划、控制和仿真等研究已经成为相关领域的热点和难点问题。本文从当前晶圆制造业的实际状况和相关最新研究成果出发，以离散事件动态系统、先进制造系统、问

题分解方法和蚁群算法等理论为依据，提出了基于问题分解的蚁群算法系统研究方法，探索出了一套能够有效解决当前晶圆制造系统建模和实时派工控制的科学生产管理方法。

然而，作为一个高度复杂的制造系统，半导体晶圆制造系统的科学生产管理问题涉及企业的供应链管理、生产计划、车间层控制、和库存管理等方方面面，在短暂的攻读博士学位研究期间是不可能囊括上述所有内容的。因此，本论文仅将研究重点集中在SWFS 的系统描述和车间层调度控制方面，但其他方面相关的研究工作均可在本文的研究成果支持下继续进行。下面，在本文的研究工作和成果基础上，就有关深入研究的课题和方向做一简要的分析和展望。

（1）本文所提出的面向批处理调度设备的蚁群算法是结合了静态调度与动态调度的混合调度方法。这种方法虽然获得了良好的调度解，但是当系统任务繁重，对调度实时性要求较强时会造成一定的时间延迟，因而，对此方法进一步深入研究，使其更适合实时动态调度环境，是未来的研究方向。

（2）本文所提出的基于问题分解的蚁群算法建立在串行问题分解理论的基础上。这种方法便于实施，但是效率较低，下一步应深入研究并行问题分解理论，其关键问题在于解决子问题调度方案的相互协调。并行分解可以更大程度地提高运算效率。

（3）下一步应从 SWFS 的车间层控制向上展开，将属于 SWFS 生产管理上层内容的生产计划、产能规划和与底层的调度控制结合起来，构建针对 SWFS 的先进生产计划与调度的整体性体系结构。同时对现有的 SWFS 实时调度仿真系统进行拓展，开发支持生产计划与供应链管理的仿真系统，从而彻底的将晶圆制造系统的整个生产管理有机结合起来。

以上就本论文的主要研究内容和贡献做了简要总结和若干研究展望。由于时间和能力所限，本文肯定存在诸多不足之处，希望能在今后的研究中不断改进和完善。

参考文献

[1] 环咨公司, 2011 年中国集成电路行业分析研究报告. 2011: 北京.

[2] 王阳元, et al., 面向产业需求的 21 世纪微电子技术的发展(上). 物理, 2004. **33**(6): p. 407-413.

[3] Kumar, P.R., Scheduling Semiconductor Manufacturing Plants. IEEE Control Systems Magazine, 1994. **14**(6): p. 33-40.

[4] Narahari, Y. and Khan, L.M., Performance analysis of scheduling polices in re-entrant manufacturing system. Computer Operations Research, 1996. **23**(1): p. 37-51.

[5] Hung, A.C., et al., Modeling, scheduling, and Prediction for Wafer Fabrication: Queueing Colored Petri-Net and GA Based Approach, in Proceedings of the 2002 IEEE International Conference on Robotics & Automation. 2002. p. 3187-3192.

[6] Bachrach, R., et al. Comparative analysis of 300 mm FAB architectures impact of equipment sets on wafer cost and dynamic performance. in Semiconductor Manufacturing Symposium, 2001 IEEE International. 2001.

[7] Leachman, R.C., Kang, J., and Lin, V., SLIM: Short cycle time and low inventory in manufacturing at Samsung electronics. Interfaces, Informs, 2002. **32**(1): p. 61-77.

[8] Zant, P.V., 芯片制造—半导体工艺制程实用教程（第四版）. 2004, 北京: 电子工业出版社.

[9] Quirk, M., 半导体制造技术. 2004, 北京: 电子工业出版社.

[10] McKay, K.N. and Wiers, V.C.S., Unifying the theory and practice of production scheduling. Journal of Manufacturing System, 1999. **18**(4): p. 241～255.

[11] 徐俊刚, 戴国忠, and 王宏安, 生产调度理论和方法研究综述. 计算机研究与发展, 2004. **41**(2): p. 257-267.

[12] 陈荣秋, 排序的理论和方法. 1987, 武汉: 华中科技大学出版社.

[13] Wein, L.M., Scheduling Semiconductor Wafer Fabrication. IEEE Transactions on Semiconductor Manufacturing, 1988. **1**(3).

[14] Chung, S.H. and Huang, H.W., The design of production activity control policy. Journal of the Chinese Institute of Industrial Engineering, 1999. **16**(1): p. 93–113.

[15] Lee, C.E. and Chen, C.W., A dispatching scheme involving move control and weighted due date for wafer foundries. IEEE Transaction on Component, Packaging, and Manufacturing Technology Part C, 1997. **20**(4): p. 268–277.

[16] Glassey, C.R. and Resende, M.G.C., Closed-loop job release control for VLSI circuit manufacturing. IEEE Transaction on Semiconductor Manufacturing, 1988. **1**(1): p. 36–46.

[17] Fowler, J.W., et al., Measurable improvements in cycle-time-constrained capacity, in

IEEE International Symposium on Semiconductor Manufacturing Conference Proceedings. 1997. p. A21-A24.

[18] Lu, S.C.H., Ramaswamy, D., and Kumar, P.R., Efficient scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plants. IEEE Transaction on Semiconductor Manufacturing, 1994. **7**(3): p. 374–388.

[19] Glassey, C.R. and Weng, W.W., Dynamic batching heuristic for simultaneous processing. Semiconductor Manufacturing, IEEE Transactions on, 1991. **4**(2): p. 77-82.

[20] Lee, L.H., Tang, L.C., and Chan, S.C., Dispatching Heuristic for Wafer Fabrication, in Proceedings of the 2001 Winter Simulation Conference. 2001. p. 1215-1219.

[21] Uzsoy, R., Church, L., and Ovacik, I., Dispatching rules for semiconductor testing operations: a computational study, in IEEE/CHMT International Electronics Manufacturing Technology Symposium. 1992. p. 272–276.

[22] Holthaus, O. and Rajendran, C., Efficient dispatching rules for scheduling in a job shop. International Journal Production Economy, 1997. **48**: p. 87–105.

[23] Dessouky, M.M. and R.C., L., Dynamic Models of Production with Multiple Operation General Processing Times. Journal of the Operational Research Society, 1997. **48**(6): p. 647-654.

[24] Kumar, S. and Kumar, P.R., Queueing Network Models in the Design and Analysis of Semiconductor Wafer Fabs. IEEE Transactions on Robotics and Automation, 2001. **17**(5): p. 548-561.

[25] Morton, T. and Pentico, D., Heuristic scheduling systems. 1993, New York: Wiley.

[26] Sung, C. and Choung, Y., Minimizing makespan on a single burn-in oven in semiconductor manufacturing. European Journal Operation Research, 2000. **120**: p. 559-574.

[27] Chen, T.-R., Chen, C.-W., and Kao, J., Due windows scheduling for ic sort and test facilities with precedence constrains via Lagrangian relaxation, in IEEE/SEMI International Semiconductor Manufacturing Science Symposium. 1993. p. 110–114.

[28] Chen, T.-R. and Hsia, T., Job shop scheduling with multiple resources and an application to a semiconductor testing facility, in Proceedings of the 33rd IEEE Conference on Decision and Control. 1994. p. 1564–1570.

[29] Chen, T.-R. and Hsia, T., Scheduling for IC sort and test facilities with precedence constrains via Lagrangian relaxation. Journal Manufacturing System, 1997. **16**(2): p. 117–128.

[30] Kaskavelis, C. and Caramanis, M. Application of a Lagrangian relaxation based scheduling algorithm to a semiconductor testing facility. in Proceedings of the Fourth International Conference on Computer Integrated Manufacturing and Automation Technology, IEEE. 1994.

[31] Kaskavelis, C. and Caramanis, M., Efficient Lagrangian relaxation algorithms for industrial size job-shop scheduling problems. IIE Transactions, 1998. **30**: p. 1085–

1097.
[32] Sun, X., James, S., and Klein, C., Single-machine scheduling with sequence dependent setup to minimize total weighted squared tardiness. IIE Transactions, 1999. **31**: p. 113–124.
[33] De, S. and Lee, A., Towards a knowledge-based scheduling system for semiconductor testing. International Journal Production Research, 1998. **36**(4): p. 1045–1073.
[34] Connors, D., Feigin, G., and Yao, D., Scheduling semiconductor lines using a fluid network model. IIE Trans Robot Autom, 1994. **10**(2): p. 88–98.
[35] Neuts, M.F., A general class of bulk queues with Poisson input. The Annals of Mathematical Statistics, 1967. **38**: p. 759-770.
[36] Medhi, J., Waiting time distribution in a Poisson queue with a general bulk service rule. Management Science, 1975. **21**(7): p. 777-782.
[37] Hopp, W., et al., Using an optimized queuing network model to support wafer fab design. IIE Solutions, 2002. **34**: p. 119–130.
[38] Huang, M.G., Chang, P.L., and Chou, Y.C., Analytic Approximations for Multiserver Batch-Service Work-Stations with Multiple Process Recipes in Semiconductor Wafer Fabrication. IEEE Transactions on Semiconductor Manufacturing, 2001. **14**(4): p. 395-405.
[39] Li, C.-L. and Lee, C.-Y., Scheduling with aggreable release time and due dates on a batch processing machine. European Jounal of Operational Research, 1997. **96**: p. 564-569.
[40] Sung, C.S., et al., Minimizing makespan on a single burn-in oven with job families and dynamics job arrivals. Computers & Operations Research, 2002. **29**: p. 995-1007.
[41] Burdick, R.G.A., A cost effective expert system for semiconductor manufacturing. Computer & Operations Research, 1987.
[42] Aytug, H., et al., A review of machine learning in scheduling. IEEE Trans Eng Manage, 1994. **41**(2): p. 165–171.
[43] Huang, C.L., et al., Construction of production performance prediction system for semiconductor manufacturing with artificial neural networks. International Journal of Production Research, 1999. **37**(6): p. 1387-1402.
[44] Liao, D.Y. and Wang, C.N., Neural-network-based delivery time estimates for prioritized 300-mm automatic material handling operations. IEEE Transactions on Semiconductor Manufacturing, 2004. **17**(3): p. 324-332.
[45] Zhang, J., et al., Fuzzy neural networks based rescheduling strategies optimization of semiconductor fabrication line. Chinese Journal of Mechanical Engineering, 2005. **41**(10): p. 75-79.
[46] Azzaro-Pantel, C., et al., A fuzzy approach for performance modeling in a batch plant: application to semiconductor manufacturing. IEEE Transactions on Fuzzy System, 1997. **5**(3): p. 338-357.
[47] Zhou, M.C. and Jeng, M.D., Modeling, Analysis, Simulation, scheduling, and Control

of Semiconductor Manufacturing Systems: A Petri Net Approach. IEEE Transactions on Semiconductor Manufacturing, 1998. **11**(3): p. 333-357.

[48] Xiong, H. and Zhou, M., Scheduling of semiconductor test facility via petri nets and hybrid heuristic search. IEEE Trans Semicond Manuf, 1998. **11**(3): p. 384–393.

[49] Cerny, V., Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of Optimization Theory and Applications, 1985. **45**(1): p. 41-52.

[50] Yim, S. and Lee, D., Scheduling cluster tools in wafer fabrication using candidate list and simulated annealing. Journal Intelligent Manufacturing, 1999. **10**(6): p. 531–540.

[51] Erramilli, V. and Mason, S.J., Multiple Orders Per Job Compatible Batch Scheduling. Electronics Packaging Manufacturing, IEEE Transactions on, 2006. **29**(4): p. 285-296.

[52] Glover, F., Tabu Search-Part I. ORSA JOURNAL ON COMPUTING, 1989. **1**(3): p. 190-206.

[53] Glover, F., Tabu Search-Part II. ORSA JOURNAL ON COMPUTING, 1990. **2**(1): p. 4-32.

[54] Geiger, C., Kempf, K., and Uzsoy, R., A Tabu search approach to scheduling an automated wet etch station. Journal of Manufacturing System, 1997. **16**(2): p. 102–116.

[55] Schwefel, H., Numerical optimization of computer models. 1981, New York: John Wiley & Sons, Inc.

[56] Fogel, L., Artificial intelligence through simulated evolution. 1966, New York: John Wiley & Sons.

[57] Holland, J., Adaptation in natural and artificial systems. 1992: MIT Press Cambridge.

[58] Goldberg, D., Genetic algorithms in search, optimization and machine learning. 1989, Boston: Wesley Longman Publishing.

[59] Liu, M. and Wu, C., Genetic algorithm using sequence rule chain for multi-objective optimization in re-entrant micro-electronic production line. Robotics and Computer-Integrated Manufacturing, 2004. **20**(3): p. 225-236.

[60] Wang, C.S. and Uzsoy, R., A genetic algorithm to minimize maximum lateness on a batch processing machine. Computers and Operations Research, 2002. **29**(12): p. 1621-1640.

[61] Melouk, S., Damodaranb, P., and Chang, P.-Y., Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing International journal of production economics, 2004. **87**(2): p. 141-147.

[62] Mönch, L., et al., Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times. Computers & Operations Research, 2005. **32**(11): p. 2731-2750.

[63] SG, K., et al., Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families. International Journal of Production Economics, 2005. **98**: p. 81–96.

[64] Dantzig, G.B. and Wolfe, P., Decomposition principle for linear programs. Operations Research, 1960. **8**(1): p. 101-111.
[65] Akker, J.M.v.d., Hurkens, C.A.J., and Savelsbergh, M.W.P., Time-indexed formulations for machine scheduling problems:column generation. INFORMS Journal on Computing, 1998. **12**: p. 111-124.
[66] Barnhart, C., et al., Brach-and-Price: column generation for huge integer programs. Operations Research, 1998. **36**: p. 316-329.
[67] Vanderbeck, F. and Savelsbergh, M.W.P., A generic view of Dantzig-Wolfe decomposition for integer programming. Operations Research Letters, 2006. **34**: p. 294-306.
[68] FISHER, M.L., The lagrangian relaxation met hod for solving integer programming problems. Management Science, 1981. **27**(1): p. 1-18.
[69] FISHER, M.L., An applications oriented guide to lagrangian relaxation. Interfaces, 1985. **15**(2): p. 10-21.
[70] Held, M. and Karp, R.M., The traveling salesman and minimum spanning trees. Operation Research, 1970. **18**: p. 1138-1162.
[71] Held, M. and Karp, R.M., The traveling salesman and minimum spanning trees:Part II. Mathematic Programming, 1971. **1**: p. 6-25.
[72] Fisher, M.L., Optimal solution of scheduling problems using Lagrange multipliers:Part I. Operation Research, 1973. **21**: p. 1114-1127.
[73] Fisher, M.L., et al., Surrogate duality relaxation for job shop scheduling. Discrete Applied Mathematics, 1983. **5**: p. 65-75.
[74] Luh, P.B. and Hoitornt, D.J., Scheduling of manufacturing system using the lagrangian relaxation technique. IEEE Transactions on Automatic Control, 1993. **38**: p. 1066-1079.
[75] Luh, P.B., et al., Lagrangian relaxation neural networks for job shop scheduling. IEEE Transactions on Robotics and Automation, 2000. **16**: p. 78-88.
[76] Zhao, X., A new generation of optimization algorithms within the lagrangian relaxation approach for job shop scheduling, in Ph.D. 1999, University of Connecticut.
[77] Fisher, M.L., The lagrangian relaxation method for solving integer programming problems. Management Science, 2004. **50**: p. 1861-1871.
[78] GUIGNARD, M. and KIM, S., Lagrangian decomposition: a model yielding st ronger Lagrangian bounds. Mathematical Programming, 1987. **39**(2): p. 215 - 228.
[79] 吴清烈 and 徐南荣, 大规模含整变量优化问题的一种分解方法. 东南大学学报, 1996. **26**(3): p. 119 - 125.
[80] Benders, J.F., Partitioning procedure for solving mixed-variables programming problems. Numerische Mathematik, 1962. **4**: p. 238-252.
[81] Bassett, M., et al., Perspectives on model based integration of process operations. Computers and Chemical Engineering, 1996. **20**(6-7): p. 821-844.
[82] Bassett, M., Pekny, J., and Reklaitis, G., Decomposition techniques for the solution of

large-scale scheduling problems. AIChE Journal, 1996. **42**(12): p. 3373-3387.

[83] Ovacik, I.M. and Uzsoy, R., Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequence-dependent setup times. International Journal of Production Research, 1994. **32**(6): p. 1243.

[84] Ovacik, I.M. and Uzsoy, R., Rolling horizon procedures for dynamic parallel machine scheduling with sequence-dependent setup times. International Journal of Production Research, 1995. **33**(11): p. 3173-3192.

[85] Upasani, A.A., Uzsoy, R., and Sourirajan, K., A Problem Reduction Approach for Scheduling Semiconductor Wafer Fabrication Facilities. Semiconductor Manufacturing, IEEE Transactions on 2006. **19**(2): p. 216 - 225

[86] Sourirajan, K. and Uzsoy, R., Hybrid decomposition heuristics for solving large-scale scheduling problems in semiconductor wafer fabrication. Journal of Scheduling, 2007. **10**(1): p. 41-65.

[87] Sannomiya, N. and Iima, H. Genetic algorithm approach to an optimal scheduling problem for a large-scale complex manufacturing system. in IEEE SMC '99 Conference Proceedings. . 1999. Tokyo.

[88] Roslöf, J., et al., A Short-Term Scheduling Problem in the Paper-Converting Industry. Computers & Chemical Engineering, 1999. **23**(S): p. S861-S874.

[89] Roslöf, J., et al., Solving a large-scale industrial scheduling problem using MILP combined with a heuristic procedure. European Journal of Operational Research 2002. **138**(1): p. 29-42.

[90] Fox, M.S., Constraint-directed search: A case study of job-shop scheduling. 1983, Pittsburgh: Carnegie Mellon University.

[91] Chu, C., Portmann, M.C., and Proth, J.M., A splitting-up approach to simplify job-shop scheduling problems. International Journal of Production Research, 1992. **30**(4): p. 859-870.

[92] Sun, D. and Batta, R., Scheduling larger job shops: a decomposition approach. International Journal of Production Research, 1996. **34**(7): p. 2019-2033.

[93] Adams, J., Balas, E., and Zawack, D., The Shifting Bottleneck Procedure for Job Shop Scheduling. Management Science, 1988. **34**(3): p. 11.

[94] Wang, C.-S., Decomposition heuristics for complex job shop scheduling. 2000, Purdue University.

[95] Dauzere-Peres, S. and Lasserre, J.B., A modified shifting bottleneck procedure for job-shop scheduling. International Journal of Production Research, 1993. **31**: p. 923-932.

[96] Balas, E., Lenstra, J.K., and Vazacopoulos, A., The one-machine problem with delayed precedence constraints and its use in job shop scheduling. Management Science, 1995. **41**: p. 94-109.

[97] Demirkol, E., Mehta, S., and Uzsoy, R., A computational study of shifting bottleneck procedures for shop scheduling problems. Journal of Heuristic, 1997. **3**: p. 111-137.

[98] Luh, P.B. and Chen, H.X., An alterative framework to lagrangian relaxation approach for job shop cheduling, in the 38th Conference on Decision & Control. 1999.

[99] Merkle, D. and Middendorf, M., Ant Colony Optimization with Global Pheromone Evaluation for Scheduling a Single Machine. Applied Intelligence, 2003. **18**(1): p. 105-111.

[100] Blum, C., ACO Applied to Group Shop Scheduling: A Case Study on Intensification and Diversification, in Ant Algorithms : Third International Workshop, ANTS 2002, Brussels, Belgium, September 12-14, 2002. Proceedings 2002, Springer Berlin / Heidelberg. p. 149-180.

[101] 梁静, 钱省三, and 马良, 基于双层蚂蚁算法的半导体炉管制程批调度研究. 系统工程理论与实践, 2005. **12**(25): p. 96-101.

[102] Song, Y., et al., Bottleneck Station Scheduling in Semiconductor Assembly and Test Manufacturing Using Ant Colony Optimization. Automation Science and Engineering, IEEE Transactions on, 2007. **4**(4): p. 569 - 578

[103] 尹文君, 刘民, and 吴澄, 进化计算在生产线调度研究中的现状与展望. 计算机集成制造系统, 2001. **7**(12): p. 1-6.

[104] Dorigo, M., Optimization, learning and natural algorithms. 1992, Politecnico di Milano.

[105] Dorigo, M., Bonabeau, E., and Theraulaz, G., Ant algorithms and stigmergy. Future Generation Computer Systems, 2000. **16**: p. 851-871.

[106] Dorigo, M. and Di Caro, G. The Ant Colony Optimization: a new meta-heuristic. in Proeeedings of the IEEE International Conference on Evolutionary Computation. 1999: IEEE Press.

[107] Dorigo, M., Di Caro, G., and Gambardella, L.M., Ant algorithms for diserete optimization. Artificial Life, 1999. **5**(2): p. 137-172.

[108] Dorigo, M., Maniezzo, V., and Colorni, A., Ant system: optimization by a colony of cooperating agents. Systems, Man and Cybernetics, Part B, IEEE Transactions on, 1996. **26**(1): p. 29 - 41.

[109] Maniezzo, V. and Colorni, A., The Ant System applied to the quadratic assignment Problem. IEEE Trans. Knowledge Data Eng, 1999. **11**(5): p. 769-778.

[110] Gambardella, L.M. and Dorigo, M. Ant-Q: A reinforCement learning apProach to the traveling salesman problem. in Proceedings of the Twelfth International Conference on Machine Learning(ML-95). 1995: Morgan Kaufimann Publishers.

[111] Gambardella, L.M. and Dorigo, M. Solving symmetric and asymmetric TSPs by ant colonies. in Proeeedings of the IEEE International Conference on Evolutionary Computation. 1996: IEEE Press.

[112] Sutton, R. and Barto, A., Reinforcement Learning: An Introduction. 1998, Cambridge, MA: MIT Press.

[113] Watkins, J.C.H., Q-learning. Machine Learning, 1992. **8**: p. 279-292.

[114] Dorigo, M. and Gambardella, L.M., Ant Colony System: A cooperative learning

approach to the traveling salesman Problem. IEEE Transactions on Evolutionary Computation, 1997. **1**(1): p. 53-66.

[115] Stutzle, T. and Hoos, H.H. The MAX-MIN ant system and local search for the traveling salesman problem. in IEEE 4th International Conference on Evolutionary Computation. 1997: IEEE Press.

[116] Stutzle, T., An ant approach to the flow shop problem in Technical report AIDA-97-07. 1997. p. 7.

[117] Stützle, T. and Hoos, H.H., MAX - MIN Ant System. Future Generation Computer Systems, 2000(16): p. 889-914.

[118] Bullnheimer, B., Hartl, R.F., and Stranss, C., A New Rank Based Version of the Ant System: A Computational Study. Central European Journal for Operations Researeh and Eeonomics, 1999. **7**(1): p. 25-38.

[119] Gutjahr, W.J., A graph-based ant system and its convergence. Future Gener. Comput.Syst., 2000. **16**(8): p. 873-888.

[120] Blum, C., Roli, A., and Dorigo, M., HC-ACO: The Hyper-Cube Framework for Ant Colony Optimization, in MIC'2001-4th Metaheuristics International Conference. 2001.

[121] 覃刚力 and 杨家本, 自适应调整信息素的蚁群算法. 信息与控制, 2002. **31**(3): p. 198-201.

[122] 谢剑英 and 王颖, 一种自适应蚁群算法及其仿真研究. 系统仿真学报, 2002. **14**(1): p. 31-33.

[123] 吴斌 and 史忠植, 一种基于蚁群算法的 TSP 问题分段求解算法. 计算机学报, 2002. **24**(12): p. 1328-1333.

[124] Dorigo, M., Maniezzo, V., and Colorni, A., The Ant System: An autocatalytic optimizing Process. Technical RePort 91-016 Revised. 1991, Dipartimento di Elettronica, Politecnico di Milano: Milan, Italy.

[125] 张纪会, 高齐圣, and 徐心和, 自适应蚁群算法. 控制理论与应用, 2000. **17**(1): p. 1-3.

[126] Stutzle, T. and Dorigo, M., ACO algorithms for the quadratic assignment Problem. New Ideas in Optimization, 1999: p. 33-50.

[127] Di Caro, G. and Dorigo, M., AntNet: Distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research, 1998. **9**: p. 317-365.

[128] Roli, A., Blum, C., and Dorigo, M., ACO for maximal constraint satisfaction Problems. Proeeedings of MIC' 2001, 2001. **1**: p. 187-191.

[129] Solnon, C., Ants can solve constraint satisfaction problems. IEEE Transactions on Evolutionary Computation, 2002. **6**(4): p. 347-357.

[130] Merkle, D., Middendorf, M., and Schmeck, H. Ant colony optimization for resource-constrained project scheduling. in Proceedings of the Genetic and Evolutionary Computation Conference. 2000.

[131] Wodrich, M. and Bilchev, G., Cooperative distributed search: the ants way.Control and

Cybernetics. 26, 1997. **3**(413-445).
[132] Jayaraman, V.K., et al., Ant colony framework for optimal design and scheduling of batch Plants. Computers & Chemical Engineering, 2000. **24**: p. 1901-1912.
[133] Mathur, M., et al., Ant colony approach to continuous function optimization. Ind.Eng.Chem.Res, 2000. **39**: p. 3814-3822.
[134] Bauer, A., et al., An ant colony optimization approach for the single machine totaltardiness problem, in Evolutionary Computation, 1999. CEC 99. 1999: Washington, DC, USA.
[135] Bauer, A., et al., Minimizing Total Tardiness on a Single Machine Using Ant Colony Optimization. Central European Journal for Operations Research and Economics, 2000. **8**(2): p. 17.
[136] Merkle, D. and Middendorf, M., An Ant Algorithm with a new Pheromone Evaluation Rule for Total Tardiness Problems. Computer Science, 2000. **1803**.
[137] Besten, M.d., Stutzle, T., and Dorigo, M., Ant Colony Optimization for the Total Weighted Tardiness Problem, in Conference on Parallel Problem Solving from Nature. 2000.
[138] 陈义保, et al., 基于蚁群系统的工件排序问题的一种新算法. 系统工程学报, 2002. **17**(5): p. 476-480.
[139] Stutzle, T., An ant approach to the flow shop problem, in proceedings of European Congress on IntelligentTechniques and Soft ComPuting. 1998. p. 1560-1564.
[140] Colorni, A., et al., Ant system for job-shop scheduling. Belgian Journal Operation Research (JORBEL), 1994. **34**(1): p. 39-53.
[141] Zwaan, S.v.d. and Marques, C., Ant Colony Optimisation for Job Shop Scheduling. 1999.
[142] 邢文训 and 谢金星, 现代优化计算方法. 2000: 清华大学出版社.
[143] 郑大钟 and 赵千川, 离散事件动态系统. 2001: 清华大学出版社.
[144] Blum, C. and Sampels, M. Ant colony optimization for FOP shop scheduling: a case study on different pheromone representations. in Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on. 2002.
[145] Blum, C. and Sampels, M. When Model Bias Is Stronger than Selection Pressure in Parallel Problem Solving from Nature - PPSN VII: 7th International Conference. 2002. Granada, Spain: Springer Berlin / Heidelberg.
[146] Balas, E., Machine Sequencing via Disjunctive Graphs: An Implicit Enumeration Algorithm. Operations Research, 1969. **17**(6): p. 941-957.
[147] Pinedo, M.L., Scheduling: Theory, Algorithms, and Systems. 2000, New Jersey: Prentice-Hall.
[148] Singer, M., Decomposition methods for large job shops. Computers & Operations Research, 2001. **28**(3): p. 193-207.
[149] Nowicki, E. and Smutnicki, C., A fast taboo search algorithm for the job shop problem. Management Science, 1996. **42**(6): p. 797 - 813.

[150] Ikura, Y. and Gimple, M., Efficient scheduling algorithms for a single batch processing machine. Operations Research Letters, 1986. **5**(2): p. 61-65.
[151] Lee, C.-Y., Uzsoy, R., and Martin-Vega, L.A., Efficient algorithms for scheduling semiconductor burn-in operations. Operations Research, 1992. **40**(4): p. 764-775.
[152] Ahmadi, J.H., et al., Batching and scheduling jobs on batch and discrete processors. Operations Research, 1992. **40**: p. 750-763.
[153] Fowler, J.W., Phillips, D.T., and Hogg, G.L., Real-time control of multiproduct bulk-service semiconductor manufacturing processes. Semiconductor Manufacturing, IEEE Transactions on, 1992. **5**(2): p. 158-163.
[154] Ham, M. and Fowler, J.W., Scheduling of wet etch and furnace operations with next arrival control heuristic. International Journal of Advanced Manufacturing Technology, 2008. **38**(9-10): p. 1006-1017.
[155] Weng, W.W. and Leachman, R.C., An improved methodology for real-time production decision at batch-process work station. IEEE Transactions on Semiconductor Manufacturing, 1993. **6**(3): p. 219-225.
[156] Chandra, P. and Gupta, S., An analysis of a last-station-bottleneck semiconductor packaging line. 1992, Faculty of Management, McGill University.
[157] Uzsoy, R. and Yang, Y., Minimizing total weighted completion time on a single batch processing machine. Production and Operations Management, 1997. **6**: p. 57-73.
[158] Lee, C.Y. and Uzsoy, R., Minimizing makespan on a single batch processing machine with dynamic job arrivals. 1996, School of Industrial Engineering, Purdue University.
[159] Chandru, V., Lee, C.Y., and Uzsoy, R., Minimizing total completion time on batch processing machines. International Journal of Producton Research, 1993. **31**: p. 2097-2121.
[160] Chandru, V., Lee, C.Y., and Uzsoy, R., Minimizing total completion time on a batch processing machine with job families. Operations Research Letters, 1993. **13**: p. 61-65.
[161] Brucker, P., et al., Scheduling a batching machine. Journal of Scheduling, 1998. **1**: p. 31–54.
[162] Li, L., Qiao, F., and Wu, Q.D., ACO-based multi-objective scheduling of parallel batch processing machines with advanced process control constraints The International Journal of Advanced Manufacturing Technology, 2009.
[163] Dobson, G. and Nambimadom, R.S., The batch loading and scheduling problem. Operations Research, 2001. **49**(1): p. 52-65.
[164] Hochbaum, D.S. and Landy, D., Algorithms and heuristics for scheduling semiconductor burn-in operations. 1994, Department of Industrial Engineering and Operations Research, University of California, Berkeley.
[165] Uzsoy, R., Scheduling a single batch processing machine with nonidentical job sizes. International Journal of Production Research, 1994. **32**: p. 1615-1635.
[166] Azizoglu, M. and Webster, S., Scheduling a batch processing machine with

incompatible job families. Computers and Industrial Engineering, 2001. **39**(3-4): p. 325-335.

[167] Sevaux, M. and Peres, S.D., Genetic algorithms to minimize the weighted number of late jobs on a single machine. European Journal of Operational Research, 2003. **151**(2): p. 296-306.

[168] Koksalan, M. and Keha, A.B., Using genetic algorithms for single machine bicriteria scheduling problems. European Journal of Operational Research, 2003. **145**(3): p. 543-556.

[169] Koh, S.G., et al., Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families. International Journal of Production Economics, 2005. **98**(1): p. 81-96.

[170] Cheng, B.-Y., et al. A chaotic Ant Colony Optimization method for scheduling a single batch-processing machine with non-identical job sizes. in Evolutionary Computation. 2008.

[171] Akturk, M.S. and Ozdemir, D., A new dominance rule to minimize total weighted tardiness with unequal release dates. European Journal of Operational Research, 2001. **135**(2): p. 394-412.

[172] El Adl, M.K., Rodriguez, A.A., and Tsakalis, K.S. Hierarchical modeling and control of re-entrant semiconductor manufacturing facilities. in Proceedings of the 35th IEEE Conference on Decision and Control. 1996. Kobe, Jpn.

[173] Liu, H., Fung, R.Y.K., and Jiang, Z., Modeling of semiconductor wafer fabrication systems by extended object-oriented Petri nets. International Journal of Production Research, 2005. **43**(3): p. 471-495.

[174] Zhang, H., et al., An EOPN-based real-time scheduling simulation platform of wafer fabrication system. Journal of Shanghai Jiao Tong University, 2006. **40**(11): p. 1857-1863.

[175] Mason, S.J., Minimizing total weighted tardiness in complex job shops. 2000, Arizona State University.

致 谢

本论文是在导师江志斌教授的悉心指导下完成的，从论文的选题、资料的准备，直到最后的审阅修改，无不凝结着江老师的心血与汗水，在此对江老师表示深深的感谢。江老师渊博的知识、高屋建瓴的学术眼光、严谨的治学态度、孜孜不倦的工作作风，令我受益匪浅。在生活上，江老师对我的关怀与帮助，特别是在我身陷困境时的鼓励与鞭策，都令我终生难忘。

还要衷心地感谢奚立峰教授、蒋祖华教授、王丽亚教授、褚学宁教授、张洁教授、刘晓教授、李娜老师、王国龙老师及其他老师几年来对我的指导和帮助。感谢叶春明教授与陈以增教授对我毕业论文的指导。

感谢刘惠然博士、李衍飞博士、何俊明博士、梁峰博士、张怀博士、耿娜博士、苌群峰博士、刘钢博士、郑俊丽博士、胡鸿韬博士及实验室的刘群亭、刘天堂、刘聪、孟峰、姚世清、王康周、林文进、李程及其他同门兄弟姐妹的帮助。

感谢我的妻子张慧中几年来对我的支持与鼓励，你灿烂的笑容是我坚持的动力。同时感谢我深爱的家人对我的最大宽容与帮助。

郭乘涛

二〇一二年五月

攻读博士学位期间已发表或录用的论文

[1] **Chengtao Guo**, Zhibin Jiang, Huai Zhang, Na Li. Decomposition-based classified ant colony optimization algorithm for scheduling semiconductor wafer fabrication system. Computers & Industrial Engineering, 2012, 62(1), 141-151, **(SCI 源&EI 源，IF: 1.543)**

[2] **郭乘涛,** 江志斌, 张怀. 基于问题分解的蚁群算法在半导体晶圆制造调度中的应用. 上海交通大学学报. 2009 年. 43 卷 11 期 1798 – 1802 **(EI: 20100112617749)**

[3] **郭乘涛,**江志斌. 应用混合蚁群算法求解并行批处理机组批与调度问题. 上海交通大学学报. 2010 年. 44 卷 8 期 1068 – 1073 **(EI:20104213309257)**

[4] **Chengtao Guo**, Zhibin Jiang, Huai Zhang. Decomposition based ant colony algorithm applied to semiconductor wafer fabrication system, 38th International Conference on Computers and Industrial Engineering. 2008. Beijing, China

[5] **Chengtao Guo**, Zhibin Jiang, Hongtao Hu. A hybrid ant colony optimization method for scheduling batch processing machine in the semiconductor manufacturing, Industrial Engineering and Engineering Management (IEEM), 2010 IEEE International Conference on. 1698 – 1701 **(EI: 20110413624867)**

[6] Huai Zhang, Zhibin Jiang, **Chengtao Guo**. An optimized dynamic bottleneck dispatching policy for semiconductor wafer fabrication. International Journal of Production Research, 2009, 47(12), 3333-3343, **(SCI 源&EI 源，IF: 1.033)**

[7] Huai Zhang, Zhibin Jiang, **Chengtao Guo**. Simulation-based optimization of dispatching rules for semiconductor wafer fabrication system scheduling by the response surface methodology. International Journal of Advanced Manufacturing Technology, 2009, 41(1-2), 110-121. **(SCI 源& EI 源，IF: 1.068)**

[8] 张怀, 江志斌, **郭乘涛**, 刘惠然. 基于 EOPN 的晶圆制造系统实时调度仿真平台. 上海交通大学学报. 2006 年 11 期 40 卷 11 期 1857-1863. **(EI: 20070310372734)**

[9] 张怀, 江志斌, **郭乘涛**. 面向瓶颈的半导体晶圆制造系统派工策略及参数优化. 上海交通大学学报. 2007 年 8 期 41 卷 8 期 1252-1257. **(EI: 20073910834797)**

[10]姚世清, 江志斌, **郭乘涛**, 胡鸿韬. 晶圆制造系统中 Lot 加工序列优化的蚁群算法. 上海交通大学学报. 2008 年 42 卷 10 期 1655 – 1659 **(EI: 20084811749958)**

[11]Huai Zhang, Zhibin Jiang, **Chengtao Guo**, and Huiran Liu. An extended object-oriented Petri nets modeling based simulation platform for real-time scheduling of semiconductor wafer fabrication, 2006 IEEE International Conference on Systems, Man and Cybernetics

3411-3416. Taipei, Taiwan. **(EI: 073510786993) (ISTP: BGK85)**

[12] Huai Zhang, Zhibin Jiang, **Chengtao Guo**. Simulation based real-time scheduling method for dispatching and rework control of semiconductor manufacturing system. 2007 IEEE International Conference on Systems, Man and Cybernetics. 2901-2905. Montreal, Quebec, Canada. **(EI: 081311169692) (ISTP: BHP06)**

[13] Hongtao Hu, Zhibin Jiang, **Chengtao Guo**, Ran Liu. A decomposition based algorithm for the scheduling problem in wafer fabrication system. Industrial Engineering and Engineering Management (IEEM), 2010 IEEE International Conference on. 2066 – 2070 **(EI: 20110413624919)**

投稿论文

[1] **Chengtao Guo**, Zhibin Jiang. A Hybrid ACO Method for Scheduling Jobs on a Parallel Batch Machine with Incompatible Job Families and Unequal Ready Times.（投稿 International Journal of Production Research）

其他研究成果

[1] 中国计算机软件著作权“半导体晶圆制造系统实时调度仿真（Real-time Scheduling Simulation, ReS^2）平台”（登记号：2005SR08364）。

1