大规模复杂生产调度问题瓶颈分解方法研究

摘 要

生产调度问题是在时间上优化分配机器资源给相互竞争的工件，由于其强大的工程应用背景和优化求解的难处理性(NP-hard)，长期受到制造领域和学术界的高度重视。随着制造规模和复杂程度的不断提高，大规模复杂生产调度问题已成为当前制造业面临的难题，存在建模和优化求解的困难。

分解方法是处理大规模复杂生产调度问题的一种常用方法，但传统的分解方法收敛速度慢，容易陷入局部最优。目前已有一些瓶颈调度方法试图从全局优化的角度处理大规模问题，但非常有限，在算法设计、理论分析和应用方面都存在很大的研究空间。本文利用瓶颈思想，结合分解方法，提出了一种瓶颈分解方法，并对大规模复杂生产调度问题进行了深入的研究。

主要工作包括以下几个方面：

- 在大系统分解思想的指导下，给出了车间调度问题机器层分解方法的一般描述，并指出传统分解方法在求解大规模调度问题时可能存在的不足。利用能力不均衡生产线存在瓶颈的特性，剖析瓶颈思想的内在机理，结合大系统分解思想提出了瓶颈分解方法的一般框架，并指出该方法的关键技术。
- 在分析单瓶颈 Flow shop 生产线特点的基础上，提出了一种单瓶颈分解算法。该方法将 Flow shop 上的机器分为瓶颈机、上游非瓶颈机和下游非瓶颈机。松弛非瓶颈机加工能力约束，原 Flow shop 问题简化为一个单机调度问题。瓶颈机的调度可以通过优化求解该简化问题获得。根据瓶颈机的调度结果，上游非瓶颈机采用有效的分派规则从后往前依次调度，而下游非瓶颈机则采用规则从前往后依次调度。随后我们给出了非瓶颈机冗余能力的条件，证明满足此条件下的 Flow shop 问题的最优调度为排列调度，即非瓶颈机上工件的加工安排与瓶颈机的调度安排一致。大量的仿真结果显示单瓶颈分解算法可以在较快的时间内获得较好的解。
- 针对单瓶颈 Job shop 调度问题，提出了一种单瓶颈分解算法。该方法利用生产线主导瓶颈的特点将 Job shop 中的机器分为瓶颈机和非瓶颈机。假设非瓶

颈机加工能力无限大，将非瓶颈机上的工序集结成一个时间迟延环节，原问题则简化为一个带到达时间和传递时间约束的单机调度问题。瓶颈机上工件的加工安排可以通过优化求解此单机问题获得，而非瓶颈机则采用有效的启发式算法调度。通过调整瓶颈机上工序的到达时间和传递时间协调瓶颈机与非瓶颈机之间的关联。仿真结果显示该算法求解瓶颈程度较高的 Job shop 问题非常有效。

- 针对能力较不均衡的 Job shop 生产线，提出了一种约束调度算法。该算法给出了一种迭代辨识约束机(包括长期瓶颈和暂时瓶颈)的方法。每次迭代，采用有效的规则调度非约束机，并根据调度结果从中辨识约束机，该过程迭代进行直至不存在新的约束机。约束机模型可描述为一个带时间迟延的多机调度问题，该问题的最优解是原问题最优解的一个下界。随着约束机不断辨识，相应的约束机模型也迭代更新，我们证明约束机子问题的最优解在迭代过程中不断提高，朝着原问题最优解的方向逐步逼近。与传统的移动瓶颈方法相比，约束调度算法在提高计算效率的同时保证了调度的优化性能。大量的仿真比较研究显示，该算法可以在调度质量和计算时间之间获得一个好的均衡。
- 针对约束调度算法无法求解的更大规模 Job shop 调度问题，提出了一种滚动瓶颈分解策略。该方法在瓶颈分解方法的基础上，结合滚动时域分解的思想，将整个调度时域分解为若干决策子窗口，每个时间窗口对应的子问题采用瓶颈分解方法求解。该方法从空间层和时间层对大规模问题进行分解，它为求解超大规模 Job shop 调度问题提供了一种新的解决思路。仿真结果验证了滚动瓶颈分解方法的有效性。
- 将滚动瓶颈分解方法应用至一个更复杂的 Job shop 调度问题，即半导体生产线，针对半导体生产线的机器加工类型不一，对分解后不同类型的子问题给出了相应的求解算法，并对经典模型 MINIFAB 进行详细的仿真测试研究。

关键词:瓶颈，生产调度，分解方法，移动瓶颈法，滚动时域法，约束理论

STUDY ON BOTTLENECK-BASED DECOMPOSITION PROCEDURE FOR LARGE-SCALE PRODUCTION SCHEDULING PROBLEMS

ABSTRACT

Production scheduling problem is to optimally allocate manufacturing machines to competitive jobs over time. Due to its important practical value and NP-hard characteristic, it draws high attention of manufacturers and academe. With the problem size and complexity increasing, the large-scale complex production scheduling has become one of the hardest problems in manufacturing system. It is difficult in modeling and optimization.

The decomposition method is a generic approach to solve the large-scale production scheduling problem. However, it may lead to a local optimum and has a low convergence speed. Although some bottleneck scheduling methods have been studied to deal with the problem from the viewpoint of global optimization, the research is elementary and limited. There still exist large spaces in both the theoretical study and practical application. In this thesis, the bottleneck-based decomposition method for large-scale complex production scheduling problems is studied.

The main research works of this dissertation are as follows.

- Based on the decomposition theory, the machine-based decomposition method for production scheduling problems is analyzed and its shortcomings are pointed out. By utilizing the bottleneck property in the unbalanced production line, the generic framework of the bottleneck-based decomposition procedure is proposed. This procedure combines bottleneck theory with decomposition method. Then its key techniques are given.
- For the flow shop problem with one bottleneck machine, a bottleneck-based decomposition procedure is proposed. In this method, the machines on the flow shop are divided into a bottleneck machine, upstream and downstream non-bottleneck machines. By relaxing the capacity constraints of non-bottleneck machines, the flow shop problem is reduced into a single-machine scheduling problem. The bottleneck scheduling can be obtained by solving the reduced problem optimally. While the upstream non-bottleneck machines are scheduled

backward based on the bottleneck scheduling, and the downstream non-bottleneck machines are scheduled forward based on the bottleneck scheduling. Then we give some conditions to describe the redundant capacities of the non-bottleneck machines, and prove that the optimal scheduling to the flow shop is a permutation scheduling under these conditions. In other words, the sequence of jobs on each non-bottleneck machine is the same as that of jobs on the bottleneck machine. Computational results show that the bottleneck-based decomposition algorithm can obtain a good solution in quite short time.

- For the job shop problem with one bottleneck machine, a bottleneck-based decomposition procedure is proposed. In this method, the machines on the job shop are divided into a bottleneck machine and non-bottleneck machines based on the bottleneck property. Under the assumption that the capacities of non-bottleneck machines are infinite, operations on the non-bottleneck machines are aggregated into a time delay. The job shop problem is then reduced into a single-machine scheduling problem with release time and delivery time constraints. The bottleneck scheduling can be obtained by solving the reduced problem optimally, while the non-bottleneck machines are scheduled around the bottleneck scheduling by some effective dispatching rules. The conflict between the scheduling of the bottleneck machine and non-bottleneck machines can be coordinated by adjusting the release times and delivery times of operations on the bottleneck machine. Computational results show that the bottleneck-based decomposition procedure can obtain a good solution for the job shop problem with high bottleneck degree.

- For the unbalanced job shop problem with multiple constraint machines (JSPMC), a constraint scheduling procedure is proposed. In this procedure, a new strategy is given to identify the constraint machines iteratively. At each iterative procedure, the non-constraint machines are scheduled with an effective dispatching rule, and a constraint machine is identified based on the scheduling results of the non-constraint machines. This procedure continues until no constraint machine is identified. Then the constraint machines can be formulated as a multiple-machine scheduling problem with time lags. Its solution provides a low bound for the JSPMC. In addition, we prove that the low bound increases iteratively with the set of constraint machines increasing, and it is close to the optimal solution of the JSPMC. Extensive computational results indicate that the proposed constraint scheduling algorithm can obtain a better tradeoff between solution quality and computation time comparing with various versions of the shifting bottleneck (SB) methods for the JSPMC.

- For the larger scale job-shop problem, which is difficult to be solved by the constraint scheduling procedure, we propose a bottleneck-based rolling scheduling method based on the bottleneck and temporal decomposition strategies. In this method, the scheduling horizon is divided into some smaller time windows. The subproblem to each time window is still a job shop problem, which can be solved by the constraint scheduling procedure. Since the large-scale problem is divided in the level of space and time, it provides a new idea to solve large-scale job shop scheduling problem. Computational results show that the method is effective.
- The bottleneck-based rolling scheduling procedure is applied to a more complex job shop problem, i.e. semiconductor manufacturing process. Since the workcenters in the semiconductor manufacturing process are different, various solution procedures are given to solve their corresponding subproblems. At last, this method is tested in a classic MINIFAB model.

KEY WORDS: Bottleneck, production scheduling, decomposition method, shifting bottleneck (SB) procedure, rolling horizon procedure (RHP), theory of constraints (TOC)

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：

日期：　　年　月　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：　　指导教师签名：

日期：　　年　　月　　日　　日期：　　年　　月

第一章 绪论

1.1 引言

调度问题是在时间上优化分配稀缺资源给相互竞争的任务[1]。它一般描述为在满足资源和任务约束条件的前提下优化某一或某些性能指标[2]。根据资源和任务的物理特性不同，调度问题包含生产调度、车辆调度、CPU 调度等诸多分支。它是一类典型的组合优化问题。

上个世纪五十年代，调度问题早期的研究工作主要依赖数学方法，Graham 和 Johnson 在调度理论方面做了大量开创性的工作[3,4]。此后调度问题成为运筹研究中最活跃的领域之一，涌现了一批重要的理论研究成果[2,5,6]。由于其强大的理论基础和重要的实用价值，调度问题长期受到数学、计算科学、管理科学以及工程应用等领域的高度重视。

本文研究的背景是生产调度问题。生产调度是实现制造业运筹、管理与优化技术的核心。它是在时间上对一组可用的制造资源(机器)进行加工任务(工件)的安排，将工件分配至相应的机器上，确定各机器上加工操作(工序)的加工次序和开工时间，使得某一性能指标最优。在执行这些加工任务时需要满足某些物理约束，如工件的加工顺序约束，机器加工能力限制等。同时也要满足某些非物理约束，如工件到达时间约束，客户交货期限需求等。因此生产调度问题可表述为在一些等式或不等式约束构成的离散解空间中，寻找目标函数值最优的解。它是一类重要的组合优化问题，运筹学称之为排序问题[7]。除少数特殊问题，大部分生产调度问题的计算复杂度为 NP-hard[5,6]，存在优化求解的困难。

在经济全球化的今天，现代制造业面临着前所未有的竞争压力。制造规模和复杂程度都在不断提高，客户需求日趋多样化，这对制造方提出了新的要求，现实的需求不断地给这一领域的研究带来大量新的急需解决又极端困难的问题。大规模复杂生产调度问题已成为当前研究的一个热点。

本章首先介绍生产调度问题的特点，并简要综述三类典型车间调度问题(Flow shop, Job shop 和半导体生产线)的研究进展。随后对大规模静态车间调度问题的分解方法进行详细分类介绍，并分析不同分解方法的特点。接着介绍了生产线瓶颈现象和特点，给出应用瓶颈方法求解大规模车间调度问题的可行性和研究思路。最后给出全文的主要研究内容和章节安排。

1.2 生产调度问题的特点及研究进展

1.2.1 生产调度问题的分类与特点

生产调度问题是在可用机器资源上，对工件进行时间安排，使得某一或某些调度优化准则最优。因此，机器、工件和优化准则是生产调度问题的三个基本要素。Graham等(1979)提出生产调度问题分类和描述的三参数表述法，即三域表示法$\alpha|\beta|\gamma$，其中参数α，β和γ分别表示机器加工环境、机器和工件的加工特性、优化准则[8]。

(1) 参数α表示机器加工环境，主要的机器环境描述如下：

- 单机(1或缺省)：单机情况是所有机器环境中最简单的一种情况，也是所有复杂机器环境中一种特殊情况。
- 同型并行机(P_m)：m台具有相同加工速度的同型号(identical)并行机。
- 同类并行机(Q_m)：m台具有不同加工速度但此速度不依赖于工件的同类(uniform)并行机。
- 非同类型并行机(R_m)：m台随工件不同加工速度也不相同的非同类型(unrelated)并行机。
- 流水作业车间(F_m)：即Flow shop，m台串联机，其中工件按相同的工艺路径依次在每台机器上加工一次。如果机器前工件队列按先进先出(first in first out, FIFO)规则依次通过机器，该流水作业车间称为排列流水作业车间(permutation flow shop)，在参数域β中包含*prmu*。
- 柔性流水作业车间(FF_c)：它是流水作业车间和并行机环境的综合，即c个串联的加工中心，工件按相同的工艺路径依次在每个加工中心上加工一次，每个加工中心有一组同型并行机。
- 异序作业车间(J_m)：即Job shop，m台串联机，其中工件按各自给定的工艺路径依次在每台机器上加工一次。每个工件也可以在同一机器上加工多次，此时在参数域β中包含*recrc*。
- 柔性异序作业车间(FJ_c)：它是异序作业车间和并行机环境的综合，即c个串联的加工中心，工件按各自给定的工艺路径依次在每台机器上加工一次，每个加工中心有一组同型并行机。
- 自由作业车间(O_m)：即Open shop，m台串联机，其中工件可按任意路径在

每台机器上加工一次。

(2) 参数β表示加工约束和特征，给出参数域β中一些主要的约束描述：

- 到达时间(r_i)：表示工件i不能在到达时间(release time)r_i之前开工。
- 顺序相关安装时间(s_{ik})：s_{ik}表示工件i和k之间需要安装时间，该时间大小与工件的加工顺序有关。
- 抢占($prmp$)：抢占(preemption)表示一个工件一旦在一台机器上开始加工，在它完工之前可以允许停止加工此工件，而加工另一工件。
- 排列($prmu$)：该约束可能出现在Flow shop环境，它表示Flow shop上所有机器上工件的加工顺序相同。
- 重入($recrc$)：重入(reentrant flow)情况一般出现在Job shop或柔性Job shop环境，它表示一台机器在同一机器或加工中心上加工多次。

(3) 参数γ表示优化准则，调度优化目标函数多采用工件完工时间(completion time)C_i的正则函数$f(C_i)$，一般分为两类：极值目标和求和目标。给出常用的优化目标函数如下：

- 调度时间表长度(makespan, $C_{\max}$)：定义为$C_{\max}=\max_{i\in N}C_i$，其中$C_i$为工件$i$的完工时间。最小化makespan意味着尽可能获得高的机器利用率。
- 最大滞后时间(maximum lateness, $L_{\max}$)：定义为$L_{\max}=\max_{i\in N}L_i$，其中$L_i=C_i-d_i$，表示工件$i$的滞后时间(lateness)，$d_i$为工件$i$的交货期限(due date)。该性能指标可以衡量破坏交货期限最严重的程度。
- 加权完工时间之和(total weighted completion time, $\sum w_iC_i$)：它可衡量调度中总库存费用。一般完工时间之和与流通时间(flow time)之和可以相互转化。
- 加权拖期时间之和(total weighted tardiness, TWT)：定义为$TWT=\sum\omega_iT_i$，其中工件i的拖期时间(tardiness)$T_i=\max(C_i-d_i,0)$，权重(weight)为ω_i。该性能指标可以用于评价不同重要程度的客户交货需求的满意度。

给出一个例子，$J_m|r_i|L_{\max}$表示机器个数为m的Job shop环境，工件具有不同的到达时间r_i，优化目标为最小化最大滞后工件的调度问题。

根据机器环境不同(α参数)，可将生产调度问题分为单机($\alpha=1$)和多机调度问题。其中多机调度问题又可分为两类：并行机和串联机。工件在m台并行机上的加工只需

要在m台并行机上任一台机器加工一次即可；而工件在m台串联机上的加工则需要每个工件在m台串联机中的每一台机器上都加工一次。其中并行机又可分成三类：同型机(P_m)、同类机(Q_m)和非同类机(R_m)。而串联机也可分为三类：自由作业车间(O_m)、流水作业车间(F_m)和异序作业车间(J_m)。

车间调度问题是生产调度问题中最复杂、也是实际生产过程中最常见的一类问题。单机调度和并行机调度问题均可以看作是车间调度问题的一类特殊情况(车间数为1)。车间调度问题的复杂性主要体现在：

- 加工环境复杂。车间中机器的加工类型不一，包括单机、并行机、批处理机等。而且不同的加工车间工艺流程不同，如Flow shop中每个工件按相同的工艺路径依次通过机器，Job shop中每个工件按照不同的工艺路径依次通过机器，最复杂的是半导体生产车间，一个工件往往需要在同一台机器加工多次。

- 加工约束繁多。车间调度问题需要满足生产制造中各种约束，包括物理约束如工件加工顺序约束、机器加工能力约束等，管理约束如订单的投放时间约束，市场约束如客户的交货期限约束等。这些约束之间相互关联，相互制约，这增加了车间调度问题建模和求解的复杂度。

- 生产规模很大。一般来说实际生产线包括几十个加工车间，每个加工车间有成百台机器，需要加工上千个工件。而且每个工件的加工步骤和加工时间都很长，即使可对该调度问题建立数学模型，该模型的维数很高。

- 生产过程不确定性。实际环境中存在调度时刻的信息不确定和动态扰动情况，如工件加工时间的动态不确定，定单的不确定(订单动态到达，订单交货期动态更改)，机器状态的不确定(可能发生机器故障)等，使得调度问题具有动态和随机的特点。

- 生产目标多样化。评价一个车间调度优劣的指标往往不是单一的，如机器利用率越高越好、在制品(work-in-process, WIP)库存越少越好、制造周期(cycle time)越短越好、客户满意度越高越好等，不同的评价指标反映了对生产的实际要求。调度问题的目标一般分为三类：基于工件加工完成时间的目标、基于工件到期时间的目标和基于生产成本的目标。而多个目标之间往往是有冲突的，这使得调度问题的建模和求解变得更为复杂。

- 计算复杂性。大多数调度问题，即使是单机调度问题，其计算复杂度都是NP-hard。很难找到一个多项式算法优化求解这类问题，而以上的五个特点更加剧了调度问题优化求解的难度。

因此，车间调度问题可视为一高维、强关联、多目标的复杂大系统问题。它存在着建模和优化求解困难。本文主要研究三类典型的车间调度问题(Flow shop, Job shop 和半导体生产线)，下面简单介绍这三类问题目前的研究进展。

1.2.2 Flow shop 调度问题的研究进展

在许多制造和装配生产线上，工件按相同的工艺路径依次在一组顺序排列的加工中心上加工，称此生产过程为 Flow shop。Flow shop 调度问题是确定每台机器上工件的加工顺序使得某一期望的目标值最优，它是一类典型的组合优化问题。

按相邻机器之间缓存的存储能力的不同，Flow shop 问题可分为无限中间存储能力的流水作业车间问题(flow shop with unlimited intermediate storage)、有限中间存储能力的流水作业车间问题(flow shop with limited intermediate storage)、有阻塞的流水作业车间问题(flow shop with blocking)和零等待流水作业车间问题(flow shop with zero wait)[6]。如果加工中心的机器个数超过一台，则该 Flow shop 问题称为柔性 Flow shop 问题。如果每台机器上工件的加工顺序都相同，则该 Flow shop 称为排列 Flow shop 问题。本文研究的问题具有无限中间存储能力、非排列、非柔性的 Flow shop 调度问题，即 Flow shop 中每个加工中心只有一台机器，相邻机器之间缓存的存储能力无限大、而且不同机器上工件的排序可以不同。我们称之为经典 Flow shop 调度问题。

自 1954 年 Johnson 对 $F_2 \| C_{\max}$ 调度问题提出一个多项式时间算法(即 Johnson 规则)以来[3]，Flow shop 调度问题进行了大量的研究工作[9-26]。Wagner(1959)对 $F_m \| C_{\max}$ 问题建立了整数规划模型[9]，随后 Garey 等(1976)证明 $F_3 \| C_{\max}$ 为 NP-hard[10]。Monma 和 Rinnooy Kan(1983)综述了具有不同特殊结构的 $F_m \| C_{\max}$ 问题[11]。而不同目标函数的 Flow shop 调度问题也随之进行了研究[12-13]。

自 Palmer(1965)对 Flow shop 调度问题提出了启发式算法以来[14]，大量的构造启发式算法相继提出[15-20]。目前 Flow shop 调度问题启发式算法的研究主要采用智能搜索算法和进化算法等改进式启发式方法[21-26]，采用禁忌搜索(tabu search, TS)[21]、模拟退火(simulated annealing, SA)[22]、遗传算法(genetic algorithm, GA)[23]、蚁群算法(colony system)[24]以及混合算法[25, 26]改进初始解。

1.2.3 Job shop 调度问题的研究进展

Job shop 调度问题是一类典型的组合优化问题，除了少数特殊结构的问题，大部分 Job shop 调度问题是强 NP-hard[6]。基于 Johnson(1954)提出的求解 Flow shop 问题的算法[4]，Akers(1956)，Jackson(1956)，Hefetz 和 Adiri(1982)分别对规模为 $2\times m$，

$n\times2$(每个工件的工序不超过两个)和$n\times2$(所有工序的加工时间为 1)的 Job shop 调度问题提出了相应的多项式求解算法[27-29]。这些工作构成了经典调度理论发展的基础。大量的研究工作集中在 Job shop 调度问题的求解上。图 1-1 列出了 Job shop 调度问题的各种算法[30]。

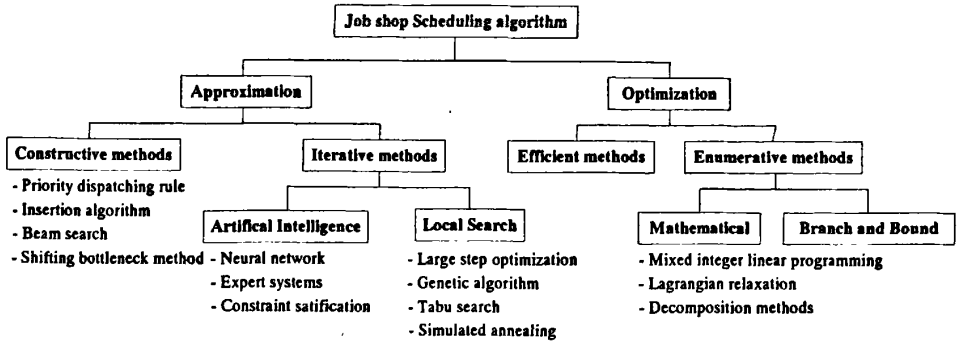


图 1-1. Job shop 调度算法

Fig 1-1. Scheduling algorithms in Job shop problems

Roy 和 Sussmann (1964)对 Job shop 问题最先提出一种析取图(disjunctive graph)描述[31]。Balas(1969)基于此析取图模型提出了一种枚举方法[32]。然而该方法在很多情况下无法获得问题的可行解。Carlier 和 Pinson(1989)提出了一种有效的分枝定界(branch and bound, B&B)算法，它首次得到标准算例 FT10 的最优解，解决了 25 年来无法求解的问题[33]。随后比 FT10 更复杂的算例也可以通过其它分枝定界方法求解[34, 35]。分枝定界算法虽然可以获得问题的最优解，但是它的计算时间随问题规模的增大呈指数增长，它只适用于小规模问题的求解。

由于精确算法的局限性，大量的研究工作集中在 Job shop 问题近似解的获取上，大量的启发式算法相继提出。Job shop 调度问题的启发式算法主要分为两类：构造型启发式算法和改进型启发式算法。

最早的构造型启发式算法是优先分派规则(priority dispatching rule)，该方法易于实现，计算量小，颇受工程人员青睐。Giffler 和 Thompson(1960)提出了一种优先分派规则框架[36]。该方法对所有工序分配优先级，每次选择优先级最高的工序最先加工，不同的分派规则相继提出[37-39]。由于分派规则只利用当前机器和工件的信息，获得解的质量往往不高。随着问题规模和复杂度的增加，解的质量变得更差。移动瓶颈(shifting bottleneck, SB)方法是一种典型的构造型启发式算法，最早由 Adams 等(1988)提出[40]，该方法将 Job shop 调度问题分解为若干单机子问题，通过依次确定每台机

器上工件的加工顺序获得 Job shop 问题的调度解。该方法可以获得较好的调度性能，但计算时间较长。束搜索(beam search)算法是另一种构造型启发式算法，它可视为一种启发式的分枝定界算法[41]。对每层的节点，该算法只保留部分好节点，而其它节点永久删除。由于搜索树上大量的分枝被砍枝，其计算时间与问题的规模呈多项式关系。在此基础上 Sabuncuoglu 和 Bayiz 提出了一种滤波束搜索算法(filter beam search)进一步改进束搜索算法[42]。

改进型启发式方法分为两类：人工智能(artificial intelligence)[43]和局部搜索(local search)[44]方法。基于人工智能的启发式方法主要包括神经网络(neural networks)[45]、专家系统(expert systems)[46]、约束满足(constraint satisfaction, CS)[47]和蚁群算法[48]等。神经网络方法采用一个简单但是具有并行处理的关联单元结构，它具有分布处理的能力，但它存在搜索震荡、收敛慢的缺陷[45]。专家系统对解决调度的实时性和智能性提供了新的辅助手段。它根据系统当前的状态和给定的优化目标，对知识库进行有效的启发式搜索和并行模糊推理机制，避开了繁琐的计算，并选择最优的调度策略，为在线决策提供支持[46]。约束满足(CS)方法是一种迭代近似算法，它应用许多规则和策略至分枝定界算法中，通过约束限制变量选择的顺序和被选变量可能的取值来减少搜索空间[47]。尽管约束满足方法试图找到一个可行调度使得满足交货截止时间(deadline)，但是存在约束描述和搜索震荡的困难。

局部搜索算法主要包括遗传算法[49]、阈值算法[50]、模拟退火算法[51]和禁忌搜索算法[52]等。遗传算法(GA)是一种模仿生物群体进化过程的优化调度方法，给定一组初始解作为一组种群，通过选择、交换和变异等遗传操作来搜索最优解。然而遗传算法的早熟和搜索效率低问题是它的主要缺陷，这类方法一般情况下收敛于最优调度的速度很慢，与多数启发式方法类似，很难确定是否或何时可以得到最优调度。模拟退火算法(SA)是一种阈值算法，它模仿晶体结晶的冷却过程。由于该算法能以一定的概率接受差值，因而可能跳出局部极小，但算法的收敛速度很慢。禁忌搜索算法(TS)从一个可行解出发，构造其邻域，选择邻域中目标函数最优的状态作为下一状态，并把这一移动的反向操作存入一个禁忌列表中，表中的移动在以后若干步内都不允许执行，直到它从表中除去。每搜索一次，更新一次禁忌列表。由于禁忌列表的限制，有可能跳出局部极小。

1.2.4 半导体生产调度问题的研究进展

半导体生产线(semiconductor manufacturing process)是目前制造业最复杂的生产过程之一，它有不同于 Flow shop 和 Job shop 的特殊性和复杂性[53]：(1) 它由不同加工特性的设备组成，按同时可加工晶圆(wafer)数可分为单片机器，单批(lot)设备以及

多批(batch)设备。而且有些加工过程(如离子注入)的物理特性使得相邻机器操作之间的安装时间与工件的加工顺序相关。(2) 产品的制造流程复杂，通常是相当数量的多种产品同时在生产线上流动，而且一个产品的完工可能需要多次访问同一台机器，生产存在重入加工流(reentrant flow)；(3) 产品的成品率(yield)随机波动性大；此外还有随机性的重做和半成品报废。由此可见，半导体生产线是不同于传统 Flow shop 和 Job shop 的第三类生产线类型。它可以描述为一个复杂的 Flow shop/Job shop 模型[54]。

Mason 建立了该问题的混合整数线性规划模型[54]，并应用 CPLEX 软件中数学规划优化算法求解该问题。由于绝大多数半导体生产线调度问题是强 NP-hard，其最优解往往很难找到。而且在实际应用中由于生产过程中各种参数的飘移，会使模型本身不具有精确性等因素，一般也没有必要去找到最优解，只需要找到满足一定要求的启发式解或近似解。

Uzsoy 等人从性能评价、生产计划和车间层控制三个方面综述了近年来半导体生产线的研究成果[55,56]。性能评价是评估现有系统的性能，而不是优化系统性能的指标。它是生产计划和车间层控制的重要输入，也是优化系统配置的主要指导。生产计划一般指高层长期生产能力的安排和分配，车间层控制是生产计划的细化和实现，它将基于能力的较长期计划转化为实际工件的实时加工序列。车间层控制主要有两类决策任务：一类是决定何时投入多少工件进入生产线，称为投料策略(release policy)。另一类是安排生产线中各类工件在各类设备上的实际加工序列和开始时间，称为实时调度策略(scheduling policy)。

输入调节方法，即投料控制策略，是保证系统平稳生产的重要前提，对投料合理控制将为后续优化调度提供重要保证，达到提高设备产量和设备利用率、增加系统产出的目的。在半导体制造车间，常用的较为简单的投料策略有固定时间间隔投料、按泊松分布随机投料[57]，这些都是静态方法。最简单的动态方法是保持 WIP 个数为常数的固定批量法(constant WIP, CONWIP)[58]。当一个工件离开系统时，一个同类的新工件有权进入系统。CONWIP 策略保持生产线 WIP 个数整体水平不变的前提下，考虑了多类工件对各设备负载均衡的影响，对于不同种类的工件，其 WIP 数可以不同。此外 Wein(1988)提出了一种基于瓶颈(bottleneck)状态的投放方法，即避免瓶颈设备饥饿法(starvation avoidance method)[59]。该方法根据瓶颈机的加工状况决定投料情况，把重点放在瓶颈工序上，保证瓶颈工序不发生停工待料，提高瓶颈加工中心的利用率，从而得到最大的有效产出。

分派规则是一种实时调度策略。当出现设备空闲时，按某一决策选择下一个竞争使用该设备的工件。Kumar 等将半导体生产线近似为多级排队网络模型[60-63]。文献[61,62]从分析特殊排队网络稳态性出发，给出了多种调度策略：LBFS(last buffer first

serve)、LWNQ(least work next queue)等。文献[63]从降低制造周期目标出发，提出了一类基于最小松弛波动光滑 FSVCT 策略(fluctuation smoothing policy for the variance of cycle-time)，它比 FIFO 降低制造周期方差 52%。基于经验的规则调度虽然使操作工在选择加工工件时有章可循，并且取得了一定的优化效果，但由于其只利用当前生产线的局部信息，获得的调度性能往往不尽如人意。因此需要耗费一定的计算费用来获得好的近似解。

Uzsoy 等最早采用移动瓶颈方法调度半导体生产线，目标函数为最小化最大滞后时间(L_{max})[64,65]。Mason 等提出了一种改进移动瓶颈方法求解半导体车间层调度问题，目标函数为最小化加权拖期时间之和(TWT)[54, 66-69]。他们根据半导体生产线加工特点，改进已有的析取图描述。在改进的析取图上考虑批处理机、重入、工件顺序相关安装时间等情况。改进的移动瓶颈方法将整个半导体生产线分解为若干子问题，每个子问题的模型对应不同的加工中心如单机、并行机、批处理等。通过依次地确定每个加工中心中机器上工件的加工安排获得半导体生产线上所有工序的加工序列和开工时间。

半导体车间层调度的迭代启发式算法主要有两类：人工智能[70-73]和局部搜索[74-76]。人工智能包括专家系统[70]、模糊逻辑[71]、Petri 网[72]和神经网络[73]。而局部搜索主要有模拟退火[74]、禁忌搜索[75]和遗传算法[76]。此类算法综合实际生产经验和理论分析，在生产实际中得到较好的应用，但它仍存在搜索速度慢的缺点。

1.3 大规模静态车间调度问题分解方法研究

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。

1.3.1 基于模型结构的分解

上世纪五六十年代，调度问题的理论研究多采用数学规划方法[2]。调度问题往往可以用一个混合整数线性规划模型描述。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，即 $A \cdot X \le B$。如果直接采用分枝定界或割平面方法(cutting plant)来求解此类大规模混合整数规划问题，可能会发生计算机内存不够的情况，无法获得原问题的解。因此需要对建立的数学规划模型进行一定的

分解，将原问题分解为一系列更易求解的小规模问题。

在实际的车间调度问题中，一个大系统通常由若干个子系统(车间)组成。这些子系统除了有其各自的内部约束关系外，各个子系统之间还有相互的关联制约作用。因此体现在其数学规划模型中，约束方程组的系数矩阵 A 往往具有对角块的形式(如图1-2所示)。

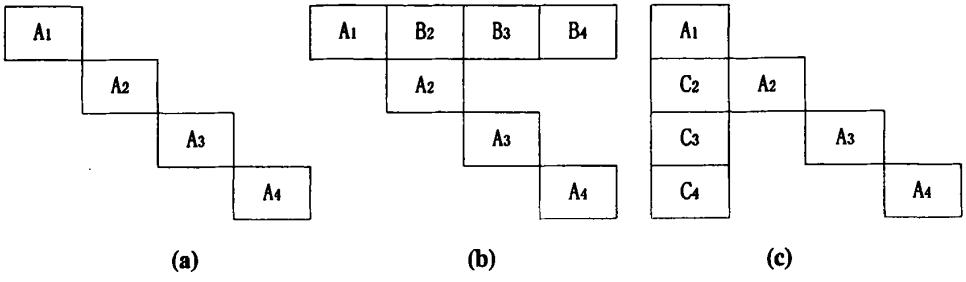


图 1-2. 约束矩阵的三种典型结构: (a) 块对角形式 (b) 主块三角形式 (c) 对偶块三角形式

Fig. 1-2. Three structures of constraint matrix: (a) Block diagonal (b) Primal block angular (c) Dual block angular

如果约束矩阵 A 具有图1-2(a)的结构特性，则该模型可自然分解为四个子系统，每个子系统对应的约束矩阵分别为 A_1, A_2, A_3 和 A_4，该模型中子系统之间没有关联作用。子问题的解可直接组合构成原问题的解。

如果约束矩阵 A 具有图1-2(b)的结构特性，它由关联约束(connecting constraints)和对角块约束构成。由于存在关联约束，该模型不能直接分解。1960 年 Dantzig 和 Wolfe 提出一种 D-W 分解方法求解此类问题，也称列生成法(column generation)[77]。D-W 分解方法的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，它可结合分枝代价(branch and price)求解大规模整数规划问题[78-80]。

拉格朗日松弛(lagrange relaxation, LR)方法是求解具有上述结构特性规划问题的另一种方法。Brooks 和 Geoffrion 描述了 LR 方法与 D-W 分解方法之间的关系[81]。LR

方法将耦合约束松弛为软约束，即将耦合约束乘以拉格朗日乘子添加至目标函数中，则原优化问题可分解为一个主优化问题和若干子问题。在子问题求解中，固定拉格朗日乘子，求解子问题得到原问题的下界，然后在主对偶问题中，根据子问题的解调整拉格朗日乘子。LR 方法最初由 Held 和 Karp 提出用于解决旅行商问题[82,83]。Fisher 将其成功地应用于求解调度问题[84,85]。随后 Peter 将其推广应用求解不同类型的生产调度问题[86-88]。LR 方法是求解整数规划问题的一种有效的优化算法[89]。

如果约束矩阵 A 具有图 1-2(c)的结构特性，约束中存在耦合变量(connecting variables)。对此通常采用 Bender 方法进行分解。它将原优化问题分解为一个主优化问题及若干子问题。在子问题中固定耦合变量，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 割约束后的耦合变量[90]。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。

1.3.2 基于调度实体的分解

按车间的基本物理单元(实体)分主要包括工序、工件和机器(或加工中心)，因此基于调度实体的分解方法包括三类：基于工序层的分解、基于工件层的分解和基于机器层的分解方法[91]。

工序是生产车间中最小的加工实体，工序层分解适应于分派机制的调度方法中。因此，基于工序层的分解方法多为分派启发式调度算法。它将整个调度问题分解为若干小规模问题，每个问题对应一道工序。一旦机器空闲，选择优先级最高的工序至该机器上加工，该过程迭代执行直至所有的工序均安排完。不同的启发式规则可用于设置工序的优先级[37-39]。

工件层分解方法中，每个子问题对应一个工件，即每个子问题的解为每个工件所有工序的开工时间。ISIS 调度系统采用基于工件层的分解方法。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，该方法可以减少库存[92]。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[86-88]。

另一种常用的分解方法是基于机器层的分解。该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。通过依次确定每台机器上工件的加工安排获得整个车间的调度安排。移动瓶颈方法是一种最典型的机器层分解方法。该方法最初是由 Adams 等(1988)针对 $J_m \| C_{\max}$ 提出[40]。Dauzere-Peres 和 Lasserre(1993)改进移动瓶颈算法，对子问题考虑迟延顺序约束(delay precedence constraints, DPC)[93]。随后 Balas 等(1995)提出了一种分枝定界算法优化求解考虑 DPC 的子问题[94]。Demirkol 等(1997)对不同加工加工类型(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[95]。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[96]。

基于调度实体的分解方法一般物理意义比较明显，每个子问题的解都有具体的物理含义。其缺点在于物理单元之间耦合作用很强，难以建立子问题的模型。一般情况下需要给定一些假设条件才能对问题进行分解，这使得子问题解合成的解可能不是原问题的最优解。

1.3.3 基于时间的分解

调度问题就是分配资源的可用时间给相互竞争的任务，所以在任何调度问题中都暗含了优化时域和决策时刻，将调度问题进行时间分解非常自然。

基于分派机制的规则调度就是一种最简单的时间分解。当机器可用时，根据某一规则从已到达该机器的待加工工序集合中选择一道工序在该机器上加工。在决策点子问题的求解只利用当前已知信息。每调度完一道工序，决策点随时间推进。如果是单机调度问题，调度 n 个工件则需要求解 n 个子问题；如果是车间调度问题，在 m 台机器上调度 n 个工件则对应 $m \times n$ 个子问题。它也是一种工序层分解方法。

滚动时域分解方法(rolling horizon procedure, RHP)是一种最典型的时间分解方法。它通过求解一系列小规模或有限时段的局部调度代替大规模或无限时段的一次性全局调度。在每个调度时刻，利用更新的当前已知信息确定一个滚动窗口，对滚动窗口内的工件优化调度。然后执行部分局部调度解。此过程重复进行，直到全部工件调度完。RHP 最初由 Ovacik 和 Uzsoy(1994)针对单机问题提出[97]，随后应用于并行机问题[98]。Wang(2005)提出了一种带终端惩罚的滚动时域方法，该方法可以保证调度性能随滚动的执行不断改进[99]。

以上三类传统的分解方法处理大规模复杂车间调度问题都是基于分而治之(divide and conquer)的思想，它将高维的整体问题转化为一系列低维的子问题，然后并行求解子问题，最后通过协调各子问题的解来获得原问题的解。分解方法与整体求解相比，由于子问题维数的降低使得计算过程大为简化，但付出的代价是需要进行多次迭代，而整体解则是一次优化完成的。因此，迭代方法的收敛速度是分解方法是否有效的关键。分解方法的优劣主要取决于如何处理子问题之间的关联，这主要体现在子问题模型的建立和协调机制。

1.4 瓶颈调度方法的研究现状和存在的问题

在传统的分解方法中，所有的子问题地位是均等的，它们需要同等对待。从直觉上看，一个系统不是所有的部分都同等重要。以车间调度问题为例，各个机器的加工能力不同，在生产过程中的重要程度也不同。因此，对不同关键程度的子问题需要区别对待。通常只有一台机器(瓶颈机)对目标函数而言是非常重要的，因此如果完全松弛其它机器来求解原问题，则可以获得目标函数的近似解，以及对原问题(未松弛问题)上该机器的近似优化排序。约束理论(theory of constraints, TOC)中指出瓶颈(bottleneck)是制约系统朝着目标前进的主要因素，瓶颈决定了整个系统的性能[100-102]。该思想可以保证从全局优化的角度来简化大规模复杂调度问题。

而且瓶颈存在任何系统中。无论是生产制造系统、交通运输系统、供电供水系统，Internet 网络系统都存在瓶颈。以生产制造过程为例，由于一个均衡的生产线容易受加工波动和外界扰动的影响，而一定程度的能力不均衡可以用于吸收扰动并保持加工的平稳流动，因此实际生产线往往设计为能力不均衡[103]。在不均衡生产线上，瓶颈机决定了整个生产线的主要性能，有效的辨识和管理瓶颈可以提高系统的有效产出(throughput)。瓶颈作为一种系统的哲学思想可以指导人们处理现实生活中面临的许多复杂问题。因此利用瓶颈思想分析求解复杂的生产过程是有一定理论意义和实际应用前景的。

瓶颈思想最初是由以色列物理学家 Goldratt 提出[100-102]，该思想最早体现在一个优化生产技术(optimized production technology, OPT)商业软件包里，逐渐发展形成一套成熟的理论，即 TOC 理论，成为处理约束的哲学思想。它主要有两个组成部分，一是 TOC 物流/调度部分，它包括五步集中改进步骤、VAT 逻辑结构分析、鼓-缓存-绳(drum-buffer-rope, DBR)调度方法和缓存管理(buffer management, BM)信息系统；另一部分是思维过程(thinking process, TP)。

五步集中改进步骤是 TOC 理论的核心，它为 TOC 思想的实现提供了基本的框架[100]：(1) 辨识系统的瓶颈；(2) 充分利用系统的瓶颈；(3) 调动其它相关因素支持上

述决策；(4) 提高系统瓶颈的能力；(5) 如果前面步骤中瓶颈发生移动，则返回至第一步，进行新一轮的瓶颈处理。它体现了 TOC 是一个连续改进的过程。DBR 方法可视为 5 步集中步骤的具体实现，它保证生产线的加工速率与瓶颈机同步[102]。对生产线存在多个约束机的情况，Simons 等对 DBR 约束调度问题构造了一个数学模型[104]，随后提出了一个目标系统法(goal system, GS)来求解该问题[105]。该算法称为 GS1.0，在该算法中约束机顺序优化求解。Simons 等改进目标系统方法，在改进算法 GS2.2 中，约束机同步优化求解[106]。

TOC 理论(包括 OPT，DBR 以及 GS)给出了一套处理生产调度的方法。在这些方法中瓶颈是与产能挂钩的，它是通过生产线上长时间运行中的某些参数(如机器利用率，机器前平均队列长度和等待时间等)来确定瓶颈的。这里瓶颈的意味多是规划层面的，也就是说给定某一产品的工艺流程可以定出该生产线上的瓶颈，由瓶颈机上的能力来定生产线的加工能力(产出量)，确定瓶颈机上工件的加工顺序，以及产品的投放时间。而非瓶颈上的工件按简单的规则如 FIFO 进行加工。该方法适用于产品类型较少，需求量较大的订单情况。该方法只需要确定瓶颈机上各操作的开工时间和产品的投放时间。它在处理其它机器上操作安排过于简单。

不同于传统的分解方法，瓶颈思想为我们处理大规模复杂问题提出了另一种解决思路。生产调度问题实际上是一种资源优化分配问题，对最终性能影响最大是瓶颈。因此，有效地辨识瓶颈，并充分利用瓶颈的能力是生产调度的目的也是手段。而且瓶颈思想也是一种系统的方法，它将整个生产线看作一个大系统，对大规模复杂生产调度问题的处理可以应用至于其它系统，具有一定的推广意义。

1.5 论文主要内容及章节安排

1.5.1 本论文的主要内容

本论文主要研究大规模能力不均衡生产线的调度问题。该问题存在于实际的生产制造过程，具有一般性。从大系统分解的角度出发，深入挖掘系统分解思想的内涵，利用该思想的基本原理和研究思路指导我们进行大规模复杂调度问题的研究。同时结合不均衡生产线的瓶颈特性，给出了车间调度问题瓶颈分解方法的一般描述。

本论文研究的瓶颈分解方法主要针对三类典型的车间调度问题：Flow shop, Job shop 和半导体生产线，考虑不同均衡程度的生产线(单瓶颈、多约束机)，给出具体的瓶颈分解算法设计、理论分析和应用研究。通过处理不同加工环境的调度问题体现瓶颈分解方法求解大规模复杂车间调度问题的有效性。

主要工作包括以下几个方面：

- 在大系统分解思想的指导下，给出了车间调度问题机器层分解方法的一般描述，并指出传统分解方法在求解大规模调度问题时可能存在的不足。利用能力不均衡生产线存在瓶颈的特性，剖析瓶颈思想的内在机理，结合大系统分解思想提出了瓶颈分解方法的一般框架，并指出该方法的关键技术。

- 在分析单瓶颈 Flow shop 生产线特点的基础上，提出了一种单瓶颈分解算法。该方法将 Flow shop 上的机器分为瓶颈机、上游非瓶颈机和下游非瓶颈机。松弛非瓶颈机加工能力约束，原 Flow shop 问题简化为一个单机调度问题。瓶颈机的调度可以通过优化求解该简化问题获得。根据瓶颈机的调度结果，上游非瓶颈机采用有效的分派规则从后往前依次调度，而下游非瓶颈机则采用规则从前往后依次调度。随后我们给出了非瓶颈机冗余能力的条件，证明满足此条件下的 Flow shop 问题的最优调度为排列调度，即非瓶颈机上工件的加工安排与瓶颈机的调度安排一致。大量的仿真结果显示单瓶颈分解算法可以在较快的时间内获得较好的解。

- 针对单瓶颈 Job shop 调度问题，提出了一种单瓶颈分解算法。该方法利用生产线主导瓶颈的特点将 Job shop 中的机器分为瓶颈机和非瓶颈机，假设非瓶颈机加工能力无限大，将非瓶颈机上的工序集结成一个时间迟延环节。原问题则简化为一个带到达时间和传递时间约束的单机调度问题。瓶颈机上工件的加工安排可以通过优化求解此单机问题获得，而非瓶颈机则采用有效的启发式算法调度。通过调整瓶颈机上工序的到达时间和传递时间协调瓶颈机与非瓶颈机之间的关联。仿真结果显示该算法求解瓶颈程度较高的 Job shop 问题非常有效。

- 针对能力较不均衡的 Job shop 生产线，提出了一种约束调度算法。该算法给出了一种迭代辨识约束机(包括长期瓶颈和暂时瓶颈)的方法。每次迭代，采用有效的规则调度非约束机，根据判定条件从中辨识约束机。该过程迭代进行直至不存在新的约束机。约束机模型可描述为一个带时间迟延的多机调度问题，该问题的最优解是原问题最优解的一个下界。随着约束机不断辨识，相应的约束机模型也迭代更新，我们证明约束机调度子问题的最优解在迭代过程中不断提高，朝着原问题最优解的方向逐步逼近。与传统的移动瓶颈方法相比，约束调度算法在提高计算效率的同时保证了调度的优化性能。大量的仿真比较研究显示，该算法可以在调度质量和计算时间之间获得一个好的均衡。

- 针对约束调度算法无法求解的更大规模 Job shop 调度问题，提出了一种滚动瓶颈分解策略。该方法在瓶颈分解方法的基础上，结合滚动时域分解的思想，

将整个调度时域分解为若干决策子窗口，每个时间窗口对应的子问题采用瓶颈分解方法求解。该方法从空间层和时间层对大规模问题进行分解，它为求解超大规模 Job shop 调度问题提供了一种新的解决思路。仿真结果验证了滚动瓶颈分解方法的有效性。

- 将滚动瓶颈分解方法应用至一个更复杂的 Job shop 调度问题，即半导体生产线，针对半导体生产线的机器加工类型不一，对分解后不同类型的子问题给出了相应的求解算法，并对经典模型 MINIFAB 进行详细的仿真测试研究。

1.5.2 本论文的结构安排

第一章绪论，介绍调度问题的研究价值和意义，对调度问题的特点、分类和已有算法进行概述，重点总结了已有文献中处理大规模静态车间调度问题的分解方法，分析它们各自的特点。给出瓶颈分解方法用于求解大规模车间调度问题的可行性和研究思路。

第二章提出了瓶颈分解方法的一般框架，建立了基于分解协调的两层模型，指出了瓶颈分解方法的关键技术。

第三章针对能力严重不均衡的 Flow shop 生产线，即单瓶颈 Flow shop 调度问题，在一定的假设条件下进行理论分析。根据该生产线存在一个主导瓶颈的特性设计具体的瓶颈分解算法。

第四章针对能力严重不均衡的 Job shop 生产线，即 Job shop 只存在一个瓶颈的情况，设计了一种单瓶颈分解算法，并给出理论分析以及具体的仿真比较研究。

第五章研究能力较不均衡的 Job shop 生产线，即 Job shop 存在多个约束机的情况，提出了一种迭代辨识约束机的方法，给出具体的约束机分解算法并进行理论分析。

第六章结合瓶颈分解和时域分解思想处理超大规模的 Job shop 调度问题，提出了一种滚动瓶颈分解方法，并将该方法应用至一个复杂的半导体生产线。

第七章对本文的工作进行归纳总结，并对今后的研究方向进行了展望。

第二章 大规模车间调度问题瓶颈分解方法一般描述

2.1 引言

为了深入研究瓶颈思想如何指导大规模车间调度问题进行分解，我们首先在策略层面建立瓶颈分解的一般框架，挖掘其通用的思想，规范在不同加工环境下的基本概念和描述。

瓶颈分解方法是大系统分解思想在调度问题的应用推广，因此在本章 2.2 节，我们首先从大系统分解基本原理在广义优化问题的应用角度出发，给出车间调度问题机器层分解方法的一般描述。随后在 2.3 节分析传统分解方法在求解大规模车间调度问题可能存在的不足，给出车间调度问题瓶颈分解方法的基本描述和框架，并指出瓶颈分解方法的关键技术，为进一步深入研究瓶颈分解方法处理大规模车间调度问题做准备。

2.2 车间调度问题机器层分解方法描述

2.2.1 大系统分解思想

传统的优化理论，总是把对象当作一个整体，所有的计算都是集中优化。但在处理高维的大系统问题时，由于计算量随系统维数的增加而急剧增长，使得很多问题的优化无法进行。

分解协调方法是处理大系统问题的一种常用方法。所谓分解，是指把复杂的高维整体问题分解为若干相互独立的低维子问题；所谓协调，是指这些子问题相互协调相互配合，以实现整体问题的总目标。由于子系统是低维的，计算量可大大减少。在一般情况下，子系统之间存在复杂的关联，因此子问题得到的解并不是整体问题的解，其差异程度取决于关联的强度。

大系统分解协调方法一般按照大系统的实际组成，例如行政单元、物理装置、地域分布等进行分解，将一个大规模复杂系统 P 分解为一系列易于处理的低阶子系统 $P_i\ (i=1,\cdots,N)$。然后对每个子系统进行独立优化求解。最后对各子系统的独立解进行协调，考虑子系统之间的关联 α，使分解后得到的各部分独立解接近并最终成为整体解[107]。因此，分解协调方法主要包括两个任务：

(1) 分解问题，即如何定义子问题 P_i。由于子问题 $P_i(\alpha)$ 是一个在关联参数 α 相对

固定时的局部优化问题，因此需要确定它的局部目标函数和约束条件。在一般情况下，大系统优化的总目标函数是可分离的，子系统的目标函数就是其一部分。但子问题中的约束条件，却与整体问题有很大的不同，这里除了整体系统的输入输出量外，还出现了子系统之间的关联约束。它们在处理整体问题时是作为内部关系出现的，但对子问题来说，却可能成为外部关系。因此，分解的核心问题是在分解中如何处理子系统之间的关联。

(2) 协调问题，即根据什么原则最终确定关联参数α^*。由于各子问题$P_i(\alpha)$的最优解是α的函数，为了协调各子问题$P_i(\alpha)$，应使它们满足某种协调规则，这样才能获得全局最优，故有必要根据协调规则改进α，通常这是通过迭代达到的。协调问题在数学上是确定α从某一初值α_0到α^*的演变规则。由于子问题$P_i(\alpha)$的优化任务是根据分解原则确定的，故协调必须适应分解原则。

按分解和协调方式的不同，大系统分解协调方法主要包括两种：可行分解和关联预估方式，不可行分解和关联平衡方式。车间调度问题分解方法采用的是可行分解和关联预估原理，下面介绍一下该分解协调方式的基本原理。

可行分解和关联预估方式是在分解中对子系统之间的关联进行预估，预估值在子优化问题中作为常数看待，这相当于在求解子问题时把其动态模型中的外界关联相对固定为常量，然后求其最优值，因此子问题的最优解取决于预估关联值。在协调级，通过修正关联项参数，使目标函数不断得到改善。在关联参数相对固定时，子问题求解获得的子系统状态也是关联参数的函数，它们可作为关联参数新的预估值，一旦预估值完全正确地反映了子系统之间的实际关联时，子问题集成的解就是整体问题的最优解。即使预估值不等于实际关联值，其中间结果也可以作为原问题的一个可行的近似解。从这个意义来说，这种分解方法称为可行的。

用分解协调方法求解与整体求解相比，由于降低了子问题的维数，使得基本求解过程大为简化，但付出的代价是要多次迭代，而整体解是一次求得。因此迭代方法的收敛速度是分解协调是否有效的关键。若能用多机并行求解子问题，即计算分散化，则用分解协调方法可望减少计算时间。

分层预估迭代方法是处理大系统问题的另一种常用方法。该方法和分解协调方法都是采用预估迭代的思想。不同之处在于分解协调方法中，底层各子问题优化是独立并行求解，它们之间不存在先后顺序关系。但在分层预估迭代方法中，它将子问题优化过程进一步分解。先估计部分子问题的关联参数，对它们优化求解。然后根据这些子问题的解修正其它子问题的关联参数，对未优化子问题进行求解。通过协调子问题之间的关联值获得各个子问题的求解。在分层预估迭代方法中，底层各子问题的求解

存在先后顺序。这种方法不但可以用来降低整体问题的复杂性，也可以处理分解过程中子问题优化求解的困难。

2.2.2 车间调度问题机器层分解方法

生产线可视为一个复杂的大系统，按车间实际组成(如机器、工件和工序等)对系统进行分解。车间调度问题可表示为一系列易于求解的小规模子问题，每个子问题对应一个具体的物理单元。每个子问题可以独立优化求解，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。本文研究的对象是大规模不均衡车间调度问题，该问题的特点是生产线上机器的加工能力不均衡，因此本节具体研究车间调度问题机器层分解。

如前所述，有效的车间调度问题分解方法有两个基本问题：分解问题和协调问题。分解问题的关键在于如何定义子问题，保证分解后子问题的解对原问题仍有意义，也就是说子问题的解必须是与整体目标一致的可行调度。而协调问题主要确定子问题解的集成机制，即如何建立子问题之间的关联，以及确定这些关联对整体解的影响，它们可以看作是在对局部子问题做全局指导。

车间调度问题可以表示为一组单机或并行机串行加工，各机器可能有其各自的加工特性，如批加工能力，顺序相关的安装时间需求等。按机器组自然分解，车间调度问题 P 可分解一组单机或并行机子问题 $P_j(j=1,\cdots,m)$。各子问题之间的关联是通过工件在不同机器上工序之间的顺序约束建立。由于每台机器 j 上每道工序 O_{ij} 最早可能的开工时间(到达时间) r_{ij} 取决于它前道工序的完工时间，而它最迟必须的完工时间(交货期限) d_{ij} 取决于它后道工序的开工时间，工序的到达时间和交货期限(或传递时间)是子问题之间的关联参数[1]。在原问题 P 中，它们不显示出现，只是隐含在工件顺序约束中，以内部关系存在。而在子问题中，它们作为外部约束条件。以经典 Job shop 调度问题 $J_m \| f(C_i)$ 为例，它可分解为 m 个单机子问题，每个子问题可以描述为带到达时间和交货期限约束的单机调度问题 $1|r_i,d_i|f(C_i)$，子问题的目标函数与原问题的目标函数保持一致。

与原车间调度问题相比，分解后的单机子问题的计算复杂度和计算规模都大大降低。这些子问题可以独立优化求解，采用多机并行计算，可大大减少计算时间。优化

[1] 关联参数也可采用工序的到达时间和传递时间。令 q_{ij} 和 d_{ij} 分别表示工序 O_{ij} 的传递时间和交货期限；d_i 表示工序 O_{ij} 所属工件 i 的交货时间，它是已知且固定不变的。由 $d_{ij}=d_i-q_{ij}$，工序的交货期限可与传递时间相互转化。

求解各子问题可确定各机器上工序的加工顺序和开工时间。

各子问题的求解只是局部优化，即每台机器上工序的加工安排只考虑该机器的能力约束、工序的到达时间和交货期限约束，而忽略了同一工件在不同机器上的工序之间的顺序约束。有可能使得各子问题解设定的工序的开工时间在时间轴上迭代，各子问题之间因实际存在的关联约束而发生冲突。一般情况下由子问题P_j的解并不能得到原问题P的解。为此，有必要通过调整关联参数r_{ij}和d_{ij}来协调各台机器上的调度安排使得局部解接近原问题的优化解。

由前可知，每个子问题P_j(对应一台机器)中的关联参数(r_{ij}和d_{ij})取决于其它机器上的调度安排。在其它机器未调度之前，该机器上工序的到达时间和交货期限是未知的，因此需要对它们预先进行估计。分解的核心问题在于准确地估计每道工序的到达时间和交货期限。

初始假设每台机器的加工能力均无限大，即每台机器前到达的工件无需等待即可加工，则每台机器上每道工序的到达时间只取决于其所有前道工序的加工时间，其初始预估值等于所有前道工序加工时间之和；同理，每台机器上每道工序的交货期限只取决于该工序所有后道工序的加工时间和工件的交货期限，其初始预估值等于该工序所属工件的交货期限减去所有后道工序加工时间之和。给定各工序的到达时间和交货期限，优化求解各子问题P_j得到局部目标函数值$z_j(r_{ij},d_{ij})$，由子问题解构成的整体解也是关联参数的函数$z'(r_{ij},d_{ij})$。一般情况它不是原问题的最优解z。

如果子问题求得的某工序的完工时间大于其后道工序的开工时间，原问题中工件顺序约束被破坏，各子问题之间因实际存在的关联约束而发生冲突。我们可以通过修正关联参数(r_{ij}和d_{ij})来协调各子问题的解。在给定工序到达时间和交货期限时，子问题获得的子系统状态(工序的开工时间或完工时间)也是关联参数的函数，即$t_{ij}=t_{ij}(r_{ij},d_{ij})$或$C_{ij}=C_{ij}(r_{ij},d_{ij})$，其中$t_{ij}$和$C_{ij}$分别表示工件$i$在机器$j$上加工工序$O_{ij}$的开工时间和完工时间。一旦所有工序的到达时间等于其前道工序的完工时间，且它们的交货期限等于其后道工序的开工时间，则预估值完全正确地反映了子系统之间的实际关联，在此条件下，子问题构成的解$z'(r_{ij},d_{ij})$就是整体问题的最优解z。即使没有达到此条件，子系统新的状态可以用来更新关联项参数，其结果也可以作为原问题的一个可行解。

基于大系统分解协调的思想，我们给出了车间调度问题机器层分解方法的一般框架。它是采用两层递阶结构：协调层(coordinator)和调度层(scheduler)(见图 2-1)。

协调层主要根据调度层各子问题P_j的解(工序开工时间t_{ij}或完工时间C_{ij})来估计

或调整每道工序的到达时间r_{ij}和交货期限d_{ij}，然后将关联参数的估计值传递至调度层各个子问题中。而调度层则根据估计的关联参数r_{ij}和d_{ij}，优化求解每个子问题，得到每台机器上每道工序的开工时间和完工时间。新的调度结果传递至协调层用于修正关联参数，该过程迭代进行直至满足停止条件。其中协调层和车间调度层之间的信息流在图中用虚线描述，工件的加工流在图中用实线表示(见图 2-1)。

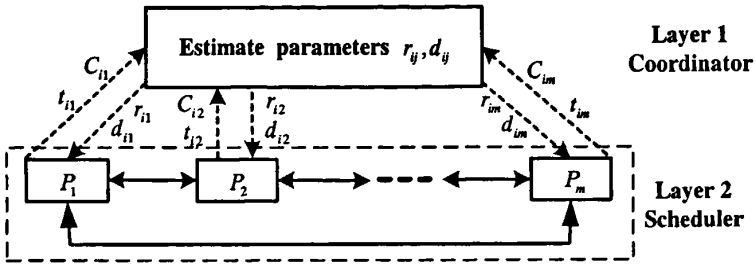


图 2-1. 车间调度问题机器层分解协调基本结构框图

Fig 2-1. The architecture of machine-based decomposition method for shop floor schedule

2.3 车间调度问题瓶颈分解方法一般框架

在传统的车间调度问题机器层分解方法中，各子问题都视为均等的，它们之间不存在主导关系，因此对每个子问题都需要精确求解。而且在协调级需要对每个子问题之间的关联项进行调整。以机器个数为m且工件个数为n的经典 Job shop 问题为例，关联参数的个数为$2 \cdot mn$。随着问题规模的增大，关联参数个数也显著增长。当车间调度问题规模很大的时候，需要协调的关联参数个数非常多，这会造成分解方法的收敛速度降低，从而影响分解方法的效率。

移动瓶颈方法是一种典型的机器层分解方法，目前已成功应用于车间调度问题的求解。该方法将一个 Job shop 调度问题分解为若干更易求解的单机子问题，基于大系统分层预估迭代的思想顺序求解各子问题，获得机器上工件的加工安排。每次调度完一台机器，根据新的调度结果对已调度的子问题重新优化。大量的研究结果显示，该方法可以获得好的近似解，但计算时间非常长[94,95,108]。它可求解问题的最大规模为 30 台机器和 50 个工件[95]。由于移动瓶颈方法对每个子问题均精确调度且重新优化，子问题求解和重新优化过程是移动瓶颈方法中最耗时的过程。

本文研究的背景是能力不均衡的生产线，该生产线上机器的加工能力不一。很多情况下，能力不均衡生产线有一个或几个固定的瓶颈机。因此利用生产线瓶颈特性，

我们提出了一种瓶颈分解方法。该方法从全局优化的角度对大规模车间调度问题进行分解，在保证求解质量的前提下可望大大提高计算效率，降低计算复杂度。

2.3.1 基于瓶颈特性的机器层分解方法

实际的制造生产线往往是能力不均衡的，每台机器的加工类型不一，加工能力也不同，不是所有的机器都同等重要。某些机器由于价格比较昂贵，出于生产成本的考虑购买的机器数量较少，使得该机器组的总加工能力最弱。如果对该机器的调度安排不合理则很容易导致整个生产线加工阻塞，影响生产线最终的产出。该机器通常被称为生产线的瓶颈。瓶颈问题已成为调度研究领域的一个难点。

对于生产制造系统来说，可以认为整个生产过程是由若干个相互联系的环节(Shop)组成的链条，一环扣一环，一个环节的产出受其前面环节的制约。只有对生产制造最薄弱环节进行改造才能真正提高生产系统的利润。一旦识别出最薄弱环节(即瓶颈)，提高该环节的加工能力就可以提高整个生产线的产出。

从调度原理的角度看，调度是稀缺资源在时间上的优化分配问题。在生产调度中，稀缺资源往往指加工机器。瓶颈机是生产线上的稀缺资源，也是竞争最激烈的机器，如果我们可以很好的安排瓶颈机上的调度，其它机器上的调度安排可以很容易获得，整个生产调度问题的求解就迎刃而解。因此，我们可以把调度重心放在瓶颈机上。

从优化求解的角度看，一般生产调度问题的约束主要包括：工件加工顺序约束、机器加工能力约束、到达时间和交货期限约束。而机器加工能力约束往往是车间调度优化求解的难约束。由于瓶颈机上的加工能力最弱，因此其加工能力约束最紧，先处理该机器上能力约束，其它机器上的能力约束则很容易满足。

Goldratt 等在其提出的约束理论中指出瓶颈并不是消极的，它在系统中起着积极的作用，瓶颈意味着系统有机会进行改进[100-102]。有效的管理和利用瓶颈可以提高系统的有效产出。因此我们利用生产线存在瓶颈机的特性，在车间调度问题机器层分解的基本框架下，提出了一种基于瓶颈特性的分解方法。

首先，辨识瓶颈机，将生产线上其它机器视为非瓶颈机。生产线上的机器分为两类：瓶颈机和非瓶颈机。与瓶颈机相比，非瓶颈机的加工能力更强。一般假设非瓶颈机有足够的冗余的能力支持瓶颈机的调度。因此对瓶颈机和非瓶颈机需要区别对待。

然后，调度瓶颈机，建立瓶颈机调度子问题并对其优化求解。传统求解单机和并行机调度问题的优化算法均可借鉴过来用于求解瓶颈机子问题。由于瓶颈机主导非瓶颈机，基于分层预估迭代思想，瓶颈机调度之后再确定非瓶颈机上的加工安排。

接着，调度非瓶颈机。为了保证瓶颈机加工能力充分利用，希望非瓶颈机的调度不破坏瓶颈机的调度安排。因此在非瓶颈机调度时，考虑瓶颈机的能力需求，根据瓶颈机的调度结果修正瓶颈机上工序的到达时间和传递时间。由于非瓶颈机加工能力相对较强，它们的能力约束相对更容易满足，因此只要采用有效的规则就可调度非瓶颈机的调度安排。

最后，协调瓶颈机与非瓶颈机。如果非瓶颈机的调度与瓶颈机的调度安排发生冲突，则调度不可行。修正关联项参数(瓶颈机上工序的到达时间和交货期限)，重新调度瓶颈机，该过程迭代进行直至调度可行。

2.3.2 瓶颈分解方法一般框架

类似传统的机器层分解方法，瓶颈分解方法也是基于大系统分解思想，主要包括协调级和调度级。由于生产线上瓶颈机主导非瓶颈机，因此底层子问题之间存在优先顺序关系，将调度级进一步分为两层：瓶颈机调度层和非瓶颈机调度层。基于大系统分层预估迭代思想，瓶颈分解方法可描述为一个三层递阶结构：协调层，瓶颈机调度层和非瓶颈机调度层(见图 2-2)。

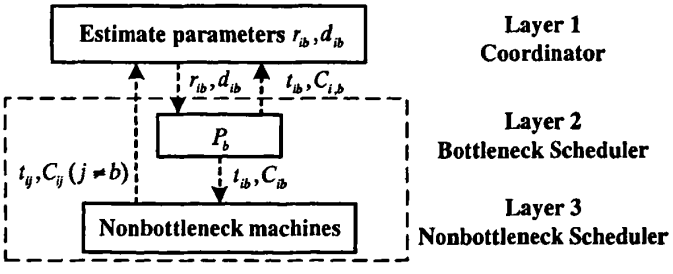


图 2-2. 车间调度问题瓶颈分解基本结构框图

Fig 2-2. The architecture of bottleneck-based decomposition method for shop floor schedule

令瓶颈机为 M_b，由于瓶颈机上每道工序 O_{ib} 最早可能的开工时间(到达时间) r_{ib} 取决于它前道工序的完工时间，而它最迟必须的完工时间(交货期限) d_{ib} 取决于它后道工序的开工时间。在瓶颈机子问题 P_b 中，它们作为外部约束条件。如果瓶颈机为单机，则子问题 P_b 描述为带到达时间和交货期限的单机调度问题 $1|r_i,d_i|f(C_i)$；如果瓶颈机为同类并行机，且并行机个数为 m，则子问题 P_b 描述为带到达时间和传递时间的并行机调度问题 $P_m|r_i,d_i|f(C_i)$。如果瓶颈机还有其它的加工特性，则子问题 P_b 在上述两类基本模型基础上添加新的约束条件。如果瓶颈机为批处理机，则子问题 P_b 可描述

为调度模型$1|B,r_i,d_i|f(C_i)$或$P_m|B,r_i,d_i|f(C_i)$；若瓶颈机有顺序相关安装时间需求，则子问题P_b可描述为调度模型$1|r_i,d_i,s_{ik}|f(C_i)$或$P_m|r_i,d_i,s_{ik}|f(C_i)$。

由车间调度问题中工件顺序约束可知，瓶颈机上每道工序的到达时间和交货期限取决于非瓶颈机的调度结果。初始时，非瓶颈机上的调度安排未知，因此需要对关联参数(r_{ib}和d_{ib})进行预估。协调层的任务主要根据调度层瓶颈机和非瓶颈机上的加工状态信息估计或修正瓶颈机上每道工序的到达时间r_{ib}和d_{ib}，然后将这些关联参数传递至瓶颈机调度器。瓶颈机调度器在固定参数r_{ib}和d_{ib}的情况下优化求解瓶颈机子问题，得到瓶颈机上每道工序的开工时间t_{ib}和完工时间C_{ib}。

为了保证工件顺序约束不破坏，瓶颈机上工序O_{ib}的前道工序(在非瓶颈机上加工)的最迟完工时间必须小于或等于工序O_{ib}的开工时间，而瓶颈机上工序O_{ib}的后道工序(在非瓶颈机上加工)的最早可能开工时间不能早于工序O_{ib}的完工时间。瓶颈机的调度结果设定了非瓶颈机上工序的到达时间和交货期限。因此，瓶颈机调度层将调度结果(t_{ib}和C_{ib})传递至非瓶颈机调度层，然后非瓶颈机围绕瓶颈机的结果而调度，得到非瓶颈机上每道工序的开工时间$t_{ij}(j\neq b)$和完工时间$C_{ij}(j\neq b)$，将瓶颈机调度层和非瓶颈机调度层得到的新的调度结果传递至协调层。如果瓶颈机与非瓶颈机的调度结果发生冲突，则根据瓶颈机调度层和非瓶颈机调度层的调度结果修正关联参数，该过程迭代进行直至瓶颈机与非瓶颈机不发生冲突。

第一个决策过程是基于协调层和瓶颈机调度层之间的垂直信息流；第二个决策过程则是基于瓶颈机调度层和非瓶颈机调度层之间的垂直信息流(见图 2-2 中虚线)。

车间调度问题瓶颈分解方法可以看作把一个大规模车间问题分解为若干小规模子问题，每个子问题对应一台机器或加工中心，其中只有一个或有限个机器(瓶颈机)对整个车间来非常关键。将重心放在这些机器上，对它建立详细的数学模型并精确求解。在优先调度瓶颈机之后，采用有效的规则就可以安排非瓶颈机上的调度。与传统的机器层分解相比，瓶颈分解方法有明显的优点：

1. 在分解层，传统的机器层分解方法将车间问题(假设机器或加工中心个数为m)的求解简化为m个单机或并行机子问题的求解，而瓶颈分解方法将该问题简化为m_b(瓶颈机个数)个单机或并行机子问题的求解。一般来说瓶颈机的个数远小于机器个数，即$m_b<m$，因此瓶颈分解方法的计算复杂度相对降低。

2. 在优化层，传统的机器层分解方法对每个子问题均建立数学模型并精确求解，而瓶颈分解方法只对瓶颈机建立模型并精确求解，对非瓶颈机则采用相

对简单的规则调度。这种对瓶颈机与非瓶颈机不同程度的处理方式可以降低计算量。

3. 在协调层，传统的机器层分解方法需要对每个子问题之间的关联项进行调整，而瓶颈分解方法只需调整瓶颈机子问题与非瓶颈机子问题的关联。协调的关联参数个数减少，提高了迭代过程的收敛速度，从而提高了分解方法的计算效率。

4. 在子问题求解顺序上，传统的机器层分解方法通常对每个子问题并行求解，而瓶颈分解方法则利用瓶颈机的主导特性优先求解瓶颈机子问题，再调度非瓶颈机。该种处理方式可以保证瓶颈机的能力得到充分利用，从而保证了整体问题的性能。

在瓶颈分解方法中，大规模能力不均衡车间调度问题可简化为一个或有限个机器(瓶颈机)的调度问题，计算复杂度大大降低。而且，由于瓶颈决定了整个生产线的主要性能，将调度的重心放在瓶颈机上，可以在提高计算效率的前提下仍能保证调度的优化性能。瓶颈分解方法为求解大规模、能力不均衡的车间调度问题提供了一种思路。而且该方法可适用于不同类型的车间调度问题，具有一般性。

2.3.3 瓶颈分解方法的关键技术

瓶颈分解方法主要有两个任务：一个是分解问题，即如何定义瓶颈子问题；另一个是协调问题，即根据什么原则最终确定关连参数。因此，准确地辨识瓶颈和精确地估计关联项参数是瓶颈分解方法的关键。

从生产线加工能力的角度出发，我们探讨瓶颈产生的原因，从而指导我们辨识瓶颈。能力一般定义为单位时间内机器、加工中心、车间或工厂的产出[109]。对一条串行生产线，如果其中某一个生产阶段中机器的加工能力低于其它生产阶段，则该生产阶段为生产线的主要制约因素，成为生产线中的瓶颈。

瓶颈机广义上称“约束资源(constraint resource)”。目前对瓶颈仍没有明确的定义。Goldratt 定义瓶颈为“任何制约系统朝着目标更好性能前进的因素”[100]。Umble 和 Srikanth 定义瓶颈是“实际生产能力小于或等于生产负荷的资源”[110]。Roser 等认为瓶颈是“对减慢或停止整个系统影响最大的阶段”[111]。这些定义可以帮助我们更好地理解瓶颈的特点，但它们不能作为辨识瓶颈的具体准则。

目前不同的瓶颈辨识方法相继提出[40, 111-113]。无论从排队网络的角度还是析取图的方式，长期瓶颈主要取决于机器的能力，而暂时瓶颈(或移动瓶颈)还与调度决策有关。大多数加工系统通常只有一个主要的瓶颈，它是加工能力最弱的机器。对能力不

均衡程度很高的生产线，非瓶颈机能力远大于瓶颈机的加工能力，因此瓶颈一旦辨识，它在整个调度时域内主导生产线的其它机器。对能力不均衡程度较低的生产线，虽然非瓶颈机总的加工能力足以完成其加工负荷，但由于时间问题可能使得该机器不能在规划的完工时间内完成它的操作。我们称这类机器为暂时瓶颈或移动瓶颈。我们将这两类机器称“约束机”。在后面的章节，我们将分别针对单瓶颈和多约束机的车间调度问题进行具体的研究。

在瓶颈分解方法中，瓶颈机调度子问题可描述为带到达时间和交货期限(或传递时间)约束的单机或并行机调度问题，其中瓶颈机上工序的到达时间和交货期限(或传递时间)为关联项。对经典车间调度而言，每个工件必须在每台机器上加工一次，瓶颈机上工序的到达时间和传递时间取决于非瓶颈上的调度决策，而非瓶颈机上的调度决策又取决于瓶颈机的调度结果。因此，瓶颈机和非瓶颈机之间的相互关联作用很复杂，很难准确估计关联参数的大小。

瓶颈分解方法采用预估迭代的思想估计关联参数。首先，对非瓶颈机做一些假设，估计关联项参数，并根据估计值求解瓶颈机子问题。然后，根据瓶颈机调度结果确定非瓶颈机的调度安排，新的调度结果可用于修正关联项参数。该过程迭代进行，直到预估值完全正确地反映瓶颈机与非瓶颈机之间的实际关联。如果能够找到一种好的引导机制使得关联预估值朝着实际值方向逼近，则可保证瓶颈分解方法的调度性能和收敛速度。

瓶颈机上工序的到达时间和传递时间实际上是工件在非瓶颈机上的流经时间(lead time)，它们可表示为：

$$r_{ib} = \sum_{ij \in pre(ib)} (p_{ij} + w_{ij}) \tag{2-1a}$$

$$q_{ib} = \sum_{ij \in suc(ib)} (p_{ij} + w_{ij}) \tag{2-1b}$$

其中 r_{ib} 和 q_{ib} 分别表示工序 O_{ib} (工件 i 在瓶颈机 M_b 上的操作)的到达时间和传递时间，$pre(j)$ 和 $suc(j)$ 分别表示工序 j 的所有前道工序集合和所有后道工序集合。p_{ij} 表示工序 O_{ij} 的加工时间，它是已知且固定不变的；w_{ij} 表示工序 O_{ij} 的等待时间，它取决于非瓶颈机的调度决策，当非瓶颈机上调度未知时，它是未知的。因此对关联参数的估计实际上是对工序在非瓶颈机上等待时间的估计。

Morton 等总结了工程上常用的 lead time 估计方法[114]。最简单的估计方法是固定系数法，该方法令等待时间等于加工时间乘以一个固定的系数 c，即 $w_{ij} = c \cdot p_{ij}$，到达时间和传递时间可以表示为：

$$r_{ib} = k \cdot \sum_{ij \in pre(ib)} p_{ij} \tag{2-2a}$$

$$q_{ib} = k \cdot \sum_{ij \in suc(ib)} p_{ij} \tag{2-2b}$$

其中$k = c + 1$，它可以根据历史记录中通过整个车间的平均时间除以所有工件加工时间的平均值。在工程领域中k一般取 3.0～6.0。该方法中k值保持不变，因此在整个调度时域内关联参数是固定不变的。由于实际生产过程存在加工波动以及外界各种扰动因素，车间的状态也在动态变化，显然固定系数法过于简单。

在此基础上 Narayan 等提出一种 lead time 迭代改进算法，也称为变系统法[115]。该方法中k值是根据车间的信息不断更新。生产过程运行一个时间段后，根据当前时间段的信息和历史信息更新系数k。该方法中k值在一定时间段内动态修正的，关联参数也随之调整。然而变系数法中工序的到达时间和传递时间的修正只是通过调整一个参数k获得，难以保证估计精读。因此有必要增加参数调整的自由度来提高关联参数估计的准确度，当然这也需要付出一定的计算代价。

本文主要采用逐步添加机器能力的思想来估计和修正每道工序的到达时间和传递时间。在能力不均衡车间调度问题中，非瓶颈机的加工能力大于瓶颈机的加工能力。基于此思想，我们首先松弛非瓶颈机的能力约束。假设非瓶颈机的加工能力无限大，非瓶颈机上的工件到达后无需等待即可加工，即工序在非瓶颈机上的等待时间为 0。由公式(2-1a)和(2-1b)可知，瓶颈机上工序的到达时间等于其所有前道工序的加工时间之和，而瓶颈机上工序的传递时间等于其所有后道工序的加工时间之和。工序的交货期限可以通过该工序所属工件的交货时间减去其传递时间获得。如果非瓶颈机和瓶颈机的调度不发生冲突，则得到的最终调度为原问题的最优调度。如果发生冲突，则表示非瓶颈机的能力不足以支持瓶颈机的调度，则考虑非瓶颈机的加工能力约束，修正瓶颈机上工序的到达时间和传递时间。我们将在后面的章节对不同加工类型、不同负载程度的车间调度问题具体讨论。

2.4 本章小结

本章在大系统分解思想的指导下，具体研究了车间调度问题机器层分解方法的一般描述，同时指出传统机器层分解方法在求解大规模车间调度问题时可能存在的不足。基于大规模能力不均衡生产调度中存在的瓶颈特性，我们提出了一种瓶颈分解方法。该方法在传统机器层分解的基础上，结合问题的瓶颈特性，给出了瓶颈分解方法的一般框架，建立了三层递阶模型。与传统的机器层分解方法相比，瓶颈分解方法可大大提高计算效率同时保证调度性能。该方法可适用于不同加工类型的不均衡车间调度问题，具有通用性。最后，本章指出了瓶颈分解方法的关键技术，它为后面章节应

用瓶颈分解方法求解不同类型的车间调度问题提供了理论框架和解决思路。

第三章 单瓶颈 Flow shop 调度问题瓶颈分解方法研究

3.1 引言

在许多制造和装配生产线上，工件按相同的工艺路径在一组顺序排列的机器上依次加工，称该生产过程为流水作业(Flow shop)生产线。Flow shop 调度问题是一类经典的生产调度问题。除少数特殊问题以外，大部分 Flow shop 调度问题均为强 NP-hard[2]，存在优化求解的困难。

本章主要研究大规模能力不均衡 Flow shop 生产线，利用流水线上机器瓶颈特性提出了一种有效的单瓶颈分解方法。

本章的主要内容安排如下：3.2 节给出 Flow shop 调度问题的一般描述并建立数学规划模型。在 3.3 节利用不均衡 Flow shop 问题的瓶颈特性对问题简化，并分析简化问题的解与原问题的解之间的关系，给出非瓶颈机冗余能力的条件。在瓶颈分解方法的框架下设计了一种单瓶颈分解算法。在 3.4 节将该方法分别应用于两类不同性能指标的 Flow shop 调度问题，进行大量的仿真测试和分析。

3.2 Flow shop 调度问题描述和分析

经典 Flow shop 调度问题可描述为：m 台机器 $M_j(j=1,2...,m)$，n 个待加工工件 $J_i(i=1,2...,n)$，每个工件 J_i 都包括一系列工序 $O_{i,j}(j=1,2...,m)$。工序 $O_{i,j}$ 必须在机器 M_j 上加工，且加工时间为 $p_{i,j}$。工件 J_i 工序之间的优先顺序为：$O_{i,1}\rightarrow O_{i,2}\rightarrow ...\rightarrow O_{i,m}$。同一时刻一台机器只能加工一个工件，同一时刻一个工件也只能在一台机器上加工。调度的目标是确定每台机器上工件的加工顺序和开工时间，使得工件完工时间 C_i 的某一目标函数值 $f(C_i)$ 最优。建立 Flow shop 调度问题数学规划模型如下：

$$(P)\qquad \min_{x_{ijk},t_{i,k}} f(C_i) \tag{3-1a}$$

$$st.\quad t_{i,k}-t_{i,k-1}\geq p_{i,k-1}\quad i=1,...,n;k=2,...,m; \tag{3-1b}$$

$$t_{i,1}\geq 0\quad i=1,...,n; \tag{3-1c}$$

$$t_{j,k}-t_{i,k}+\mathrm{H}(1-x_{ijk})\geq p_{i,k}\quad i=1,\cdots,n-1;j=i+1,\cdots,n;k=1,...,m; \tag{3-1d}$$

$$t_{i,k}-t_{j,k}+\mathrm{H}x_{ijk}\geq p_{j,k}\quad i=1,\cdots,n-1;j=i+1,\cdots,n;k=1,...,m; \tag{3-1e}$$

$$C_i=t_{i,m}+p_{i,m}\quad i=1,...,n; \tag{3-1f}$$

$$x_{ijk} \in \{0,1\} \quad i,j=1,...,n; i \neq j; k=1,...,m; \tag{3-1g}$$

其中决策变量$t_{i,k}$表示工件J_i在机器M_k上的开工时间；x_{ijk}为二进制变量，表示机器M_k上工件J_i和J_j的加工顺序。如果机器M_k上工件J_i在工件J_j之前加工，则x_{ijk}=1；否则x_{ijk}=0。H为一充分大的正整数；$p_{i,k}$表示工件J_i在机器M_k上的加工时间，d_i表示工件J_i的交货期限，它们是给定且固定不变的。C_i表示工件J_i的完工时间，它取决于机器上工件的调度安排，可按公式(3-1f) 计算。

目标函数(3-1a)为工件完工时间C_i的一个正则函数。加工顺序约束(3-1b)保证每个工件的工序必须在前一道工序完工后才能开始加工。到达时间约束(3-1c)保证每个工件必须在工件投入生产线之后才能开始加工，在本章中每个工件的到达时间均为0。机器能力约束(3-1d)和(3-1e)保证同一时刻每台机器最多只能加工一个工件。Flow shop调度问题相当于在约束(3-1b)~(3-1g) 构成的可行解集内搜索一个解，使得目标函数(3-1a)最小。

自1954年Johnson对$F_2 \| C_{\max}$调度问题提出一个多项式算法之后[4]，Flow shop调度问题一直备受理论研究者和实际工程人员的关注。除了特殊问题，大部分Flow shop调度问题的计算复杂度为NP-hard[2]。由模型P可知，决策变量和约束个数均为$m \cdot n^2$。如果采用精确算法如分枝定界算法，其计算量与问题的规模呈指数关系。随着问题规模的增大，精确算法存在计算困难，它只适用于小规模问题的求解。而实际工程人员多采用启发式规则来确定每台机器上工件的加工优先级。该方法计算效率很高而且易于实现，但是它只利用了生产线的局部信息，获得的调度性能往往较差。因此大量的研究工作主要集中在近似解的获取上。

移动瓶颈(SB)方法是求解车间调度问题一种非常有效的启发式算法。该方法将车间调度问题分解为若干更易求解的单机子问题，通过依次求解这些单机子问题确定每台机器上工件的加工顺序和开工时间。Demirkol 和 Uzsoy(1997)对不同的加工环境(Flow shop, Job shop, Two sets Job shop)，不同性能指标($C_{\max}$和$L_{\max}$)的车间调度问题采用SB算法进行详细的仿真比较研究[95]。大量的仿真结果显示SB方法可以获得好的近似解，但计算时间非常长。而且SB方法求解Flow shop问题的质量远不如求解Job shop的性能好。产生这一现象的原因分析如下：

首先，SB算法的性能受工艺路径结构的影响。在Job shop问题中工件的工艺路径是机器的随机排列，因此加工负荷或多或少均匀分布到各个机器上，在某一时间间隔内到达给定机器上工件的个数很少，减少了该时间段对机器资源的竞争程度。而Flow shop问题中所有工件的工艺路径均相同，在某一时间段内竞争同一机器资源的

工件数相对较多，使得该时间段上工件的加工难以安排。

其次，SB 算法框架限制了它对 Flow shop 问题求解的改进空间。SB 算法是通过顺序迭代求解一组单机子问题来获得机器上工件安排。如果先调度的机器上调度决策不正确，错误的决策将会依次传播给未调度的机器，从而使得最终解的性能变差。因此，子问题的优先求解顺序对最终解的性能影响很大。SB 算法机制决定了关键机器的调度决策对 Flow shop 调度性能的影响最大。尽管 SB 算法耗费大量的计算时间对每个子问题均精确求解并重新优化，但是调度的性能改善不大。

到目前为止，SB 算法可求解问题的最大规模为 30 台机器 50 个工件的情况[95]，它只适用于中等规模问题。而实际的流水作业生产线往往规模很大，而且机器的加工能力也不同，因此本文针对大规模能力不均衡的 Flow shop 调度问题提出了一种有效的分解方法，该方法在大系统分解思想的指导下，利用生产线机器的瓶颈特性，希望在较快的时间内获得好的调度结果。

3.3 单瓶颈 Flow shop 调度问题瓶颈分解方法

3.3.1 瓶颈辨识

流水作业车间可以看作一条串行的加工链，每个工件按照相同的工艺路径在一组机器上依次加工。当工件被源源不断地投入生产线时，某些机器由于加工能力有限无法及时处理到来的工件，使得该机器前堆积了大量工件，造成整个生产线的阻塞。其中加工能力最弱的机器前阻塞的程度最严重，成为整个流水线的瓶颈。

对一个单产品的流水作业生产线，我们很容易就能找到该生产线的瓶颈，即加工时间最长的机器为瓶颈机。而对一个多产品的流水作业生产线，由于不同产品在机器上的加工时间不同，每个产品对应的加工能力最弱的机器也不同，因此难以判定生产线的瓶颈。由前所述，瓶颈机定义为加工能力小于或等于加工负荷的机器[110]。加工负荷越大的机器越容易发生拥塞现象而成为整个生产线的瓶颈。因此我们选择加工负荷最大的机器为瓶颈机。

首先计算每台机器总的加工负荷为该机器上加工的所有工序加工时间之和：

$$WL_k = \sum_{i=1}^{n} p_{i,k} \quad k = 1,\cdots,m; \tag{3-2}$$

选择加工负荷最大的机器 M_b 为瓶颈机，即

$$b = \arg\max_{k=1,\cdots,m} WL_k \tag{3-3}$$

3.3.2 单瓶颈 Flow shop 问题模型简化与分析

一旦确定好瓶颈机，Flow shop 中其它机器称为非瓶颈机。由于 Flow shop 特殊的工艺路径，瓶颈机工序之前的所有操作均在瓶颈机之前的非瓶颈机上加工，我们称这些机器为上游非瓶颈机，而瓶颈机工序之后的所有操作均在瓶颈机之后的非瓶颈机上加工，我们称这些机器为下游非瓶颈机。整个 Flow shop 上的机器分为三部分：上游非瓶颈机、瓶颈机和下游非瓶颈机。每个工件 J_i 流经上游非瓶颈机和下游非瓶颈机的时间 r_i 和 q_i 分别取决于上游非瓶颈机和下游非瓶颈机上工序的加工时间和等待时间，即

$$r_i = \sum_{j=1}^{b-1}(p_{ij} + w_{ij}) \quad i = 1,\cdots,n; \tag{3-4a}$$

$$q_i = \sum_{j=b+1}^{m}(p_{ij} + w_{ij}) \quad i = 1,\cdots,n; \tag{3-4b}$$

其中 w_{ij} 表示工件 J_i 在机器 M_j 上的等待时间，它取决于工件在非瓶颈上的调度安排。

Flow shop 问题可简化为一个单机问题 $1|r_i,q_i|f(C_i)$。如图 3-1 所示，一个工件 J_i 至少需要经过 r_i 个时间单元才能到达瓶颈机，在瓶颈机上加工 $p_{i,b}$ 个时间单元之后至少需要经过 q_i 个时间单元才能完成工件 J_i 的全部加工。在简化问题中，工件 J_i 在上游非瓶颈机和下游非瓶颈机上的加工时间分别用时间迟延 r_i 和 q_i 替代。

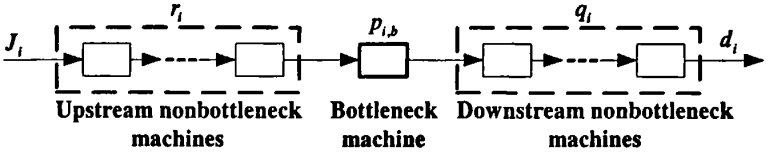


图 3-1. 流水车间调度问题简化模型框图

Fig 3-1. The reduced flow chart of the flow shop with a bottleneck machine

当非瓶颈机上加工安排未知时，需要对 r_i 和 q_i 进行估计。一般来说能力不均衡的 Flow shop 中非瓶颈机的加工能力远大于瓶颈机的加工能力，假设非瓶颈机的加工能力无限大，则每个工件到达非瓶颈机之后无需等待即可开始加工，即工件在非瓶颈机上的等待时间为 0。每个工件 J_i 通过上游非瓶颈机和下游非瓶颈机的流经时间分别等于工件 J_i 在上游非瓶颈机和下游非瓶颈机上所有工序加工时间之和，即

$$r_i = \sum_{j=1}^{b-1} p_{ij} \quad i = 1,\ldots,n; \tag{3-5a}$$

$$q_i = \sum_{j=b+1}^{m} p_{ij} \quad i = 1,\cdots,n; \tag{3-5b}$$

在假设非瓶颈机无限大的情况下，松弛问题 P 中非瓶颈机的机器能力约束，得到单机调度问题 $1|r_i,q_i|f(C_i)$，模型 P_b 描述如下：

$$(P_b) \qquad \min_{x_{ijb},t_{i,b}} f(C_i) \tag{3-6a}$$

$$s.t. \qquad t_{i,b} \geq r_i \quad i = 1,...,n; \tag{3-6b}$$

$$C_i \geq t_{i,b} + p_{i,b} + q_i \quad i = 1,...,n; \tag{3-6c}$$

$$t_{j,b} - t_{i,b} + \mathrm{H}(1 - x_{ijb}) \geq p_{i,b} \quad i = 1,\cdots,n-1; j = i+1,\cdots,n; \tag{3-6d}$$

$$t_{i,b} - t_{j,b} + \mathrm{H}x_{ijb} \geq p_{j,b} \quad i = 1,\cdots,n-1; j = i+1,\cdots,n; \tag{3-6e}$$

$$x_{ijb} \in \{0,1\} \quad i,j = 1,...,n; i \neq j; \tag{3-6f}$$

其中决策变量 $t_{i,b}$ 表示工件 J_i 在瓶颈机 M_b 的开工时间，x_{ijb} 为二进制变量，表示瓶颈机 M_b 上工件 J_i 和 J_j 的加工顺序，如果瓶颈机 M_b 上工件 J_i 在工件 J_j 之前加工，则 x_{ijb}=1；否则 x_{ijb}=0。$p_{i,b}$ 表示工件 J_i 在瓶颈机 M_b 上的加工时间，它们是已知且固定不变的。r_i 和 q_i 分别表示工件 J_i 在瓶颈机 M_b 上最早可能的开工时间(或到达时间)和最大可能的剩余加工时间(或传递时间)，它们分别按公式(3-5a)和(3-5b)计算。约束(3-6b)表示工件 J_i 至少需要等待 r_i 个单位的时间迟延才能开始在瓶颈机上加工；约束(3-6c)表示工件 J_i 在瓶颈机上加工完之后至少需要经过 q_i 个单位的时间迟延才能完成该工件所有的工序。目标函数(3-6a)与原问题 P 中目标函数(3-1a)一致。

在模型 P_b 中，变量和约束个数为 n^2。与原问题相比，计算复杂度大大降低。

性质 3-1 令 z 和 z_b 分别为问题 P 和 P_b 的最优解，如果每个工件在非瓶颈机上的加工时间均为 0，则 $z_b \equiv z$。

证明：如果非瓶颈机上每个工件的加工时间均为 0，问题 P 等价于问题 P_b，则 $z_b \equiv z$。

性质 3-1 说明了当非瓶颈机的加工能力无限大时，简化的单机调度问题的最优值为原 Flow shop 调度问题的最优值。此时，非瓶颈机调度对整个系统性能的影响可以被忽略，非瓶颈机上工件可按任意顺序进行加工。瓶颈机调度问题的目标为主导目标，与整个问题的目标保持一致。性质 3-1 暗含了当 Flow shop 生产线加工能力严重不均衡时，瓶颈主导整个系统的性能。

定理 3-1 令 z 和 z_b 分别为问题 P 和 P_b 的最优解，如果瓶颈机上每个工件的到达时间

和传递时间按(3-5a)和(3-5b)计算，则有 $z_b \le z$。

证明：令 S 和 S_b 分别表示问题 P 和 P_b 的可行解的集合。通过松弛机器能力约束(3-1d)和(3-1e)中非瓶颈机的能力约束，得到瓶颈机上工件的到达时间和传递时间分别等于公式(3-5a)和(3-5b)的右项，因此问题 P 的可行集合 S 属于问题 P_b 的可行集合 S_b，即 $S \subseteq S_b$。由于问题 P 和 P_b 的目标函数相同，且为最小化问题，则 $z_b \le z$，即简化问题的最优解是原问题最优解的一个下界。

到达时间 r_i 与传递时间 q_i 是瓶颈机与非瓶颈机之间的关联项，它们取决于 Flow shop 中非瓶颈机上工件的调度安排。由公式(3-4a)和(3-4b)可知，工件 J_i 的到达时间 r_i 和传递时间 q_i 不小于它在上游非瓶颈机和下游非瓶颈机上的加工时间之和，即 $r_i \ge \sum_{j=1}^{b-1} p_{i,j}, q_i \ge \sum_{j=b+1}^{m} p_{i,j}$。当简化问题中关联参数($r_i$ 和 q_i)分别取其下界时，它的最优解是原问题最优解的一个下界。定理 3-1 为原问题最优解的下界计算提供了一种方法。

3.3.3 瓶颈机与非瓶颈机的调度

瓶颈机上工件的调度安排可以通过优化求解简化问题 P_b 获得。尽管问题 $1|r_i,q_i|f(C_i)$ 仍为强 NP-complete[3]，但是很多有效的启发式算法可以用于求解该问题。

令瓶颈机 M_b 上工件的排序为 $\pi_b(1),\pi_b(2),\cdots,\pi_b(n)$，其中 $\pi_b(i)$ 表示瓶颈机 M_b 上第 i 个加工的工件；$t^*_{\pi_b(i),b}$ 为瓶颈机 M_b 上工件 $\pi_b(i)$ 的开工时间，由公式(3-1b)可知

$$t_{\pi_b(i),b-1} + p_{\pi_b(i),b-1} \le t^*_{\pi_b(i),b} \quad i = 1,\cdots,n; \tag{3-7a}$$

$$t_{\pi_b(i),b+1} \le t^*_{\pi_b(i),b} + p^*_{\pi_b(i),b} \quad i = 1,\cdots,n; \tag{3-7b}$$

由公式(3-7a)可知，瓶颈机 M_b 上工件的开工时间决定了上游非瓶颈机 M_{b-1} 上工件的最迟完工时间(或截止时间)；由公式(3-7b)可知，瓶颈机 M_b 上工件的完工时间决定了下游非瓶颈机 M_{b+1} 上工件的最早可能的开工时间(或到达时间)。上游非瓶颈机和下游非瓶颈机的加工安排受瓶颈机调度决策的影响，即非瓶颈机围绕着瓶颈机的调度结果而调度。而且非瓶颈机器之间需要满足一定的加工顺序约束，因此上游非瓶颈机的调度问题可以描述为 $b-1(b>1)$ 台机器，带截止时间约束的优化问题。而下游非瓶颈机的调度问题则可描述为 $m-b(b<m)$ 台机器，带到达时间约束的优化问题。我们希望非瓶颈机的调度安排在满足加工约束的条件下尽可能地服从瓶颈机的调度决策。

由于瓶颈机上工件到达时间和传递时间的初始估计过于松弛非瓶颈机上机器能力约束，使得基于瓶颈机调度结果设置的截止时间太紧，难以满足工件顺序约束(3-7a)，造成上游非瓶颈机无法找到可行的调度。为了保证分解算法的可执行性，我们将截止时间约束放松为软约束，即设置瓶颈机上工件的开工时间为上游非瓶颈机 M_{b-1} 上工件的交货期限，希望工件最大迟延时间最小。瓶颈机调度结果为上游非瓶颈机上的每道工序设置了一个期望的完工时间(工序交货期限)，并希望每个工序尽可能在设定的交货期限之前完工。

上游非瓶颈机的调度问题可描述为 $b-1(b>1)$ 台机器，目标为最小化最大迟延时间的优化问题，记为 $F_{b-1}\|L_{\max}$。设置工件的交货期限如下：

$$d_{i,b-1}=t_{i,b}^{*} \quad i=1,\cdots,n; \tag{3-8a}$$

而下游非瓶颈机的调度问题可描述为 $m-b(b<m)$ 台机器，带到达时间约束，目标为最小化最大迟延时间的优化问题，记为 $F_{m-b}|r_j|L_{\max}$。设置工件的到达时间如下：

$$r_{i,b+1}=t_{i,b}^{*}+p_{i,b}^{*} \quad i=1,\cdots,n; \tag{3-8b}$$

当非瓶颈机上的调度结果满足瓶颈机设定的约束条件时，则整个 Flow shop 调度可行。具体而言，上游非瓶颈机上每个工序的完工时间都在期望的工序交货期限之前完工，而下游非瓶颈机上每个工序的开工时间都在期望的到达时间之后开工，则瓶颈机和非瓶颈机上的调度安排为原问题的一个最优调度，此时非瓶颈机有足够的冗余能力支持瓶颈机的调度。

然而非瓶颈机上的加工能力冗余程度究竟多大才能足以支持瓶颈机的调度安排，即 Flow shop 生产线上工件的加工时间满足什么条件时才能保证非瓶颈机有足够的冗余的能力来支持瓶颈机的调度。对此我们进行了详细分析，给出了非瓶颈机冗余能力的条件。

当 Flow shop 中工件的加工时间 $p_{i,k}$ 满足以下条件时，非瓶颈机有足够的冗余能力来支持瓶颈机 M_b 的调度。

$$\sum_{k=r}^{b} p_{i,k} \geq \sum_{k=r}^{b} p_{j,k-1} \quad i,j=1,\cdots,n, i\neq j; r=2,\cdots,b; \tag{3-9a}$$

$$\sum_{k=b}^{r} p_{i,k} \geq \sum_{k=b}^{r} p_{j,k+1} \quad i,j=1,\cdots,n, i\neq j; r=b,\cdots,m-1; \tag{3-9b}$$

条件(3-9a)和(3-9b)分别给出了上游非瓶颈机和下游非瓶颈机的冗余加工能力，由(3-9a)和(3-9b)可得：

$$\sum_{i=1}^{n} p_{i,b} \geq \sum_{i=1}^{n} p_{i,k} \quad k=1,\cdots,b-1,b+1,\cdots,m; \tag{3-9c}$$

公式(3-9c)与瓶颈辨识条件(3-3)一致，而且公式(3-9a)和(3-9b)给出了不均衡 Flow shop 中非瓶颈机上工件的加工特性，它们保证了非瓶颈机有足够的冗余能力支持瓶颈机的调度。

定理 3-2. 如果 Flow shop 中的瓶颈机为 $M_b(b\in\{1,\cdots,m\})$，且该机器上调度安排为 π_b。当 Flow shop 上工件的加工时间 $p_{i,k}$ 满足条件(3-9a)和(3-9b)时，对任何正则目标函数，排列排序(Permutation schedule) $\tau=(\pi_b,\pi_b,\cdots,\pi_b)$ 为最优调度。

证明. 已知 Flow shop 中的瓶颈机为 $M_b(b\in\{1,\cdots,m\})$，且该机器上调度安排为 π_b，给出 Flow shop 的一个调度表为 $\pi=(\pi_1,\pi_2,\cdots,\pi_m)$，其中瓶颈机上的加工安排为 π_b。下面我们证明每台机器按 π_b 中工件的顺序排列，工件的完工时间不会增加。

首先我们对每台上游非机器 $M_k(k=1,\cdots,b-1)$ 按 π_b 的顺序排列工件，其余机器上保持原加工安排，得到新的调度 $\pi'=(\pi_b,\cdots,\pi_b,\pi_{b+1},\cdots,\pi_m)$，则有

$$\sum_{k=1}^{b-1} p_{\pi_b(1),k} + \sum_{j=1}^{i} p_{\pi_b(j),b} = C_{\pi_b(i),b}(\pi') \leq C_{\pi_b(i),b}(\pi) \quad i=1,\cdots,n;$$

因此，对任何 $i\in N$，有 $C_i(\pi')\leq C_i(\pi)$。

然后，对每台下游非机器 $M_k(k=b+1,\cdots,m)$ 按 π_b 的顺序排列工件，构造一个新的排列调度为 $\tau=(\pi_b,\pi_b,\cdots,\pi_b)$，则有

$$C_{\pi_b(i)}(\tau)-t_{\pi_b(1),b}(\tau) \leq C_{\pi_b(i)}(\pi')-t_{\pi_b(1),b}(\pi') \quad i=1,\cdots,n;$$

由 $t_{\pi_b(1),b}(\tau)=t_{\pi_b(1),b}(\pi')$，则有

$$C_{\pi_b(i)}(\tau) \leq C_{\pi_b(i)}(\pi') \leq C_{\pi_b(i)}(\pi)$$

由于目标函数为完工时间的正则函数，即目标函数是完工时间 C_i 的单调非减函数。因此对任何正则目标函数，排列排序 $\tau=(\pi_b,\pi_b,\cdots,\pi_b)$ 为最优调度。

由定理 3-2 可知，如果 Flow shop 上工件的加工时间满足条件(3-9a)和(3-9b)，那么最优调度为一排列排序。具体而言，Flow shop 中所有机器上工件的加工顺序与瓶颈机上工件的加工顺序相同。我们可以设计一个最优排列调度算法如下：

算法 3-1. 最优排列调度算法(Optimal Permutation Scheduling Algorithm)

Step 1： 按公式(3-5a)和(3-5b)估计瓶颈机上工件的到达时间和传递时间。

Step 2：对瓶颈机调度问题$1|r_i,q_i|f(C_i)$设计一个最优排序π_b，确定工件在瓶颈机上的开工时间。

Step 3：令非瓶颈机上工件的加工顺序都等于π_b，计算非瓶颈上工件的开工时间：

Step 3a. 采用后向 (Backward) 递推公式依次计算各工件在上游非瓶颈机上的开工时间：

$$t_{\pi_b(n),k}=t_{\pi_b(n),k+1}-p_{\pi_b(n),k}\quad k=b-1,\cdots,1;$$

$$t_{\pi_b(i),k}=\min(t_{\pi_b(i+1),k},t_{\pi_b(i),k+1})-p_{\pi_b(i),k}\quad i=n-1,\cdots,1;k=b-1,\cdots,1;$$

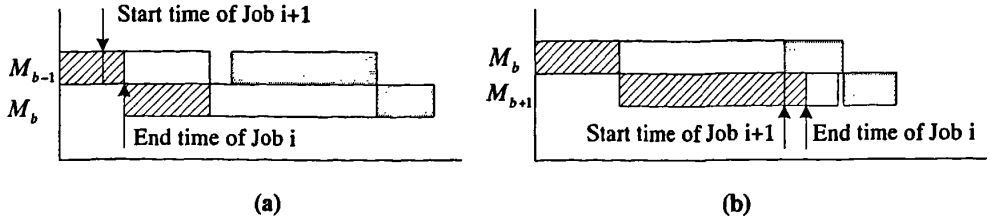
Step 3b. 采用前向 (Forward) 递推公式依次计算各工件在下游非瓶颈机上的开工时间：

$$t_{\pi_b(1),k}=t_{\pi_b(1),k-1}+p_{\pi_b(1),k-1}\quad k=b+1,\cdots,m;$$

$$t_{\pi_b(i),k}=\max(t_{\pi_b(i-1),k}+p_{\pi_b(i-1),k},t_{\pi_b(i),k-1}+p_{\pi_b(i),k-1})\quad i=2,\cdots,n;k=b+1,\cdots,m;$$

由算法 3-1 可知，我们只需要确定瓶颈机上工件的最优排序就可得到整个 Flow shop 问题的最优调度。如果瓶颈机调度算法(Step2)是多项式时间内可解的，则算法 3-1 也是在多项式时间内可解。大规模 Flow shop 调度问题可简化为单机调度问题求解，计算复杂度大大降低。此时非瓶颈机上工件的加工顺序与瓶颈机上工件的加工顺序相同。

然而公式(3-9a)和(3-9b)给出的非瓶颈机冗余能力条件过于严格，在实际生产过程中，上游非瓶颈机(或下游非瓶颈机)可能存在某些工序不能在瓶颈机设定的期望交货期限内完工，造成上游非瓶颈机(或下游非瓶颈机)上工序冲突(见图 3-2)。在此情况下，条件(3-9a)和(3-9b)不满足，此时排列调度不一定是原问题的最优调度。



(a) (b)

图 3-2. (a) 上游非瓶颈机工件冲突示意图 (b) 下游非瓶颈机工件冲突示意图

Fig 3-2. (a) Collision window on upstream non-bottleneck machines (b) Collision window on downstream non-bottleneck machines

尽管不均衡 Flow shop 问题中工件的加工特性不满足以上两个条件，但非瓶颈机的加工能力仍大于瓶颈机的加工能力。一旦确定好瓶颈机上的调度安排，则非瓶颈机上的能力约束相对更容易满足。因此我们采用列表调度(list scheduling, LS)算法来求解非瓶颈机调度问题，该方法是求解 Flow shop 调度问题的一种有效启发式算法。列表调度可简单描述为：当一台机器空闲时，从该机器上已到达的工件中选择优先级最高的工件加工，其中工件优先级可以按某些规则设置[37-39]。

算法 3-2. 列表调度算法 (List Scheduling algorithm, LS)

令 $N_k=\{1,\cdots,n\}$ 为机器 $k\in M$ 上所有工件的集合。U_k 为机器 M_k 上已调度的工件集，$\bar{U}_k$ 为机器 M_k 未调度的工件集，$\bar{U}_k=N_k\setminus\{U_k\}$。

Step 1：初始化。令 $k=1$。

Step 2：调度机器 k。

Step 2a: 令 $t_k=\min_{i\in N_k} r_{i,k}$，$U_k=\Phi$，$\bar{U}_k=N_k$。

Step 2b: 在 t_k 时刻，对未调度工件集合 $\bar{U}_k$ 中已到达的工件 i(满足 $r_{i,k}\le t_k$)设置优先级，选择优先级最大的工件 j 调度。

Step 2c: 更新 $U_k:=U_k\cup\{j\}$，$t_k:=\max\{t_k+p_{j,k},\min_{i\in\bar{U}} r_{i,k}\}$，$r_{j,k+1}=t_k(k\neq m)$。如果机器 k 上所有工序均调度完，转 Step 3；否则，转 Step 2b。

Step 3：停止条件。，如果 $k=m$，则所有机器上的工件均调度完，算法停止；否则更新 $k:=k+1$，转 Step 2a。

该算法的计算复杂度为 $O(mn\cdot\log n)$

由于经典 Flow shop 中每个工件都是按照相同的工艺路径依次经过机器 $M_1,M_2,\cdots,M_m$，每个工件必须在其前道工序完工之后才能开始下一道工序的加工，因此我们也可通过顺序求解一组单机子问题 $1|r_i|L_{\max}$ 获得。每个工序的到达时间和交货期限分别设置如下：

$$r_{i,k}=\begin{cases}0 & k=1;i=1,\cdots,n;\\ t_{i,k-1}+p_{i,k-1} & k=2,\cdots,m;i=1,\cdots,n;\end{cases} \tag{3-10a}$$

$$d_{i,k}=\begin{cases}d_{i,k+1}-p_{i,k+1} & k=b-2,\cdots,1;i=1,\cdots,n;\\ t_{i,b} & k=b-1;i=1,\cdots,n;\\ t_{i,b}+p_{i,b} & k=b;i=1,\cdots,n;\\ d_{i,k-1}+p_{i,k} & k=b+1,\cdots,m;i=1,\cdots,n;\end{cases} \tag{3-10b}$$

算法 3-3. 基于优先分派规则的启发式算法(Priority Dispatching Heuristic)

令 $N_k=\{1,\cdots,n\}$ 为机器 $k\in M$ 上所有工件的集合。U_k 为机器 M_k 上已调度的工件集，$\bar{U}_k$ 为机器 M_k 未调度的工件集，$\bar{U}_k=N_k\setminus\{U_k\}$。

Step 1：令 $t=\min_{i\in N_k} r_{i,k}$；$U_k=\Phi$，$\bar{U}_k=N_k$；

Step 2：在 t 时刻，对未调度工件集合 $\bar{U}_k$ 中已到达的工件 i(满足 $r_{i,k}\le t$)设置优先级，选择优先级最高的工件 j 调度。

Step 3：更新 $U_k:=U_k\cup\{j\}$，$t:=\max\{t+p_{j,k},\min_{i\in\bar{U}_k} r_{i,k}\}$。如果所有工件均调度完，算法停止；否则，转 Step 2。

该算法的计算复杂度为 $O(n\log n)$。其中工序优先级设置可以采用一些与交货期限相关的分派规则确定，如改进工序交货期限最小优先(modified operation due date, MOD)，近似拖期费用最大优先(apparent tardiness cost, ATC)，剩余加工时间最大优先(most work remaining, MWR) 以及关键比最大优先 (critical ratio, CR) 规则[39]。

3.3.4 瓶颈机与非瓶颈机之间的协调

如前所述，瓶颈机的调度安排可以通过优化求解一个单机子问题 $1|r_i,q_i|f(C_i)$ 获得，其调度结果(工件在瓶颈机上的开工时间和完工时间)限制了非瓶颈机上工件的最早可能的开工时间和最迟的完工时间，从而非瓶颈机围绕着瓶颈机而调度。其中瓶颈机上工件的到达时间体现了瓶颈机与上游非瓶颈机之间的关联，而瓶颈机上工件的传递时间则体现了瓶颈机与下游非瓶颈机之间的关联。

当上游非瓶颈机上有一个工件不能在瓶颈机给定的期望交货期限之前完工，工序开工时间在时间轴上存在迭代，即瓶颈机与非瓶颈机上的工序产生冲突，调度不可行。为了保证最终调度的可行，需要重新调整瓶颈机上工序的到达时间。单瓶颈分解方法的协调机制如图 3-3 所示。

它基于三层递阶模型：协调层，瓶颈机调度层和非瓶颈机调度层。协调层的目标在于根据车间调度层(瓶颈机调度层和非瓶颈机调度层)的信息估计(或调整)瓶颈机上每个工件的到达时间 r_i 和传递时间 q_i，并将估计值传递至瓶颈机调度层。瓶颈机调度层的目标在于根据估计的关联参数(r_i 和 q_i)调度瓶颈机，获得瓶颈机上工件的开工时间 $t_{i,b}$ 和完工时间 $C_{i,b}$。由于瓶颈机上工件的开工时间决定了上游非瓶颈机上工件的交货期限，将 $t_{i,b}$ 传递至上游非瓶颈机调度器，从后往前依次调度上游非瓶颈机。同时

瓶颈机上工件的完工时间决定了上游非瓶颈机上工件的最早可能的开工时间，将$C_{i,b}$传递至下游非瓶颈机调度器，从前往后依次调度下游非瓶颈机。第一个决策过程是基于协调层和调度层(瓶颈机调度层和非瓶颈机调度层)之间的垂直信息流；第二个决策过程是基于瓶颈机调度层和非瓶颈机调度层的垂直信息流(见图 3-3 中虚线)。

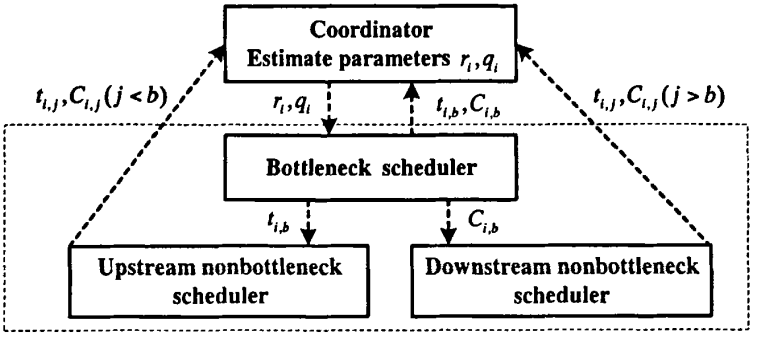


图 3-3. Flow shop 瓶颈分解方法三层递阶结构图

Fig 3-3. The three-level architecture of the BDP for flow shop problem

本文采用一种简单但直接有效的方式协调瓶颈机调度结果与非瓶颈机调度结果之间的冲突。如果上游非瓶颈机上至少存在一个工件J_i其完工时间小于瓶颈机调度确定的开工时间，则调整瓶颈机上工件J_i的到达时间为其前道工序(在非瓶颈机上加工)的完工时间。然后根据更新的到达时间重新调度瓶颈机，这相当于右移重调度策略。

为了保证迭代过程的收敛速度，每重新调度一次瓶颈机则固定一个工件的到达时间。令Ω表示已固定到达时间的工件集合，则瓶颈机上每个工件的到达时间更新如下：

$$r_i = \begin{cases} \max\left\{\sum_{j=1}^{b-1} p_{i,j}, C^*_{i,b-1}\right\} & if\ i \notin \Omega\ and\ C^*_{i,b-1} > t^*_{i,b} \\ r_i & otherwise \end{cases} \tag{3-11a}$$

如果瓶颈调度可行，则可以调整瓶颈机上工序的传递时间来改进整个调度的性能。如果某个工件的完工时间小于其交货期限，则该工件早到，可以减少该工件在瓶颈机上的传递时间，使得该工件在瓶颈机上的加工安排推后；反之如果某一工件在交货期限之后完工，则可以提高该工件在瓶颈机上的传递时间，使得该工件在瓶颈机上的加工安排提前。更新瓶颈机上工件的传递时间如下：

$$q_i = \max\left\{q_i + (C^*_{i,m} - d_i), \sum_{j=b+1}^{m} p_{i,j}\right\} \quad i = 1,\ldots,n; \tag{3-11b}$$

为了保证改进算法的收敛速度，当算法没有改进时则停止迭代，选择最好值为最终调度。给出单瓶颈分解算法如下。

算法 3-4. Flow shop 瓶颈分解算法(Bottleneck-based Decomposition Procedure, BDP)

Step 1：辨识瓶颈机。选择加工负荷最严重的机器为瓶颈机 M_b，令瓶颈机上已固定到达时间工件集合 $\Omega = \Phi$。

Step 2：估计关联项。按公式(3-5a)和(3-5b)设置瓶颈机上工件的初始到达时间 r_i 和传递时间 q_i。

Step 3：调度瓶颈机。建立并优化求解瓶颈机调度模型 P_b。固定工件 i^* 的到达时间，其中 $i^* = \arg\max_{i\notin\Omega}\{t_{i,b}^*\}$，更新 $\Omega := \Omega \cup \{i^*\}$。

Step 4：调度上游非瓶颈机。按公式(3-10a)和(3-10b)分别设置上游非瓶颈机上工件的到达时间和交货期限。采用基于优先分派规则的启发式算法求解上游非瓶颈机。

Step 5：检测可行性。如果调度不可行，则按公式(3-11a)更新瓶颈机上工件的到达时间 r_i，转 Step 3；否则转 Step 6。

Step 6：调度下游非瓶颈机。按公式(3-10a)和(3-10b)分别设置下游非瓶颈机上工件的到达时间和交货期限。采用基于优先分派规则的启发式算法求解下游非瓶颈机。

Step 7：迭代停止准则。如果解没有改进或 $|\Omega| = n$，停止迭代。否则按公式(3-11b) 更新瓶颈机上工件的传递时间 q_i，转 Step 3。

3.4 仿真测试与分析

随着制造业对客户满意程度的日益提高，希望生产方尽可能满足客户需求。一般选择两类性能指标来衡量客户满意程度：最大滞后时间与总拖期时间之和。本节将分别研究单瓶颈分解算法求解两类不同性能指标的 Flow shop 调度问题(即 $F_m \| L_{\max}$ 和 $F_m \| \sum T_i$ 问题)的性能。

1. $F_m \| L_{\max}$ 问题仿真测试

此问题目标函数为最大滞后时间，即 $f(C_i) = \max_{i\in N}(C_i - d_i)$，它可以衡量客户不满意度最差的程度。当所有工件的交货期限 d_i 相同时，最大滞后时间 $L_{\max}$ 与最大完工时间 $C_{\max}$ 等价。为了测试单瓶颈分解算法在调度问题 $F_m \| L_{\max}$ 中的性能，按文献[108]的方式随机生成大量测试数据，实验参数如表 3-1 所示。

表 3-1. 测试数据生成参数表

Table 3-1. Experimental design for randomly generated test problem

Problem parameter	Value	Total Value
Number of machines m	10, 20, 30	3
Number of jobs n	30, 50, 100, 200	4
Percentage tardy jobs T	0.3, 0.6	2
Due date range R	0.5, 2.5	2
Workload difference l	0.25(low), 0.5(medium), 0.75 (high)	3
Position of bottleneck b	The first 1/4, the second 1/4, the third 1/4 and the last 1/4	4
Total problem combinations	$3\times4\times2\times2\times3\times4 = 576$	

所有工件均在零时刻到达，非瓶颈机上每个工件的加工时间服从[1,50]均匀分布，瓶颈机上每个工件的加工时间服从$[1+l,50\cdot(1+l)]$均匀分布，机器之间负荷差异l越大，瓶颈程度越高。瓶颈机的位置分别选择在流水车间的前 1/4 位置，第 2 个 1/4 的位置，第 3 个 1/4 的位置，最后一个 1/4 的位置。工件的交货期限服从$[P'(1-T-R/2),P'(1-T+R/2)]$均匀分布，其中$P'$是最大完工时间的下界。所有参数组合生成 576 种类型的问题，每种类型分别生成 10 组数据，共 5760 组测试数据。

对问题$F_m \| L_{\max}$，对应瓶颈机子问题为$1|r_i,q_i|L_{\max}$。虽然该问题计算复杂度为 NP-complete[3]，但 Carlier (1982) 提供了一个有效的分枝定界算法精确求解该问题[117]。Carlier 算法可在较短的时间内求解最大规模为 1,000 个工件的问题[94]。本文提出的单瓶颈分解算法(BDP)中，瓶颈机子问题采用 Carlier 算法精确求解，非瓶颈机采用基于 MOD 规则的启发式算法(具体见算法 3-3)。

将本文提出的单瓶颈分解算法的调度性能与移动瓶颈方法进行比较。其中移动瓶颈方法分别采用不同的子问题建模和重新优化过程。子问题建模分别采用带到达时间和传递时间约束的单机子模型[40]和考虑迟延顺序约束的单机子模型[94]，分别记为 SBI 和 SBII。同时考虑两种不同的重新优化过程，即无重新优化过程和完全重新优化过程，分别记为 SB(NR)和 SB(FR)。两种不同的子问题建模和两种不同的重新优化方式组合生成四种不同的移动瓶颈算法，分别记为 SBI(NR)，SBI(FR)，SBII(NR)和 SBII(FR)。在移动瓶颈方法中，所有的子问题均采用 Carlier 算法。为了评价不同非瓶颈机的求解算法对本文提出的单瓶颈分解算法性能的影响，我们分别采用 Carlier 算法和 MOD 规则，记为算法 BDPI 和 BDPII。

为了评价单瓶颈分解算法的解的质量，采用两个评价指标[95]：目标值与所有算法中最好值的相对比值$\rho(H,S)$和目标值与下界的相对比值$\eta(H,S)$。

令$f(H,I)$为对给定实例I下采用启发式算法H求得的目标函数$f(C_i)$值，且所有

算法中最好值 $BEST(I) = \min_H \{f(H,I)\}$。定义一组满足相同特征的问题 S，对每类问题 S 计算目标值与最好值平均比值为：

$$\rho(H,S) = \sum_{I\in S} f(H,I) / \sum_{I\in S} BEST(I) \tag{3-12}$$

令 $LB(I)$ 为问题 I 的下界，对每类问题 S 计算目标值与下界平均比值 $\eta(H,S)$ 为：

$$\eta(H,S) = \sum_{I\in S} f(H,I) / \sum_{I\in S} LB(I) \tag{3-13}$$

其中 $LB(I)$ 是 BDP 算法第一次迭代求解简化问题 P_b 获得的最优值(参见定理 3-1)。计算时间的评价指标采用所有实例的平均 CPU 计算时间(average CPU time, ACT)，启发式方法 H 求解问题 S 的平均 CPU 计算时间记为 $ACT(H,S)$。

所有的算法均采用 C 语言编程，运行在 Celeron 处理器 (CPU 速度 2GHZ，缓存 128kB，内存 256MB) 的机器上。

首先，比较单瓶颈分解算法和移动瓶颈方法。仿真结果如表 3-2，3-3 和 3-4 所示。表 3-2 显示不同算法求解不同类型问题的平均比值 ρ，表 3-3 显示不同算法求解不同类型问题的平均比值 η，表 3-4 为平均计算时间 ACT。

表 3-2. 不同类型问题下算法的性能 ρ

Table 3-2. Solution quality ratio ρ of the algorithms for randomly generated problems

Problem S	$\rho(H,S)$ Heuristic H					
	SBI (NR)	SBI (FR)	SBII (NR)	SBII (FR)	BDP I	BDP II
Number of machines m						
10	1.056	1.010	1.032	1.001	1.027	1.020
20	1.074	1.029	1.047	1.002	1.038	1.031
30	1.088	1.037	1.059	1.001	1.041	1.038
Number of jobs n						
30	1.072	1.012	1.036	1.002	1.032	1.031
50	1.041	1.007	1.028	1.001	1.021	1.009
Due date (T, R)						
(0.3, 0.5)	1.070	1.008	1.046	1.005	1.031	1.007
(0.3, 2.5)	1.065	1.052	1.019	1.001	1.023	1.074
(0.6, 0.5)	1.056	1.008	1.041	1.002	1.030	1.010
(0.6, 2.5)	1.067	1.016	1.042	1.000	1.032	1.051
Workload difference l						
0.25	1.070	1.033	1.053	1.004	1.030	1.048
0.50	1.062	1.011	1.045	1.002	1.029	1.032
0.75	1.038	1.002	1.021	1.001	1.025	1.027
Position of bottleneck b						
The first 1/4	1.082	1.035	1.028	1.001	1.023	1.033
The second 1/4	1.057	1.021	1.037	1.003	1.027	1.033
The third 1/4	1.059	1.018	1.040	1.003	1.033	1.042
The final 1/4	1.057	1.019	1.036	1.001	1.031	1.046

表 3-3. 不同类型问题下算法的性能η

Table 3-3. Solution quality ratio η of the algorithms for randomly generated problems

Problem S	$\eta(H,S)$ Heuristic H					
	SBI (NR)	SBI (FR)	SBII (NR)	SBII (FR)	BDP I	BDP II
Number of machines m						
10	1.121	1.064	1.092	1.050	1.085	1.077
20	1.252	1.177	1.208	1.162	1.191	1.188
30	1.355	1.254	1.289	1.227	1.270	1.256
Number of jobs n						
30	1.090	1.155	1.187	1.143	1.182	1.179
50	1.030	1.130	1.156	1.123	1.147	1.133
Due date (T, R)						
(0.3, 0.5)	1.253	1.179	1.223	1.174	1.205	1.176
(0.3, 2.5)	1.204	1.190	1.154	1.135	1.158	1.225
(0.6, 0.5)	1.182	1.121	1.162	1.109	1.170	1.118
(0.6, 2.5)	1.222	1.162	1.193	1.243	1.181	1.204
Workload difference l						
0.25	1.217	1.173	1.197	1.139	1.170	1.191
0.50	1.199	1.131	1.170	1.120	1.151	1.155
0.75	1.189	1.120	1.142	1.119	1.147	1.149
Position of bottleneck b						
The first 1/4	1.182	1.179	1.170	1.039	1.165	1.176
The second 1/4	1.219	1.148	1.166	1.127	1.155	1.162
The third 1/4	1.198	1.152	1.178	1.135	1.160	1.180
The final 1/4	1.185	1.137	1.158	1.116	1.152	1.163

如表 3-2 和 3-3 所示，带完全重新优化的移动瓶颈方法 (SBI(FR) 和 SBII(FR)) 的调度值最优，其次是单瓶颈分解方法 (BDPI 和 BDPII)，无重新优化的移动瓶颈方法 (SBI(NR)) 和 SBII (NR)) 的调度值最差。而且算法 BDPI 和 BDPII 的调度性能几乎差不多。算法 BDPII 比算法 SBI(NR) 和 SBII(NR) 平均好 5% 和 2%，但它比算法 SBI(FR) 和 SBII(FR) 平均差 1% 和 3%。如表 3-2 和 3-3 所示，子问题模型和重新优化过程影响移动瓶颈方法的性能。考虑迟延顺序约束的子问题建模的移动瓶颈算法 (SBII) 比不考虑迟延顺序约束的子问题建模的移动瓶颈算法(SBI)更优，即 $\rho(SBII,S) < \rho(SBI,S)$ 和 $\eta(SBII,S) < \eta(SBI,S)$。而带重新优化过程的移动瓶颈方法比无重新优化过程的移动瓶颈方法性能更优，即 $\rho(SB(FR),S) < \rho(SB(NR),S)$ 和 $\eta(SB(FR),S) < \eta(SB(NR),S)$。算法 BDPI 和 BDPII 相对最优值比值 ρ 几乎相等，它们相对下界比值 η 也相差不大。因此非瓶颈机的调度对单瓶颈机调度算法的性能影响不大。

为了研究不同参数对单瓶颈分解算法影响，对表 3-2 和 3-3 的数据进行纵向比较分析。表 3-2 显示算法 BDP 的性能随机器个数的增加而下降。当瓶颈机个数从 10 升至 30，算法 BDP 的调度值相对最优值的比值平均增加了 3.8%。原因在于随着机器

个数的增加，瓶颈机对流水作业车间的主导作用减弱。表 3-3 显示当机器个数从 10 升至 30，算法 BDP 的调度值相对下界的比值平均增加了 17.9%。下界性能也随着机器个数的增加而下降。

然而算法 BDP 的性能随工件个数的增加而提高。表 3-2 显示当工件个数从 30 升至 50，算法 BDP 的调度值相对最优值的比值平均减少了 2.2%。原因可能是随着工件个数的增加，流水作业车间的瓶颈变得更稳定。表 3-3 显示当工件个数从 30 升至 50，算法 BDP 的调度值相对下界的比值平均减少了 4.6%。下界性能也随着工件个数的增加而提高。

瓶颈程度是影响算法 BDP 性能的重要参数。由表 3-2 和 3-3 可知，算法 BDP 的性能随着机器加工负荷差异 l 的增加而提高。当 l 从 0.25 增至 0.5，算法 BDP 的调度性能平均改进 2%。瓶颈程度越高，算法 BDP 性能越好。而且算法 BDP 的下界也随 l 的增加而提高。当机器加工负荷差异 l 从 0.25 增至 0.5，算法 BDP 的调度值相对下界的比值平均减少了 4%。但是机器位于流水作业车间的位置对算法 BDP 的解的质量影响不大。

交货期分布也影响算法 BDP 的性能。算法 BDP 的性能随着交货期范围 R 值的增加而下降。当交货滞后程度 T 为 0.3，R 从 0.5 提高至 2.5 时，算法 BDP 的调度值相对最优值的比值 ρ 平均增加 6%；当交货滞后程度 T 为 0.6，R 从 0.5 提高至 2.5 时，算法 BDP 的调度值相对最优值的比值 ρ 平均增加 4%。而且交货期限越紧，获得的下界性能越差。当交货滞后程度 T 为 0.3，R 从 0.5 提高至 2.5 时，算法 BDP 的调度值相对下界的比值 η 平均增加 5%；当交货滞后程度 T 为 0.6，R 从 0.5 提高至 2.5 时，算法 BDP 的调度值相对下界的比值 η 平均增加 9%。

表 3-4. 不同问题规模下算法的平均计算时间

Table 3-4. Average computation time of the algorithms for different problem sizes

Problem S	ACT (s)					
(m,n)	SBI (NR)	SBI (FR)	SBII (NR)	SBII (FR)	BDP I	BDP II
(10, 30)	2.5	45.8	2.6	45.7	5	0
(10, 50)	11.8	200.5	12.5	195.3	40.2	0.10
(20, 30)	40.3	1268.5	42.1	1255.6	76.8	0.01
(20, 50)	172.2	5677.3	168.8	5642.7	402.1	0.20
(30, 30)	208.1	8536.4	185.4	8407.9	459.5	0.05
(30, 50)	828.8	39757.8	828.4	39442.6	2551.8	0.30

尽管带完全重新优化的移动瓶颈算法的性能优于算法 BDP，但其 CPU 计算时间很长。如表 3-4 所示，带完全重新优化的移动瓶颈算法(算法 SBI(FR)和 SBII(FR))求

解规模为 30 台机器 50 个工件的问题计算时间需要 11 个多小时，这在实际生产过程中是无法接受的。而且移动瓶颈方法的计算时间随着机器和工件的增加指数增长。然而本文提出的单瓶颈分解算法(算法 BDPII)的计算时间非常快，它求解 30 台机器 50 个工件的问题只需要 0.03 秒。因此单瓶颈分解算法可以在很短的时间内获得一个较好的解。该算法可以用于求解大规模 Flow shop 调度问题。

到目前为止，移动瓶颈分解算法可求解问题规模最大为 30 台机器 50 个工件。为了研究算法 BDP 求解大规模问题的性能，我们选择后 1920 组数据对该算法进行测试。在这些数据中机器的个数分别为 20 和 30，工件的个数分别为 100 和 200，工序的个数在 2000 至 6000 之间。表 3-5 显示算法 BDPII 的调度值相对下界的平均比值 η 和平均计算时间。

表 3-5. 大规模 Flow shop 调度问题算法 BDPII 的性能

Table 3-5. Performance of the algorithm BDPII for large-scale Flow shop problems

Problem S (m,n)	$\eta(H,S)$	ACT (s)
(20, 100)	1.190	0.46
(20, 200)	1.281	3.30
(30, 100)	1.254	0.48
(30, 200)	1.343	3.46

如表 3-5 所示，算法 BDPII 相对下界的比值 η 随着机器和工件个数的增加而增加。原因可能是随着机器和工件个数的增加下界性能下降。仿真结果显示算法 BDPII 可以在较短的时间内获得一个好的解，本文提出的单瓶颈分解算法适合求解大规模 Flow shop 调度问题。

综合分析以上仿真结果可得：

(1) 带完全重新优化的移动瓶颈算法的调度性能优于本文提出的单瓶颈分解算法，但前者的计算时间很长，不适合求解大规模问题。

(2) 本文提出的单瓶颈分解算法的调度性能优于无重新优化过程的移动瓶颈算法，而且前者的计算时间远比后者更快。算法 BDPII 可以在较短的时间内获得好的解。

(3) 算法 BDPI 和 BDPII 的性能几乎差不多。非瓶颈机的调度与瓶颈机调度相比，它对性能的影响相对较小。

(4) 瓶颈程度对单瓶颈分解算法的性能影响很大。瓶颈程度越高，算法越有效。

(5) 单瓶颈分解算法的性能随机器个数的增加而下降，原因可能是瓶颈主导作用随着机器个数的增加而减弱。然而该算法的性能随工件个数的增加而提高，原因可能是随着工件个数的增加，Flow shop 中的瓶颈机更加稳定。

2. $F_m \| \sum T_i$ 问题仿真测试

此问题目标函数为总拖期时间之和，即 $f(C_i) = \sum_{i \in N} T_i$，其中 $T_i = \max(C_i - d_i, 0)$ 为工件 J_i 的拖期时间，它可以衡量所有客户总的满意程度。

为了测试单瓶颈分解算法在调度问题 $F_m \| \sum T_i$ 中的性能，按文献[116]的方式随生成大量的测试数据，实验参数如表 3-6 所示。所有的工件均在零时刻到达，瓶颈机上每个工件的加工时间服从[1,50]均匀分布，非瓶颈机上每个工件的加工时间服从 $[1 \cdot l, 50 \cdot l]$ 均匀分布。每个工件的交货期限服从 $[P'(1-T-R/2), P'(1-T+R/2)]$ 均匀分布，其中 P' 是最大完工时间 $C_{\max}$ 的下界。所有参数组合生成 432 种类型的问题，每种类型分别生成 10 组数据，共 4320 组测试数据。

表 3-6. 测试数据生成参数表

Table 3-6. Experimental design for randomly generated test problem

Problem parameter	Value	Total Value
Number of machines m	10, 20, 30	3
Number of jobs n	50, 100, 200	3
Percentage tardy jobs T	0.1, 0.5	2
Due date range R	0.8, 1.8	2
Workload difference l	0.3 (high), 0.5 (medium), 0.8 (low)	3
Position of bottleneck b	The first 1/4, the second 1/4, the third 1/4 and the last 1/4	4
Total problem combinations	3×3×2×2×3×4 = 432	

对问题 $F_m \| \sum T_i$，对应瓶颈机子问题为 $1 | r_i | \sum T_i$。该问题计算复杂度为强 NP-hard[6]，目前为止没有一种有效的分枝定界算法可以在较短的时间内获得最优解，单机问题 $1 | r_i | \sum T_i$ 多采用启发式算法。因此本文提出的单瓶颈分解算法(BDP)中瓶颈机子问题 $1 | r_i | \sum T_i$ 采用基于 MOD 规则的启发式算法(见算法 3-3)，同时上游非瓶颈机与下游非瓶颈机分别采用基于 MOD 规则的列表调度算法(具体见算法 3-2)。

将单瓶颈分解算法分别与基于分派规则的列表调度算法进行比较[39]。6 种分派规则用于列表调度算法中：先到先加工(first come first served, FCFS)，最短加工时间先加工(shortest processing time, SPT)，最早交货期限先加工(earliest due date, EDD)，最早工序交货期限先加工(modified operation due date, MOD)，拖期费用最大者先加工(apparent tardiness cost, ATC)和改进型 PSK 规则(modified PSK rule, MPSK)。

由于问题 $F_m \| \sum T_i$ 的下界很多情况为 0，因此只采用目标值与最好值平均比值

$\rho(H,S)$来评价算法解的质量(按公式(3-12)计算)。计算时间的评价指标采用平均 CPU 计算时间$ACT(H,S)$。

所有的算法均采用 C 语言编程，运行在 Celeron 处理器 (CPU 速度 2GHZ，缓存 128kB，内存 256MB)的机器上。仿真结果如表 3-7，3-8，3-9 和 3-10 所示。

表 3-7. 不同加工负荷差异下算法的性能比较

Table 3-7. Performance of the algorithms for different workload difference

Problem S	$\rho(H,S)$						
	BDP	FCFS	SPT	EDD	MOD	ATC	MPSK
l = 0.3	1.000	1.359	1.213	1.338	1.211	1.280	1.219
l = 0.5	1.000	1.545	1.348	1.432	1.282	1.373	1.323
l = 0.8	1.000	3.254	2.665	1.756	1.507	1.648	2.507

表 3-8. 瓶颈机位于生产线不同位置下算法的性能比较

Table 3-8. Performance of the algorithms for different positions of the bottleneck machine

Problem S	$\rho(H,S)$						
	BDP	FCFS	SPT	EDD	MOD	ATC	MPSK
The first 1/4	1.000	2.355	1.719	1.505	1.330	1.432	1.731
The second 1/4	1.000	1.450	1.762	1.515	1.335	1.437	1.718
The third 1/4	1.000	3.095	1.799	1.506	1.330	1.431	1.705
The last 1/4	1.000	1.311	1.688	1.509	1.338	1.435	1.579

表 3-9. 不同交货期限范围和滞后程度下算法的性能比较

Table 3-9. Performance of the algorithms for different due date ranges and tardiness factors

Problem S	$\rho(H,S)$						
(R,T)	BDP	FCFS	SPT	EDD	MOD	ATC	MPSK
(0.8, 0.1)	1.000	2.355	1.890	1.650	1.496	1.606	2.208
(1.8, 0.5)	1.000	1.450	1.273	1.396	1.250	1.334	1.253
(0.8, 0.1)	1.000	3.095	2.602	1.728	1.427	1.588	2.112
(1.8, 0.5)	1.000	1.311	1.203	1.262	1.161	1.207	1.161

由表 3-7, 3-8, 3-9 和 3-10 可以看出，单瓶颈分解算法(BDP)的调度性能优于基于优先分派规则的列表调度算法。表 3-7 和表 3-8 给出不同瓶颈程度下瓶颈分解算法的性能。随着瓶颈机与非瓶颈机之间的负荷差异程度增大(对应 l 值减小)，单瓶颈分解算法的调度性能提高。直觉上瓶颈机与非瓶颈机负荷差异越大，非瓶颈机的冗余能力越高，瓶颈的主导作用越强。该算法对能力不均衡程度的流水作业生产线尤其有效。瓶颈机位于生产线的位置对瓶颈分解算法的调度质量影响不大。表 3-9 给出不同交货期限范围和滞后程度下单瓶颈分解算法的性能。仿真结果显示交货期限越松，单瓶颈

分解算法的性能越好。

表 3-10. 不同问题规模下算法的性能比较

Table 3-10. Performance of the algorithms for different problem sizes

问题 S (m,n)	$\rho(H,S)$							ACT(s)
	BDP	FCFS	SPT	EDD	MOD	ATC	MPSK	BDP
(10, 50)	1.000	1.867	1.533	1.584	1.355	1.456	1.493	0.001
(20, 50)	1.000	1.497	1.344	1.474	1.326	1.382	1.327	0.002
(30, 50)	1.000	1.380	1.282	1.388	1.280	1.314	1.279	0.003
(10, 100)	1.000	2.580	2.071	1.616	1.370	1.514	1.914	0.005
(20, 100)	1.000	1.712	1.480	1.487	1.330	1.423	1.426	0.008
(30, 100)	1.000	1.513	1.361	1.430	1.310	1.379	1.329	0.010
(10, 200)	1.000	3.938	3.156	1.612	1.362	1.536	3.065	0.010
(10, 200)	1.000	2.134	1.813	1.486	1.318	1.443	1.748	0.030
(10, 300)	1.000	1.852	1.638	1.503	1.351	1.458	1.568	0.040

由表 3-10 可知，随着问题规模的增大单瓶颈分解算法相对规则调度的比值差减少，性能下降。但该算法的计算效率非常快，对规模为 30 台机器和 100 个工件的问题只需要 0.01 秒，该方法适用于求解大规模 Flow shop 调度问题。

因此，本文提出的单瓶颈分解算法是一种有效的启发式算法，它可以在很快的时间内获得较好的解。而且瓶颈程度越高，算法的性能越好，它适合求解大规模不均衡 Flow shop 调度问题。

3.5 本章小结

本文以大规模能力不均衡 Flow shop 调度问题为背景，针对该生产线存在一个主导瓶颈的加工特点提出了一种单瓶颈分解方法。该方法利用不均衡流水作业车间机器能力差异，将流水线上的机器分为瓶颈机、上游非瓶颈机和下游非瓶颈机。在假设非瓶颈机加工能力无限大的前提下，将原问题简化为一单机调度问题，通过优化求解该问题可以获得瓶颈机上工件的加工安排。随之给出了非瓶颈机冗余能力的约束条件，证明满足此条件的 Flow shop 问题的最优调度为排列调度，即非瓶颈机上工件的加工安排与瓶颈机的调度安排一致。因此，我们只需要确定瓶颈机上工件的最优排序就可得到整个 Flow shop 问题的最优调度。大规模 Flow shop 调度问题的优化求解可简化为单机调度问题的求解，计算复杂度大大降低。

由于给出的非瓶颈机冗余能力条件过于苛刻，很多情况下非瓶颈机的加工能力不满足上述条件，因此需要确定非瓶颈机上的加工安排。由于瓶颈机的调度结果决定了非瓶颈机上工序最早可能的开工时间和最迟必须的完工时间。根据瓶颈机调度结果对

非瓶颈机依次调度，其中上游非瓶颈机采用有效的分派规则从后往前依次调度，而下游非瓶颈机则采用规则从前往后依次调度。通过协调瓶颈机和非瓶颈机之间的关联可以保证整个 Flow shop 调度的可行性。该方法分别应用于两类不同性能指标的 Flow shop 调度问题。大量的仿真结果显示，本文提出的单瓶颈分解算法可以在较短的时间内获得较好的解，该方法适用于大规模不均衡 Flow shop 调度问题的求解。

第四章 单瓶颈 Job shop 调度问题瓶颈分解方法研究

4.1 引言

随着客户需求的多样化、大量的研究工作从面向库存的生产问题转向面向订单的生产调度问题，它们通常描述为一个 Job shop 调度问题。不同于 Flow shop 调度问题，经典 Job shop 调度问题中每个工件按照给定的工艺路径依次通过每台机器，且每个工件的工艺路径不同。

上一章研究了大规模不均衡 Flow shop 调度问题，并提出了一种有效的单瓶颈分解算法。在此基础上，本章研究大规模能力不均衡的 Job shop 调度问题，根据该生产线的瓶颈特性提出一种有效的分解方法。

本章的主要内容安排如下：4.2 节给出 Job shop 调度问题的一般描述并建立数学规划模型。在 4.3 节针对不均衡 Job shop 生产线的瓶颈特点提出了一种单瓶颈分解方法。在 4.4 节将该方法分别应用于两类不同性能指标的 Job shop 调度问题，并进行大量的仿真测试并进行分析。

4.2 Job shop 调度问题描述和分析

经典 Job shop 调度问题可描述如下。n 个工件 $J_i(i=1,\cdots,n)$ 需要在 m 台机器 $M_k(k=1,\cdots,m)$ 上加工。每个工件 J_i 在每台机器上仅加工一次，每个工件需要完成一组操作 $O_{i1},O_{i2},\cdots,O_{im}$，其中 O_{ik} 表示工件 J_i 在机器 M_k 上的工序。每个工件的工艺路径已知且不同，一个工序一旦开始加工则不允许中断。同一时刻每台机器最多只能加工一个工件，同一时刻每个工件也最多只能在一台机器上加工。我们希望确定每台机器上工件的加工顺序和开工时间使得与工件完工时间 C_i 相关的性能指标 $f(C_i)$ 最小。建立 Job shop 调度问题数学规划模型如下：

$$(P) \qquad \min_{x_{ijk},t_{i,k}} f(C_i) \tag{4-1a}$$

$$\text{s.t.} \quad t_{i,s_i(k)} - t_{i,s_i(k-1)} \geq p_{i,s_i(k-1)} \quad i=1,...,n; k=2,...,m; \tag{4-1b}$$

$$t_{i,s_i(1)} \geq 0 \quad i=1,...,n; \tag{4-1c}$$

$$t_{j,k} - t_{i,k} + \mathrm{H}(1-x_{ijk}) \geq p_{i,k} \quad i=1,\cdots,n-1; j=i+1,\cdots,n; k=1,\cdots m; \tag{4-1d}$$

$$t_{i,k} - t_{j,k} + \mathrm{H}x_{ijk} \geq p_{j,k} \quad i=1,\cdots,n-1; j=i+1,\cdots,n; k=1,\cdots m; \tag{4-1e}$$

$$C_i = t_{i,s_i(m)} + p_{i,s_i(m)} \quad i = 1,\cdots,n; \tag{4-1f}$$

$$x_{ijk} \in \{0,1\} \quad i,j = 1,...,n; i \neq j; k = 1,...,m; \tag{4-1g}$$

其中 $s_i(k)$ 表示工件 J_i 第 k 道工序加工所在机器的标号。决策变量 $t_{i,k}$ 表示工件 J_i 在机器 M_k 上的开工时间；x_{ijk} 为二进制变量，它表示机器 M_k 上工件 J_i 和 J_j 的加工顺序。如果机器 M_k 上工件 J_i 在工件 J_j 之前开工，则 $x_{ijk}=1$；否则 $x_{ijk}=0$。$p_{i,k}$ 表示工件 J_i 在机器 M_k 上的加工时间，d_i 表示工件 J_i 的交货期限，它们是已知且固定的。H 是一个非常大的正整数。每个工件的完工时间 C_i 按公式(4-1f)计算。目标函数(4-1a)为工件完工时间的一个正则函数。约束(4-1b)为工件加工顺序约束，约束(4-1c)表示每个工件必须在其到达之后才能开始加工。约束(4-1d)和(4-1e)为机器能力约束，它们保证同一时刻每台机器最多加工一个工件。

除了特殊问题，大部分 Job shop 调度问题是强 NP-hard 问题，面临的计算求解困难[6]。由模型 P 可知决策变量和约束个数均为 $m \cdot n^2$。尽管精确算法如分枝定界算法可以获得最优调度，但计算时间随问题规模的增加呈指数增长，因此它只适用于小规模问题的求解。到目前为止，它可优化求解的最大规模为 10 台机器和 20 个工件的问题[35]。实际上，大多数 Job shop 调度问题通常采用启发式分派规则，该方法计算效率高且易于实现，但它只利用局部信息进行决策，得到的调度性能往往较差。因此大量的研究工作主要集中在近似解的获取上。

移动瓶颈(SB)方法是求解 Job shop 问题的一种非常有效的启发式方法。它将 Job shop 问题分解为若干更易求解的单机子问题。每次迭代优化求解每个未调度的机器，并选择最大滞后时间 $L_{\max}$ 值最大的机器为关键机(或瓶颈机)，然后根据新固定的瓶颈机的调度结果重新优化已调度的机器。该方法最初由 Adams 等(1988)针对 $J_m \| C_{\max}$ 问题提出[40]，Dauzere-Peres 和 Lasserre(1993)改进 SB 方法，对分解后的子问题添加迟延顺序约束，并提出一种启发式算法求解该单机子问题[93]。Balas 等(1995)对该单机子问题提出一种精确算法[94]。随后 SB 方法推广至不同的性能指标问题，如最大迟延时间($L_{\max}$)[95]，总加权拖期时间(TWT)[118]。Mason 等将 SB 方法推广至更复杂的 Job shop 调度问题[54, 66-69]。大量的仿真实验显示 SB 方法可以获得高质量的解，但需要耗费大量的计算时间[91]。原因在于子问题求解和重新优化过程是 SB 方法中最耗时的过程。每次迭代，SB 对每个子问题均优化求解并对已调度的机器重新优化。随着机器和工件个数的增加，SB 算法的计算量显著增长。目前移动瓶颈分解方法可求解的最大规模为 30 台机器 50 个工件的问题[95]。

SB 方法的大量仿真研究多是针对均衡生产线而言，即所有机器上工序的加工时

间均服从同一均匀分布。这暗含了所有机器的利用率几乎相等，因此不可能存在一台瓶颈机主导其它机器。在实际生产线中，机器加工能力不可能完全均衡。相反，少数机器的利用程度要远比其它机器高。Uzsoy 和 Wang(2000)指出移动瓶颈方法可以很好地求解不均衡 Job shop 调度问题[108]。但由于该方法对每台机器均详细求解并重新优化，需要耗费大量的计算时间而调度值没有改进。因此本文针对不均衡 Job shop 调度问题提出了一种有效的分解方法，该方法利用生产线瓶颈特性和大系统分解思想，希望在较快的时间内获得好的调度结果。

4.3 单瓶颈 Job shop 调度问题瓶颈分解方法

4.3.1 瓶颈辨识

不同于 Flow shop 调度问题，所有工件的入口和出口只有一个。Job shop 问题中工件的入口可以不同，它们按不同的工艺路径在机器上加工完后从不同的出口产出。因此 Job shop 的调度性能不仅取决于机器的加工能力，而且取决于工件的工艺路径。

尽管经典 Job shop 问题中每个工件的工艺路径不同，但每个工件都必须在每台机器上加工一次。因此加工负荷越大的机器由于能力有限无法及时处理到达的工件(可能来自不同的机器)，造成该机器前大量工件的堆积而阻塞整个生产线。一般而言生产线上加工负荷最大的机器最可能成为该生产线的瓶颈。我们选择加工负荷最大的机器为瓶颈机。

首先计算每台机器总的加工负荷为该机器上所有工序加工时间之和，即

$$WL_k = \sum_{i=1}^{n} p_{i,k} \quad k = 1,\cdots,m; \tag{4-2}$$

选择加工负荷最大的机器 M_b 为瓶颈机，即

$$b = \arg\max_{k=1,\cdots,m} WL_k \tag{4-3}$$

4.3.2 单瓶颈 Job shop 问题模型简化与分析

令机器 M_b 为瓶颈机，Job shop 中其它机器为非瓶颈机，则 Job shop 中的机器可分为两类：瓶颈机和非瓶颈机。在 Flow shop 生产线中，每个瓶颈机上工序的所有前道工序都是在瓶颈机的上游机上加工，而每个瓶颈机上工序的所有后道工序都是在瓶颈机的下游机上加工。与 Flow shop 生产线不同，Job shop 生产线中每个工件的工艺路径不同，瓶颈机上不同工序的前道工序加工所在机器可能不同，而且瓶颈机上某一道工序的前道工序与瓶颈机上另一道工序的后道工序可能在同一台机器上加工。因此

工件流经非瓶颈机上的时间估计必须具体到每道工序。

我们称瓶颈机上的每道工序为瓶颈工序，在每道瓶颈工序开始之前所需要完成的所有工序记为瓶颈工序的前序，而瓶颈工序加工完后到该工序对应工件加工完之间所需要加工的所有工序记为瓶颈工序的后序。每个瓶颈工序O_{ib}的所有前序加工所需时间$r_{i,b}$和所有后序加工所需时间$q_{i,b}$分别为：

$$r_{i,b} = \sum_{j=1}^{\sigma_i(b)-1}(p_{i,s_i(j)} + w_{i,s_i(j)}) \quad i = 1,\cdots,n; \tag{4-4a}$$

$$q_{i,b} = \sum_{j=\sigma_i(b)+1}^{m}(p_{i,s_i(j)} + w_{i,s_i(j)}) \quad i = 1,\cdots,n; \tag{4-4b}$$

其中$\sigma_i(k)$表示工件J_i在机器M_k上加工工序标号，$w_{i,k}$表示工件J_i在机器M_k上的等待时间，它取决于机器的调度安排。

整个 Job shop 调度问题可以简化为一个带到达时间和传递时间约束的单机调度子问题$1|r_i,q_i|f(C_i)$。如图 4-1 所示，一个工件J_i至少需要经过$r_{i,b}$个时间单元才能到达瓶颈机，在瓶颈机上加工$p_{i,b}$个时间单元之后至少需要经过$q_{i,b}$个时间单元才能完成工件J_i的全部加工。

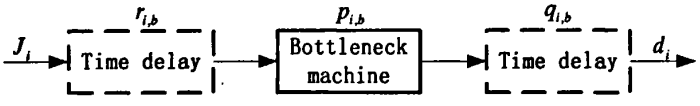


图 4-1. Job shop 调度问题简化模型框图

Fig 4-1. The reduced flow chart of the job shop with a bottleneck machine

初始时，非瓶颈机上的加工安排未知，因此需要在给定某些假设的条件下来估计$r_{i,b}$和$q_{i,b}$。由(4-3)可知非瓶颈机的加工能力大于瓶颈机的加工能力。假设非瓶颈机的加工能力无限大，工件通过非瓶颈机时无需等待即可在此机器上进行加工，即非瓶颈机上工件的等待时间为 0。在此假设条件下，每个瓶颈工序所有前序和后序所需时间$r_{i,b}$和$q_{i,b}$可按下式计算：

$$r_{i,b} = \sum_{j=1}^{\sigma_i(b)-1} p_{i,s_i(j)} \quad i = 1,\cdots,n; \tag{4-5a}$$

$$q_{i,b} = \sum_{j=\sigma_i(b)+1}^{m} p_{i,s_i(j)} \quad i = 1,\cdots,n; \tag{4-5b}$$

由上式可知，$r_{i,b}$和$q_{i,b}$分别等于工件J_i对应瓶颈工序O_{ib}的所有前序加工时间之和与所有后序加工时间之和。

松弛问题 P 中非瓶颈的机器能力约束，得到瓶颈机调度子问题。该问题可描述为一个带到达时间和传递时间约束，目标函数为最小化工件完工时间正则指标 $f(C_i)$ 的单机调度问题 $1|r_i,q_i|f(C_i)$。瓶颈机子问题的目标为主导目标，与原 Job shop 问题的优化目标一致。整个 Job shop 调度问题可以简化为一个单机问题求解，数学规划模型 P_b 描述如下：

$$(P_b) \qquad \min_{x_{ijb},t_{i,b}} f(C_i) \tag{4-6a}$$

$$\text{s.t.} \qquad t_{i,b} \geq r_{i,b} \quad i=1,...,n; \tag{4-6b}$$

$$C_i \geq t_{i,b} + p_{i,b} + q_{i,b} \quad i=1,...,n; \tag{4-6c}$$

$$t_{j,b} - t_{i,b} + \mathrm{H}(1-x_{ijb}) \geq p_{i,b} \quad i=1,\cdots,n-1; j=i+1,\cdots,n; \tag{4-6d}$$

$$t_{i,b} - t_{j,b} + \mathrm{H}x_{ijb} \geq p_{j,b} \quad i=1,\cdots,n-1; j=i+1,\cdots,n; \tag{4-6e}$$

$$x_{ijb} \in \{0,1\} \quad i,j=1,...,n; i \neq j \tag{4-6f}$$

其中决策变量 $t_{i,b}$ 为工件 J_i 在机器 M_b 的开工时间；x_{ijb} 为二进制变量，表示机器 M_b 上工件 J_i 和 J_j 的加工顺序。若机器 M_b 上工件 J_i 在工件 J_j 之前加工，则 $x_{ijb}=1$；否则 $x_{ijb}=0$。$p_{i,b}$ 为工件 J_i 在机器 M_b 上的加工时间，d_i 为工件 J_i 的交货期限，它们是已知且固定的。$r_{i,b}$ 和 $q_{i,b}$ 分别表示工件 J_i 在瓶颈机 M_b 上最早可能的开工时间(或到达时间)和最大可能的剩余加工时间(或传递时间)，它们分别按公式(4-5a)和(4-5b)计算。目标函数(4-6a)与原问题 P 中目标函数(4-1a)一致。约束(4-6b)表示工件 J_i 至少需要等待 $r_{i,b}$ 个单位的时间迟延才能到达瓶颈机 M_b；约束(4-6c)表示工件 J_i 在瓶颈机上加工完之后至少需要等待 $q_{i,b}$ 个单位的时间迟延才能完成该工件所有的工序。每个瓶颈工序 $\sigma_i(b)$ 的到达时间 $r_{i,b}$ 和传递时间 $q_{i,b}$ 反映了瓶颈机与非瓶颈机之间的相互关联。约束(4-6d)和(4-6e)是瓶颈机能力约束，它保证同一时刻机器最多加工一个工件。

在单机调度模型 P_b 中，变量和约束的个数均为 n^2，计算复杂度显著下降。

性质 4-1　令 z 和 z_b 分别为问题 P 和 P_b 的最优解。如果每个工件在非瓶颈机上的加工时间为 0，则 $z_b \equiv z$。

证明：如果非瓶颈机上每个工件的加工时间为 0，问题 P 等价于问题 P_b，则 $z_b \equiv z$。

性质 4-1 说明了瓶颈机决定了整个 Job shop 的性能。当非瓶颈机加工能力无限大时，非瓶颈机调度对系统性能的影响可以被忽略。瓶颈机调度子问题的目标为主导目标，与原问题保持一致。

定理 4-1 令 z 和 z_b 分别为问题 P 和 P_b 的最优解，如果瓶颈机上每个工件的到达时间和传递时间按(4-5a)和(4-5b)计算，则有 $z_b \le z$。

证明：令 S 和 S_b 分别表示问题 P 和 P_b 的可行解的集合。通过松弛(4-1d)和(4-1e)中非瓶颈机加工能力约束，得到瓶颈机上工件的到达时间和传递时间分别等于公式(4-5a)和(4-5b)的右项。因此问题 P 的可行集合 S 属于问题 P_b 的可行集合 S_b，即 $S \subseteq S_b$。由于问题 P 和 P_b 的目标函数相同，且为最小化问题，则有 $z_b \le z$ 成立。

定理 4-1 为原问题最优解的下界提供了一种计算方法。

4.3.3 瓶颈机与非瓶颈机的调度

如前所述，瓶颈机上工件的调度安排可以通过优化求解简化问题 P_b 获得。令瓶颈机 M_b 上工件的排序为 $\pi_b(1),\pi_b(2),\cdots,\pi_b(n)$，其中 $\pi_b(i)$ 表示瓶颈机 M_b 上第 i 个加工的工件。$t^*_{\pi_b(i),b}$ 表示瓶颈机 M_b 上工件 $\pi_b(i)$ 的开工时间，则有：

$$t_{i,s_i(\sigma_i(b)-1)} + p_{i,s_i(\sigma_i(b)-1)} \le t^*_{i,b} \quad i=1,\cdots,n; \tag{4-7a}$$

$$t_{i,s_i(\sigma_i(b)+1)} \le t^*_{i,b} + p^*_{i,b} \quad i=1,\cdots,n; \tag{4-7b}$$

为了保证分解后子问题的调度结果在时间轴上不冲突，瓶颈机与非瓶颈机上工序之间必须满足上述条件。由公式(4-7a)可知，瓶颈机上工序的开工时间决定了其前道工序(在非瓶颈机上加工)最迟必须的完工时间；同理，由公式(4-7b)可知，瓶颈机上工序的完工时间决定了其后道工序最早可能的开工时间。瓶颈机的调度结果设置了非瓶颈机上工序的到达时间和期望的完工时间(交货期限)，我们希望每个工序尽可能在设定的工序交货期限之前完工。非瓶颈机的调度问题可以描述为一个带到达时间和交货期限约束的单机调度问题 $1|r_i|L_{\max}$，非瓶颈工序的到达时间和交货期限设置如下：

$$r_{i,s_i(k)} = \begin{cases} 0 & k=1;i=1,\cdots,n \\ \max\left\{\sum_{j=1}^{\sigma_i(k)} p_{i,s_i(j)}, C_{i,s_i(k)}\right\} & k=2,\cdots,m;i=1,\cdots,n \end{cases} \tag{4-8a}$$

$$d_{i,s_i(k)} = \begin{cases} d_{i,s_i(k+1)} - p_{i,s_i(k+1)} & k=\sigma_i(b)-2,\cdots,1;i=1,\cdots,n \\ t_{i,b} & k=\sigma_i(b)-1;i=1,\cdots,n \\ d_{i,s_i(k+1)} - p_{i,s_i(k+1)} & k=m-1,\cdots,\sigma_i(b)+1;i=1,\cdots,n \\ C_{i,b} + q_{i,b} & k=m;i=1,\cdots,n \end{cases} \tag{4-8b}$$

其中 $r_{i,s_i(k)}$ 和 $d_{i,s_i(k)}$ 分别表示工件 J_i 在机器 M_k 上的工序 $O_{i,k}$ 的到达时间和交货期限。

每个非瓶颈机调度问题$1|r_i|L_{\max}$可以通过与交货期限相关的分派规则有效求解，如改进工序交货期限最小最先加工规则(MOD)，近似拖期费用最大最先加工规则(ATC)，剩余加工时间最大最先加工规则(MWR)和关键比最大最先加工规则(CR)等[39]。本文采用基于 MOD 规则的启发式算法，具体步骤如下：

算法 4-1 基于 MOD 规则的启发式调度算法 (Heuristic based on MOD rule)

令$N_k=\{1,\cdots,n\}$为机器$k\in M$上所有工件的集合。U_k为机器M_k上已调度的工件集，$\bar{U}_k$为机器M_k未调度的工件集，$\bar{U}_k=N_k\setminus\{U_k\}$。

Step 1： 令$t=\min_{i\in N_k} r_{i,k}$；$U_k=\Phi$，$\bar{U}_k=N_k$；

Step 2： 在t时刻，对未调度工件集合$\bar{U}_k$中已到达的工件 i(满足$r_{i,k}\le t$)计算工序的改进交货期限$d_{i,k}^{\text{mod}}=\max\{t+p_{i,k},d_{i,k}\}$，选择改进工序交货期限最小的工序 j 调度，其中$j=\arg\min_{r_{ik}\le t \text{ and } i\in\bar{U}_k} d_{i,k}^{\text{mod}}$。

Step 3： 更新$U_k:=U_k\cup\{j\}$，$t:=\max\{t+p_{j,k},\min_{i\in\bar{U}_k} r_{i,k}\}$。如果所有工件均调度完，算法停止；否则转 Step 2。

该算法的计算复杂度为$O(n\log n)$。

4.3.4 瓶颈机与非瓶颈机之间的协调

如果非瓶颈机上的某一工序不能在期望的完工时间内完成，则会造成非瓶颈工序和瓶颈工序开工时间在时间轴上迭代，使得调度不可行。因此需要调整关联参数来协调瓶颈机与非瓶颈机的调度结果。通过调整瓶颈工序的到达时间和传递时间，对瓶颈机重新调度。随后根据瓶颈机上新的调度结果重新设定非瓶颈工序的到达时间和交货期限，非瓶颈机的调度随之调整。

Job shop 调度问题单瓶颈分解方法基于三层递阶模型：协调层、瓶颈机调度层和非瓶颈机调度层(见图 4-2)。协调层的目标在于根据调度层(瓶颈机调度层和非瓶颈机调度层)的已知信息来估计每个瓶颈工序的到达时间$r_{i,b}$和传递时间$q_{i,b}$(关联项)，并将估计后的关联项参数传递至瓶颈机调度层。瓶颈机调度层的任务是根据估计的参数值($r_{i,b}$和$q_{i,b}$)优化求解瓶颈机子问题$1|r_i,q_i|f(C_i)$，获得瓶颈机上工件的开工时间$t_{i,b}$和完工时间$C_{i,b}$。由于瓶颈机上工件的开工时间和完工时间决定了非瓶颈机上工件的最早可能的开工时间(到达时间)和最迟必须的完工时间(交货期限)，将瓶颈机的调度结果($t_{i,b}$和$C_{i,b}$)传递至非瓶颈机调度层，依次调度非瓶颈机。第一个决策过程是基于协

调层和调度层(瓶颈机调度层和非瓶颈机调度层)之间的垂直信息流；第二个决策过程是基于瓶颈机调度层和非瓶颈机调度层之间的垂直信息流(见图 4-2 中虚线)。

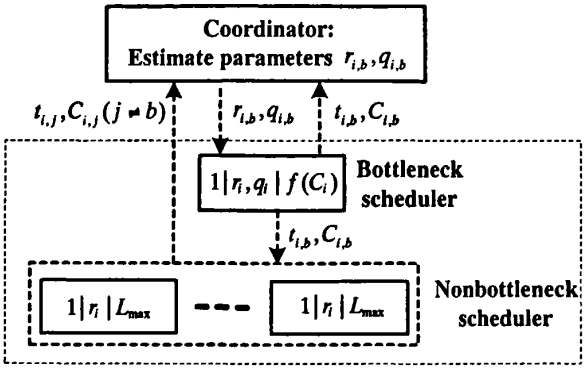


图 4-2. Job shop 瓶颈分解方法三层递阶结构图

Fig 4-2. The three-level architecture of the BDP for job shop problem

关联参数初始估计是在假设非瓶颈机加工能力无限大的情况下，由公式(4-5a)和(4-5b)计算而得。然而实际 Job shop 生产线中非瓶颈机的加工能力并不是无限大，因此非瓶颈机上的工序在到达后不一定能立即开始加工。由于非瓶颈机的能力限制，可能存在一个瓶颈工序其前面某一道工序无法在瓶颈调度设定的完工时间内完工，则该工序所属工件对应的瓶颈工序会与其前道工序的开工时间在时间轴上发生迭代，调度不可行。为保证整个调度的可行，在此我们采用一种简单但有效的调整策略，即调整每个冲突瓶颈工序的到达时间等于其前道工序的完工时间，相当于对冲突瓶颈工序在时间轴上右移一定的量。下面给出具体的调整过程。

如果瓶颈工序$O_{i,b}$的前一道工序$O_{i,s_i(\sigma_i(b)-1)}$的完工时间$C^{\bullet}_{i,s_i(\sigma_i(b)-1)}$大于瓶颈工序的开工时间$t^{\bullet}_{i,b}$，则调整瓶颈工序$O_{i,b}$的到达时间等于工序$O_{i,s_i(\sigma_i(b)-1)}$的完工时间。

为了保证迭代过程的收敛速度，每重新调度一次瓶颈机固定一个瓶颈工序的到达时间，则瓶颈工序到达时间参数修正的次数(迭代次数)不超过瓶颈机上工序的个数n。令Ω表示已固定到达时间的工件集合，瓶颈机上每个工件的到达时间更新如下：

$$r_{i,b} = \begin{cases} \max\left\{ \sum_{j=1}^{\sigma_i(b)-1} p_{i,s_i(j)}, C^{\bullet}_{i,s_i(\sigma_i(b)-1)} \right\} & if\ i \notin \Omega\ and\ C^{\bullet}_{i,s_i(\sigma_i(b)-1)} > t^{\bullet}_{i,b} \\ r_{i,b} & otherwise \end{cases} \tag{4-9a}$$

如果瓶颈调度可行，则可以通过调整每个瓶颈工序的传递时间来改进整个调度的

性能。提高延期最严重的工件的传递时间，使得该工件在瓶颈机上加工工序提早安排，该瓶颈工序的传递时间修正如下：

$$q_{i,b} = \max\left\{q_{i,b} + (C_i^* - d_i), \sum_{j=\sigma_i(b)+1}^{m} p_{i,s_i(j)}\right\} \quad i = \arg\max_{i \in N}(C_i^* - d_i) \tag{4-9b}$$

为了保证改进过程的收敛性，如果调度没有改进则停止迭代，最好调度解选为最终调度。给出 Job shop 调度问题单瓶颈分解算法如下：

算法 4-2. 单瓶颈分解算法(Bottleneck-based Decomposition Procedure, BDP)

Step 1：辨识瓶颈机。选择加工负荷最严重的机器为瓶颈机 M_b，令瓶颈机到达时间已固定的工件集合 $\Omega = \Phi$。

Step 2：估计初始关联项。按公式(4-5a)和(4-5b)设置瓶颈机上工序的初始到达时间和传递时间。

Step 3：调度瓶颈机。建立瓶颈机调度模型 P_b，优化求解该问题得到瓶颈机上的调度安排。令 $\Omega = \Omega \cup \{i^*\}$，其中 $i^* = \arg\max_{i \in \Omega}\{t_{i,b}^*\}$。

Step 4：调度非瓶颈机。按公式(4-8a)和(4-8b)分别设置非瓶颈机上工序的到达时间和交货期限。然后采用 MOD 启发式规则调度非瓶颈机。

Step 5：检测可行性。如果调度不可行，则按公式(4-9a)更新瓶颈机上工序的到达时间，转 Step 3；否则转 Step 6。

Step 6：迭代停止准则。如果解没有改进或 $|\Omega| = n$，停止迭代。否则按公式(4-9b)更新瓶颈机上工序的传递时间，转 Step 3。

4.4 仿真测试与分析

1. 仿真测试平台

为了测试 Job shop 调度问题单瓶颈分解算法的性能，按文献[108]的方式随机生成大量测试数据，实验参数设计如表 4-1 所示。所有工件均在零时刻到达，非瓶颈机上每个工件的加工时间服从[1,50]均匀分布，瓶颈机上每个工件的加工时间服从 $[1+l, 50\cdot(1+l)]$ 均匀分布，机器之间负荷差异 l 越大，瓶颈程度越高。工件交货期限由参数 R 和 T 决定，其中 R 为交货期范围，T 为交货期滞后程度，它表示滞后工件个数的百分比。工件的交货期限服从 $[P'(1-T-R/2), P'(1-T+R/2)]$ 均匀分布，其中 P' 是最大完工时间的下界。

表 4-1. 测试数据生成参数表

Table 4-1. Experimental design for randomly generated test problem

Problem parameter	Value	Total Value
Number of machines m	10, 20, 30	3
Number of jobs n	30, 50, 100, 200	4
Percentage tardy jobs T	0.3, 0.6	2
Due date range R	0.5, 2.5	2
Workload difference l	0.50(low), 1.0(medium), 2.0(high)	3

将本文提出的单瓶颈分解算法(BDP)与移动瓶颈算法和 TOC 算法进行比较。在移动瓶颈方法中，所有的子问题均采用 Carlier 算法精确求解[117]。按 TML 规则依次调度每台机器。每调度完一次机器，已调度的机器则按滞后时间从大到小的顺序重新优化。考虑两种不同的重新优化过程，即无重新优化过程和完全重新优化过程，分别记为 SB(NR)和 SB(FR)。TOC 算法具体参见文献[108]。

移动瓶颈方法最初针对 $J_m \| C_{\max}$ 问题提出，随后应用于 $J_m \| L_{\max}$ 问题。大量的仿真结果显示移动瓶颈方法对这两类问题获得的调度质量非常好。为了更好地评价本文提出的单瓶颈分解算法的性能，我们将分别对 $J_m \| C_{\max}$ 和 $J_m \| L_{\max}$ 问题进行研究。

我们采用目标值与下界的相对比值 η 来评价算法的调度质量[95]，令 $f(H,I)$ 为给定实例 I 下采用启发式算法 H 求得的目标函数 $f(C_i)$ 值，$LB(I)$ 为问题 I 的下界，有：

$$\eta(H,S) = \sum_{I \in S} f(H,I) / \sum_{I \in S} LB(I) \tag{4-10}$$

其中 $LB(I)$ 是 BDP 算法第一次迭代求解简化问题 P_b 获得的最优值(参见定理 4-1)。计算时间的评价指标采用平均 CPU 计算时间 $ACT(H,S)$。所有的算法均采用 C 语言编程，运行在 Celeron 处理器(CPU 速度 2GHZ，缓存 128kB，内存 256MB)的机器上。

2. $J_m \| C_{\max}$ 问题仿真测试

首先研究 $J_m \| C_{\max}$ 问题的单瓶颈分解算法的性能。此问题目标函数为最大完工时间(makespan)，即 $f(C_i) = \max_{i \in N} C_i$，它可以衡量机器的利用程度。考虑 10 种不同问题规模，3 种不同的负荷差异程度(0.5、1.0、2.0)。所有参数组合生成 30 种类型的问题，每种类型分别随机生成 10 组数据，共 300 组测试数据。

问题 $J_m \| C_{\max}$ 对应的子问题为 $1|r_i,q_i|C_{\max}$。虽然该问题计算复杂度为 NP-complete[3]，但 Carlier (1982) 提供了一个有效的分枝定界算法精确求解该问题[117]。Carlier 算法可求解最大规模为 1,000 个工件的问题[94]。本文提出的单瓶颈分解

算法(BDP)中，瓶颈机子问题采用 Carlier 算法精确求解，非瓶颈机采用基于 MOD 规则的启发式算法(算法 4-1)。由于该问题中不考虑工件的交货期限，我们设瓶颈工序传递时间修正公式(4-9b)中的工件交货期限为该瓶颈工序所属工件最后一道工序的交货期限$d_{i,s_i(m)}$，即根据最后一道工序的完工时间和交货期限信息来调整瓶颈工序的传递时间。

对随机生成的 300 组数据进行仿真测试。表 4-2 显示了调度问题$J_m \| C_{\max}$的调度值与下界比值平均值η和平均计算时间(ACT)，所有算法的平均计算时间以秒为单位。

表 4-2. 不同问题规模下$J_m \| C_{\max}$问题算法性能比较

Table 4-2. Performance of algorithms for problem $J_m \| C_{\max}$ with different problem sizes

Problem S	SB (NR)		SB (FR)		BDP		TOC
(m, n)	η	ACT(s)	η	ACT(s)	η	ACT(s)	η
(10, 30)	1.010	5.7	1.009	49	1.040	0.04	1.079
(10, 50)	1.006	16	1.003	209	1.035	0.13	1.066
(20, 30)	1.137	40.5	1.121	1367	1.148	0.2	1.204
(20, 50)	1.067	191	1.050	5742	1.087	0.9	1.128
(30, 30)	1.176	205	1.157	8516	1.181	0.4	1.223
(30, 50)	1.130	887	1.108	39881	1.142	1.5	1.194
(20, 100)	N/A	N/A	N/A	N/A	1.050	3.6	1.077
(20, 200)	N/A	N/A	N/A	N/A	1.031	7.5	1.057
(30, 100)	N/A	N/A	N/A	N/A	1.080	4.8	1.103
(30, 200)	N/A	N/A	N/A	N/A	1.045	10.5	1.065

由表 4-2 可知，带完全重新优化过程的移动瓶颈算法(SB(FR))的调度质量最好，它比单瓶颈分解算法(BDP)平均好 3.1%。但它的计算时间比较长，对规模为 30 台机器 50 个工件的问题，完全重新优化的移动瓶颈算法需要 11 个小时，这在实际调度中是不可接受的。而且由于机器内存限制，移动瓶颈算法无法求解大规模问题，它最大可求解问题的规模为 30 台机器 50 个工件。

TOC 算法的计算时间非常快，只需要几毫秒(在表 4-2 中略去)，但它的性能相对较差，它比算法 BDP 平均差 4.3%。单瓶颈分解算法可以在求解质量和计算时间之间获得一个好的均衡。该算法的调度值(上界)和下界之差平均在 10%范围内，而且它求解规模为 30 台机器 200 个工件的大规模问题只需要 10.5 秒。单瓶颈分解算法可在较快的时间内获得较好的调度解，它可应用于大规模$J_m \| C_{\max}$调度问题的求解。

3. $J_m \| L_{\max}$问题仿真测试

接着研究$J_m \| L_{\max}$问题单瓶颈分解算法的性能。此问题目标函数为最大滞后时

间，即 $f(C_i)=\max_{i\in N}(C_i-d_i)$，它可以衡量客户不满意度最差的程度。考虑 10 种不同的问题规模，3 种不同的负荷差异程度(0.5、1.0、2.0)，2 种不同的交货期范围(0.5 和 2.5)以及 2 种不同的滞后程度(0.3 和 0.6)。所有参数组合生成 120 种类型的问题，每种类型分别随机生成 10 组数据，共 1200 组测试数据。

问题 $J_m\|L_{\max}$ 对应的子问题为 $1|r_i,q_i|L_{\max}$。虽然该单机问题计算复杂度为 NP-complete[3]，但 Carlier 算法可以在较快的时间内获得最优解[117]。本文提出的单瓶颈分解算法(BDP)中，瓶颈机子问题采用 Carlier 算法精确求解，非瓶颈机采用基于 MOD 规则的启发式算法(算法 4-1)。

对随机生成的 1200 组数据进行仿真测试。表 4-3 显示了调度问题 $J_m\|L_{\max}$ 的调度值与下界比值平均值 η 和平均计算时间(ACT)。所有算法的平均计算时间(ACT)以秒为单位。表 4-4 显示了调度问题 $J_m\|L_{\max}$ 不同参数下调度值与下界比值的平均值 η。

表 4-3. 不同问题规模下 $J_m\|L_{\max}$ 问题算法性能比较

Table 4-3. Performance of algorithms for problem $J_m\|L_{\max}$ with different problem sizes

Problem S	SB (NR)		SB (FR)		BDP		TOC
(m,n)	η	$ACT(s)$	η	ACT(s)	η	ACT(s)	η
(10, 30)	1.081	5.6	1.053	50	1.083	0.06	1.113
(10, 50)	1.061	16	1.049	215	1.059	0.13	1.091
(20, 30)	1.183	42	1.115	1319	1.124	0.18	1.174
(20, 50)	1.134	207	1.084	5832	1.098	0.5	1.110
(30, 30)	1.207	223	1.161	8527	1.172	0.31	1.199
(30, 50)	1.191	921	1.133	39796	1.144	1.0	1.175
(20, 100)	N/A	N/A	N/A	N/A	1.085	2.0	1.091
(20, 200)	N/A	N/A	N/A	N/A	1.074	8.0	1.085
(30, 100)	N/A	N/A	N/A	N/A	1.123	4.5	1.167
(30, 200)	N/A	N/A	N/A	N/A	1.102	12.0	1.151

表 4-4. 不同参数下 $J_m\|L_{\max}$ 问题算法性能比较

Table 4-4. Performance of algorithms for problem $J_m\|L_{\max}$ with different factors

Problem S	SB (NR)	SB (FR)	BDP	TOC
Due date (T,R)				
(0.3, 0.5)	1.281	1.265	1.271	1.405
(0.3, 2.5)	1.141	1.073	1.067	1.059
(0.6, 0.5)	1.146	1.137	1.181	1.236
(0.6, 2.5)	1.154	1.086	1.088	1.114
Workload difference l				
0.5	1.213	1.190	1.222	1.233
1.0	1.124	1.102	1.110	1.139
2.0	1.081	1.051	1.057	1.077

如表 4-3 所示，带完全重新优化过程的移动瓶颈算法(SB(FR))的调度质量最好，它比单瓶颈分解算法(算法 BDP)平均好 1.5%，但其计算时间非常长。对规模为 30 台机器 50 个工件的问题，完全重新优化的移动瓶颈算法需要 11 个小时，这在实际情况中是无法接受的。而且，由于计算机内存限制使得移动瓶颈方法无法求解更大规模的问题，它最大可求解问题的规模为 30 台机器 50 个工件。

TOC 算法的计算时间非常快，只需要几毫秒(在表 4-3 中略去)，但它的性能相对较差，它比算法 BDP 平均差 3.1%。单瓶颈分解算法可以在求解质量和计算时间之间获得一个好的均衡，它求解规模为 30 台机器 200 个工件的大规模问题只需要 12 秒。因此单瓶颈分解算法是一种非常有效的启发式算法，它可以应用于大规模 $J_m \| L_{\max}$ 问题的求解。

为了分析不同参数对单瓶颈分解算法的影响，对表 4-3 和 4-4 进行具体分析。随着机器个数的增加，单瓶颈分解算法性能下降。当机器个数从 10 增加至 20 时，单瓶颈分解算法的调度值与下界相对值 η 平均增加了 4%。原因可能是瓶颈机的主导作用随着机器个数的增加而减弱。但是单瓶颈分解算法的调度性能随着工件个数的增加而提高。当工件个数从 30 增至 50 时，单瓶颈分解算法的调度值与下界相对值 η 平均减少了 2.6%。原因可能是随着工件个数的增加，Job shop 中的瓶颈机更为稳定。

从表 4-4 可以看出，机器负荷差异 l 是一个影响瓶颈分解算法的一个重要参数。随着机器负荷差异 l 的增加，单瓶颈分解算法的性能改进。当瓶颈程度 l 从 0.5 升至 1.0，算法 BDP 平均改进 11.2%。瓶颈程度越高，单瓶颈分解算法的调度性能越好。

同时交货期分布也影响单瓶颈分解算法的性能。交货期范围 R 越小，单瓶颈分解算法性能越差。当工件滞后程度 T 为 0.3，而交货期范围 R 从 0.5 增至 2.5 时，算法 BDP 性能平均下降 20.4%；当工件滞后程度 T 为 0.6，而交货期范围 R 从 0.5 增至 2.5 时，算法 BDP 性能平均下降 9.3%。

针对大规模 Job shop 问题，单瓶颈分解算法可以在较快的时间内获得一个好的调度。而且瓶颈程度越高，单瓶颈分解算法性能越好。单瓶颈分解算法由于其高效的计算效率可适用于实时调度环境。

4.5 本章小结

本文以大规模能力不均衡 Job shop 调度问题为背景，针对该生产线存在一个主导瓶颈的特点提出了一种单瓶颈分解方法。该方法利用不均衡 Job shop 生产线机器能力差异，将 Job shop 上的机器分为瓶颈机和非瓶颈机。首先假设非瓶颈机加工能力无限大的前提下，将原问题简化为一单机调度问题，通过求解该简化问题可以获得瓶颈机

上工件的调度安排。瓶颈机的调度结果决定了非瓶颈机工序最早可能的开工时间和最迟必须的完工时间。而非瓶颈机随后采用基于 MOD 规则的启发式算法围绕瓶颈机而调度。通过调整瓶颈机上工序的到达时间和传递时间协调瓶颈机与非瓶颈机之间的相互关联，从而保证整个 Job shop 调度的可行。仿真结果显示，本文提出的单瓶颈分解算法可以有效求解大规模不均衡 Job shop 调度问题。

第五章 多约束机 Job shop 调度问题瓶颈分解方法研究

5.1 引言

前两章我们研究了生产线只存在一个瓶颈机的情况，对大规模单瓶颈 Flow shop 调度问题和单瓶颈 Job shop 调度问题分别提出了有效的瓶颈分解算法。该方法的主要思想：生产线只存在一个瓶颈机，瓶颈决定了整个生产调度问题的性能，如果完全松弛其它机器来求解该问题，则可以很容易的获得原问题的近似解和原问题中瓶颈机的近似优化排序。

然而实际生产线上，即使非瓶颈机总的加工能力足以完成其加工负荷，但是由于时间问题，它仍可能制约系统的最终产出。一个生产线上的机器受限主要来自两方面：一个是加工负荷超过其可用加工能力的机器，该类机器我们称为瓶颈机。另一种情况是工件不能在足够早的时间内到达某些非瓶颈机，使得该机器不能在规划的完工时间内完成它的操作。我们称这类机器为暂时瓶颈或移动瓶颈。TOC 理论定义能力受限资源(capacity constraint resource, CCR)为制约系统产能的任何资源，无论它是否为瓶颈机[110]。在 Job shop 调度问题中，能力受限资源多为加工机器。我们将这两类机器统称为“约束机”，它包括长期瓶颈和暂时瓶颈，可以看作一种广义瓶颈。本章主要研究多约束机 Job shop 调度问题(Job-shop Scheduling Problem with Multiple Constraint Machines, JSPMC)。

由 TOC 理论可知，约束机制约系统的最终产出，有效地辨识和利用这些机器可以改善整个系统的性能。TOC 五步集中过程和鼓-缓存-绳子(drum-buffer-rope, DBR)方法处理生产调度问题主要是使生产线的加工速率与瓶颈机保持同步[100-102]。Glassey 和 Resende(1988)提出一种避免瓶颈“饥饿”(starvation avoidance, SA)的投放策略来保证瓶颈机能力的充分利用[119]。在这些方法中，假设只存在一个瓶颈机，而非瓶颈机有足够的冗余能力来支持瓶颈机的调度安排。

对生产线存在多个约束机的情况，Simons 等(1996)对 DBR 约束调度问题构造了一个数学模型[104]，随后提出了一个目标系统法来求解该问题[105]。该算法称为 GS1.0，在该算法中约束机顺序优化求解。Simons 等(1999)改进目标系统方法，在改进算法 GS2.2 中，约束机并行优化求解[106]。GS 算法提供了一种辨识约束机的方法，但该过程过于粗略。

移动瓶颈(SB)方法是处理约束调度问题的另一种有效的方法。它将一个 Job shop 调度问题分解为若干更易求解的单机子问题，通过顺序求解各子问题获得机器上工件

的加工安排[91]。Uzsoy 和 Wang(2000)详细分析了 SB 和 GS 方法的异同，并应用 SB 求解不均衡 Job shop 调度问题。尽管 SB 方法能很好地解决带瓶颈机的 Job shop 调度问题，但是它对每台机器精确求解并重新优化，需要消耗一定的计算代价但调度质量改善不大[108]。

在多约束机的 Job shop 调度问题中，非约束机往往有足够的冗余的能力完成其所需的工序而不影响其它机器的调度，因此没有必要对所有的机器都精确调度。对约束机和非约束机采用不同程度的调度策略不但可以保持较好的调度性能，而且还可以大大提高计算效率。

本章的主要内容安排如下：5.2 节分析多约束机 Job shop 问题特点，对该问题进行简化。5.3 节提出一种新的迭代辨识瓶颈的方法，并给出一个具体的瓶颈分解算法。在 5.4 节中进行了大量的仿真测试并给出详细的结果分析。

5.2 Job shop 调度问题描述与模型简化

5.2.1 Job shop 调度问题描述

目标函数为最小化最大滞后时间($L_{\max}$)的 Job shop 调度问题 $J_m \| L_{\max}$ 描述如下：n 个待加工工件按给定的工艺路径依次在 m 台机器上加工，每个工件在每台机器上加工一次。工件在机器上的加工称为一个操作或工序，每个操作一旦开始不允许中断。我们希望确定每台机器上每道工序的加工顺序和开工时间，使得在满足以下的约束条件情况下最小化目标函数：(a) 同一时刻每个工件只能在一台机器上加工；(b) 同一时刻每台机器最多只能加工一个工件。

$J_m \| L_{\max}$ 问题可用一个析取图 $G(N,A,E)$ 描述，其中 $N=\{0,1,\cdots,m\cdot n,n_1,\cdots,n_n,*\}$ 为工序集(包括虚拟起点 0，虚拟终点 *，$m\cdot n$ 个工序节点和 n 个虚拟完工节点 n_i)。A 表示满足顺序关系(a)的工序对集合；E_k 表示在机器 k 上加工的工序对集合，如条件(b)所述，它们在时间轴上不允许迭代。Job shop 调度问题的数学规划模型描述如下：

$$(P) \qquad \min \max_{i\in J}(t_{n_i}-d_i) \tag{5-1a}$$

$$st \quad t_j - t_i \geq p_i \qquad (i,j)\in A \tag{5-1b}$$

$$t_i \geq 0 \qquad i\in N \tag{5-1c}$$

$$t_j - t_i \geq p_i \vee t_i - t_j \geq p_j \qquad (i,j)\in E_k, k\in M \tag{5-1d}$$

其中决策变量为 t_j，它表示工序 j 的开工时间。p_j 为工序 j 的加工时间，d_i 为工件 i 的

交货期限，它们是固定已知的。$J=\{1,2,\cdots,n\}$ 为工件集，$M=\{1,2,\cdots,m\}$ 为机器集。目标函数(5-1a)为最大滞后时间，它可以衡量客户不满意最差的程度。约束(5-1b)描述了同一工件上工序之间的顺序关系。约束(5-1c)保证工件必须在到达后开始加工。约束(5-1d)则描述了机器能力约束，它保证同一时刻每台机器最多只能加工一个工件。

图 5-1 给了 Job shop 调度问题的析取图描述 $G(N,A,E)$。在此析取图中，每个节点出弧的长度等于该节点对应工序的加工时间的大小。从每个虚拟完工时间 n_i 到虚拟终点*弧线的长度设为 $K-d_i$，其中 K 为一个常数，$K \geq \max_i\{d_i\}$，则从虚拟起点 0 到虚拟终点*的最长路径大小等于 $L_{max}+K$。因此，确定一个非循环图使从点 0 到*的最长路径大小最小对应找到一个优化调度最小化目标函数 L_{max}。因此 Job shop 调度问题对应确定析取图 $G(N,A,E)$ 上析取弧对的方向。

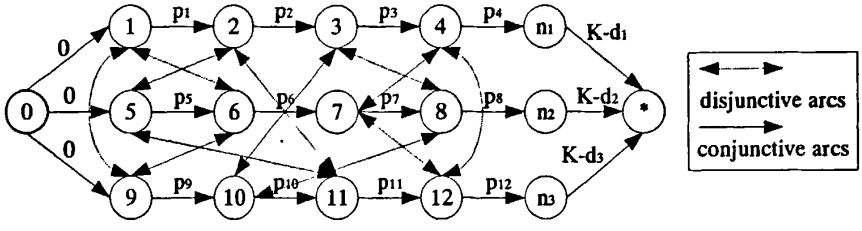


图 5-1. Job shop 调度问题析取图描述(4×3 例子)

Fig 5-1. Disjunctive graph of Job shop scheduling problem for a 4×3 instance

该问题计算复杂度为强 NP-hard[6]。尽管精确算法如分枝定界可以优化求解该问题[33, 120]，但计算时间随问题规模的增大呈指数增长。到目前为止，它可优化求解的最大规模为 10 台机器和 20 个工件的问题[35]。SB 方法是求解 Job shop 问题非常有效的一种启发式方法，但是它的计算时间很长[95]，它只适用于中等规模问题。因此有必要提出一种有效的启发式方法，希望它可在合理的时间内获得一个好的调度结果。

5.2.2 Job shop 调度问题简化模型及分析

由于 SB 算法对每个子问题(对应一台机器)均精确求解并重新优化，因此该算法的计算代价主要来自机器的个数。对此本文利用带多约束机的 Job shop 生产线的机器特性，提出一种问题简化方法。

该方法将 Job shop 中的机器分为两类：约束机和非约束机。约束机制约整个系统的性能，对它们精确调度，保证这些机器的能力得以充分利用。而非约束机有足够的

冗余能力来支持约束机的调度，因此它可以采用有效规则调度。假设非约束机加工能力无限大，则松弛问题 P 中非约束机的机器能力约束。称在约束机上加工的工序为约束工序，而在非约束机上加工的工序为非约束工序。非约束机上的工序的具体加工安排可以简化为一个时间迟延单元，得到 Job shop 调度问题简化模型，具体描述如下：

$$(P_{CCR}) \qquad \min \max_{i \in J} (t_{n_i} - d_i) \tag{5-2a}$$

$$s.t \quad t_j - t_i \geq p_i + TL_{ij} \qquad (i,j) \in A_C \tag{5-2b}$$

$$t_i \geq TL_{0i} \qquad i \in N_C \tag{5-2c}$$

$$t_j - t_i \geq p_i \vee t_i - t_j \geq p_j \qquad (i,j) \in E_k, k \in M_C \tag{5-2d}$$

其中 M_C 表示约束机集合，N_C 表示约束工序集合，A_C 为满足顺序关系的约束工序对集合。TL_{ij} 为约束工序 i 和 j 之间非约束工序加工的时间迟延，TL_{0i} 表示从虚拟起始节点 0 到约束工序 i 之间的时间迟延。

尽管简化问题的计算复杂度仍是强 NP-hard[6]，但由于约束机个数远小于原问题机器的个数，问题的计算规模显著下降。简化问题 P_{CCR} 的优化解可以为非约束工序设定期望的完工时间。当某一非约束机上的工序不能在其期望的完工时间内完工，则问题 P_{CCR} 最优解确定的约束机调度是不可行的，此时简化问题 P_{CCR} 的解不能构成原问题 P 的解。图 5-2 给出了简化问题的析取图描述 $G(N_C, A_C, E_C)$。在此析取图中，只描述约束工序，而非约束工序的节点完全删除。而且同一工件相邻约束工序节点的长度等于该约束工序的加工时间加上它与相邻约束工序之间的非约束工序加工所需的时间迟延。

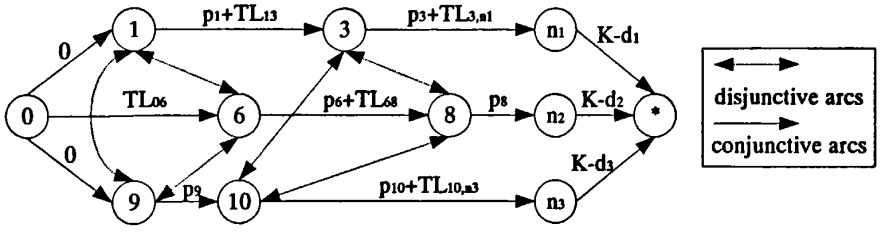


图 5-2. Job shop 调度问题简化模型析取图描述

Fig 5-2. Disjunctive graph of the reduced Job shop problem

在问题 P_{CCR} 中，时间迟延 TL_{ij} 表示同一工件相邻约束工序 i 和 j 之间非约束工序所需的加工时间。它保证在约束工序 i 之后加工的非约束工序有充足的时间在下一个约束工序 j 开始之前完成其加工操作。在经典 Job shop 问题中，由于每个工件必须在每

台机器上加工一次，同一工件的不同约束工序分别在不同的约束机上加工。此时在约束工序之间的加工顺序约束条件中考虑时间迟延，它保证约束工序i必须在约束工序j开始前TL_{ij}个时间单元完工。

时间迟延的大小取决于非约束机的加工安排。初始时，非约束机的调度安排未确定，因此时间迟延的大小也未知，需要对其进行估计。为了保证问题P_{CCR}的解对原问题仍有意义，同一工件相邻约束工序必须放置地足够远，这样可以保证有足够的时间完成它们之间的非约束工序。具体来说，时间迟延TL_{ij}的长度至少要等于约束工序i和j之间的非约束工序的加工时间之和。然而时间迟延长度过大会推迟工件的完工，使得最终性能变差。因此需要合理地估计时间迟延的大小来获得好的调度性能。

目前一些研究者提出了不少估计时间迟延大小的方法[121,122]。本文基于析取图模型，采用最长路径算法来估计时间迟延的大小。时间迟延T_{ij}表示从约束工序节点i到约束工序节点j的最长路径的长度。令$L(p,q)$表示析取图上节点p到节点q的最长路径长度，则时间迟延大小设置如下：

$$TL_{ij} = L(i,j) - p_i \qquad (i,j) \in A_C \tag{5-3}$$

假设机器k的加工能力无限大，则从析取图上删除该机器的所有析取弧。初始时，删除析取图上所有的析取弧。一旦约束机的调度已知，根据其调度结果添加约束机调度决策对应的有向弧，更新析取图。本文采用 Label 算法来计算有向图的最长路径。该算法计算复杂度为$O(\alpha)$，其中α为图中弧的个数[40]。

本文提出的约束调度方法的一个显著特点是采用时间迟延来代替非约束工序的具体加工安排。时间迟延不但可以描述经典 Job shop 生产线中同一工件在不同约束机上工序之间的关联，而且还可以描述重入 Job shop 生产线中同一工件在同一台约束机上不同工序之间的关联。时间迟延也可以描述动态过程不确定因素带来的生产波动，因此本文提出的方法也可推广至动态不确定调度问题。

定理 5-1 令z和z_{CCR}分别为问题P和P_{CCR}的最优解，如果相邻约束工序的时间迟延大小按公式(5-3)计算，则有$z_{CCR} \le z$。

证明：令S和S_{CCR}分别表示问题P和P_{CCR}的所有可行解的集合。松弛问题P中非约束机的能力约束得到问题P_{CCR}。在此假设条件下相邻约束工序的时间迟延大小等于式(5-3)的右项。因此问题P的可行集合S属于问题P_{CCR}的可行集合S_{CCR}，即$S \subseteq S_{CCR}$。由于问题P和P_{CCR}的目标函数相同，且为最小化问题，则问题P的最优解不小于问题P_{CCR}的最优解，即$z_{CCR} \le z$。

5.3 多约束机 Job shop 调度问题约束调度算法

如果约束机已知，则可以直接建立 JSPMC 问题的简化模型。约束机上工件的调度安排可以通过优化求解简化模型 P_{CCR} 获得，而非约束机则采用有效的启发式算法围绕约束机而调度。然而初始时约束机通常是未知的。如前所述，约束机的产生有两种情况：一是当机器的加工负荷超过该机器可利用的能力，我们称之为长期瓶颈；另一种情况是由于某机器上工件不能在足够早的时间内到达，使得它在机器上的操作不能在期望的完工时间内加工完，我们称之为暂时瓶颈。因此约束机的辨识不仅取决于机器的可利用能力，而且还取决于调度决策。约束机的辨识不可能一次完成，它需要迭代地进行，该过程嵌入至机器调度问题求解过程。具体细节将在下面章节详细介绍。

5.3.1 约束机辨识

在 SB 方法中，一些指标如机器总加工负荷最大优先(Total Machine Load, TML)，子问题最大解优先(Sub-problem Largest Value, SLV) 等可用于评价机器关键程度[123]，然而它们仅仅给出机器的优先级，它不能用于区分约束机与非约束机。

如前所述，约束机通常定义为制约系统产能的任何机器，无论它是否为瓶颈机或非瓶颈机[110]。它的辨识不仅取决于机器可用能力，还取决于调度决策。目前不存在一种定量的指标来严格界定约束机和非约束机。因此，我们提出一种启发式方法迭代地辨识约束机。

令 M_C^i 表示第 i 次迭代过程已辨识的约束机集合，M_{NC}^i 表示第 i 次迭代过程非约束机集合，其中 $M_{NC}^i = M \setminus M_C^i$。$N_k$ 表示在机器 k 上加工的所有工序的集合。N_C^i 和 N_{NC}^i 分别表示集合 M_C^i 和 M_{NC}^i 上工序的集合。令 P_{CCR}^i 表示第 i 次迭代过程辨识的约束机 M_C^i 对应的简化模型。初始时无约束机，即 $M_C^0 = \varnothing$，$M_{NC}^0 = M$，$N_C^i = \varnothing$，$N_{NC}^i = N$。删除析取图上所有的析取弧，得到一个有向图。

在第 i 次迭代过程，如果已辨识的约束机集合非空，即 $M_C^{i-1} \neq \varnothing$，则对每台约束机机器 $k \in M_C^{i-1}$ 的调度决策可以通过优化求解其对应的简化问题 P_{CCR}^{i-1} 获得，获得约束机上工序的加工顺序 $S_k^i (k \in M_C^{i-1})$。根据约束机的调度结果，固定相应析取弧的方向，更新有向图为 $G(N, A \cup \{S_k^i \mid k \in M_C^{i-1}\}, E \setminus \{E_k \mid k \in M_C^{i-1}\})$。

基于当前的有向图，采用最长路径算法设置非约束工序的最早可能的开工时间(到达时间)和期望的完工时间(交货期限)：

$$r_l^i = L(0,l) \quad l \in N_{NC}^{i-1} \tag{5-4a}$$

$$dd_l^i = \max\left(d_j, L(0,n_j)\right) - L(l,n_j) + p_i \quad l \in N_{NC}^{i-1}, j = \sigma(l) \tag{5-4b}$$

其中 r_l^i 和 dd_l^i 分别表示第 i 次迭代过程工序 l 的到达时间和交货期限。$\sigma(l)$ 表示工序 l 所属工件的标号。公式(5-4a)和(5-4b)暗含了非约束机围绕约束机而调度。具体而言，非约束机的调度受已辨识约束机加工能力约束的制约。

每个非约束机 $k \in M_{NC}^{i-1}$ 采用基于工序交货期限最早最先加工规则(earliest operation due date first，EODD)调度，其调度决策记为 $S_k^i (k \in M_{NC}^{i-1})$。如果非约束机 k 上工序 l 的完工时间超过其设定的交货期限，即 $C_l^i(S_k^i) > dd_l^i$，则非约束机 k 没有足够的能力在约束机设定的期望完工时间内完成相应的操作，约束机的调度被破坏。因此此非约束机 k 没有足够的能力支持约束机的调度，选择它为候选约束机。我们采用最大滞后时间指标来检测所有可能的候选约束机，即 $L\max_k = \max_{l \in N_{NC}^{i-1}} (C_l^i(S_k^i) - dd_l^i)$，其中 $k \in M_{NC}^{i-1}$。如果非约束机 k 最大滞后时间为正值，该机器被选为候选约束机，更新候选约束机集合 $M_{Cand}^i = \{k \mid L\max_k > 0, k \in M_{NC}^{i-1}\}$。

如果检测出多个候选约束机，则从中选择一台机器作为新辨识的约束机。SB 方法中确定机器关键度的规则可用于从所有的候选约束机中选择关键的一台机器，该机器为新辨识的约束机。更新已辨识的约束机集合 $M_C^i := M_C^{i-1} \cup \{b^i\}$，已辨识约束机 M_C^i 的简化模型 P_{CCR}^i 也相应更新。该过程迭代进行直至没有新的约束机辨识。

定理 5-2. 令 z_{CCR}^i 和 z_{CCR}^{i+1} 分别表示第 i 次迭代过程简化问题 P_{CCR}^i 和第 $i+1$ 次迭代过程简化问题 P_{CCR}^{i+1} 的最优值，则有 $z_{CCR}^i \le z_{CCR}^{i+1}$。

证明： 令 S_{CCR}^i 和 S_{CCR}^{i+1} 分别表示问题 P_{CCR}^i 和 P_{CCR}^{i+1} 所有可行解的集合。因为 $M_C^{i+1} = M_C^i \cup \{b^{i+1}\}$，对简化问题 P_{CCR}^i 添加机器 b^{i+1} 的机器能力约束得到简化问题 P_{CCR}^{i+1}，因此问题 P_{CCR}^{i+1} 的所有可行解的集合 S_{CCR}^{i+1} 必属于问题 P_{CCR}^i 的所有可行解的集合 S_{CCR}^i，即 $S_{CCR}^{i+1} \subseteq S_{CCR}^i$。由于问题 P_{CCR}^i 和 P_{CCR}^{i+1} 目标函数相同且为最小化问题，则有 $z_{CCR}^i \le z_{CCR}^{i+1}$。

由定理 5-1 和 5-2 可知 $z_{CCR}^1 \le z_{CCR}^2 \le \cdots \le z_{CCR}^c \le z$，其中 c 为迭代循环次数(即辨识的约束机个数)。在约束机辨识过程中，随着辨识约束机个数的增加，下界逐步提高。

基于不同的规则，从候选约束机中选出的约束机不同，对应的简化模型也随之不同，它会影响约束机和非约束机上工件的加工安排。由于约束机辨识过程是迭代进行，非约束机的结果又将影响下一次迭代过程中候选约束机的辨识，从而影响最终调度的性能。我们将在后面仿真测试中研究约束机选择对调度解的影响。

5.3.2 简化问题的求解

如果 Job shop 中仅包含一个约束机，则 Job shop 调度问题可以简化为一个单机调度问题。传统求解单机问题的调度方法可直接用来求解简化问题。但对 Job shop 存在多个约束机的情况，Job shop 问题简化为一个多机调度问题 P_{CCR}。在问题 P_{CCR} 中，非约束机的工序用时间迟延代替。与原问题 P 相比，简化问题规模显著下降。然而该问题的计算复杂度仍为强 NP-hard[6]。简化问题的求解过程是约束调度算法中最耗时的过程，而且约束调度算法中问题 P_{CCR} 随约束机的迭代辨识不断更新，因此有必要找到一种有效的方法求解该问题。

本文采用一种类 SB 的启发式方法来求解问题 P_{CCR}。该方法通过顺序地求解一组单机调度问题从而降低计算复杂度。一旦一个约束机被调度后，根据该约束机的调度结果固定相应析取弧的方向(见图 5-3 中粗线)。通过依次确定各约束机上工序的加工顺序得到问题 P_{CCR} 的调度结果。

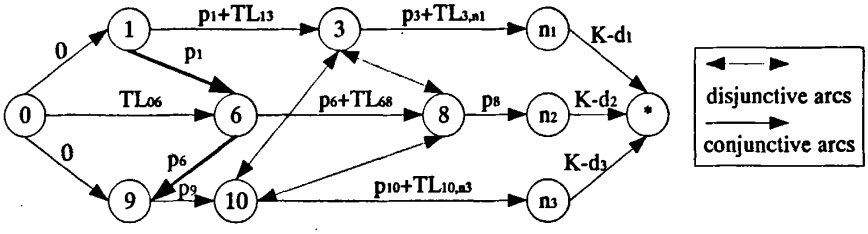


图 5-3. 单约束机调度问题析取图描述

Fig 5-3. Disjunctive graph for single constraint scheduling problem

由图 5-3 可知，已调度约束机的加工安排通过顺序弧添加至析取图上，它使得未调度约束机上工序最早可能的开工时间增加，最迟必须的完工时间值减少，从而缩小了未调度约束机上工序加工安排可用的时间范围。未调度约束机的加工安排必须在满足已调度约束机的加工安排，一个约束机的调度牺牲了其它约束机的性能，该算法可能会产生解的次优性。因此可以通过重新优化已调度约束机来改进调度解的性能。

令 S_l 为约束机 l 的调度决策，M_0 为已调度约束机集合。每个约束机 k 对应子问题的数学模型可描述如下：

$$(P(k,M_0)) \qquad \min \max_{i\in J}(t_{n_i}-d_i) \tag{5-5a}$$

$$st \qquad t_j - t_i \geq p_i \qquad (i,j)\in \bigcup(S_l : l\in M_0)\cup A \tag{5-5b}$$

$$t_i \geq 0 \qquad i \in N \tag{5-5c}$$

$$t_j - t_i \geq p_i \vee t_i - t_j \geq p_j \qquad (i,j) \in E_k \tag{5-5d}$$

问题 $P(k,M_0)$ 等价于单机调度问题 $1|r_i,q_i|L_{\max}$ [40]，它可以用 Carlier 算法优化求解[117]。到目前为止，Carlier 算法求解规模最大为 1,000 个工件[94]。

定理 5-3. 令 z_{CCR} 和 z_k 分别表示问题 P_{CCR} 和 $P(k,\varnothing)$ 的最优值，其中 $\varnothing$ 表示空集，则有 $z_k \leq z_{CCR}$。

该定理显然成立，问题 $P(k,\varnothing)$ 的最优解为问题 P_{CCR} 提供了一个下界。

5.3.3 约束调度算法

约束机的调度决策可以通过求解简化问题 P_{CCR} 获得，而非约束机采用有效的分派规则围绕着约束机而调度。非约束机的调度嵌入至约束机辨识过程，其调度结果用于判定当前非约束机是否有足够能力支持已辨识约束机的调度。给出一个具体的约束调度算法如下：

算法 5-1. 约束调度算法(Constraint Scheduling Algorithm, CSA)

Step 1：初始化。令 $M_C^0 = \varnothing$，$M_{NC}^0 = M$，$N_C^i = \varnothing$，$N_{NC}^i = N$，$i = 1$。

Step 2：约束机迭代辨识过程。

Step 2a. 关联项参数估计。按公式(5-4a)和(5-4b)估计当前每道非约束工序 $l \in N_{NC}^i$ 的到达时间 r_l^i 和交货期限 dd_l^i。

Step 2b. 非约束机调度。按EODD启发式算法调度当前每一台非约束机 $k \in M_{NC}^{i-1}$，并获得每一台非约束机 k 的调度决策 S_k^i。

Step 2c. 候选约束机辨识。最大滞后时间为正值的任一台非约束机被选为候选约束机，即 $M_{Cand}^i = \{k \mid L\max_k > 0, k \in M_{NC}^{i-1}\}$。如果当前迭代不存在候选约束机，则转 Step 4；否则转 Step 2d。

Step 2d. 约束机选择。如果存在多台候选约束机，则按 TML 或 SLV 规则从中选择一台机器 $b^i \in M_{Cand}^i$ 作为约束机。更新 $M_C^i := M_C^i \cup \{b^i\}$，$M_{NC}^i = M \setminus M_C^i$。约束工序 N_C^i 和非约束工序 N_{NC}^i 也随之更新。

Step 3：约束机建模和求解。

Step 3a. 时间迟延估计。对每个相邻约束工序i和j($(i,j)\in A_C, i,j\in N_C^i$)按公式(5-3)估计其时间迟延$TL_{ij}$的大小。

Step 3b. 约束机调度。对当前已辨识的约束机$k\in M_C^i$构造约束机调度问题P_{CCR}^i，采用类 SB 启发式算法求解该问题。更新$i:=i+1$，转 Step 2。

Step 4: 迭代停止过程。获得最终调度，停止迭代。

本章提出的约束调度算法和传统的移动瓶颈方法都是基于瓶颈的分解方法。他们将 Job shop 问题分解为若干更易求解的单机子问题，然后依次顺序求解这些子问题获得最终的调度结果。然而这两种方法有其各自不同的特点：

首先，每次迭代子问题的调度程度不同。每次迭代移动瓶颈方法对约束机和非约束机均详细调度，而本章提出的约束调度算法对它们分别采用不同的调度策略。由于迭代辨识过程中非约束机的调度结果仅仅用于判定是否产生新的约束机(见 Step 2)，因此没有必要耗费大量的计算时间对每个非约束机精确调度。在每次迭代过程中，本章提出的约束调度算法采用一类 SB 算法优化调度约束机(见 Step 3b)，而对非约束机采用相对简单的 EODD 算法(见 Step 2b)。这种处理方式不仅可以保留好的调度质量，而且可以大大提供计算效率。对不均衡问题效果更显著。

其次，迭代停止条件不同。在移动瓶颈方法中，整个迭代过程持续直到所有的机器都调度完。而在我们提出的约束调度算法中，如果不存在新的约束机则迭代停止。此时非约束机有足够的能力满足已辨识约束机的调度。因此没有必要像移动瓶颈方法一样耗费大量的计算时间对非约束机精确调度。

最后，重新优化的过程不同。在重新优化过程中，移动瓶颈方法根据当前新调度机器的调度决策对所有已调度的机器重新优化。因此可能会产生约束机围绕着非约束机调度的现象，即非约束机的调度将牺牲部分约束机的性能。重新优化过程结束后可能会产生性能恶化。而约束调度算法则是基于当前调度约束机的加工安排对所有已调度约束机重新优化。非约束机的调度安排总是在约束机调度之后进行，约束机一直主导非约束机，从而保证了约束机的调度性能。

总而言之，本章提出的约束调度算法可以保证算法执行过程中约束机一直主导非约束机，它不但可以大大提高计算效率，而且可以保持好的调度性能。

5.4 仿真测试与分析

为了评价带多约束机 Job shop 调度问题约束调度算法的性能，按文献[108]的方式随机生成大量测试数据，实验参数设计如表 5-1 所示。所有工件均在零时刻到达，

非瓶颈机上每个工件的加工时间服从[1,50]均匀分布，瓶颈机上每个工件的加工时间服从$[1+l,50\cdot(1+l)]$均匀分布，其中l为机器负荷差异程度。瓶颈机上总加工负荷比非瓶颈机平均高l。机器之间负荷差异l越大，瓶颈程度越高。工件的交货期限服从$[P'(1-T-R/2),P'(1-T+R/2)]$均匀分布，其中$P'$是最大完工时间的下界。$T$和$R$分别表示交货滞后程度和交货期范围。我们分别考虑3种不同的机器个数(10, 20, 30)，2种不同的工件个数(30, 50)，4种不同约束机个数(1, 2, 4, m)，3种不同机器负荷差异程度(0.5, 1.0, 2.0)，2种交货期范围(0.4, 1.2)和2种交货期滞后程度(0.1, 0.3)。所有参数组合生成288种类型的问题，每种类型分别生成10组数据，共2880组测试数据。

表 5-1. 测试数据生成参数表

Table 5-1. Experimental design for randomly generated test problem

Problem parameter	Value	Total Value
Number of machines m	10, 20, 30	3
Number of jobs n	30, 50	2
Percentage tardy jobs T	0.1, 0.3	2
Due date range R	0.4, 1.2	2
Workload difference l	0.50(low), 1.0(medium), 2.0(high)	3
Number of constraint machines b	1, 2, 4, m	4
Total problem combination	$3\times2\times2\times2\times3\times4 = 288$	

首先研究约束机的选择对约束调度算法(CSA)性能的影响。分别采用TML和SLV规则从候选约束机中选择关键的机器，构造两种约束调度算法，记为CSA_TML和CSA_SLV。对10台机器和30个工件类型的问题生成的480组数据进行测试，性能指标为最大滞后时间L_{max}。如果算法S对算例I的调度解(上界)等于其下界，则该算法的解必为最优解。表5-2给出了算法CSA_TML和CSA_SLV求解问题$J_m \| L_{max}$获得的目标值的均值(Mean)、标准方差(SDV)、平均计算时间(ACT)，以及该算法找到最优解的个数(NOS)。所有算法的平均计算时间以秒为单位。

表 5-2. 不同约束机选择规则算法的性能比较

Table 5-2. Performance of the algorithms with constraint machine selection rules

Problems	CSA_TML				CSA_SLV			
	Mean	SDV	ACT(s)	NOS	Mean	SDV	ACT(s)	NOS
All data	656.3	18.29	4.32	32.7%	661.9	18.25	4.55	32.5%
Number of the constraint machines b								
1	428.6	15.84	4.23	55.8%	430.0	15.80	4.58	57.5%
2	561.8	16.72	4.17	43.3%	568.4	16.61	4.35	44.2%
4	683.5	17.05	4.11	30.0%	693.2	16.94	4.12	25.8%
m	951.2	18.67	5.06	1.7%	956.2	18.67	5.56	2.5%
Workload difference l								
0.5	376.5	12.16	6.68	15.0%	382.9	12.14	7.11	15.0%
1.0	582.7	15.29	4.16	38.1%	588.4	15.23	4.83	36.9%
2.0	1009.6	19.32	2.31	45.0%	1014.5	19.29	2.34	45.6%

如表 5-2 所示，选择加工负荷最严重的约束机最先加工(TML)一般来说效果最好，但不是对所有情况都是最好的。采用 TML 和 SLV 规则得到调度性能差异不大，因此在求解多约束机 Job shop 问题的约束调度算法中采用任何一种选择规则都可以。表 5-2 中 NOS 列的数据显示机器负荷差异越大，算法找到最优解个数越多，约束调度算法的性能越好，而且随瓶颈程度的提高算法的平均计算时间也相应减少。

然后，采用 2880 个实例来测试本章提出的约束调度算法(CSA)的性能。将它与移动瓶颈(SB)方法和 TOC 算法进行比较。SB 算法中每个子问题描述为带到达时间、传递时间以及迟延顺序约束的单机调度问题[94]，均采用 Carlier 算法精确求解[117]。按机器关键程度依次调度每台机器。每调度完一次机器，已调度的机器则按滞后时间从大到小的顺序重新优化。SB 方法分别采用不同的关键机器选择规则和重新优化过程。其中关键机器选择分别采用 TML 和 SLV 规则，对应的 SB 算法分别记为 SB(TML)和 SB(SLV)。考虑三种不同的重新优化过程，即无重新优化过程，局部重新优化过程以及完全重新优化过程，它们分别表示为 NR, LR 和 FR。不同子问题优先级和重新优化过程的组合得到六种不同的 SB 算法，记为 SB(TML)_NR, SB(TML)_LR, SB(TML)_FR 和 SB(SLV)_NR, SB(SLV)_LR, SB(SLV)_FR。TOC 算法具体见文献[108]。

我们采用目标值与下界的相对比值η来评价算法的调度质量[95]，令$L_{\max}(H,I)$为给定实例I下采用启发式算法H求得的目标函数值，$LB(I)$为问题I的下界，则有：

$$\eta(H,S)=\sum_{I\in S}L_{\max}(H,I)/\sum_{I\in S}LB(I) \tag{5-6}$$

其中$LB(I)$是约束调度算法中第一次迭代求解简化问题P_{CCR}获得的最优值(见定理 5-1)。计算时间的评价指标采用平均 CPU 计算时间$ACT(H,S)$。所有的算法均采用 C 语言编程，运行在 Celeron 处理器(CPU 速度 2GHZ，缓存 128kB，内存 256MB)的机器上。仿真结果见表 5-3，5-4 和 5-5。表 5-3 和 5-4 分别表示不同参数对算法性能的影响。表 5-5 给出不同算法的平均计算时间。所有算法的计算时间以秒为单位。

如表 5-3，5-4 和 5-5 显示，在所有的算法中，带完全重新优化的 SB 算法的性能最好，它比约束调度算法平均好 2.6%。但它的计算时间非常长。对 30 台机器 50 个工件的问题该算法需要耗费近 11 个时间，这在实际生产调度过程中是无法接受的。TOC 算法计算效率非常高，但其调度质量较差，它比约束调度算法平均差 25%。约束调度算法可以在调度质量和计算时间之间获得一个好的均衡。它比带局部重新优化的 SB 算法和无重新优化过程的 SB 算法平均好 12.3% 和 15.1%，而且它只需要带局部重新优化的 SB 算法计算时间的 1/30 到 1/3。产生这一现象的原因可能是移动瓶颈方法重新优化过程中产生约束机围绕着非约束机调度的情况，即在调度非约束机的

时候牺牲了约束机的性能，这不但需要耗费额外的计算代价，而且调度的质量反而下降。而我们提出的约束调度算法的机制保证算法执行过程中约束机一直主导非约束机，因此它可以在提高计算效率的前提下仍保持好的调度性能。

表 5-3. 不同问题规模下算法性能比较

Table 5-3. Performance of the algorithms for different problem sizes

Algorithms		Problem size (m, n)					
		(10, 30)	(10, 50)	(20, 30)	(20, 50)	(30, 30)	(30, 50)
SB(TML)	NR	1.285	1.194	1.341	1.295	1.403	1.352
	FR	1.096	1.050	1.156	1.119	1.211	1.160
	LR	1.260	1.181	1.304	1.267	1.355	1.317
SB(SLV)	NR	1.337	1.256	1.379	1.344	1.431	1.384
	FR	1.095	1.047	1.152	1.113	1.207	1.153
	LR	1.275	1.197	1.313	1.281	1.369	1.325
TOC		1.341	1.292	1.428	1.351	1.593	1.467
CSA	TML	1.135	1.084	1.187	1.143	1.214	1.185
	SLV	1.137	1.091	1.189	1.142	1.217	1.188

表 5-4. 不同瓶颈程度下算法性能比较

Table 5-4. Performance of the algorithms for different bottleneck factors

Algorithms		Workload difference (l)			Number of constraints (b)			
		0.5	1.0	2.0	1	2	4	m
SB(TML)	NR	1.428	1.237	1.095	1.125	1.141	1.165	1.482
	FR	1.122	1.082	1.002	1.046	1.049	1.063	1.192
	LR	1.379	1.227	1.068	1.123	1.176	1.233	1.320
SB(SLV)	NR	1.538	1.241	1.109	1.151	1.206	1.268	1.496
	FR	1.126	1.086	1.003	1.043	1.050	1.068	1.188
	LR	1.401	1.249	1.078	1.117	1.178	1.257	1.351
TOC		1.511	1.295	1.159	1.209	1.225	1.265	1.548
CSA	TML	1.161	1.091	1.003	1.052	1.055	1.090	1.249
	SLV	1.182	1.094	1.003	1.054	1.062	1.086	1.235

表 5-5. 不同问题规模下算法的平均计算时间

Table 5-5. Average computation time of the algorithms for different problem sizes

Algorithms		Problem size (m, n)					
		(10, 30)	(10, 50)	(20, 30)	(20, 50)	(30, 30)	(30, 50)
SB(TML)	NR	1.2	7.5	10	66.9	155	270
	FR	21	125	339	2298	6536	29757
	LR	14	103	294	1681	4093	20171
SB(SLV)	NR	3.5	12.5	42	168	285	628
	FR	26	195	1255	3124	8407	39442
	LR	23	137	486	2537	6447	28763
TOC		0.01	0.5	1.5	3.7	5.6	8.4
CSA	TML	4.3	18	44	191	311	641
	SLV	4.5	19	44	190	312	644

为了研究不同参数对约束调度算法的影响，我们对表 5-3 和 5-4 的结果进行具体的分析。如表 5-3 所示，约束调度算法的性能随着机器个数的增加而下降。当机器个数从 10 个增至 30 个，约束调度算法目标值与下界的比值平均增加了 9%。原因在于 Job shop 中约束机的主导特性随机器个数的增加而减弱。相反的，约束调度算法的性能随着工件个数的增加而改进。当工件个数从 30 个增至 50 个，约束调度算法目标值与下界的比值平均减少了 4%。原因可能是随着工件个数的增加，Job shop 中的约束机变更为稳定，有效调度约束机的重要性也相应提高。

而且约束调度算法的性能也受机器负荷差异影响。如表 5-4 所示，约束调度算法的性能随机器负荷差异 l 的增加而提高。当机器负荷差异 l 从 0.5 增至 1.0，约束调度算法目标值与下界的比值平均减少了 7.9%。当机器负荷差异 l 从 1.0 增至 2.0，约束调度算法目标值与下界的比值平均减少了 9%。这些结果显示机器负荷差异是影响约束调度算法一个非常重要的参数。机器负荷差异越大，瓶颈程度越强，约束调度算法越有效。同时负荷差异也影响约束调度算法的计算时间，计算时间随机器负荷差异的增加而减少。负荷差异越大，非约束机的冗余能力越大，辨识出的约束机的个数也相应减少。

对不均衡问题(约束机个数 $b=1,2,4$)，约束机个数对约束调度算法的影响不大。但它比处理均衡问题(约束机个数 $b=m$)的调度性能要好。而且约束调度算法求解不均衡问题的计算时间也要比求解均衡问题要快。对某些均衡问题，约束调度算法与带重新优化过程的 SB 算法的计算时间差不多，但约束调度算法对不均衡问题的求解时间远小于带重新优化过程的 SB 算法。

总之，约束调度算法在调度质量和计算时间上可以获得一个好的均衡。Job shop 生产线瓶颈程度越高，约束调度算法的性能越有效。

5.5 本章小结

本章主要研究带多约束机 Job shop 调度问题，在瓶颈分解的基本框架下，提出了一种新的约束机调度算法。其中约束机为广义的瓶颈，它包括长期瓶颈和暂时瓶颈。该方法利用约束机与非约束机能力差异将 Job shop 问题简化为一个更易求解的多机调度问题，简化问题中原非约束机上的加工工序用一个时间迟延环节替代。约束机上的调度可以通过优化求解该简化问题获得，而非约束机则采用 EODD 规则围绕约束机的调度结果而调度。该方法可以大大减少计算量，同时又可以保证获得好的调度性能。大量的仿真比较研究显示，与 SB 方法和 TOC 方法相比，本文提出的约束机调度算法可以在调度性能和计算时间之间获得一个好的均衡。它对不均衡 Job shop 问题的求解优势更加明显。

第六章 滚动瓶颈分解方法研究及其在半导体生产中的应用

6.1 引言

在上一章我们针对多约束机 Job shop 调度问题提出了一种约束调度算法。与传统的移动瓶颈方法相比，该方法不仅可以大大减少计算时间，同时也保证了调度的优化性能。随着 Job shop 生产线规模的不断增大和产品加工复杂程度的增加，由于内存的限制，单一的空间层分解可能无法有效地求解更大规模的 Job shop 调度问题，因此有必要找到一种有效的方法处理超大规模的 Job shop 调度问题。

实际 Job shop 生产线的调度通常按一个月、一周、一天或一个班次(shift)进行安排。类似这种分段决策思想，我们将整个调度时域分解为若干决策子窗口。每个时间窗口对应的子问题仍是一个 Job shop 问题，但问题规模减小。子问题可以采用上一章提出的约束调度算法优化求解。

本文在前面提出的瓶颈分解策略的基础上，结合滚动时域分解方法，提出了一种滚动瓶颈分解方法。该方法从时间层和空间层两个方面同时对大规模问题进行分解，它为求解超大规模 Job shop 调度问题提供了一种新的解决思路。随后将该方法应用至一个更复杂的生产调度问题，即半导体生产线。针对半导体生产线上机器加工类型不一，对分解后不同类型的子问题给出了相应的求解算法。

本章的内容安排如下：6.2 节在研究滚动时域分解算法的基础上，针对大规模 Job shop 调度问题提出了一种滚动瓶颈分解算法，并进行详细的仿真测试研究。6.3 节分析了半导体生产调度问题的特点，改进滚动瓶颈分解算法，应用于半导体生产线经典 MINFAB 模型。

6.2 大规模 Job shop 问题滚动瓶颈分解方法研究

6.2.1 Job shop 问题滚动时域分解方法

如 5.2.1 节所述，Job shop 调度问题可以描述为一个析取图 $G(N,A,E)$(见图 6-1)。其中 N 表示工序集合，对应析取图上的节点。节点 ij 表示工件 i 在机器 j 上加工的工序，节点 0 和 * 分别表示虚拟的起点和终点。集合 A 表示满足工件加工顺序关系的工序对集合，在析取图中采用有向弧描述。而集合 $E=\bigcup\{E_k:k\in M\}$ 表示在同一台机器上加工的工序对集合，其中 E_k 为机器 k 上加工的工序对集合，在析取图中采用析取弧(双向弧)描述。在此析取图中，每个节点 ij 出弧的长度等于该节点对应工序的加工时

间 p_{ij}。令每个工件最后一道工序 O_{im} 到虚拟终点*弧线的长度为 $K-d_i+p_{im}$，其中 K 为一个常数，$K \geq \max_{i \in J}\{d_i\}$，则从起点0到终点*的最长路径的长度等于 $L_{\max}+K$。因此，$J_m \| L_{\max}$ 问题最优调度相当于确定析取图中析取弧的方向使得构造的一个非循环有向图从起点0到终点*的最长路径的长度最小。

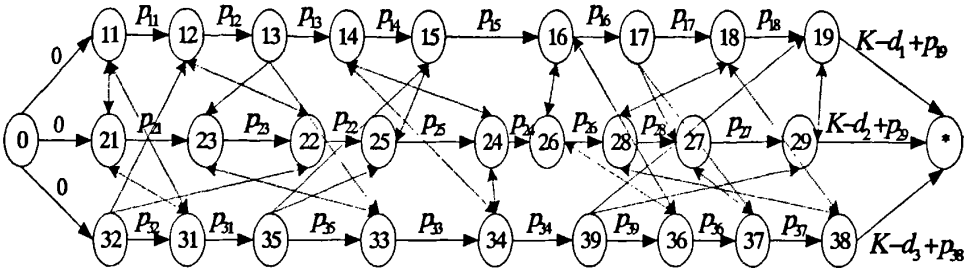


图 6-1. Job shop 调度问题析取图描述(9×3 例子)

Fig 6-1. Disjunctive graph of Job shop scheduling problem for a 9×3 instance

$J_m \| L_{\max}$ 问题的计算复杂度为强 NP-hard[6]，精确算法只能求解小规模问题。由图 6-1 可知，随着问题规模的增大，析取图中集合 N 中的节点个数和集合 A 中的有向弧对的个数线性增加，且集合 E 中的析取图弧对的个数呈指数增长，这需要耗费大量的存储空间。由于内存限制，上一章提出的约束调度算法可能无法求解更大规模的 Job shop 问题。因此下面我们在瓶颈分解的基础上，借鉴滚动时域分解的思想来处理超大规模 Job shop 调度问题。

滚动时域方法(RHP)是一种时间分解策略，对 Job shop 问题来说，分解后的调度子问题仍然是 Job shop 问题。通常子问题中的机器数目与原问题相差不大，但所包含工件的操作数大大减少，从而降低了子问题的求解规模。在实际生产调度中，整个调度决策也是分段进行的如一个月、一周、一天或一个班次，它和 RHP 中的时间窗口可以一一对应起来。因此结合时间和空间分解处理超大规模 Job shop 调度问题是有现实意义的。

RHP 算法将整个调度时域分解为若干小规模的时间窗口，每个子问题对应一个时间窗口。沿时间轴推进的方向依次优化分解后的小规模或有限时段的局部问题。在每个决策时刻，利用当前已知信息确定一个时间窗口，并对该时间窗口内的工序进行局部优化。执行该时间窗口的部分调度，这部分解的完成时刻就是新的决策时刻，如此重复迭代，直到全部工序的调度安排均执行完。因此 RHP 算法主要包括两个决策过程[124]：(1) 如何分配工序至每个时间窗口，即子问题建模；(2) 如何优化调度每个时间窗口，即子问题求解。

第一个决策问题表示如何对原问题进行分解。目前 RHP 算法中工序分配方式主要有两种[97,98]：基于工序个数和基于时间窗口大小的分解方式。在第一种方式中，每次选择每个工件未调度的前 x 道工序至当前时间窗口。除了最后一个时间窗口，其它时间窗口内工序的个数均为 $n \cdot x$ 个，其中 n 为工件的个数，这种工序分配方式保证每个子问题优化规模相差不大。在第二种方式中，任意决策时刻 t，选择在时域 $[t, t+\mathrm{FW}]$ 内已到达的工序至当前的决策子窗口中。其中 FW 为预测窗口大小，它决定了当前调度决策待优化时间窗口内工序的个数；

第二个决策问题是确定子问题求解算法以及子问题的求解顺序。由于分解后的子问题之间相互关联，因此在求解每个子问题时必须考虑到其它子问题的加工信息。如果子问题之间的关联是单向耦合的，即子问题 1 的解影响子问题 2，但子问题 2 的解不影响子问题 1，则可以先优化子问题 1，然后再优化子问题 2。在 RHP 算法中，每个子问题对应一个时间窗口，每个时间窗口只受前一个时间窗口的影响，子问题是单向耦合的。对两个相邻的时间窗口而言，第一个时间窗口内工序的完工时间决定了第二个时间窗口内工序所需机器的可用时间(available time)，因此必须先优化第一个时间窗口。在 RHP 算法中，我们按时间窗口在时间轴上的先后顺序依次优化相应的子问题。

由于每个时间窗口 l 对应的子问题仍为 Job shop 调度问题，因此求解 Job shop 问题的任何优化算法均可用于子问题求解，获得时间窗口内工序的调度安排。给出具体的滚动时域算法如下。

算法 6-1. 滚动时域算法(Rolling Horizon Procedure, RHP)[125]

Step 1：初始化。设置预测时间窗口 FW 和调度时间窗口 SW 的大小，$t=0$，$l=1$。

Step 2：子问题辨识。选择时域 $[t, t+\mathrm{FW}]$ 内已经到达的工序至当前预测窗口，其中 t 为当前时刻，即决策点。对预测窗口内的所有工序建立子问题。如果 $l>1$，则根据前一窗口 $l-1$ 的调度结果更新当前调度窗口 l 内工序的有关参数；

Step 3：子问题求解。优化求解子问题，获得时间窗口 l 内工序的调度安排。

Step 4：状态更新。根据 Step 3 获得的调度结果，固定时域 $[t, t+\mathrm{SW}]$ 内已经开始加工的工序。从未调度工序集中删除这些工序，更新未调度工序的到达时间。更新 $t := t+\mathrm{SW}$。如果所有的工序均调度完，则算法停止；否则，返回 Step 2。

该算法中有两个决策参数：预测窗口大小 FW 和调度窗口大小 SW。当预测窗口大小为零，相当于对未来信息完全未知，此时为在线调度的情况；当预测窗口大小等于整个调度时域长度时，未来信息全部已知，滚动时域算法变成了一次全局静态调度。

算法 6-1 中对预测窗口中所有工序建立子问题，因此预测窗口的大小表征局部调度子问题的求解规模。FW 越大，子问题规模越大。而且算法 6-1 采用周期性滚动机制，子问题调度的次数和系统对动态因素的适应能力由调度窗口SW 决定。SW 越小，求解的子问题个数增加，计算时间也相应增加；SW 越大，调度的次数就越少，但系统对动态事件的发生不能及时做出反应。因此针对不同问题，应权衡调度的次数和系统对动态因素的适应能力二者的矛盾，选择合适的预测窗口和调度窗口大小。

6.2.2 大规模 Job shop 问题滚动瓶颈分解算法

大规模 Job shop 问题的析取图模型 $G(N,A,E)$ 描述了 Job shop 生产线中所有工件的所有工序的加工状态，随着问题规模的增大，由于内存不足可能导致单一的空间层分解方法对其仍无法求解。对此情况，从空间和时间两个层面对问题进行分解，提出了一种滚动瓶颈分解方法。

该方法将整个调度时域分解为若干时间段，每个子问题对应一个时间窗口。分解后的子问题仍是一个 Job shop 问题。由于子问题的求解规模远小于原 Job shop 问题，它可以采用瓶颈分解方法优化求解。在滚动瓶颈分解方法中，每个时间窗口是单向耦合，因此按时间窗口的先后顺序依次优化子问题就可处理它们之间的相互关联。

由于时间窗口内子问题优化时不考虑调度窗口之外的工序，因此没有必要耗费大量的存储空间在析取图上对所有工件的所有工序建模。在每个调度决策点只需建立当前时间窗口内工序的析取图模型(见图 6-2)，这样可以大大节省存储空间而不影响调度性能。

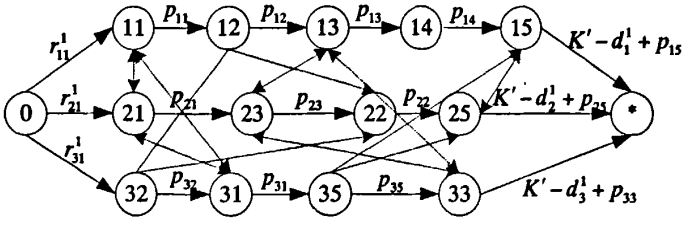


图 6-2. 时间窗口子问题析取图描述

Fig. 6-2. Disjunctive graph of the sub-problem in time window

时间窗口 l 的析取图模型 $G(N^l,A^l,E^l)$ 描述如图 6-2 所示：节点集 N^l 为调度窗口 l 内的工序集合；集合 A^l 表示节点集 N^l 中满足顺序关系的工序对集合，在图中用有向弧描述；集合 E^l 表示节点集 N^l 中满足在同一机器上加工的工序对集合，在图中用析取弧描述。每个节点 $ij\in N^l$ 出弧的长度等于该节点对应工序的加工时间的大小 p_{ij}。

令滚动子窗口个数为w，n_i^l表示子窗口l内属于工件i的工序个数，m_k^l表示子窗口l内属于机器k的工序个数，则有$\sum_{i=1}^{n} n_i^l = \sum_{k=1}^{m} m_i^l$，它表示子窗口$l$问题的求解规模，且$\sum_{l=1}^{w}\sum_{i=1}^{n} n_i^l = \sum_{l=1}^{w}\sum_{k=1}^{m} m_i^l = mn$；

令pN_i^l表示子窗口l内工件i已调度的工序个数，当$l=1$时，$pN_i^l = 0$；否则，$pN_i^l = \sum_{j=1}^{l-1} n_i^j$；$pM_k^l$表示子窗口$l$内工件$i$已调度的工序个数，当$l=1$时，$pM_k^l = 0$；否则$pM_k^l = \sum_{j=1}^{l-1} m_k^j$；$s_i(j)$表示工件$J_i$第$j$道工序加工的机器的标号，$\sigma_i(k)$表示工件$J_i$在机器$M_k$上加工工序标号；$C_{ik}$表示工件$J_i$在机器$M_k$上的完工时间；$T_k^l$表示子窗口$l$内机器$k$的完工时间，它等于子窗口$l$内机器$k$上第$m_k^l$个加工工序的完工时间，初始时$T_k^1 = 0(k=1,\cdots,m)$。

由于滚动时域方法中每个子问题的调度取决于前一个子问题的调度结果，为了保证分解后子问题不破坏原问题的约束关系，时间窗口l内每个工件i第j道工序的到达时间$r_{i,s_i(j)}^l$必须大于该工序前一道工序的完工时间和该工序所需加工机器k上的可用开工时间。令当前时间窗口为l，则该时间窗口内所有工序以及该时间窗口以外未调度的所有工序的到达时间可按如下方式估计：

$$r_{i,s_i(pN_i^{l-1}+j)} = \begin{cases} 0 & i=1,\cdots,n;\ j=1;\ l=1; \\ T_{s_i(j)}^{l-1} & i=1,\cdots,n;\ j=1;\ l>1;\ pN_i^{l-1}=0; \\ \max(T_{s_i(pN_i^{l-1}+j)}^{l-1}, C_{i,s_i(pN_i^{l-1})}) & i=1,\cdots,n;\ j=1;\ l>1;\ pN_i^{l-1}>0; \\ r_{i,s_i(pN_i^{l-1}+j-1)} + p_{i,s_i(pN_i^{l-1}+j-1)} & i=1,\cdots,n;\ j=2,\cdots,m-pN_i^{l-1}; \end{cases} \tag{6-1}$$

由于滚动时域方法是沿着时间轴方向依次调度子窗口，在时间窗口l调度时，其后的时间窗口内的工序均未调度，因此我们可以通过以下方式估计当前时间窗口l工件i的局部交货期限d_i^l：

$$d_i^l = \begin{cases} d_i & i=1,\cdots,n;\ l=w; \\ d_i - \sum_{j=pN_i^l+1}^{m} p_{i,s_i(j)} & i=1,\cdots,n;\ l<w; \end{cases} \tag{6-2}$$

令每个工件最后一道工序O_{ik}到虚拟终点*弧线的长度为$K' - d_i^l + p_{ik}$，其中K'为一个常数，$K' \geq \max_{i\in J}\{d_i^l\}$，则从起点0到终点*的最长路径大小等于$L_{\max} + K'$。因此，时间窗口$l$的最优调度相当于确定析取图$G(N^l, A^l, E^l)$中析取弧的方向使得构造的一个非循环有向图从起点0到终点*的最长路径的长度最小。

在滚动瓶颈分解方法中，需要顺序求解一组子窗口问题。为了保证该方法的计算

效率，需要子问题的求解过程计算时间不要过长。移动瓶颈方法是求解 Job shop 调度问题非常有效的一种算法，但该算法的计算时间比较长。由于移动瓶颈分解方法的计算困难主要来自机器的个数，对此我们对时间分解后的子问题进一步分解，采用前面提出的瓶颈分解方法处理时间窗口子问题优化问题。本文采用约束调度算法(算法 5-1)来优化求解子问题，该方法已经验证可以在合理的时间内获得一个好的调度。

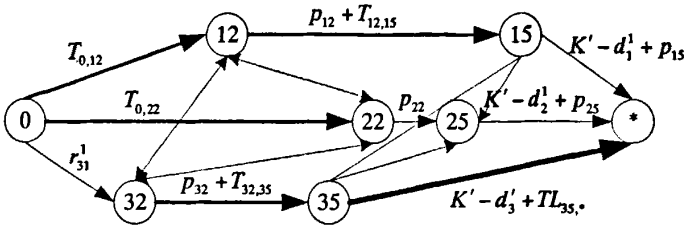


图 6-3. 时间窗口简化问题析取图描述

Fig. 6-3. Disjunctive graph of the reduced problem in time window

基于瓶颈分解思想，利用机器负荷的差异将子窗口内的机器分为两类：约束机和非约束机。由于非约束机有足够的能力支持约束机的调度，在优先调度约束机的前提下非约束机的加工约束很容易满足。因此对非约束机上的加工工序用一个简单的时间迟延代替，得到滚动时间窗口内的简化问题。图 6-3 给出了滚动时间窗口的简化问题的析取图描述$G(N_C,A_C,E_C)$。在该图中，非约束机工序的节点完全删除，同一工件相邻约束工序i和j对应节点的长度等于约束工序i的加工时间加上它与相邻约束工序j之间的非约束工序加工所需的时间迟延TL_{ij}，其中时间迟延按公式(5-3)计算。时间窗口内约束机的调度可以通过优化求解简化问题获得，而非约束机则可采用 EODD 规则调度，具体的优化算法见算法 5-1。给出具体的滚动瓶颈调度算法如下：

算法 6-2. 滚动瓶颈调度算法(Rolling Bottleneck Scheduling Algorithm, RBS)

Step 1：初始化。设置预测时间窗口 FW 和调度时间窗口 SW 的大小，$t=0$，$l=1$。

Step 2：子问题辨识。按公式(6-1)估计所有未调度工序的到达时间。选择到达时间小于$t+$FW 的未调度工序至当前时间窗口l，其中t为当前时刻。按公式(6-2)设置时间窗口l内工件的局部交货期限，建立时间窗口子问题$G(N^l,A^l,E^l)$。

Step 3：子问题求解。采用算法 5-1 求解子问题，得到瓶颈机上的工序的加工顺序和开工时间。

Step 4：状态更新。根据子窗口获得的调度安排，固定在时域$[t,t+\mathrm{SW}]$内开始加工的工序。从未调度工序集中删除这些工序。更新$t=\max_{k\in M} T_k^l$，$l:=l+1$。如果

所有的工序均调度完，则算法停止；否则，返回 Step 2。

在上述提出的滚动瓶颈调度算法中，瓶颈机是针对每个时间段而言，相当于该时间窗口内的关键机器。在某个时间窗口内，某机器可能因为在该时间段内加工任务比较集中造成该机器上的加工负荷过重而成为瓶颈。在另一个时间段内，该机器的加工任务相对较少，在其调度子窗口内有足够的时间完成其加工，该机器成为非瓶颈机。因此对不同时间子窗口辨识的瓶颈也可能不同。Job shop 的工艺路径、机器的加工能力以及预测窗口的大小都会影响影响瓶颈机的辨识。在后面的仿真研究中，我们将对此现象进行研究。

6.2.3 仿真测试与分析

为了评价带大规模不均衡 Job shop 调度问题滚动瓶颈分解算法的性能，按文献[108]的方式随机生成大量的测试数据。所有工件均在零时刻到达，非瓶颈机上每个工件的加工时间服从[1,50]均匀分布，瓶颈机上每个工件的加工时间服从$[1+l,50\cdot(1+l)]$均匀分布，其中l为机器负荷差异程度。机器之间负荷差异越大，瓶颈程度越高。l分别取 0.5，1.0 和 2.0，分别表示低、中和高三种程度。工件的交货期限服从$[P'(1-T-R/2),P'(1-T+R/2)]$均匀分布，其中$P'$是最大完工时间的下界。$T$和$R$分别表示交货滞后程度和交货期范围。我们分别考虑 2 种不同的机器个数(30, 50)，3 种不同的工件个数(50, 100, 200)，3 种不同约束机个数(1, 4, m)，3 种不同机器负荷差异程度(0.5, 1.0, 2.0)，2 种交货期范围(0.4, 1.2)和 2 种交货期滞后程度(0.1, 0.3)。所有参数组合生成 216 种类型的问题，每种类型分别生成 10 组数据，共 2160 组测试数据。

将滚动瓶颈分解算法(RBS)分别与算法 4-2(BDP)、算法 5-1(CSA)以及有效的规则调度(EDD, CR)[39]进行比较。RBS 算法中预测窗口 FW 的大小分别选择为 500 和 1000，调度窗口 SW 的大小分别选择为 250 和 500。

我们采用目标值与最好值相对比值的平均值ρ来评价算法的调度质量[95]，令$L_{\max}(H,I)$为给定实例I下采用启发式算法H求得的目标函数值，所有算法中最好值$BEST(I)=\min_H\{L_{\max}(H,I)\}$。定义一组满足相同特征的问题$S$，对每类问题$S$计算目标值与最好值平均比值为：

$$\rho(H,S)=\sum_{I\in S}L_{\max}(H,I)\Big/\sum_{I\in S}BEST(I)$$

计算时间的评价指标采用平均 CPU 计算时间$ACT(H,S)$。所有的算法均采用 C 语言编程，运行在 Celeron 处理器(CPU 速度 2GHZ，缓存 128kB，内存 256MB)的机器上。仿真结果见表 6-1 和 6-2。表 6-1 给出不同问题规模下算法的性能比较值，在

该表中 RBS 的调度值取不同决策参数(预测窗口和调度窗口)获得的最好值。表 6-2 显示预测窗口 FW 和调度窗口 SW 大小对滚动瓶颈算法的影响。

表 6-1. 不同问题规模下算法的性能比较

Table 6-1. Performance of the algorithms for different problem sizes

Problem	$\rho(H,S)$		Heuristic H		
(m,n)	RBS	BDP	CSA	EDD	CR
(30, 50)	1.08	1.17	1.01	1.26	1.25
(50, 50)	1.10	1.22	1.02	1.31	1.31
(30, 100)	1.01	1.19	N/A	1.29	1.29
(50, 100)	1.02	1.25	N/A	1.37	1.36
(30, 200)	1.03	1.26	N/A	1.39	1.37
(50, 200)	1.02	1.30	N/A	1.42	1.42

由表 6-1 可以看出，瓶颈分解算法(算法 RBS，算法 BDP 和算法 CSA)都优于规则调度(算法 EDD 和算法 CR)。对中等规模的调度问题， 约束调度算法 CSA 的性能最好，滚动瓶颈算法 RBS 次之。对规模为 30×50 的问题，算法 CSA 的调度值比算法 RBS 的调度值平均好 8%；对规模为 50×50 的问题，算法 CSA 的调度值比算法 RBS 的调度值平均好 10%。原因可能在于算法 CSA 的求解是对整个调度时域而言，全局指导意义更显著。而算法 RBS 则是通过求解多个时间窗口子问题依次获得各机器上工序的加工安排，在优化每个子问题时只是利用了部分信息。而且在时域分解过程中不能保证每个子窗口的工序分配合理，当预测窗口和调度窗口的大小等于整个调度时域长度时，算法 RBS 与算法 CSA 等价。算法 CSA 虽然非常有效，但对规模超过 50×50 的调度问题，由于内存限制，它无法对更大规模的问题优化求解。

对大规模的调度问题，滚动瓶颈算法 RBS 性能最好，单瓶颈分解算法 BDP 次之。RBS 的调度值比算法 BDP 的调度值平均好 20.2%，原因可能是算法 BDP 只是将整个 Job shop 问题简化为一个单机调度问题来求解，非瓶颈机采用 EODD 规则求解。它在能力不均衡程度很高的情况下与 RBS 算法相差不大，但对能力不均衡程度较低的问题，它比算法 RBS 要更差。

在大规模静态问题优化求解中，滚动瓶颈分解方法将无法求解的大规模问题转化为若干小规模问题来求解，它降低了问题的求解规模。该方法还可以推广至动态不确定问题，它不仅可以降低问题规模，还可以实时跟踪动态信息。

在滚动瓶颈分解方法中，预测窗口 FW 和调度窗口 SW 的大小是决策参数。我们研究了不同预测窗口和调度窗口的大小下算法 RBS 的性能，仿真结果见表 6-2。

如表 6-2 所示，预测窗口 FW 越大，算法 RBS 的性能越好。对调度窗口大小为

250 的情况，当 FW 从 500 增至 1000 时，算法 RBS 平均改进了 2.5%。对调度窗口大小为 500 的情况，当 FW 从 500 增至 1000 时，算法 RBS 平均改进了 4.7%。原因在于预测窗口 FW 的大小决定了利用未来信息的多少。预测窗口越大，利用的未来信息越多，调度性能越好。但是预测窗口越大，相应付出的计算代价也越大，子问题的求解复杂程度增加。

表 6-2. 不同预测窗口和调度窗口参数下滚动瓶颈算法的性能

Table 6-2. Performance of the RBS algorithms with different FW and SW

Problem	$\rho(H,S)$ Algorithm RBS			
(m, n)	FW=500,SW=250	FW=500,SW=500	FW=1000,SW=250	FW=1000,SW=500
(30, 50)	1.08	1.18	1.04	1.12
(50, 50)	1.09	1.19	1.06	1.13
(30, 100)	1.11	1.19	1.08	1.15
(50, 100)	1.14	1.23	1.12	1.19
(30, 200)	1.15	1.24	1.13	1.20
(50, 200)	1.21	1.29	1.2	1.25

同时，调度窗口 SW 的大小也影响算法 RBS 的性能。调度窗口 SW 越小，算法 RBS 的性能越好。对预测窗口的大小为 500 的情况，当 SW 从 500 减至 250 时，算法 RBS 平均改进了 9%。对预测窗口大小为 1000 的情况，当 SW 从 500 减至 250 时，算法 RBS 平均改进了 6.8%。原因在于调度窗口 SW 越小，窗口迭代的程度越高，部分工序利用新的信息重新调度，性能更好。但是 SW 越小，子问题求解的个数增加，计算量也相应增加。因此需要选择合适的预测窗口和调度窗口的大小，使得算法 RBS 可以在调度性能和计算时间获得一个好的均衡。

在仿真的过程，我们发现对同一实例，预测窗口和调度窗口的大小不同，子问题中辨识的瓶颈也不同。这与我们之前的分析一致。由于时间窗口大小的不同，分配至各个子窗口的工序也不同，造成子窗口内机器上的加工负荷分配也不同，从而影响瓶颈机的选择。滚动瓶颈分解方法中的瓶颈是对具体时间段而言，它与整个调度时域的长期瓶颈意义不一样。该瓶颈机总的加工能力是足以完成其总的加工任务，只是在某个时间段由于工序过于集中使得它们无法在规定的时间段内完成。滚动瓶颈分解方法中瓶颈的辨识与工件的工艺路径和车间的调度决策有关。这一现象在实际生产调度中非常有意义。如果我们可以根据实际生产调度的决策时域确定预测时域的长度，则可以保证滚动瓶颈分解策略总是把重心放在该时段中关键的机器上。

滚动瓶颈分解方法为求解超大规模的 Job shop 调度问题提供了一种有效的解决思路。下面我们尝试将该方法应用于一类更复杂生产线中，即半导体生产线，研究该方法处理复杂生产调度问题的可行性。

6.3 半导体生产线滚动瓶颈分解方法研究

6.3.1 半导体生产调度问题一般描述

半导体生产线又称大规模集成电路制造生产线，它是目前制造业中最复杂的生产过程。该过程主要分为四个阶段：晶圆制作(wafer fabrication)，晶圆探测(wafer probe)，晶圆装配/封装(assembly or packaging)，以及终端产品检测(final testing)，其中晶圆制作过程是所有阶段中资金最密集、技术最复杂以及调度最困难的阶段。本节主要研究晶圆制造生产过程的调度问题。

不同于传统的 Flow shop 和 Job shop 问题，晶圆制造生产过程有其独特的生产工艺特点[55, 56]：

- 重入生产流：半导体生产线中的机器极其昂贵，几乎占所有建厂投入成本75%的费用。由于工艺要求，某些操作需要重复多次，这样晶圆可能会多次访问同一台机器，称之为重入加工流。例如一个晶圆必须多次访问光刻加工设备来完成多层电路制备。
- 生产设备多样性：按同时可加工的晶圆个数可分为单晶圆(wafer)加工设备(如光刻设备)，单批(lot)加工设备(如清洗槽)，和多批(batch)加工设备(如氧化扩散高温炉)。而且有的加工设备(如离子注入设备)需要额外的安装时间，安装时间的大小与工件的加工顺序有关(sequence-dependent setup times)。
- 生产规模很大，加工路径复杂：一般半导体生产线上流动着成百上千个工件，它们需要在 50-100 个机器组上加工近 200 至 400 道操作。而且每类产品的工艺路径可能也各不相同。

因此，半导体生产过程可描述为一个复杂的 Flow shop 或 Job shop 调度问题[54]：

(1) $FF_m \mid r_j, s_{i,j}, B, recrc \mid f$ (专用半导体生产线)

(2) $FJ_m \mid r_j, s_{i,j}, B, recrc \mid f$ (代工厂半导体生产线)

令 $J=\{1,\cdots,n\}$ 为工件集，$M=\{1,\cdots,m\}$ 为机器集，$N=\{0,1,\cdots,o,n_1,\cdots,n_n,*\}$ 为工序集(包括虚拟起点0，虚拟终点*，o 个工序节点和 n 个虚拟完工节点 v_i)。p_j 表示工序 j 的加工时间，d_i 表示工件 i 的交货期限，q_k 表示批处理机 k 的加工时间，s_{ij} 表示工序 i 加工完之后，在加工工序 j 之前所需安装时间，它们都是已知且固定不变。图 6-4 给出一个典型半导体生产调度问题的析取图描述 $G(N,A,E)$。在此析取图中，每道工序上的标号 i,j 表示该工序是工件 i 在机器 j 上的操作。每个节点出弧的长度等于

该节点对应工序的加工时间的大小。

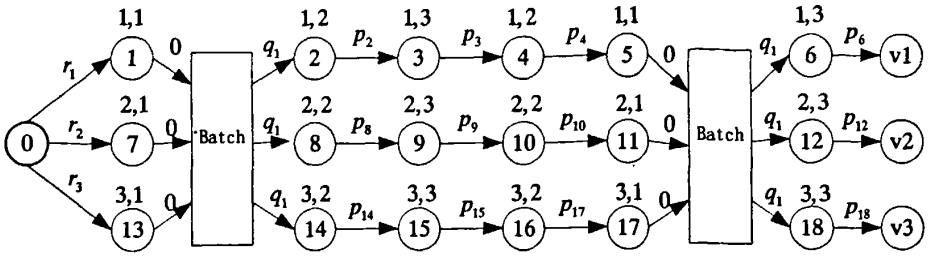


图 6-4. $FF_m\,|r_j,s_{i,j},B,recrc\,|TWT$ 调度问题析取图描述(3×3 例子)

Fig. 6-4. Disjunctive graph for problem $FF_m\,|r_j,s_{i,j},B,recrc\,|TWT$ (3×3 instance)

该析取图对半导体生产线的加工特性做了相应的描述，具体如下[66]：

(1) 批处理机描述：如图 6-4 所示，机器组 1 为一台批处理设备，每台机器可以同时加工的工件个数最多为 b (最大批次)个。令可组批的工件个数为 n，则所有可能的批组合个数 n_b 为：

$$n_b = \sum_{k=1}^{b} C_n^k, \quad C_n^k = \frac{n!}{b!(n-b)!} \tag{6-3}$$

图 6-5 给出了具体的批组合的方式，其中工件 1 和工件 2 是同类产品，可以一起组合加工。当批处理机未调度时，工件批组合的方式也未确定，因此在构造初始析取图时，考虑所有可能的批组合节点，在析取图中添加 n_b 个虚拟的批组合节点。连接工序节点和该工序对应的批组合节点，从工序节点到批组合节点的弧长为 0，从批组合节点到该工序节点的后序节点的弧长为批处理机的加工时间 q_1。

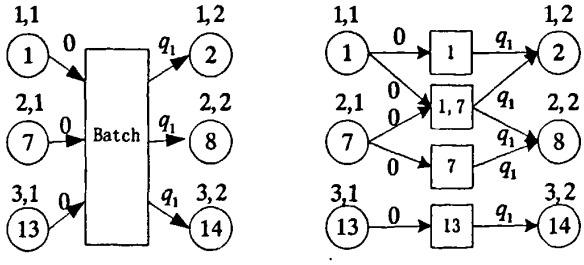


图 6-5. 批加工机器组 1 可能的批组合方式

Fig. 6-5. Potential batching configurations for batching tool group 1

一旦批处理机调度之后，确定批组合方式和批加工顺序。保留确定的批组合节点，从析取图中删除其余的批组合节点以及这些批组合节点与其它节点的顺序弧。

(2) 顺序相关安装时间机器描述：如图 6-4 所示，机器组 3 上工件加工需要考虑顺序相关的安装时间。在构造初始析取图时，由于机器上工件加工顺序未知，从而无法确定安装时间大小，该机器上工序节点出弧的长度为该工序的加工时间。一旦该机器上的加工顺序确定后，添加对应的安装时间至相应的顺序弧。

(3) 并行机描述：如图 6-4 所示，机器组 2 为并行机。在构造初始析取图时，由于机器上加工顺序未知，该机器组上加工的工序之间存在析取弧。当机器组 2 上加工顺序确定之后，如果两个工序在机器组同一台机器上加工，则按加工顺序固定这两个工序的析取弧对的方向；如果它们在不同的机器上加工，则从析取图中完全删除该析取弧对。

(4) 重入加工流描述：由图 6-4 可以看出，在机器组 1 上加工的工序分为 1, 5, 7, 11, 13 和 17，其中工序 1 和 5 都属于工件 1，工序 7 和 11 属于工件 2，工序 13 和 17 属于工件 3。在构造初始析取图时，属于同一工件的工序之间存在顺序关系，它们之间没有析取弧对。由于重入加工流的存在，该机器上所有工序的析取弧不能构成一个圈。

由于所有可能的批组合个数为 $O(n^b)$，随着工件个数 n 的增加，析取图中的批节点个数呈指数增长，对应的顺序弧也指数增长。而且重入加工流特性使得加工工序的个数也有所增加，析取图中析取弧随工序个数的增加呈指数增长。即使对一个简单的半导体生产线(如图 6-4 描述)，精确算法可求解最大规模问题中工件个数不超过 7 个[54]。Uzsoy 等最早应用移动瓶颈方法调度半导体生产线，目标函数选择最小化最大滞后时间(L_{max})[64,65]。Mason 等提出了一种改进移动瓶颈方法求解半导体车间层调度问题，目标函数选择最小化加权拖期时间之和(TWT)[66-69]。然而传统的机器层分解方法对所有的机器均详细调度，需要耗费大量的计算时间。因此，有必要提出一种有效的算法在较快的时间获得较好的调度。

6.3.2 半导体生产调度问题滚动瓶颈分解算法

类似上一节提出的滚动瓶颈分解算法(算法 6-2)，我们对半导体生产调度问题提出了一种滚动瓶颈分解算法。算法分两步进行：

首先，时域分解。将整个调度时域分解为若干时间段，每个子问题对应一个时间窗口，对每个时间窗口建立子模型，即对该时间窗口内的工序进行析取图建模。通过时间层分解，问题规模降低，而且存储空间大大降低。

然后，瓶颈分解。对每个时间窗口内的子问题，辨识瓶颈机，松弛其它机器的能

力约束，建立瓶颈机调度子问题。非瓶颈机采用有效的分配规则计算，通过调整瓶颈机上工序的到达时间和交货期限来协调瓶颈机与非瓶颈机之间的关联。

在滚动瓶颈分解算法中，每个时间窗口辨识的瓶颈机可能不同。由于半导体生产线上机器的加工类型不一，每台机器对应的子问题也可能不同。对此，我们对不同的子问题给出了相应的求解算法。结合一个例子，我们具体研究滚动瓶颈分解方法如何求解半导体生产调度问题。

目前对半导体生产调度问题的算法研究大多针对一个典型的半导体模型进行测试比较研究，该模型只包括 3 个加工中心，6 道加工步骤，但它具有半导体生产线的复杂特点，称之为微小半导体模型(MINIFAB)(见图 6-6)[126]。每个工件流经每个加工中心两次，具有重入加工流特性。机器组 1 和 2 包括两台并行机，而且机器组 1 中的机器为批处理机，它可以同时加工 3 个工件。在步骤 1 任何产品类型的工件都可以组合在一起进行批加工，而步骤 5 中只有同类的工件才能组合进行批处理。机器组 3 需要顺序相关的安装时间，而且安装时间的大小与工件类型和加工步骤有关。当机器组 3 需要对一个新产品类型的工件加工时，安装时间为 5 分钟；当它需要对一个不同加工步骤的工件加工时，安装时间 10 分钟，如果待加工的工件与前一个工件加工类型和步骤均不同，则需要耗费安装时间 12 分钟。

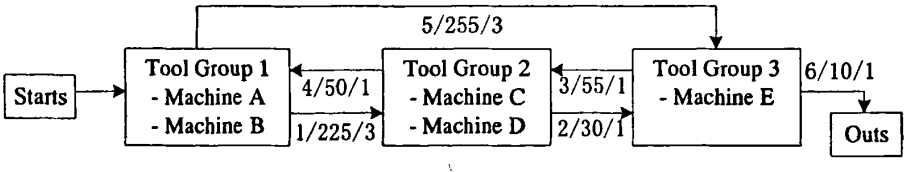


图 6-6. 微小半导体模型(弧线标号表示加工步骤/加工时间/最大批次)

Fig. 6-6. MINIFAB model (Arc labels indicate process step number / processing time / lot size)

(1) 如果辨识的瓶颈机为机器组 1，则瓶颈子问题可描述为 $P_2|r_j,B,recrc|TWT$。它包括两个决策过程：批组合和批调度。目前主要有两种启发式求解方法。

一种方法为基于 ATC 规则的满批调度。令子窗口 l 内机器组 1 上的工序个数为 n_1^l，在决策时刻 t，对子窗口 l 内机器组 1 上的每道工序 j 计算其 ATC 值[39]：

$$I_j(t)=\frac{w_j}{p_j}\exp\left(-\frac{(d_j^l-p_j+(r_j^l-t))^+}{K\cdot\bar{p}}\right) \tag{6-4}$$

其中子窗口 l 内工序 j 的到达时间 r_j^l 和交货期限 d_j^l 分别按公式(6-1)和(6-2)估计。$\bar{p}$ 为工序的平均加工时间，K 为一固定参数。当机器组 1 上某台机器可用时，按工序

的 ATC 值从高到低依次选择 $\min(b,n_1^l)$ 个工序组成一个批 b_k，批 b_k 的到达时间 $r_{b_k}^l$ 为该批中所有工序到达时间的最大值，加工时间 p_{b_k} 为该批中所有工序加工时间的均值，交货期限 $d_{b_k}^l$ 为该批中所有工序交货期限的最小值，即

$$r_{b_k}^l = \max_{j\in b_k} r_j^l,\quad p_{b_k} = \sum_{j\in b_k} p_j / |b_k|,\quad d_{b_k}^l = \min_{j\in b_k} d_j^l \tag{6-5}$$

另一种方法为基于 BATC 规则的调度。该方法对子窗口 l 内机器组 1 上的工序构造所有可能的批组合，批组合的个数按公式(6-3)计算。当 t 时刻当子窗口 l 内机器组 1 上的某台机器可用，对所有批 b_j 计算其 BATC 值[66]：

$$I_{b_j} = \frac{w_{b_j}}{p_{b_j}} \exp\left(-\frac{(d_{b_j}^l - p_{b_j} + (r_{b_j}^l - t))^+}{K\cdot\bar{p}}\right)\cdot\frac{|b_j|}{b} \tag{6-6}$$

其中子窗口 l 内批 b_j 的到达时间 $r_{b_k}^l$ 和交货期限 $d_{b_k}^l$ 按公式(6-5)计算。当机器组 1 上某台机器可用时，从未调度的批中选择 BATC 值最大的批 b_k，在 $\min(r_{b_k}^l, t)$ 时刻在该机器上加工批 b_k。

(2) 如果辨识的瓶颈机为机器组 2，则瓶颈子问题可描述为 $P_2|r_j, recrc|TWT$。采用基于 ATC 规则的列表调度求解此问题。该方法对子窗口 l 内机器组 2 上的每道工序 j 按公式(6-4)计算其 ATC 值，当机器组 2 上某台机器空闲，从未调度工序中选择 ATC 值最大的工序 j^*，在 $\min(r_j^l, t)$ 时刻在此机器上加工工序 j^*，其中子窗口 l 内工序 j 的到达时间 r_j^l 按公式(6-1)计算。

(3) 如果辨识的瓶颈机为机器组 3，则瓶颈子问题可描述为 $1|r_j, s_{ij}, recrc|TWT$。在决策时刻 t，对子窗口 l 内机器组 3 上的每道工序 j 计算其 ATC 值[127]：

$$\cdot\ I_j(t,l) = \frac{w_j}{p_j}\exp\left(-\frac{(d_j^l - p_j + (r_j^l - t))^+}{k_1\cdot\bar{p}}\right)\exp\left(-\frac{s_{b_i,b_j}}{k_2\cdot\bar{s}}\right) \tag{6-7}$$

其中 $\bar{s}$ 为平均安装时间，k_1 和 k_2 为一固定参数。

当机器 3 空闲时，从未调度工序中选择 ATC 值最大的工序 j^*，在 $\min(r_j^l, t)$ 时刻在机器 3 上加工工序 j^*，其中子窗口 l 内工序 j 的到达时间 r_j^l 按公式(6-1)计算。

6.3.3 例子及仿真测试

为了研究滚动瓶颈分解算法如何求解半导体生产调度问题，我们以一个 3 个工件的 MINIFAB 模型为例。工件的加工特性如表 6-6 所示。

表 6-3. 3 个工件的 MINIFAB 模型

Table 6-3. Three-job instance of a MINIFAB model

Job	Type	Weight	Release time	Due date
1	A	10	0	650
2	A	7	100	675
3	B	3	50	625

按图 6-4 的方式建立该问题的析取图模型，令预测窗口 PW=450，SW= 450，该问题分解为 2 个时间窗口(如图 6-7 所示)。

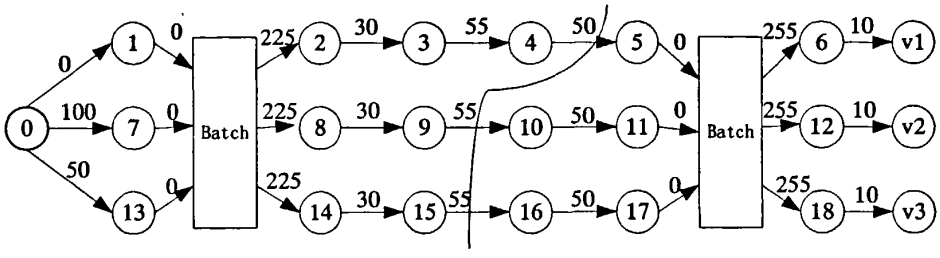


图 6-7. 调度问题析取图描述(3 个工件的例子)

Fig. 6-7. Disjunctive graph for the instance with 3 jobs

第一个子窗口内工序为：1，2，3，4，7，8，9，13，14 和 15。它仍可描述为一个带动态到达时间、顺序相关安装时间、批加工和重入加工流约束的柔性 Flow shop 调度问题 $FF_m \mid r_j, s_{i,j}, B, recrc \mid \sum \omega_j T_j$。按下式计算子窗口 l 内每个加工中心 k 的平均加工负荷 WL_k^l：

$$WL_k^l = \sum_{j \in O_k^l} p_j / (b \times m_k) \tag{6-8}$$

其中 O_k^l 表示子窗口 l 内加工中心 k 的工序集，m_k 表示加工中心 k 中机器的个数。b 为机器同时可加工工件的最大批次。加工中心 1，2 和 3 的平均加工负荷分别为 112.5，70 和 165。选择平均加工负荷最大的机器为瓶颈机，即加工中心 3 为瓶颈机。

按(6-1)和(6-2)估计加工中心 3 上加工工序 3，9，15 的到达时间和交货期限，得 $r_3^1 = r_9^1 = r_{15}^1 = 355$，$d_3^1 = 335$，$d_9^1 = 360$，$d_{15}^1 = 310$。令 $k_1 = 8$，$k_2 = 1$，$\bar{p} = 55$，$\bar{s} = 13.5$，在决策时刻 $t = 0$ 按公式(6-7)计算每道工序的 ATC 值，按 ATC 值从大到小的顺序依次排列。得到子窗口 1 内加工中心 3 上工序的加工顺序为：3→9→15。

基于加工中心 3 的调度结果，更新析取图，按公式(5-4a)和(5-4b)估计子窗口 1 内加工中心 1 和 2 上工序的到达时间和交货期限。对加工中心 1 采用 BATC 规则调

度，得到子窗口 1 内加工中心 1 的调度结果为：(1,13),7，即工序 1 和工序 13 构成一个批在加工中心 1 上的机器 A 上加工，工序 7 在在加工中心 1 上的机器 B 上加工。对加工中心 2 采用 ATC 规则调度，得到子窗口 1 内加工中心 2 的调度结果为：2→8，14→4，其中工序 2 和 8 在加工中心 2 上的机器 C 上加工，工序 14 和 4 在加工中心 2 上的机器 D 上加工。执行在[0, 450]窗口内开工的工序，按图 6-8(a)的方式更新析取图。更新当前决策点 $t=\max_{i\in J} C_i=475$。

第 2 个子窗口内工序为：5，6，10，11，12，16，17，18。由于步骤 5 中只有同类的工件才能组合进行批加工，此时批处理机具有不相容性(imcompatible)，它可描述为一个柔性 Flow shop 调度问题 $FF_m \mid r_j, s_{i,j}, B, recrc, imcompatible \mid \sum \omega_j T_j$。

修正子窗口 2 内工序 O_{ij} (工件 i 第 j 道工序)的到达时间 $r_{i,j}$，它为其前道工序完工时间 $C_{i,j}$ 和它可用机器的完工时间的最大值，即

$$r_{i,j}=\max(C_{i,j-1}, \min_{k\in m_{ij}} T_k^1) \tag{6-9}$$

其中 m_{ij} 表示工序 O_{ij} 可用的加工中心标号，T_k^1 表示加工中心 m_{ij} 上机器 k 在子窗口 1 内的完工时间。

按公式(6-8)计算子窗口 2 内加工中心 1，2 和 3 的平均加工负荷，它们分别为 127.5，50 和 30。选择平均加工负荷最大的机器为瓶颈机，得到加工中心 1 为瓶颈机。

按(6-9)和(6-2)估计加工中心 1 上加工工序 5，11，17 的到达时间和交货期限，得 $r_5^2=410$，$r_{11}^2=465$，$r_{17}^2=525$；$d_5^2=385$，$d_{11}^2=410$，$d_{17}^2=360$。令 $K=1$，$\bar{p}=255$，在决策时刻 t 按公式(6-6)计算每个批组合的 BATC 值，选择 BATC 值最大的批最先加工排列。得到子窗口 2 内加工中心 1 上的调度结果为：(5,11),17。即工序 5 和工序 11 构成一个批在加工中心 1 上的机器 A 上加工，工序 17 在在加工中心 1 上的机器 B 上加工。

基于加工中心 1 的调度结果，更新析取图，按公式(5-4a)和(5-4b)估计子窗口 2 内加工中心 2 和 3 上工序的到达时间和交货期限。对加工中心 2 采用 ATC 规则调度，得到子窗口 2 内加工中心 2 的调度结果为：10，16。其中工序 10 在加工中心 2 上的机器 C 上加工，工序 16 在加工中心 2 上的机器 D 上加工。。对加工中心 3 采用 ATC 规则调度，得到子窗口 2 内加工中心 3 的调度结果为：6→12→18。执行在[475, 925]窗口内开工的工序，按图 6-8(b)的方式更新析取图。得到整个 MINIFAB 模型的调度结果。其中每个工件的完工时间分别为 $C_1=742$，$C_2=752$，$C_3=795$，目标值为：$\sum w_j T_j=10(742-650)+7(752-675)+3(795-625)=1969$。，该结果与文献[68]中移动

瓶颈方法整体求解得到目标值一样。

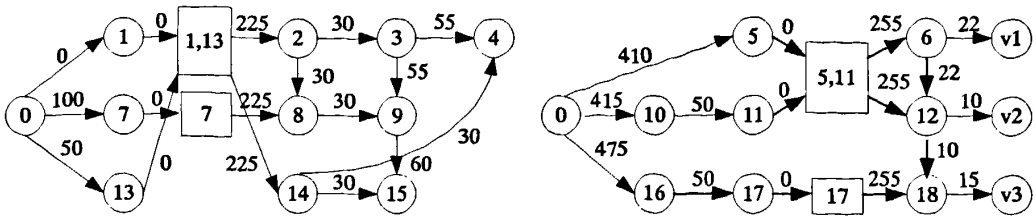


图 6-8. (a) 子窗口 1 调度析取图　　(b) 子窗口 2 调度析取图

Fig.6-8. (a) Disjunctive graph of subproblem 1　　(b) Disjunctive graph of subproblem 2

由此例子我们可以看出每个子窗口辨识的瓶颈机不同。在子窗口 1 中，由于该时间窗口内机器组 3 上的加工负荷较为集中，它成为该时间段内的瓶颈。而在子窗口 2 中，机器组 1 的加工负荷最严重，该时间段内的瓶颈为机器组 1。这一现象与 6.2 节中采用滚动瓶颈分解方法处理大规模 Job shop 调度问题的现象是一致。采用滚动瓶颈分解方法总是把调度的重心放在该时段中关键的机器上。

下面我们将通过大量的仿真测试来研究滚动瓶颈分解算法的性能，按文献[66]的方式生成测试数据：工件个数 n 分别取 3～12。50％的工件到达时间为零，即 $r_j=0$；50％的工件到达时间服从[1,300]均匀分布，即 $r_j \sim U[1,300]$。而且工件的权重服从[1, 10]均匀分布，即 $w_j \sim U[1,10]$，工件的交货期限服从[250, 1000]均匀分布，即 $d_j \sim U[250,1000]$。如果 $w_j>5$，则工件 j 为产品 A；而当 $w_j \le 5$，则工件 j 为产品 B。两类产品的工艺路径如图 6-6 所示。每种类型分别生成 10 组数据，共 100 组算例。

滚动瓶颈分解算法中预测窗口和调度窗口分别取 800 和 500，该算法与四种规则调度(FIFO, EDD, WSPT, CR)进行比较[39]。采用目标值与最好值相对比值的平均值 ρ 来评价算法的调度质量[95]，令 $TWT(H,I)$ 为给定实例 I 下采用启发式算法 H 求得的目标函数值，所有算法中最好值 $BEST(I)=\min_H\{TWT(H,I)\}$。定义一组满足相同特征的问题 S，对每类问题 S 计算目标值与最好值平均比值为：

$$\rho(H,S)=\sum_{I\in S}TWT(H,I)/\sum_{I\in S}BEST(I)$$

计算时间的评价指标采用平均 CPU 计算时间 $ACT(H,S)$。所有的算法均采用 C 语言编程，运行在 Celeron 处理器(CPU 速度 2GHZ，缓存 128kB，内存 256MB)的机器上。表 6-4 给出了不同问题规模下调度算法的性能比较值。

表 6-4. 不同问题规模下算法的性能比较

Table 6-4. Performance of the algorithms for different problem sizes

Problem S	Heuristic H				
(n)	FIFO	EDD	WSPT	CR	RBS
3	1.46	1.49	1.22	1.11	1.00
4	1.35	1.36	1.35	1.36	1.03
5	1.25	1.46	1.26	1.64	1.02
6	1.42	1.27	1.3	1.28	1.05
7	1.52	1.32	1.34	1.33	1.00
8	2.24	1.8	1.81	1.95	1.00
9	1.37	1.12	1.06	1.04	1.00
10	1.85	1.42	1.35	1.30	1.00
12	2.15	1.46	1.61	1.57	1.00
13	1.98	1.40	1.40	1.38	1.00

由表 6-4 可以看出滚动瓶颈分解方法比规则调度的性能好，它能在较快的时间内获得一个较好的调度性能，该方法可以应用于实际半导体生产调度问题。

在仿真中，我们发现机器组 1 成为瓶颈机的次数多于其它机器。这是由于批处理机的加工时间远比其它机器的加工时间长，它对整个半导体生产线的性能影响最大，因此有必要进一步研究批处理机的调度，找到一种更有效的调度算法。

对更复杂的半导体生产线，不同子窗口下辨识的瓶颈机可能不同，它可处理暂时瓶颈发生移动的问题。而且滚动瓶颈分解方法采用滚动机制执行调度决策，它还可以处理动态不确定问题。

6.4 本章小结

约束调度算法由于内存限制无法求解更大规模 Job shop 调度问题，对此本章从空间层和时间层对大规模问题进行分解，提出了一种滚动瓶颈分解方法。该方法将整个调度时域分为若干决策子窗口，对每个子窗口建立数学模型。时域分解后对应的子问题仍是一个 Job shop 问题，对此采用前面提出的瓶颈分解求解子问题。大量的仿真结果显示对约束调度算法(算法 5-1)无法求解的大规模问题，滚动瓶颈分解可以获得较好的解。而且该方法可以处理瓶颈机随时间发生移动的情况。

随后将此方法应用至一个半导体生产线。针对半导体生产线加工特性建立了半导体调度析取图描述。由于半导体生产线上机器加工类型不一，对分解后不同类型的子问题给出了相应的求解算法。该方法最后应用于一个典型半导体生产调度问题(MINIFAB)，仿真验证了该算法的有效性。

第七章 总结与展望

在经济全球化的今天，现代制造业面临着前所未有的竞争压力。制造规模和复杂程度都在不断提高，客户需求日趋多样化，这对制造方提出了新的要求。大规模复杂生产调度问题成为当前调度研究的一个热点。

分解方法是处理大规模复杂调度问题的一种常用方法，但传统的分解方法将每个子问题视为均等，容易陷入局部最优。虽然目前已有一些瓶颈调度方法相继提出，但处理的方式过于简单，在建模、算法设计和应用方面都存在很大的研究空间。本文从全局优化的角度出发，利用瓶颈思想，对大规模复杂生产调度问题进行了深入的研究，得到了如下成果：

- 在大系统分解思想的指导下，给出了车间调度问题机器层分解方法的一般描述，并指出传统分解方法在求解大规模调度问题时可能存在的不足。利用能力不均衡生产线存在瓶颈的特性，剖析瓶颈思想的内在机理，结合大系统分解思想提出了瓶颈分解方法的一般框架，并指出该方法的关键技术。

- 针对单瓶颈 Flow shop 调度问题，提出了一种单瓶颈分解算法。该方法将 Flow shop 上的机器分为瓶颈机、上游非瓶颈机和下游非瓶颈机。松弛非瓶颈机加工能力约束，原 Flow shop 问题简化为一个单机调度问题。瓶颈机的调度可以通过优化求解该简化问题获得。根据瓶颈机的调度结果，上游非瓶颈机采用有效的分派规则从后往前依次调度，而下游非瓶颈机则采用规则从前往后依次调度。随后我们给出了非瓶颈机冗余能力的条件，证明满足此条件下的 Flow shop 问题的最优调度为排列调度，即非瓶颈机上工件的加工安排与瓶颈机的调度安排一致。大量的仿真结果显示单瓶颈分解算法可以在较快的时间内获得较好的解。

- 针对单瓶颈 Job shop 调度问题，提出了一种单瓶颈分解算法。该方法利用生产线主导瓶颈的特点将 Job shop 中的机器分为瓶颈机和非瓶颈机。假设非瓶颈机加工能力无限大，将非瓶颈机上的工序集结成一个时间迟延环节，原问题则简化为一个带到达时间和传递时间约束的单机调度问题。瓶颈机上工件的加工安排可以通过优化求解此单机问题获得，而非瓶颈机则采用有效的启发式算法调度。通过调整瓶颈机上工序的到达时间和传递时间协调瓶颈机与非瓶颈机之间的关联。仿真结果显示该算法求解瓶颈程度较高的 Job shop 问题非常有效。

- 研究带多约束机的 Job shop 调度问题，提出了一种约束调度算法。该算法给出了一种迭代辨识约束机(包括长期瓶颈和暂时瓶颈)的方法。每次迭代，采用有效的规则调度非约束机，并根据调度结果从中辨识约束机，该过程迭代进行直至不存在新的约束机。约束机模型可描述为一个带时间迟延的多机调度问题，该问题的最优解是原问题最优解的一个下界。随着约束机不断辨识，相应的约束机模型也迭代更新，我们证明约束机子问题的最优解在迭代过程中不断提高，朝着原问题最优解的方向逐步逼近。与传统的移动瓶颈方法相比，约束调度算法在提高计算效率的同时保证了调度的优化性能。大量的仿真结果显示，该算法可以在调度质量和计算时间之间获得一个好的均衡。

- 针对超大规模的 Job shop 调度问题，提出了一种滚动瓶颈分解方法。该方法从空间和时间层对大规模问题进行分解，按滚动机制执行瓶颈分解算法，可避免内存不足带来问题求解困难。而且不同时间子窗口内辨识的瓶颈机也不同，它可以处理暂时瓶颈移动的问题。随后将滚动瓶颈分解方法应用至半导体生产线，针对半导体生产线的机器加工类型不一，对分解后不同类型的子问题给出了相应的求解算法，并对 MINFAB 模型进行详细的仿真测试研究。

本文提出了一种新的瓶颈分解方法，并在大规模复杂调度问题中进行了一些尝试和探索，也获得一些成果。但是瓶颈分解方法还有很多问题值得深入研究下去，主要有以下几个方面：

- 将单瓶颈分解方法应用于更复杂的加工环境，如柔性 Flow shop 调度问题、柔性 Job shop 调度问题以及装配生产线调度问题中。

- 在第六章中滚动瓶颈分解方法在半导体生产线的研究只是对 MINFAB 模型进行，SEMATECH 研究组提供了一组描述“实际”半导体生产线的测试数据，进一步的工作可以对 SEMATECH 数据进行详细仿真测试，研究滚动瓶颈分解方法求解大规模复杂半导体生产线问题。

- 本论文研究的瓶颈分解方法主要是针对大规模静态调度问题，瓶颈子问题中的关联参数是对加工波动的估计。而生产加工中的动态不确定性也可以通过关联参数的估计获得，该方法可推广至求解动态不确定性车间调度问题。

参考文献

[1] Karger D., Stein C., Wein J., Scheduling algorithms, Handbook of Algorithms and Theory of Computation, 2001, 1-52

[2] Bellman R., Esogbue A.O., Nabeshima I., Mathematical aspects of scheduling and applications, New York: Pergamon Press, 1982

[3] Garey M.R., Johnson D.S., Computers and intractability: A guide to the theory of NP-completeness, San Francisco, CA: W. H. Freeman, 1979

[4] Johnson S.M., Optimal two- and three-stage production schedules with setup times included, Naval Research Logistics Quarterly, 1954, 1:61-68

[5] Peter Brucker, Scheduling algorithm (3rd Edition), Springer-Verlag, New Youk, Inc, 2001

[6] Pinedo M., Scheduling: Theory, algorithms, and systems (2nd Edition), Englewood Cliffs, NJ: Prentice Hall, 2002

[7] 唐恒永，赵传立，排序引论，北京：科学出版社，2002

[8] Graham R.L., Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Optimization and approximation in deterministic sequencing and scheduling: A survey, Annals of Discrete Mathematics, 1979, 5:287-326

[9] Wagner H., An integer linear programming model for machine scheduling, Naval Research Logistics Quarterly, 1959, 6:131-140

[10] Garey M.R., Johnson D.S., Sethi R., The complexity of flow shop and job shop scheduling, Mathematics of Operations Research, 1976, 1:117-129

[11] Monma C.L., Rinnooy Kan A.H.G., A concise survey of efficiently solvable special cases of the permutation flow-shop problem, RAIRO Recherche Operationelle, 1983, 17:105-119

[12] Gonzalez T., Sahni S., Flow shop and job shop schedules: Complexity and approximation, Operation Research, 1978, 26:26-52

[13] Du J., Leung J.Y., Minimizing mean flow time with release time and deadline constraints, Journal of Algorithms, 1993, 14:45-68

[14] Palmer D.S., Sequencing jobs through a multi-stage process in the minimum total time-a quick method of obtaining a near optimum, Operations Research Quarterly, 1965, 16:101-107

[15] Campbell H.G., Dudek H.G., Smith R.A., A heuristic algorithm for the n-jobs, m-machine sequencing problem, Management Science, 1970, 16:630-637

[16] Gupta J.N.D., Heuristic algorithm for multi-stage flow shop scheduling problem, AIIE Transaction, 1972, 4:11-18

[17] Baker K.R., A comparative study of flow-shop algorithm, Operation Research, 1975, 23: 62-73

[18] Dannenbring D.G., Evaluation of flow shop sequencing heuristics, Management Science, 1977, 23:1174-1182

[19] Widmer M., Hertz A., A new heuristic method for the flow shop sequencing problem, European Journal of Operational Research, 1989, 41:186-193

[20] Taillard E., Some efficient heuristic methods for the flow shop sequencing problem, European Journal of Operational Research, 1990, 47:65-74

[21] 李霄峰、邵惠鹤、任德祥，求解混合 Flow shop 调度问题的简化禁忌搜索方案，上海交通大学学报，2003, 37:48-51

[22] 陈雄、杨凤霞、吴启迪，Flow shop 调度问题的自适应模拟退火算法，控制理论与应用，2003, 20:131-134

[23] 庞哈利、郑秉霖，多阶段混合 Flow shop 调度问题及其遗传求解算法，控制与决策，1999, 14:86-89

[24] Feng Y., Feng Z., Ant colony system hybridized with simulated annealing for flow-shop scheduling algorithms, WSEAS Transactions on Business and Economics, 2004, 1:133-138

[25] Zheng D.-Z., Wang L., An effective hybrid heuristic for flow shop scheduling, The International Journal of Advanced Manufacturing Technology, 2004, 21:38-44

[26] 徐震浩、顾幸生，用混合算法求解 Flow shop 调度问题，华东理工大学学报，2004, 30:114-118

[27] Akers S.B., A graphical approach to production scheduling problems, Operations Research, 1956, 4:244-245

[28] Jackson J.R., An extension of Johnson's result on job lot scheduling, Naval Research Logistics Quarterly, 1956, 3:201-203

[29] Hefetz N., Adiri I., An efficient optimal algorithm for the two-machines unit-time job-shop schedule-length problem, Mathematics of Operations Research, 1982, 7:354-360

[30] Jain A.S., Meeran S., Deterministic job-shop scheduling: past, present and future, European Journal of Operational Research, 1999, 13:390-434

[31] Roy B., Sussmann B., Les problems d'ordonnancement avec constraints disjonctives, Note D.S., SEMA, Paris, France, 1964, 9

[32] Balas E., Machine scheduling via disjunctive graphs: An implicit enumeration algorithm, Operations Research, 1969, 17:941-957

[33] Carlier J., Pinson E., An algorithm for solving the job shop problem, Management Science, 1989, 35:164-176

[34] Carlier J., Pinson E., A practical use of Jackson's preemptive schedule for solving the job-shop problem, Annals of Operations Research, 1990, 26:269-287

[35] Applegate D., Cook W., A computational study of the job-shop scheduling problem, ORSA Journal on Computing, 1991, 3:149-156

[36] Giffler B., Thompson G.L., Algorithms for solving production scheduling problems, Operations Research, 1960, 8:487-503

[37] Panwalkar S.S., Iskander W., A survey of scheduling rules, Operations Research, 1977, 25:45-61

[38] Blackstone J.H., Philips D.T., Hogg G.L., A state-of-the-art survey of dispatching rules for manufacturing job shop operations, International Journal of Production Research, 1982, 20:27-45

[39] Vepsalainen A.P.J., Morton T.E., Priority rules for job shops with weighted tardiness costs, Management Science, 1987, 33:1035-1047

[40] Adams J., Balas E., Zawack D., The shifting bottleneck procedure for job shop scheduling, Management Science, 1988, 34:391-401

[41] Morton T.E., Pentico D.W., Heuristic scheduling systems, Wiley Series in Engineering and Technology Management, Wiley, New York, 1993

[42] Sabuncuoglu I., Bayiz M., Job shop scheduling with beam search, European Journal of Operational Research, 1999, 118:390-412

[43] Golver F., Greenberg H.J., New approaches for heuristic search: A bilateral linkage with artificial intelligence, European Journal of Operational Research, 1989, 39:119-130

[44] Vaessens R.J.M., Aarts E.H.L., Lenstra J.K., Job-shop scheduling by local search, INFORMS Journal on Computing, 1996, 8:302-317

[45] 张长水，阎平凡，解 Job-shop 调度问题的神经网络方法，自动化学报，1995, 21:706-712

[46] [46] Alexander S.M., Expert system for the selection of scheduling rules in a job-shop, Computers and Industrial Engineering, 1987, 12:167-171

[47] Yang, S., Dingwei Wang, Constraint satisfaction adaptive neural network and heuristics combined approaches for generalized job-shop scheduling, IEEE Transactions on Neural Networks, 2000, 11:474-486

[48] 陈知美、顾幸生，改进型蚁群算法在 Job shop 问题中的应用，华东理工大学学报，2006, 32:107-111

[49] 方剑、席裕庚，基于遗传算法的 Job shop 静态调度算法，上海交通大学学报，1997, 31:51-54

[50] Aarts E.H.L., Van Laarhooven P.J.M., Lenstra J.K., Ulder N.L.J., A computational study of local search algorithms for job-shop scheduling, ORSA Journal on Computing, 1994, 6:118-125

[51] Nowicki E., Smutnicki C., A fast taboo search algorithm for the job-shop problem, Management Science, 1996, 42:797-813

[52] Sadeh N., Nakakuki Y., Focused simulated annealing search- an application to job-shop scheduling, Annals of Operations Research, 1996, 63:77-103

[53] 王然，吴澄，半导体制造中的车间层控制，信息与控制，1997, 26:192-203

[54] Mason S.J., Minimizing total weighted tardiness in complex job shops, Ph.D. dissertation, Arizona State University, 2000

[55] Uzsoy R., Lee C.Y., Martin-Vega L.A., A review of production planning and scheduling models in the semiconductor industry, Part I: system characteristics, performance evaluation and production planning, IIE Transaction, 1992, 24:47-60

[56] Uzsoy R., Lee C.Y., Martin-Vega L.A., A review of production planning and scheduling models in the semiconductor industry, Part II: shop floor, IIE Transaction, 1994, 26:44-55

[57] Melnyk S.A., Ragatz G.L., Order review/release: research issues and perspectives, International Journal of Production Research, 1989, 27:1081-1096

[58] Spearman M.L., Woodruff D.L., Hopp W.J., CONWIP: a pull alternative to kanban, International Journal of Production Research, 1990, 28:879-894

[59] Wein L.M., Scheduling semiconductor wafer fabrication, IEEE Transactions on Semiconductor Manufacturing, 1988, 1:115-130

[60] Kumar P.R., Scheduling Semiconductor Manufacturing Plants, IEEE Control Systems Magazine, 1994, 14:30-40

[61] Kumar P.R., Scheduling Manufacturing Systems of Re-Entrant Lines, Stochastic Modeling and Analysis of Manufacturing Systems, Springer-Verlag, New York, 1994,325-360

[62] Kumar P.R., Re-entrant Lines, Queuing Systems: Theory and Applications, Special Issue on Queuing Networks, 1993, 13:87-110

[63] Lu S.H., Kumar P.R., Distributed Scheduling Based on Due Dates and Buffer Priorities, IEEE Transactions on Automatic Control, 1991, 36:1406-1416

[64] Uzsoy R., Martin-Vega L.A., Lee C.Y., Leonard P.A., Production scheduling algorithms for a semiconductor test facility, IEEE Transactions on Semiconductor Manufacturing, 1991,4:270-280

[65] Ovacik I.M., Uzsoy R., A shifting bottleneck algorithm for scheduling semiconductor testing operations, Journal of Electronics Manufacturing, 1992, 2:119-134

[66] Mason S.J., Fowler J.W., Carlyle W.M., A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops, Journal of Scheduling, 2002,5:247-262

[67] Mason S.J., Oey K., Scheduling complex job shops using disjunctive graphs: a cycle elimination procedure, International Journal of Production Research, 2003, 41:981-994

[68] Mason S.J., Jin S., Wessels C.M., Rescheduling strategies for minimizing total weighted tardiness in complex job shops, International Journal of Production Research, 2004, 42:613-628

[69] Mason S.J., Fowler J.W., Carlyle W.M., Montgomery D.C., Heuristics for minimizing total weighted tardiness in complex job shops, International Journal of Production Research, 2005, 43:1943-1963

[70] De S., Lee A., Towards a knowledge-based scheduling system for semiconductor testing, International Journal of Production Research, 1998, 36:1045–1073

[71] Azzaro-Pantel C., Floquet P., Pibouleau L., Domenech S., A fuzzy approach for performance modeling in a batch plant: application to semiconductor manufacturing, IEEE Transactions on Fuzzy Systems, 1997, 5:338–357

[72] Zhou M.C., Jeng M.D., Modeling, analysis, simulation, scheduling, and control of semiconductor manufacturing systems: a Petri net approach, IEEE Transactions on Semiconductor Manufacturing, 1998, 11:333–357

[73] Tong L.I., Lee W.I., Su C.T., Using a neural network-based approach to predict the wafer yield in integrated circuit manufacturing, IEEE Transactions Components, Packaging, and Manufacturing Technology, 1997, 20:288-294

[74] Yim S.J., Lee D.Y., Scheduling cluster tools in wafer fabrication using candidate list and simulated annealing, Journal of Intelligent Manufacturing, 1999, 10:531–540

[75] Geiger C.D., Kempf K.G., Uzsoy R., A Tabu search approach to scheduling an automated wet-etch station, Journal of Manufacturing systems, 1997, 16:102–116

[76] Cavalieri S., Crisafulli F., Mirabella O., A genetic algorithm for job-shop scheduling in a semiconductor manufacturing system, IEEE Transactions on Engineering Management, 1999, 41:957–961

[77] Danzig G.B., Wolfe P., Decomposition principle for linear programs, Operations Research, 1960, 8:101-111

[78] Akker van den J.M., Hurkens C.A.J., Savelsbergh M.W.P., Time-indexed formulations for machine scheduling problems: column generation, INFORMS Journal on Computing, 1998, 12:111-124

[79] Barnhart C., Johnson E.L., Nemhauser G.L., Savelsbergh M.W.P., Vance P.H., Brach-and- Price: column generation for huge integer programs, Operations Research, 1998, 36:316-329

[80] Vanderbeck F., Savelsbergh M.W.P., A generic view of Dantzig-Wolfe decomposition for integer programming, Operations Research Letters, 2006, 34:294-306

[81] Brooks R., Geoffrion A., Finding Everett's Lagrange multipliers by linear programming, Operation Research, 1966, 14:1149-1153

[82] Held M., Karp R.M., The traveling salesman and minimum spanning trees, Operation Research, 1970, 18:1138-1162

[83] Held M., Karp R.M., The traveling salesman and minimum spanning trees: Part II, Mathematic Programming, 1971, 1:6-25

[84] Fisher M.L., Optimal solution of scheduling problems using Lagrange multipliers: Part I, Operation Research, 1973, 21:1114-1127

[85] Fisher M.L., Lageweg B.J., Lenstra J.K., Rinnooy Kan A.H.G., Surrogate duality relaxation for job shop scheduling, Discrete Applied Mathematics, 1983, 5:65-75

[86] Luh P.B., Hoitomt.D.J., Scheduling of manufacturing system using the lagrangian relaxation technique, IEEE Transactions on Automatic Control, 1993, 38:1066-1079

[87] Luh P.B., Zhao X., Wang Y., Thakur L.S., Lagrangian relaxation neural networks for job shop scheduling, IEEE Transactions on Robotics and Automation, 2000, 16:78-88

[88] Zhao X., A new generation of optimization algorithms within the lagrangian relaxation approach for job shop scheduling, Ph.D. dissertation, University of Connecticut, 1999

[89] Fisher M.L., The lagrangian relaxation method for solving integer programming problems, Management Science, 2004, 50:1861-1871

[90] Benders J.F., Partitioning procedure for solving mixed-variables programming problems, Numerische Mathematik, 1962, 4:238-252

[91] Ovacik I.M., Uzsoy R., Decomposition methods for complex factory scheduling problems, Boston/Dordrecht/London: Kluwer Academic Publishers, 1997

[92] Fox M.S., Constraint-directed search: A case study of job-shop scheduling, Morgan Kaufmann, 1987

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

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

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

[96] Luh P.B., Chen H.X., An alterative framework to lagrangian relaxation approach for job shop scheduling, Proceedings of the 38th Conference on Decision & Control, Phoenxi, Arizona USA, 1999

[97] Ovacik I.M., Uzsoy R., Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequence-dependent setup times, International Journal of Production Research, 1994, 32:1243-1263

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

[99] Wang B., Xi Y., Gu H., Terminal penalty rolling scheduling based on an initial schedule for single-machine scheduling problem, Computers & Operations Research, 2005, 32:3059-3072

[100] Goldratt E.M., Cox J., The Goal: a Process of Ongoing Improvement, Croton-on-Hudson, NY: North River Press, 1984

[101] Goldratt E.M., Fox R.E., The Race, Croton-on-Hudson, NY: North River Press, 1986

[102] Goldratt E.M., Shifting information out of the data ocean: the haystack syndrome, Croton-on-Hudson, NY: North River Press, 1990

[103] Spearman M.L., On the theory of constraints and the goal system, Production and Operations Management, 1997, 6:28-33

[104] Simons Jr J.V., Simpson III W.P., Carlson B.J., James S.W., Lettiere C.A., Mediate Jr B.A., Formulation and solution of the drum-buffer-rope constraint scheduling problem (DBRCSP), International Journal of Production Research, 1996, 34:2405-2420

[105] Simons Jr J.V., Simpson III W.P., An exposition of multiple constraint scheduling as implemented in the goal system (formerly DISASTER™), Production and Operations Management, 1997, 6:3-22

[106] Simons Jr J.V., Stephens M.D., Simpson III W.P., Simultaneous versus sequential scheduling of multiple resources which constrain system throughput, International Journal of Production Research, 1999, 37:21-33

[107] 席裕庚，动态大系统方法导论，北京：国防工业出版社，1988, 106-109

[108] Uzsoy R., Wang C.S., Performance of decomposition procedures for Job shop scheduling problems with bottleneck machines, International Journal of Production Research, 2000, 38:1271-1286

[109] Cox J.F., Blackstone J.H., APICS Dictionary (9th), APICS: The educational society for resource management, 1998, 72

[110] Umble M.M., Srikanth, M.L., Synchronous manufacturing: principles for world class excellence, Cincinnati, OH: South-Western Publishing, 1990

[111] Roser C., Nakano M., Tanaka M., Shifting bottleneck detection, Proceedings of the 2002 Winter Simulation Conference, 2002

[112] Chiang S.Y., Kuo C.T., Meerkov S.M., c-Bottlenecks in serial production lines: identification and application, Mathematical Problems in Engineering, 2001, 7:543-578

[113] Lawrence S.R., Buss A.H., Shifting production bottlenecks: causes, cures, and conundrums, Journal of Production and Operations Management, 1994, 3:21-37

[114] Morton T., Narayan V., Ramnath P., A tutorial on bottleneck dynamics: a heuristic scheduling methodology, Production and Operations Management, 1995, 4:94-107

[115] Narayan V., Morton T., Ramnath P., X-Dispatch methods for weighted tardiness job shops, Working Paper #1994-14, G.S.I.A., Carnegie Mellon University, Pittsburgh, PA

[116] Lee G.C., Kim Y.D., Choi S.W., Bottleneck-focused scheduling for a hybrid Flow shop, International Journal of Production Research, 2004, 42:165-181

[117] Carlier J., The one-machine sequencing problem, European Journal of Operational Research, 1982, 11:42-47

[118] Pinedo M., Singer M., A shifting bottleneck heuristic for minimizing the total weighted tardiness in a Job shop, Naval Research Logistics, 1999, 46:1-17

[119] Glassey C.R., Resende M.G.C., Closed-loop job release control for VLSI circuit manufacturing, IEEE Transactions on Semiconductor Manufacturing, 1988, 1:36-46

[120] Brucker P., Jurisch B., Sievers B., A branch and bound algorithm for the job-shop scheduling problems, Discrete Applied Mathematics, 1994, 49:107-127

[121] Demeulemeester E.L., Herroelen W.S., Modeling setup times, process batches and transfer batches using activity network logic, European Journal of Operational Research, 1996, 89:355-365

[122] Ivens P., Lambrecht M., Extending the shifting bottleneck procedure to real-life applications, European Journal of Operational Research, 1996, 90:252-268

[123] Aytug H., Kempf K., Uzsoy R., Measures of subproblem criticality in decomposition algorithms for shop scheduling, International Journal of Production Research, 2002, 41:865-882

[124] Singer M., Decomposition methods for large Job shops, Computers & Operations Research, 2001, 28:193-207

[125] Upasani A.A., Uzsoy R., Sourirajan K., A problem reduction approach for scheduling semiconductor wafer fabrication facilities, IEEE Transactions on Semiconductor Manufacturing, 2006, 19:216-225

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

[127] Lee Y.H., Bhaskaran K., Pinedo M.L., A heuristic to minimize the total weighted tardiness with sequence-dependent setups, Technical Report, Department of Industrial Engineering and Operations Research, Columbia University, New York, 1991

附录 1：论文中主要符号说明

m	机器个数	M	机器集，$M=\{1,2,\cdots,m\}$
n	工件个数	N	工件集，$N=\{1,2,\cdots,n\}$

r_i 工件i的到达时间 (release time)

q_i 工件i的传递时间 (delivery time)

d_i 工件i的交货期限 (due date)

ω_i 工件i的权重 (weight)

C_i 工件i的完工时间 (completion time)

L_i 工件i的滞后时间 (lateness)，$L_i=C_i-d_i$

T_i 工件i的拖期时间 (tardiness)，$T_i=\max(C_i-d_i,0)$

$O_{i,j}$ 工序ij，表示工件i在机器j上的加工工序

$p_{i,j}$ 工序ij的加工时间 (processing time)

$t_{i,j}$ 工序ij的开工时间 (start time)

$C_{i,j}$ 工序ij的完成时间 (end time)

$r_{i,j}$ 工序ij的到达时间

$q_{i,j}$ 工序ij的传递时间

$d_{i,j}$ 工序ij的交货期限 (due date)

$w_{i,j}$ 工序ij的等待时间(waiting time)

$C_{\max}$ 工件的最大完工时间，$C_{\max}=\max_{i\in N}C_i$

$L_{\max}$ 工件的最大滞后时间，$L_{\max}=\max_{i\in N}L_i$

$s_i(j)$ 工件i第j道工序加工机器的标号(index)

$\sigma_i(k)$ 工件i在机器k上加工工序的标号

$\pi_k(i)$ 机器k上第i个加工的工件的标号

附录 2：图列表

附录 3：表格列表

致谢

值此论文完成之际，特别感谢我的导师席裕庚教授。感谢他几年来在学术上给予我的精心指导和严格要求。感谢他在我身处困境时给予的理解、鼓励和支持。他渊博的学识、敏捷的思维和卓越的眼光使我受益终生。他严谨的治学态度、忘我的工作热情、虚怀若谷的学士风范更将成为我终生为人治学的楷模。

感谢谷寒雨副教授，在他的指导下使我得以顺利完成瓶颈分解调度的研究工作。他对调度优化问题的深刻理解和睿智的思维总是使我获益良多。和他共同探讨学习、生活和工作问题的这段日子，令人难忘，他是我人生中难得的良师益友。

感谢卢俊国副教授、巢志俊老师，感谢他们给予的无私帮助。感谢复杂系统与控制开放实验室的李少远、汪小帆、苏剑波、陈卫东教授、李柠副教授和吴沂军老师。

感谢实验室刘斌、陈庆、王冰、贾永基、张宏远、王长军、孙波、蔡自立、张燕等师兄师姐，感谢刘琳、张群亮、郭世亮、王勇军同学，感谢岑丽辉、刘勇、李晓丽、郑鹏远、李德伟、林姝、陶继平、刘宏、熊俊、刘学英、姜政、何佳、罗鸣、杜军君、秦辉、李静、葛荣容、石磊、王晓峰、张颖、周俊杰、应惟伟、薛拾贝和夏凌等师弟师妹。和他们在一起的日子是快乐的，真诚地祝福他(她)们工作顺利、生活幸福。

感谢 A0303222 班的同学、感谢室友祁永敏、杨华、范瑾、欧林林、陈爱玲、张颖颖、尚丽辉，永远也忘不了大家在一起的这段美好时光。

感谢我的硕士导师侯国莲教授，感谢华北电力大学自动化系的金慰刚、张建华、李国光教授，感谢所有曾经关心和培育过我的老师。

感谢我的弟弟，他的乐观开朗一直陪伴着我成长！

最后深深感谢我的父母，是他们无私的爱和关怀伴我走过二十多年的求学之路。我的每一次成功和失败背后都有他们的支持和鼓励。

谨以此文献给所有关爱和帮助过我的人！

攻读博士学位期间发表的学术论文（第一作者）

[1] Study on constraint scheduling algorithm for job shop problems with multiple constraint machines. To appear in International Journal of Production Research. (SCI/ EI)

[2] A modified bottleneck-based procedure for large-scale flow-shop scheduling problems with a bottleneck. Chinese Journal of Mechanical Engineering, 2006, 19(3): 356-361. (EI: 064610239807)

[3] A modified bottleneck-based heuristic for large-scale job-shop scheduling problems with a bottleneck. To appear in Journal of Systems Engineering and Electronics. (EI)

[4] 大规模流水线调度的瓶颈分解算法研究. 控制与决策，2006, 21(4): 425-429. (EI: 06259949218)

[5] Study on the shifting bottleneck procedure for flow-shop problem with makespan. The 6-th Asian Control Conference, Bali-Indonesia, 2006: 1234-1236. (EI)

[6] 移动瓶颈机子问题优先级确定方法研究. 二十四届中国控制会议论文集, 广州，2005 年 8 月, 1195-1199. (ISTP)

攻读博士学位期间承担的研究项目

[1] 国家自然科学基金项目，“Job Shop 调度问题的滚动时域方法研究及性能分析” 项目号：60274013

[2] 国家自然科学基金项目，“基于非合作博弈的多目标生产调度研究” 项目号：60474002

[3] 国家自然科学基金项目，“基于离散事件系统控制理论的预测调度算法研究” 项目号：60504026