具有时间不确定性的 Job Shop 问题研究

专 业：计算机软件与理论

硕 士 生：孙素君

指导老师：姜云飞教授

摘 要

机器调度问题来源于不同的领域，如柔性制造系统、生产计划、计算机设计、后勤及通信等，这些问题的共同特性是没有一个有效的算法能在多项式时间内求出最优解。古典的作业车间问题（Job Shop Problem，JSP）是最著名的机器调度问题之一。在过去的 40 年里，机器调度问题吸引了无数研究者的浓厚兴趣，大量的研究成果相继问世，但是对于调度问题的研究一般都在于静态调度问题的研究。可是在实际世界中调度很少是静态的，在调度执行过程中一些事件的发生往往是不可预测的，而一些事件的不可预测往往引起的是时间的不确定。

本文研究了具有时间不确定性的的作业车间问题，以古典的作业车间问题为背景，加入处理时间不确定性因素——处理时间在一个闭区间的范围内变化。为了求解含有该类不确定性时间因素的作业车间问题，我们以确定性车间调度问题的求解为基础，对一个确定性的作业车间问题进行分离逻辑（Difference Logic）建模，使用分离逻辑求解器求出一个近似最优的调度结果，在调度执行过程中使用插空策略（Hole-Filling Strategy），在每个作业处理结束时，根据该调度策略动态地调整调度方案，缩短生产时间。并通过实验证明了利用分离逻辑求解确定性的作业车间问题的有效性，同时也证明了在一般情况下使用插空策略动态调整调度结果可以有效地缩短调度执行时间。所有的测试结果是在 1.8GHz 的 Pentium 4 处理器，512 兆内存的硬件环境和 Linux RedHat 9 的软件环境下取得的。

关键词：作业车间问题、不确定性、分离逻辑、TSAT++、调度策略

Study on Job Shop Problem with Uncertain Processing Time

Major: Computer Software and Theory

Name: Sun Sujun

Supervisor: Professor Jiang Yunfei

Abstract

Scheduling problems are found in such diverse fields as manufacturing, computing, communications, construction, transportation, health-care, service delivery, and project management, and so on. The classical Job Shop Problem (JSP) has been one of the most studied scheduling problems. The research community has devoted decades of effort trying to find efficient methods for this problem. But most of these research focus on static instance. And in the production scheduling of industrial processes, there are various uncertainties, and the uncertainty of processing time is the most typical one.

This article presents a survey on Job Shop Problem with uncertain processing time. In this work we present Difference Logic as the natural tool for modeling and solving of Job Shop Problem (JSP). We also define appropriate scheduling strategies to develop algorithm for finding optimal scheduling based on the solving of static Job Shop Problem. The approach is experimentally verified on a Job Shop Problem where the duration of each task is only known to be inside an interval of the form [l, u]. All experiments are performed on a 1.8 GHz Pentium Ⅳ, 512M Memory and Linux RedHat 9

Key Words: Job Shop Problem、Uncertainty、Difference Logic、TSAT++、Scheduling Strategy

第一章 引言

1.1 简介

机器调度问题来源于不同的领域，如柔性制造系统、生产计划、计算机设计、后勤及通信等，这些问题的共同特性是没有一个有效的算法能在多项式时间内求出最优解。在过去的 40 年里，机器调度问题吸引了无数研究者的浓厚兴趣，大量的研究成果相继问世。古典的作业车间问题（Job Shop Problem，JSP）是最著名的机器调度问题之一，它最初是由 Johnson 在 1954 年在车间工作流任务调度问题上提出，并对两台机器的情况进行研究。随着对它的研究逐步加深，人们发现它的理论模型可以应用于很多的现实问题中，如路面的车辆管理，火车的时刻表制定，物流，排课以及企业的人力资源管理等问题都能用类似的 Job Shop 问题模型描述和求解，Job Shop 问题还对目前较热的企业资源规划（ERP）的研究和应用有着推动作用，所以对 Job Shop 问题研究的现实意义是巨大的。

但是对于调度问题的研究一般都在于静态调度问题的研究，即调度的相关信息包括作业数目，作业之间的优先关系和作业的处理时间等确定而且在调度执行过程中不会发生改变。可是在实际世界中调度很少是静态的，在调度执行过程中一些事件的发生往往是不可预测的，而一些事件的不可预测往往引起的是时间的不确定。为此，必须考虑存在不确定性情况的生产调度问题，保证生产过程正常、满意地进行。

本文主要就是研究具有时间不确定性的作业车间问题，以古典的作业车间问题为背景，加入处理时间不确定性。为了求解具有时间不确定性的作业车间问题，我们以确定性作业车间问题的求解为基础，给出近似最优的调度结果，在调度过程中使用适当的调度策略（Scheduling Strategy），每个作业处理结束时，根据该调度策略调整调度结果缩短生产时间。

在求解确定性车间调度问题的众多方法中，将作业车间问题编码表示成一个命题可满足问题（the Propositional Satisfiability problem, SAT），再用 SAT 求解方法求解的方法因为求解 SAT 问题的高效算法而得到众多研究者的重视，可是如

果用 Crawford 的编码方法对 Job Shop 问题进行编码，随着 Job Shop 问题的规模的增大，编码后所得的布尔表达式急速增加；而且随着问题规模的增大，编码所需时间会大大多于用 SAT 算法求解时间。为了解决这些问题，我们使用分离逻辑（Difference Logic）对 Job Shop 问题建模，使用分离逻辑求解器求解得出调度结果。

1.2 本论文各章节安排

本文的第一章是引言，从整体上介绍了本文的研究背景和研究内容。本文之后的内容里将详细介绍具有时间不确定性的 Job Shop 问题的研究，具体的章节安排如下：

第二章详细介绍了古典作业车间问题，包括问题模型描述，问题模型，问题的复杂性和求解方法。

第三章分析了生产调度过程中所存在的各种不确定性，阐述了不确定性的分类，不确定性的数学描述，含有确定性因素的生产调度的数学模型，各种调度策略和求解方法。

第四章介绍利用SAT模型求解古典作业车间问题的不足之处和改进的办法，在本文中引入分离逻辑对作业车间问题进行建模来降低模型的规模，本章介绍了分离逻辑的基本概念和求解器 TSAT++。

第五章介绍了用分离逻辑求解作业车间问题时问题的建模，并在作业车间问题的分离逻辑模型中引入领域约束以提高求解的速度。

第六章主要介绍了一类处理时间不确定条件下的作业车间问题，以第五章介绍的确定性作业车间问题的分离逻辑求解方法为基础，运用适当的调度策略求解该问题。

第二章 古典作业车间问题

调度问题是一个复合的优化问题，作业车间问题（Job Shop Problem，JSP）是调度问题领域中最著名，最难求解的问题之一，其特点是没有一个有效的算法能在有效的多项式时间内求出其最优解。它最初是由 Johnson 在 1954 年在车间工作流任务调度问题上提出，并对 2 台机器的情况进行研究。随着对它的研究逐步加深，人们发现它的理论模型可以应用于很多的现实问题中，如路面的车辆管理，火车的时刻表制定，物流，排课以及企业的人力资源管理等问题都能用类似的 Job Shop 问题模型描述和求解，Job Shop 问题还对目前较热的企业资源规划（ERP）的研究和应用有着推动作用，所以对 Job Shop 问题研究的现实意义是巨大的。

人们对 Job Shop 调度问题的研究自从上世纪 50 年代到目前为止都从未中断过，这个领域的研究产生了大量杰出的求解思想和研究成果，对现实生产的促进起到了巨大的推动。随着人类的计算能力不断增强，人们对 Job Shop 问题的本质理解的逐步深刻，提出的求解方法不断更新和提高，人们就有希望解决更大规模的 Job Shop 问题。

从文字上描述 Job Shop 问题似乎很简单，就是给定一组任务和一组机器，每个机器在某一个时间段只能执行一个任务，每个任务由一系列的操作组成，操作之间有先后关系，并且每个操作在机器上的执行不能被打断，除非该操作完成退出。调度的目的就是如何把这些操作分配在给定的这组机器上，并在最短时间内完成所有的任务。

本章详细介绍了古典作业车间问题，包括问题描述，问题模型，问题的复杂性和求解方法。

2.1 古典作业车间问题描述

古典作业车间问题可以表述为：有 m 台不同的机器和 n 个不同的工件，每个工件是一个由多道工序组成的工序集合，工件的工序顺序是预先给定的。每道工序用它所要求的机器和固定的加工时间来表示。此外对工件和机器有一些约束，例如：

(1) 一个工件不能两次访问同一机器；
(2) 不同工件的工序之间没有先后约束；
(3) 工序一旦进行不能中断；
(4) 每台机器一次只能加工一个工件；
(5) 下达时间和交货期都不是给定的。

形式上表达如下：给定任务集合 J，有 n 个任务组成 $J=\{J_1,J_2,...J_n\}$，机器集合 M，有 m 台机器，$M=\{M_1,...M_m\}$，一个包含 l 个操作的操作集合 O，$O=\{O_1,O_2,...O_l\}$；对 $\forall v\in O,\exists p(v)\in IN$，$IN$ 是每个操作在机器上的间隔，p(v) 是操作 v 的执行时段，在这个时段中，$M(v)\in M$ 是唯一的，$J(v)\in J$ 也唯一；另外在操作集合 O 上定义一个二元关系 A 来表示两个操作之间的序关系：如果 $(v,w)\in A$ 则表示 v 在 w 前执行，于是 A 引入了一个任务的所有操作的序排列，但属于不同任务的操作之间没有 A 关系。进一步，如果 $(v,w)\in A$，并且不存在 $u\in O$，使得 $(v,u)\in A\wedge(u,w)\in A$，那么 $M(v)\neq M(w)$。

一个调度就是一个函数 $S:O\rightarrow IN\cup\{0\}$，即操作集合到时间段的映射，其中对每个操作 v 都定义初始时间 s(v)，那么合法的调度即：

$$\forall v\in O: s(v)\geq 0$$

$$\forall v,w\in O,(v,w)\in A: s(v)+p(v)\leq s(w)$$

$$\forall v,w\in O,\ \ v\neq w, M(v)\neq M(w): s(v)+p(v)\leq s(w) 或 s(w)+p(w)\leq s(v)$$

调度的长度 Make-span 就是 $Max_{v\in O}s(v)+p(v)$，最优解就是使得合法调度的长度 Make-span 最短。

求解目标：$C^*_{\max}=\min(C_{\max})=\min(\max_{feastibles\_chedule}(s_{ik}+p_{ik}):\forall j\in J,\forall m_k\in M)$

我们也可以用 $n/m/G/C_{max}$ 表示一个以最大流程时间最小为目标的的n个工件，m台机器的作业车间问题。

表 2-1 描述了一个 3 台机器、3 个工件的作业车间问题 *J1 = (m1 ,3), (m2, 3), (m3,2)* , *J2 = (m1, 1), (m3, 5) ,(m2, 3), J3 = (m2, 3), (m1, 2) ,(m3, 3)*的机器序列和加工时间表。

表 2-1　3 台机器、3 个工件的问题

加工时间				机器序列			
工件	工序			工件	工序		
	1	2	3		1	2	3
J_1	3	3	2	J_1	m1	m2	m3
J_2	1	5	3	J_2	m1	m3	m2
J_3	3	2	3	J_3	m2	m1	m3

我们用甘特图表示作业车间问题的一个可行调度，有两类甘特图：机器甘特图和工件甘特图。甘特图中用下标jom来命名每一道工序，这里j指出工件，o表示工件的工序，m表示加工这道工序的机器，例如，下标321表示工件3的第2道工序在机器1上进行。图2-1是表2-1描述的作业车间问题的一个可行调度的机器甘特图，表示每台机器上不同工件的加工时间。图2-2是该可行调度的工件甘特图，表示每个工件的多道工序进行的时间。

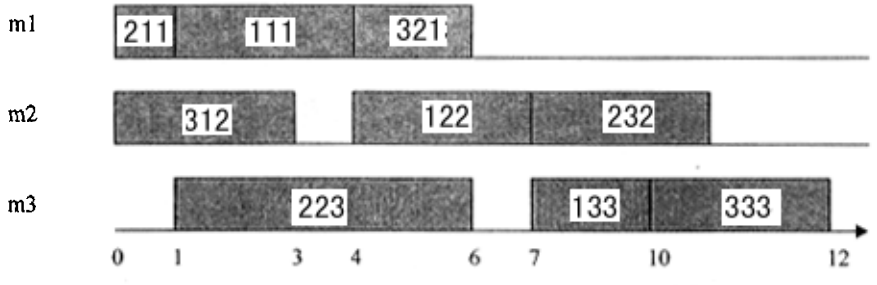


图 2-1　机器甘特图

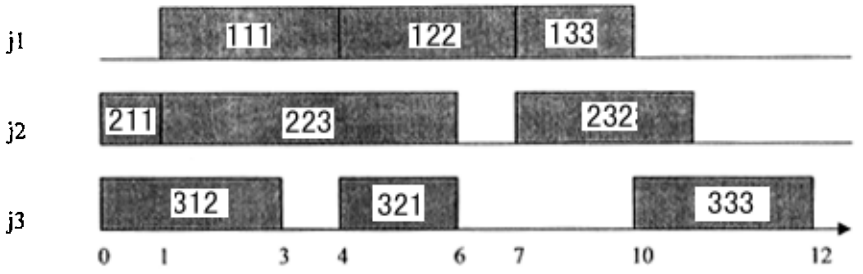


图 2-2 工件甘特图

2.2 古典作业车间问题模型

2.2.1 整数规划模型

整数规划模型（IP）由 Baker 提出。在作业车间问题的整数规划模型中考虑两类约束：

(1) 给定工件的工序前后约束；

(2) 给定机器的工序非堵塞约束。

对于一个 n 个工件、m 台机器的作业车间问题，一般地，令 c_{jk} 表示工件 j 在机器 k 上的完工时间，t_{jk} 表示工件 j 在机器 k 上的加工时间。

首先考虑给定作业车间问题的工序前后约束。

对于工件 i, 如果在机器 h 上的加工先于机器 k，有如下约束：

$$c_{ik} - t_{ik} \geq c_{ih}$$

另一方面，如果在机器 k 上的加工首先出现，有约束

$$c_{ih} - t_{ih} \geq c_{ik}$$

由于约束中的一个或多个都必须满足，因此称它们为分离性约束（Disjunctive Constraints）。定义如下的指示参数：

$$a_{ihk}=\begin{cases}1,\text{对于给定的工件i，如果在机器h上的加工先于机器k}\\0,\text{其他}\end{cases}$$

以上约束也可以写成

$$c_{ik}-t_{ik}+M(1-a_{ihk})\geq c_{ih};\qquad i=1,2,\cdots,n;\qquad h,k=1,2,\cdots,m$$

其中 M 是一个很大的数。上面的不等式很巧妙地吸收了前后约束，如果先在机器 h 上加工，$M(1-a_{ihk})$ 为 0，给定的不等式正是所要的。否则 $M(1-a_{ihk})$ 变得很大，不等式同样为真。

现在来考虑给定机器的工序非堵塞约束：

对于两个工件 i 和 j 都需要在机器 k 上加工，如果 i 先于工件 j 来到，有以下约束：

$$c_{jk}-c_{ik}\geq t_{jk}$$

另一方面，如果工件 j 先来到，有约束：

$$c_{ik}-c_{jk}\geq t_{ik}$$

定义指示器变量 x_{ijk} 为

$$x_{ijk}=\begin{cases}1,\text{如果工件i先于工件j来到机器k}\\0,\text{其他}\end{cases}$$

以上约束可以写成

$$c_{jk}-c_{ik}+M(1-x_{ijk})\geq t_{jk};\qquad i=1,2,\cdots,n;\qquad h,k=1,2,\cdots,m$$

以最大流程时间最小为目标的 n 个工件、m 台机器的作业车间问题的整数规划模型可以描述为：

$$\min\ \max_{1\leq k\leq m}\left\{\max_{1\leq i\leq n}\{c_{ik}\}\right\}\qquad(2\text{-}1)$$

$$c_{ik}-t_{ik}+M(1-a_{ihk})\geq c_{ih};\qquad i=1,2,\cdots,n;\qquad h,k=1,2,\cdots,m\qquad(2\text{-}2)$$

$$c_{jk}-c_{ik}+M(1-x_{ijk})\geq t_{jk};\qquad i=1,2,\cdots,n;\qquad h,k=1,2,\cdots,m\qquad(2\text{-}3)$$

$$c_{ik}\geq 0,\quad i=1,2,\cdots,n;\quad k=1,2,\cdots m\qquad(2\text{-}4)$$

$$a_{ihk}=0\text{或}1;\quad i=1,2,\cdots,n;\quad h,k=1,2,\cdots,m\qquad(2\text{-}5)$$

$$x_{ijk}=0\text{或}1;\quad i,j=1,2,\cdots,n;\quad k=1,2,\cdots,m\qquad(2\text{-}6)$$

约束（2-2）保证每个工件的工序的加工顺序满足预先的要求；约束（2-3）保证

每台机器一次只能加工一个工件。

如果考虑平均流程时间问题，目标可以写成：

$$\min \sum_{i=1}^{n} \max_{1\le k\le m} \{c_{ik}\} \tag{2-7}$$

2.2.2 线性规划模型

Akers 等人在文献[3]中讨论了作业车间问题的线性规划（LP）模型。对于一个 n 个工件、m 台机器的作业车间问题，令 $N=\{0,1,2,\cdots, m\times n+1\}$ 表示工序的集合，其中 0 和 $m\times n+1$ 表示虚设的起始工序和完成工序；$M=\{1,2,\cdots, m\}$ 是机器的集合；A 是表示每个工件的工序前后关系约束的工序对集合；E_k 是机器 k 上加工的工序对集合。对于每个工序 i，加工时间 d_i 是一定的，工序的起始时间 t_i 是优化过程中待确定的变量。因此，作业车间问题的线性规划模型可以描述为：

$$\min \ t_n \tag{2-8}$$

$$t_j - t_i \ge d_i, \qquad (i,j)\in A \tag{2-9}$$

$$t_j - t_i \ge d_i, \ \text{或}\ t_i - t_j \ge d_j, \qquad (i,j)\in E_k, \quad k\in M \tag{2-10}$$

$$t_i \ge 0, \qquad i\in N \tag{2-11}$$

约束（2-9）保证每个工件的工序顺序满足预先的要求，约束（2-10）保证每台机器一次只能加工一个工件。问题的一个可行解称为一个调度。

2.2.3 图模型

作业车间问题可以用非连接图来表示，非连接图 $G=(N,A,E)$ 定义为：N 包含代表所有工序的节点，A 包含连接同一工件的邻接工序的边，E 包含连接同一机器上加工工序的非连接边，所谓非连接边是可以有两个可能方向的边。调度将固定所有非连接边的方向，以确定同一机器上的工序顺序；一旦对于某台机器的顺序确定下来，连接工序的非连接边将被通常的带优先箭头的连接边取代。非连

接边集 E 分解成子集系 E_k：$E = E_1 \cup E_2 \cup E_3 \cdots E_m$，每台机器一个系。图 2-3 是一个 3 个工件、3 台机器作业车间问题的非连接图实例，其中每个工件包含 3 道工序，节点 $N = \{0,1,3,4,5,6,7,8,9,10\}$ 对应工序，其中 0 和 10 是虚设的起始工序和终止工序。连接边 $A = \{(1,2),(2,3),(4,5),(5,6),(7,8),(8,9)\}$ 对应同一工件的工序前后约束。非连接边（虚线表示）$E_1 = \{(1,5),(5,9),(9,1)\}$ 对应机器 1 上加工的工序，非连接边 $E_2 = \{(4,2),(2,8),(8,4)\}$ 对应机器 2 上加工的工序，非连接边 $E_3 = \{(7,3),(3,6),(6,7)\}$ 对应机器 3 上加工的工序。

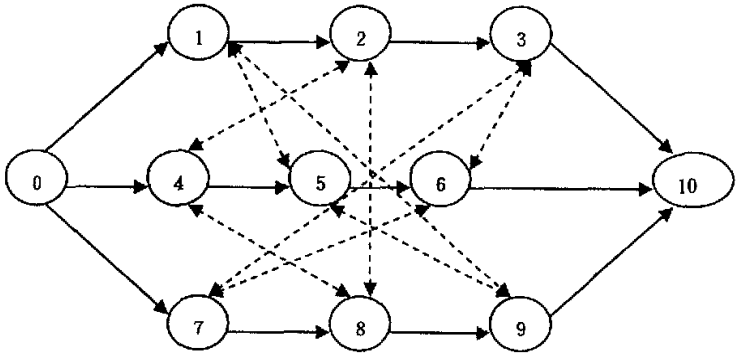


图 2-3 非连接图表达法

该作业车间问题的机器加工序列也可以用表 2-2 来描述：

表 2-2 图 2-3 表示的作业车间问题的机器序列表

机器序列			
工件	工序		
	1	2	3
J_1	m1	m2	m3
J_2	m2	m1	m3
J_3	m3	m2	m1

作业车间问题是找出每台机器上的工序的加工顺序，即确定非连接边的加工顺序，即确定非连接边的方向，使得产生的结果图是非循环的（工序间无先后冲突），并且起始点到终止点的最大权路径的长度最小。最大权路径的长度对应于最大流程时间。图 2-4 给出了图 2-3 的一个合法调度的情况。

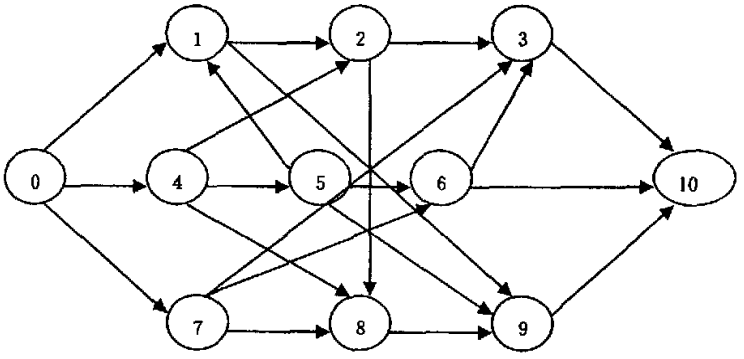


图 2-4 一个合法调度

2.3 古典作业车间问题的复杂性

Job Shop 问题的复杂性一般是 NP-hard 的，对于$n \times m$维的 Job Shop 问题复杂度的上界就是$(n!)^m$，所以对于20×30的问题而言，其计算量就是7.2651×10^{183}。显然，计算量随着 m 或 n 呈指数激增的。在低维情况下可以在分支界定方法中使用枚举，高维情况下就只能用近似求解方法了。Lenstra 等人 1977 年证明3×3的问题、每个任务不超过 3 个操作的$n \times 2$的问题或者每个任务不超过 2 个操作的$n \times 3$问题都是 NP-hard；1979 年，Lenstra 和 Rinnooy 证明了所有的操作不超过 2 个单元的延时的$n \times 2$的问题是 NP 问题；Sotskov 于 1991 年证明了$3 \times m$问题的 NP 特征。正是由于 Job Shop 问题的复杂性，French 在 1982 年预言永远不可能找到有效的算法解决答多数 m>4 且 n>4 的调度问题。

2.4 古典作业车间问题求解方法

自从 1954 年 Johnson 提出作业车间调度的问题开始，学术界对 Job Shop 问题的研究已经有半个世纪之久，Job Shop 问题研究的规模也从一两个机器和任务到今天的成百上千，计算时间随着计算机硬件和算法发展的突飞猛进而不再望而生畏。

通过对大量文献的分析可以将 JSP 的研究方法分为两大类[1]：最优化方法和

近似/启发式方法。最优化方法主要包括混合整数线性规划(MILP)、分枝定界(B&B) 法以及拉氏松弛法等；近似/启发式最初是由于计算量小并且算法易实现而引入 JSP 问题的，主要包括优先分派规则(PDRs)、人工智能(AI)、神经网络(NN)及邻域搜索法(NS)，邻域搜索法又包括禁忌搜索(TS)、遗传算法(GA) 和模拟退火(SA) 等可以称之为亚启发式(Meta-heuristic) 的近似优化方法。

2.4.1 最优化方法

最优化方法是能够产生一个精确最优解的方法，Johnson 最早提出的针对(n/2/G/Cmax)问题的 Johnson 规则[2]，虽然是针对流水作业的求解方法，但它对以后的研究有很大的影响。此后相继有利用多项式时间算法求解 2/m/G/Cmax[3]、$J_2|J\leq 2|Cmax$ 等特殊的 JSP 问题，这些研究奠定了经典调度理论的基础。

在 20 世纪 60 年代，针对整数规划问题的求解，提出利用更加复杂的数学结构消除隐含非最优解的搜索空间以提高搜索效率的枚举算法，分枝定界法(B&B)在理论研究上有很大成果。对一个 n×m 的 JSP 问题有(n!)m 种可能的解，因此大规模问题用完全枚举法在计算上是不可能的。70 年代后侧重计算复杂性方面的研究，Lenstra 等证明了 3/3/G/Cmax , J 2|J≤3|Cmax 和 J 3|J≤2|Cmax 三种情况都是 NP 难问题。Balas 最早提出 JSP 问题的 B&B 算法，此后 Carlier 等基于 Jackson 的剩余总加工时间最长(MWR)规则提出预占先调度(JPS)，取得了较好的结果。为克服数学表示和软件方法的局限性，近期 Davis 等提出基于数学规划的分解策略，将调度问题分解为多个子问题，分别考虑各子问题及其限制，提高了计算能力。

2.4.2 近似/启发式方法

早期人们对调度问题的研究大都局限于数学方法，理论证明和问题求解，偏重于理论方面，并且企图获得全局的最优解。随着 70 年代到 80 年代初以 Cook 等计算理论专家为首的学者对可计算性和计算复杂性进行了深入研究，证明了绝大多数的调度问题是 NP-hard 问题，在实际中不可能求得精确解，人们便开始寻求另一种方法来解决调度问题。

1．优先分派规则

最早的近似解求法是优先级分派规则（Priority Dispatching Rule），它的思想是对所有的操作赋予一个优先级，使得在调度选择的时候能按照优先级的顺序进行“贪心”安排，使得优先级高的操作先得到调度，比如最短作业优先法等，操作的优先级可以根据优先级函数在调度过程中动态改变，以便更好的利用贪心策略。这种方法容易实现，计算量小。

最早的分派规则由 Jackson[5]和 Smith[6]等提出，JSP 的分配规则有 SPT (最短加工时间)、L PT (最长加工时间)、MWR (剩余总加工时间最长)、LWR (剩余总加工时间最小)、MOR (剩余工序数最多)、LOR(剩余工序数最小)、EDD (最早交货期)和 FCFS (选择同一机器上工件队列中的第一道工序) 等。Panwalkar 等通过性能指标对 113 个分派规则归类总结，Wu 把调度规则分为三大类，即同作业信息相关的优先级规则、优先级规则的组合和切换以及加权规则。优先分派规则的近似优化方法，关键在于如何针对给定问题的性能，选择最好的规则。从各规则的优化效果看，SPT 能够减小所有作业的平均流程时间，EDD 用于优化最大延期相关的目标。对规则之间的切换及由此产生的问题（如出错修复）是近期活跃的研究领域。

2．人工智能技术

利用人工智能领域现有的研究成果对调度问题求取近似解或采用某些智能优化技术是行之有效的方法。80 年代出现的人工智能和专家系统在调度研究中占据重要地位，可以产生比优先规则更复杂的基于对整个调度系统的启发式，并能从特殊数据结构中获取大量信息，缺点是计算比较耗时。因此，真正用人工智能的方法来研究调度问题是从 20 世纪 80 年代开始的，并出现智能调度（AI Scheduling），并作为一个专门的领域来研究。

它首先开始于 Fox et al.的 ISIS 约束引导的调度系统[5]。ISIS 是最早基于 AI 技术解决特定 JSP 的专家系统之一，使用约束指导的搜索方法，其目标约束基于交货期和在制品，把资源的处理能力作为物理约束。ISIS 分为三层结构，分别为选择订单、能力分析、执行调度，同时加入重构和修改调度的学习功能。对大规模问题限于单个专家系统的有限知识和能力，加入资源代理、任务代理及代理间的协作机制，近期出现了分布式调度系统。但是，AI 技术对如何协调各代理机制，目前还没有统一的设计和指导思路，对作业调度的集中化和分散化思想还在

讨论之中。

同时，随着智能规划的研究成果在实际生产应用中发挥出巨大的作用，越来越多的人工智能专家转向智能调度的研究，并且把已有的算法改进优化应用在智能调度问题上，如早期的模拟退火、以及后期提出的神经元网络等算法都发挥了显著的作用。90 年代初，在约束满足（Constraint Satisfaction）局部搜索（Local Search）技术的发展基础上，人工智能领域的学者们提出了很多优秀的算法如约束满足算法、遗传算法、Tabu 搜索等。目前人们在这些算法上不断的优化改进，使得它们的实用性和效率得到更大的提高。

3．神经网络方法

神经网络应用于调度问题已有十几年的历史，利用指导学习神经网络找到系统输入、输出之间的关系，输入特性包含作业特征(如数量、路径、交货期和处理时间等)，输出为相关排序和性能指标。目前应用最多的是 BP 网，Rabelo 针对不同的到达模式、过程计划和程序排序提出使用后增值神经网解决 JSP 问题，针对由能量函数定义的基于松弛模型的神经网提出的 Hopfield 网解决了一类经典的调度问题。由于计算时产生大量不可行解且计算时间较长，NN 解决实际调度问题的效率不高，而指导学习神经网试图通过训练类型找到输入输出之间的关系，随着问题规模的增大，网络的规模也急剧增大。

Hopfied 和 Tank 提出对称性内连接的网络结构之后，Foo 和 Takefuji 在 1988 年提出的一种建立在这种确定性神经网络模型基础上的解描述方法，他们将 Job-Shop 问题表示成 2 维神经元矩阵。Cherkassky 和 Zhou 等人提出一种使用线性 cost 函数编程（linear programming）方法取代了原来的“夸克”函数的神经元网络算法，对每个操作都用网络中的神经元来表示，操作之间的存在连接。这个算法在解的质量上和神经网络的复杂程度上都较原有 Foo 等人提出的方法上很大提高和改进。Chang 和 Nam（1993）结合全局优化技术实现了线性编程，能很好地解决一些小规模问题，对于大规模的计算很难保证求得最优解。Willems（1994）使用带有阈值约束的线性编程方法能通过反馈连接（feedback connection）快速地找出可行解。Foo 在 1995 年修改了 Hopfield 模型，避免使用了局部优化的夸克能量函数解决 Job-Shop 问题。Sabuncuoglu 等人（1996）提出了修改 3D Hopfield 模型解决单机器调度问题。

4．领域搜索方法

求解 JSP 的亚启发式方法是基于邻域搜索策略发展起来的，启发式并不企图在多项式时间内求得最优解，而是在计算时间和调度效果上进行折衷。下面是三种有代表性的亚启发式：

模拟退火算法（SA）

SA 是基于 Mente Carlo 迭代求解的一种全局概率型搜索算法，是一种串行优化算法，其收敛性要求初温应足够高，使解空间各状态能以几乎相同的概率出现。Vanarhooven 等[9]提出对于 JSP 问题，邻域函数的重要标志是相邻关键操作工序处理顺序的改变，且必须服从在同一机器上处理的工序条件。LKolonko[10]证明标准的 JSP 的邻域具有非对称性，且由于 SA 的弱收敛性，提出了在 SA 基础上结合 GA 的混合启发式。 文献[11]利用 SA 以一定的概率接受较差个体的性质，结合遗传算法而改进了选择机制,使收敛速度有所提高。近期在调度问题研究领域的一大趋势是将不同的邻域搜索法结合形成混合启发式。

禁忌搜索(TS)

TS 是 Glover[12]提出的模拟智能过程的一种具有记忆功能的全局逐步优化算法，对变动的排序在其可行解的所有空间中进行搜寻。通过设置禁忌表，避免陷入局部最优或重复过去的搜索，利用中、长期的存储机制进行强化和多样化搜索，对 JSP 问题 Laguna 等[13, 14]提出三个基于简单移动的 TS 排序策略。Taillard[15]结合加速搜索策略防止重复计算工序的开始时间，提出一种快速估值策略，但只对半活动调度有效。而 Nowicki 等[16]考虑到解的质量和计算时间，结合 Taillard[17]的 TS 算法，在 Vanaarhooven 邻域的基础上，把单个关键路径分割为不同块应用于严格限制的邻域中，计算效率大大提高。

遗传算法(GA)

遗传算法是 Holland 基于自然遗传进化的绝对模型提出的并行搜索机制[18]，GA 的 5 个要素是编码、适应值函数、初始种群、遗传算子和参数设置。由于对 JSP 问题要考虑工序的前后约束和非堵塞约束，使得染色体的编码表示非常重要。如果编码不当，会在遗传算子操作时产生不可行解，失去了交叉或变异算子的有效性。Cheng 等[19]把编码表示分为两类共 9 种，分别是直接的方法(基于工序、基于工件、基于工件对关系、基于完成时间和基于随机键表示法) 和间接的方法(基于优先规则、基于优先表、基于非连接图和基于机器的表示法)。Davis[20]首先基于优先表的间接编码表示求解 JSP, Falkenauer 等[21]则利用把工序编码为字符串的方法。Tamaki[22]基于非连接图的间接编码，其染色体用非连接边

顺序表的二进制串表示。Bean[23]采用随机键表示法，每个基因由两部分组成：整数部分表示机器的分配，分数部分以非减顺序排列的(0, 1) 之间的随机数来确定每个机器上的工序。曹承煜等[24]提出结合拉氏松弛法的遗传算法，能够克服震荡并减小计算量。纪树新等[25]对 JSP 应用连锁基因编码法与遗传进化算子相结合，显示算法的可行性。王海英等[26]采用定界遗传算法与逆序调度策略，在收敛速度上有所提高。GA 对初始种群很敏感，交叉算子和编码表示对算法结果会产生不利影响。研究表明，随机产生的初始种群不如用其他启发式(如 SA)产生的初始种群好，而修复非法染色体相对容易些，需要对遗传算法进行改进。调度问题中使用的 GA 技术，大多只强调了遗传算法的独立使用，限制了问题的复杂性。

5. 其他方法

定理证明方法是一种结合逻辑程序设计（Logic Programming）求解调度问题的方法，把求解调度的过程形式化为证明由初始状态和动作序列推理到目标状态成立的过程。它基于 Robinson 的消解原理（refutation resolution），把求解过程形式化为证明由初始状态和动作序列可以证明目标状态的过程，其证明过程的解释就是一个调度，以命题逻辑、一阶谓词逻辑等规范逻辑和各种非规范逻辑为理论基础。它有两种谓词组成，一个是由 if-and-only-if 定义的普通谓词，另一个是由约束表示的原子“Atom”定义的原始谓词。推导由一组开始的目标，一般是事实开始，应用规则，如果目标的析取范式的一个子句是解，当且仅当它仅包括原始原子并且满足 ALP（Adductive Logic Programming）约束。

第三章 含有不确定性因素的生产调度问题

在过去的40年里，作业车间问题吸引了无数研究者的浓厚兴趣，大量的研究成果相继问世。但是对于调度问题的研究一般都在于静态调度问题的研究，即调度的输入确定而且在调度执行过程中不会发生改变，可是在实际世界中调度很少是静态的，生产过程中存在事先无法预料的不确定因素是不可避免的，例如机器故障，工件加工时间事先未知，用户改变交付期等。为此，必须考虑存在不确定性情况的生产调度问题，保证生产过程正常、满意地进行。

本章分析了生产调度过程中所存在的各种不确定性，阐述了不确定性的分类，不确定性的数学描述，含有确定性因素的生产调度的数学模型，各种调度策略和求解方法。

3.1 不确定性的分类

在企业的经营和生产过程中，会存在各种各样的不确定因素[28, 29]，如产品的产量、原材料的价格和供应量、劳动力因素、每一道生产工序中产品的处理量、处理时间、中间存储单元的存储量、中间产品的稳定存放时间等都可能发生变化；生产中原材料或能源的暂时短缺也是不确定因素；另外，生产过程中往往会发生一些事先无法预料的突发事件，如生产设备的损坏、仪器仪表的故障、操作工的误操作等，这些不确定的因素往往会导致生产调度方案无法按预定的目标正常执行。企业经营和生产过程的不确定因素可以分为以下四类[30, 31, 32, 33]。

1．系统固有的不确定性

这类不确定性参数主要包括各种动力学、热力学常数和传热、传质系数等。由于实际工业生产工艺过程是十分复杂的，各种化学、物理、热力学等常数与实验室数据具有比较大的差别，或者这些数据本身就很难获得。这些不确定性，造成了对生产过程进行精确、有效地建模的极大困难，因而常常影响生产过程的控制水平和控制性能指标。

2．生产过程中产生的不确定性

这类不确定性主要包括生产过程中各种流体介质的流速、温度、压力等的变化和设备的处理能力。如在生产过程中，各生产工序对各种物流的处理时间、某一设备的生产能力、中间产品的稳定存放时间等。这类不确定性往往影响生产调度系统的性能。

3．外部环境的不确定性

在市场经济体制的条件下，企业的生产不再是独立的行为，而是受外部环境的影响，产品的需求量、产品的价格、能源、原材料的供应以及其它外部环境因素构成不确定因素。这类不确定性往往影响生产计划和生产调度方案的正常执行。

4．离散不确定性

这方面的不确定性主要是设备的故障，仪器、仪表的失效，人工误操作等，或者是关键操作人员的短缺等。这类不确定性对企业组织正常生产会造成很大困难。

3.2 不确定性的描述

1．不确定性参数服从一定的概率分布规律，使用随机变量进行描述[34, 35, 36, 37, 38]

应用独立的随机变量来表示不确定的参数，根据对实际生产的历史数据的分析和对市场的预测，统计出不确定的参数所服从的概率分布。在许多情况下，采用随机变量描述调度系统的不确定性是一种有效的方法。如处理时间(包括产品的加工时间、设备的清洗时间、原材料或中间产品的装载时间、传输时间等）的不确定性，我们可以通过对历史操作数据的分析研究，统计出某一产品在某一道工序中的处理时间服从某种统计分布规律。常用的概率分布有均匀分布、正态分布和指数分布。

2．不确定性参数在某一区间内变化[39, 40, 49]

在生产实际中，要依据历史的操作数据分析、归纳出某一参数满足何种概率分布，有时是困难的。最简便的方法就是统计、整理出操作数据位于某个区间内。用区间表示方法来模拟参数的不确定性具有很大的适用性，如鲁棒控制理论中表示参数的不确定性一样。这时，不确定的参数可以表示为：$Hi \in [Hi_{\min}, Hi_{\max}]$，其

中 Hi_{min}，Hi_{max} 为参数 Hi 可能取值的下限和上限，这两个量可以通过对历史操作数据的分析整理获得。

3．不确定性变量为离散的值[41-45]

这类不确定性主要是设备的故障，仪器、仪表的失效等，或者是误操作等。在数学描述上可以用一些离散的量来表示这类不确定性，但实际处理时往往采用动态反馈策略，一旦发生故障，及时进行检测，对调度模型重新进行优化计算，以确保整个调度方案的最优性。另外，还可以应用模糊数学描述调度过程中的不确定性。

3.3 本文研究的不确定性描述

目前对不确定因素下的调度的研究主要集中在处理时间不确定下的调度问题（该问题在时间应用也是最普遍的），在不确定性因素的描述上，主要采用随机变量和模糊数学的方法，但是实际上常常很难对一些不确定性因素作出概率分布上的估算，而只能凭经验或历史数据进行大致的区间估算。本文主要研究处理时间不确定的作业车间问题，以古典的作业车间问题为背景，加入时间不确定性因素。我们在本文中考虑的时间不确定就在于作业的处理时间处于一个时间区间 $[l,u]$ 之内，即不确定性参数（处理时间）处于一个区间之中。只有在作业处理后才能确定处理时间。在下面的例子中，就是在古典的作业车间问题中引入了这种时间不确定性因素。

表 3-1 描述了一个处理时间不确定条件下的作业车间问题的机器序列和加工时间表，问题可以也表示为 $J_1=(m_1,[4,10])\prec(m_3,[2,4])\prec(m_2,[1,5])$

$J_2=(m_2,[2,8])\prec(m_1,[1,3])\prec(m_3,[3,7])$

表 3-1 一个处理时间不确定性条件下的作业车间问题

加工时间				机器序列			
工件	工序			工件	工序		
	1	2	3		1	2	3
J_1	[4,10]	[2,4]	[1,5]	J_1	m1	m3	m2
J_2	[2,8]	[1,3]	[3,7]	J_2	m2	m1	m3

在上面的例子中，包含 2 个工件 J_1，J_2 和 3 台机器 $M=\{m_1,m_2,m_3\}$，其中不确定性涉及到每个工件的每道工序的处理时间 d，例如第一个工件的第一道工序 j_{11} 的处理时间 $d_{11}\in[4,10]$，即 d_{11} 在区间[4,10]内变化。

3.4 不确定条件下的调度策略

生产过程中出现的各种不确定性（如产品的处理时间）对调度问题的目标函数的影响是不同的。另一方面，不同的调度策略对不确定因素的灵敏度是不同的，所以，我们在制定调度方案时应该根据生产过程中出现的不确定性，综合考虑目标函数的优化与调度方案的鲁棒性之间的矛盾，选择适当的调度策略，制定出满足需要的调度方案。

3.4.1 鲁棒调度策略（Robust Scheduling）

鲁棒调度是指对不确定因素的变化具有鲁棒性的调度方案，即该调度方案在任何情况下都满足对应的调度问题的各种约束。当存在不确定因素时，按该调度方案执行，既能保证生产过程的正常进行，又能保证具有优化(满意) 的目标函数。文献[29]给出了一个在工件加工时间不确定情况下的鲁棒调度定义，并以单机流程时间总和的调度问题为例给出了鲁棒调度解的求法。李明切和顾幸生也研究了Job shop调度的鲁棒性。

一般地说，对于不确定因素变化频繁，但变化幅度较小的情况，应该采用鲁棒调度方案，一旦调度方案制订完成，只要不出现大的扰动，按此方案组织生产，能获得比较满意的结果。实际中，一般采用两种方法来实现这种调度，一种是一次性调度方案，一种是Reactive调度方案。

李明切，顾幸生，李琦等[39][40][41]是采用一次性调度方案，他们采用随机变量描述处理时间的不确定性，将调度模型描述成随机规划问题，采用遗传算法进行寻优，取得了较好的调度效果。在调度过程中，如果对过程进行深入的分析研究，可以总结出一些规则，在调度过程中充分发挥这些规则的作用，对加快调度

算法的寻优起到很大的作用，他们研究并比较了加入调度规则时对一条具有8个加工工序，生产5 个产品的Job Shop 生产线的调度结果，说明了调度规则对减少优化计算量起到很大的作用。而Honkomp、Mockus、Reklaitis等采用Reactive调度方案[51][52][53]，他们所建立的调度模型为确定性模型，但调度系统监测生产过程对调度方案的执行情况，将其反馈给调度器，一旦生产过程发生不确定的变化，如处理时间的变化，调度器及时修正调度方案，以保证整个调度系统的稳定性。他们采用PLASTIC和BATCH4 仿真软件包进行仿真计算，研究了调度策略对调度系统性能指标的影响。

3.4.2 适应性调度策略（Adaptive Scheduling）

对于不确定性因素变化幅度较大，如处理单元失效，重要的仪器仪表发生故障，处理时间大幅度改变，或者是由于市场的动荡而引起的需求量的大幅改变等这些未预期事件，此时应采取适应性调度策略。适应性调度可以分为重调度(Rescheduling)、滚动调度(Rolling scheduling)、动态调度(Reactive scheduling 或 Dynamic scheduling)、在线调度(Online scheduling) 等。

对于未预期事件的调度问题，在柔性制造系统(FMS) 中研究较多，研究的方法是重调度和滚动调度。重调度可分为连续性和周期性两类。连续性重调度指当引起系统的状态变化的事件发生时，就立即进行重调度，因而能极快跟踪系统的变化，但计算量极大，对于复杂系统的实时性要求较难满足；方剑和席裕庚[54]提出了周期性和事件驱动的滚动调度策略。周期性调度指在间隔一定的生产时间后进行重调度，但在这段时间区间内，对“未预期”事件不敏感，积累该时间段内各种偏差，可能导致系统整体性能的恶化，其优点是计算量较小，易于实现在线重调度。滚动调度则主要是借鉴预测控制的滚动优化的思想，跟踪系统的动态变化，可分为 Time-based 和 Job-based 两大类，但如何选择时间窗或工作窗的规模，针对具体的对象仍然是有待进一步研究的课题。

吴受章等[34]应用自适应控制理论的思想，提出了一种 FMS 的模型参考自适应调度方案，并对 Job shop 情况分别研究了建模、摄动计算、调度器设计等问题；Daeho和Moon[35]研究了在操作单元失效情况下的重调度技术，他们针对SP88 标准间歇生产过程讨论了 DSMM (Dynamic Shift Modification Method)，PUOM (Parallel Unit Operation Method)和 UVVM(Unit Validity Verification Method) 重调

度技术，提出了时间调整因子的概念，并以化妆品生产过程为实例，研究了这三种调度技术的应用。

Ierapetritou 和 Pistikopoulo 在其发表的论文中，研究了不确定性条件下生产过程设计和调度的一体化问题。在进行生产过程最优设计过程中考虑了生产过程的调度问题，建立了二级(设计模型和调度模型相互关联的) 随机规划模型，然后将其转化为一个优化问题，这样既增加了生产过程的操作柔性，又使生产过程在不确定情况下的调度能取得满意的结果。

对于流程工业，由于具有复杂性和各产品任务间的强关联特性，对“未预期”事件的调度问题，无法照搬 FMS 的动态调度的方法，如滚动调度方法，对某些存在不稳定中间产品的生产过程，很难采用 Time-based 滚动调度方法，但可借鉴 Job-based 重调度策略，人们相应的提出了 Product-based 重调度策略。也有学者提出实时在线调度、Reactive 调度等方法。

3.4.3 智能调度方案

根据处理实际生产过程的不确定因素对调度系统的影响，人们根据人工智能原理，提出了智能调度方案。常见的智能调度方案主要有：专家系统、基于规则的调度方法、基于智能优化方法的调度以及几种调度方法的结合[37]。Badell 等[38]研究了基于实时网络知识对生产过程和资金的调度，并研究了信息集成技术。由于智能调度方法可以模拟人的思想，对生产过程出现的问题进行分析、推理，并作出重调度决策，所以智能调度对于处理不确定性具有独特的优势。自动控制系统中处理不确定性时，研究了系统的鲁棒控制、预测控制和自适应控制，以克服不确定性对系统的影响。在生产调度中，为了克服不确定性对调度方案的影响，借鉴自动控制理论的思想，研究具有鲁棒性的、具有预测功能和自适应能力的反馈调度策略，对于解决实际问题具有重要的意义。

3.5 不确定性条件下的调度问题求解

对于生产调度，已有多种类型的建模和优化方法。采用数学规划理论如排队网络方法、线性规划、非线性规划、动态规划、混合整数线性规划等，和采用

Petri 网、极小极大代数、时序逻辑等，取得了许多研究成果。然而流程工业的生产过程是高维对象，采用规划模型求解调度问题，随着维数的增加，计算量呈指数增长，因而为了提高求解效率、减少计算工作量，提出了不少基于规则的优化方法。如根据实际生产过程的特点，总结一定的调度经验，提出一些调度规则，对于提高计算效率起到了重要的作用；采用人工智能的方法(如各种搜索的方法、专家系统的方法等) 对于解决具体的调度问题，不仅可以简化问题，而且能获得合乎实际的满意解。近几年来，随着优化技术的发展，各种新颖的优化方法被应用到调度系统中去，如采用遗传算法、模拟退火算法、趋化性算法等，为获得全局最优解提供了可能性。

具体来说，顾幸生等[39-41]将Flow shop和Job shop问题的处理时间不确定性采用模糊数学的方法进行描述，建立基于模糊规划的处理时间不确定性条件下的调度模型，并采用一种改进的模糊优化算法将模糊模型化为确定性模型，结合遗传算法进行求解，取得了较好的调度效果。具体来说是采用三角模糊数 $O=\{O_L,O_M,O_U\}$ 表示在一定区间变化的不确定量 O（比如加工时间），其中 O_L 表示最理想情况下的值，O_M 表示最可能或最常见的值，O_U 表示最长可能的加工时间，即最不理想情况下的加工时间，建立了一个含有不确定性因素的调度问题模型。其中对模糊调度问题的求解结果是使完成时间Make-span在最理想、最可能和最不理想三种情况下的值都尽可能小。为了求解该调度模型，采用了改进的模糊优化算法将模糊模型化为确定性模型，结合遗传算法进行求解。

第四章 分离逻辑

在求解确定性车间调度问题的众多方法中，将作业车间问题编码表示成一个命题可满足问题（the Propositional Satisfiability problem, SAT）因为求解 SAT 问题的高效算法而得到众多研究者的重视[56][57]。Crawford & Baker [56]详细研究了 JSP 问题的 SAT 编码方法。可是如果用 Crawford 的编码方法对 JSP 问题进行编码，随着 JSP 问题的规模的增大，编码后所得的布尔表达式急速增加；而且随着问题规模的增大，编码所需时间会大大多于用 SAT 算法求解时间。为了解决这些问题，我们使用分离逻辑（Difference Logic）[67][68]对 JSP 问题建模，使用分离谓词求解器[69]求解得出调度结果。

在本章我们将会介绍利用 SAT 模型求解古典作业车间问题的不足之处和改进的办法，在本文中引入分离逻辑对作业车间问题进行建模来降低模型的规模，介绍了分离逻辑的概念和求解器 TSAT++。

4.1 满足调度(Scheduling as SAT)

人工智能认为调度问题是结合优化要求的约束可满足问题，所以解决调度问题常用的一种方法就是将其表示为约束可满足问题模型，人工智能学者们把命题可满足问题（Propositional Satisfiability，SAT）作为约束可满足问题的分支来研究。命题可满足问题是约束可满足问题的一种，当然也有它自身的特点：它的每个变量都是布尔型，每个约束在形式上如果用一阶逻辑表示的话，则是表示变量或其否定的文字析取的子句。

一个命题可满足性问题(Propositional Satisfiability，SAT)是确定一个布尔表达式是否可满足。问题表示成一组合取范式$C_1 \wedge C_2 \cdots \wedge C_m$，每个字句（Clause）$C$是文字（Literal）的析取$l_1 \vee l_2 \cdots \vee l_n$，每个文字$l$是一个布尔变量$x$或者$\neg x$。每个布尔变量的取值为*true*或者*false*，一个满足赋值是使得每个字句中至少有一个文字为真，从而使得整个布尔表达式为真。该领域的工作已经产生了许多快速有效的满足算法，SAT 规划器正是基于这种方法设计的。因为一般的满足算法

大都利用了可满足问题的统一结构，为了利用这些快速的算法，人们就需要把调度问题转化成可满足问题。这个工作是艰巨和困难的，因为要把时态信息表示成布尔变量，并且所有的约束都要表示成子句形式。而且一些问题表示成为 SAT 问题后命题规模很巨大，也可能使得求解器无法在可接受的时间内求得解。很多组合优化问题都可以被成功地表示成命题可满足性问题，但是并不是所有的组合优化问题都可以表示成为命题可满足性问题。

大概二十年前 Crawford & Baker[56]研究了如何将 JSP 问题表示成 SAT 问题，并给出了用不同的 SAT 算法(Tableau, Isamp, Gsat)求解该问题的实验结果对比。Cadoli & Schaerf [57]也提出了 Job Shop 问题的 SAT 编码方法。但是使用 Crawford 和 Cadoli 的 SAT 编码方法对 Job Shop 问题进行转换，转换时间甚至大于使用其他好的 JSP 算法的求解时间。

把调度问题表示成可满足问题的局限性是很大的。首先，离散的时态表示很难有效地表达代数关系和函数，进而对时态约束进行表示的时间空间消耗开销就已经抵消了快速算法带来的节省；其次，一些问题表示成为 SAT 问题后命题规模很巨大，也可能使得求解器无法在可接受的时间内求得解；再次，在传统的编码方法中，没有涉及到一些领域约束的编码，而领域约束的应用可以提高求解的速度；最后，Bayardo & Schrag [59]发现 Job Shop 问题是一个非典型的 SAT 问题，用迭代枚举（Iterative Sampling）算法比回朔算法能更有效地求解。所以使用广义的 SAT 来降低问题的规模是一个有效的方法。

一些研究者提出了表达能力更强的一些逻辑语言来减小编码的规模，例如语言 nested equivalences[60], exclusive-or [61], disjunctions of conjunctions of literals [62], cardinality constraints[63], finite-domain literals[64], Quantified Boolean Formulae (several papers),lifted models [65] ,linear pseudo-Boolean (PB) models [66]和分离逻辑 [67~69]。Steven Prestwich 和 Colin Quirke 在文献[58]中就提出用线性的 Pseudo-Boolean 语言模型来对 JSP 进行建模的方法。

其中分离逻辑模型是整数线性规划（ILP）问题的一个特殊形式。将作业车间问题表示成 DL 模型可以大大地减小问题编码的规格，而且可以把一些领域约束引入模型，加快问题的求解。

4.2 分离逻辑（Difference Logic）

4.2.1 分离逻辑定义

分离逻辑（Difference Logic，或者 Separation Logic）[67]也被称为分离谓词（Separation Predicate）[68]，是命题逻辑的简单扩展，在分离谓词中除了命题变量外，该逻辑的原子公式是由不等式 x-y < c 组成，其中 x，y 是实数变量，而 c 是整型常量。分离逻辑理论的形式化定义如下：

定义4.1 分离逻辑（Difference Logic）语法

设 $B=\{b_1,b_2,\cdots,b_n\}$ 是一组命题（布尔）变量，$X=\{x_1,x_2,\cdots,x_n\}$ 是一组数值变量，DL(P,X) 可用如下文法表示

$$\phi ::= b \mid (x-y<c) \mid (x-y\le c) \mid \neg\phi \mid \phi\vee\phi \mid \phi\wedge\phi$$

其中 $b\in B$，$x,y\in X$，c 是一个常数 $c\in D$，D 是整数集合 Z 或者实数集合 R。

Remark1 其他的布尔连接符：

- 等价 $\Leftrightarrow$：$\phi_1\Leftrightarrow\phi_2\equiv(\phi_1\vee\neg\phi_2)\wedge(\neg\phi_1\vee\phi_2)$
- 蕴涵 $\Rightarrow$：$\phi_1\Rightarrow\phi_2\equiv(\neg\phi_1\vee\phi_2)$
- 异或 $\oplus$：$\phi_1\oplus\phi_2\equiv(\phi_1\wedge\neg\phi_2)\vee(\neg\phi_1\wedge\phi_2)$

Remark2 如果 D = Z，DL可以简化为：

$$\phi ::= b \mid (x-y\le c) \mid \neg\phi \mid \phi\vee\phi \mid \phi\wedge\phi$$

因为在整数域 $(x-y<c)$ 等价于 $(x-y\le c-1)$

定义4.2 赋值

给定一个B赋值 v_B 和X赋值 v_X，$v_B: B\to \mathrm{B}$，$v_X: X\to D$，其中 $\mathrm{B}=\{true, false\}$

一个 $\{B,X\}$ 赋值是函数 $v: DL(X,B) \to \mathrm{B}$，其中

$$v(b) = v_B(b)$$

$$v(x-y<c) = \begin{cases} true & iff\ v_X - v_Y < c \\ false & otherwise \end{cases}$$

$$v(x-y \le c) = \begin{cases} true & iff\ v_X - v_Y \le c \\ false & otherwise \end{cases}$$

$$v(\neg\phi) = \neg v(\phi)$$

$$v(\phi_1 \wedge \phi_2) = v(\phi_1) \wedge v(\phi_2)$$

$$v(\phi_1 \vee \phi_2) = v(\phi_1) \vee v(\phi_2)$$

其中 $b \in B$， $x, y \in X$， $c \in D$, and $\phi, \phi_1, \phi_2 \in DL(X,B)$

和命题逻辑相比，分离逻辑有着强大的表达的能力，对一些组合优化问题的编码有着很大的优势。近年来国际上对分离逻辑的求解有着很大的进展。 文献[67, 68]详细论述了用基于 SAT 的方法求解分离逻辑公式的方法，一些研究小组也开发出了独立的分离逻辑（DL）求解器[MathSAT, SEP, DLSAT, TSAT++]，我们采用了 TSAT++作为我们的分离逻辑求解器，该求解器输入多个分离逻辑表达式组，如果表达式组可满足则给出每个变量的赋值，否则给出不可满足的提示。

4.2.2 TSAT++

TSAT++是一个求解 Satisfiability Modulo Theories (SMT)问题[69][70]的求解器。Satisfiability Modulo Theories (SMT)问题是确定一个背景理论 T 的一组表达式是否可满足，其中 T 是简单的一阶理论，比如 the theory of arrays, full linear arithmetic (real and integer)和 Difference Logic[68]。

TSAT++的主要模块中是一个枚举模块（Enumerators）和一个满足性检查模块（Satisability Checker），TSAT++的架构如图 4-1。文献[69]详细论述了求解器的求解过程和求解过程中的优化技术。

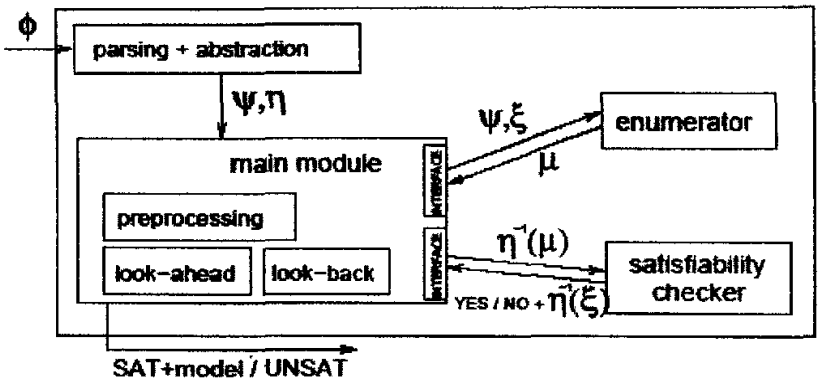


图 4-1 TSAT++的架构

TSAT++的最新版本 TSAT++ v0.5 主要是为了求解 SMT 的一个实例：分离逻辑（Separation Logic）。一个析取时态问题（Disjunctive Temporal Problem，DTP）可以用分离逻辑表示，一个 DTP 用分离逻辑表示如图 4-2。

```
# k = 2  每个字句（Clause）的文字（Literal）数目，
# n = 5  变量的数目，
# m = 10 字句数目，
x3-x1<=68 v x1-x3<=-15 ^
x0-x3<=42 v x0-x1<=80 ^
x0-x4<=-42 v x2-x4<=34 ^
x3-x2<=-60 v x0-x2<=15 ^
x0-x3<=-60 v x0-x4<=-4 ^
x4-x3<=21 v x1-x4<=-18 ^
x2-x4<=50 v x1-x4<=-20 ^
x4-x0<=-38 v x3-x4<=50 ^
x0-x1<=52 v x1-x0<=27 ^
x0-x1<=88 v x0-x3<=82
```

图 4-2 一个 DTP 问题的分离逻辑表示

其中 x_0~x_4 是整型变量，^ 和 v 分别表示 OR 和 AND 逻辑符号。

在文献中对 TSAT++求解用分离逻辑表示的含有 20 个变量的 DTP 问题时，根据问题中字句数目的不同，在求解过程中的一致性检查的次数的对比。图 4-3 给出了实验结果。

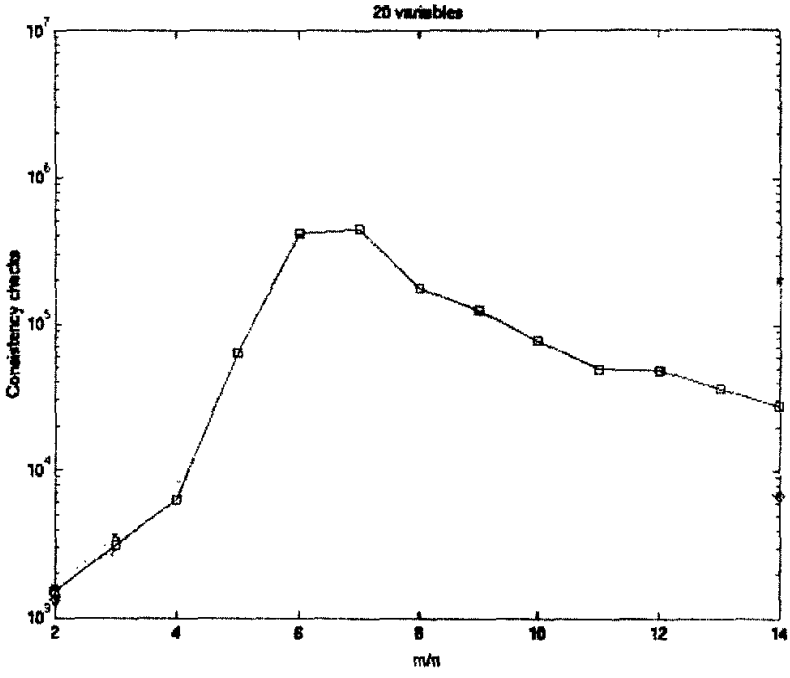


图 4-3　TSAT++求解效率

文献中还对比了 TSAT++和其他两个分离逻辑求解器 SEP 和 MathSEP 求解的速度。求解问题为钻石问题（Diamond Problem）和其他实际问题。比较结果如图 4-4 和图 4-5。可以看到和 SEP 和 MathSEP 相比，TSAT++的求解效率较高。

D	S	unique	TSAT++	SEP	SEP (no c.m.)	MathSAT
50	4	N	0	0.03	0.12	0.05
50	4	Y	0.01	0.84	0.07	TIME
100	5	N	0.01	0.13	1.18	0.61
100	5	Y	0.04	10.20	0.17	TIME
250	5	N	0.08	0.95	52.20	5.4
250	5	Y	0.21	288.30	0.77	TIME
500	5	N	0.29	5.92	742.99	21.22
500	5	Y	1.05	TIME	4.85	TIME

图 4-4　Diamond Problem

Instance (cat.)	TSAT++	SEP (no c.m.)	MathSAT
abz5-900 (a)	1.9	TIME	0.82
ring2-10 (a)	0.02	0.01	0.01
ring2-100 (a)	2.07	0.18	1.07
post-office 4/10 (a)	0.51	TIME	1.06
post-office 4/11 (a)	1.01	TIME	2.13
post-office 4/12 (a)	0.58	TIME	0.91
LD-ST-neg.3*step* (b)	0.03	0.42	-
OOO-neg.3*step* (b)	0.01	0.23	-

图 4-5 其他实际问题

TSAT++的求解形式是 TSAT++ (*problem*, *time*)，其中，*problem* 是组分离逻辑公式的集合，*time* 可选，表示如果在 *time* 时间内还没有求出可满足解则认为问题不可满足。

第五章 分离逻辑求解作业车间问题

在上一章我们介绍了分离逻辑和分离逻辑求解作业车间问题的优点。在本章中我们将详细介绍用分离逻辑求解作业车间问题时问题的建模，并在模型中引入领域约束以提高求解的速度。

5.1 古典作业车间问题建模

为了利用分离逻辑求解作业车间问题，我们对作业车间问题建立最小整数线性规划（ILP）模型如下。

在一个$n\times m$（n 个工件，m 台机器）的古典作业车间问题中，我们称每个工件的每道工序为一个作业，作业J_{ij}表示第i个工件的第j道工序，设定每个作业的开始时间是个正整数，取值范围是$\{0,1,\cdots T-1\}$，其中T是最大可能流程时间(Maximum Allowed Make-span)。我们为每个作业J_{ij}的开始时间定义一个整数变量s_{ij}，其中$1\le i\le n$，$1\le j\le m$。每个开始时间变量的取值范围是$dom(s_{ij})=\{0,1,\cdots T-1\}$，每个作业$J_{ij}$的加工时间是由问题给出的常量$d_{ij}$。

古典作业车间问题中需要考虑的两类约束

(1) 给定工件的工序前后约束（Sequencing Constraints）；

(2) 给定机器的工序非堵塞约束（Resource Capacity Constraints）。

首先考虑给定作业车间问题的工序前后约束（Sequencing Constraints）。对于工件i的前后两个相邻的工序有

$$s_{ij}-s_{ij-1}\ge d_{ij-1} \qquad (5\text{-}1)$$

其中$1\le i\le n$，$2\le j\le m$。在 ILP 模型中表示这些约束条件比用 SAT 公式表示要简单得多。

现在来考虑给定机器的工序非堵塞约束，即资源能力约束。

对于工件i,i'，$1\le i<i'\le n$,机器$1\le r\le m$，如果作业$J_{ij},J_{i'j'}$争用机器r，则有如下的资源能力约束：

$$s_{ij}+d_{ij}\le s_{i'j'}\oplus s_{i'j'}+d_{i'j'}\le s_{ij} \qquad (5\text{-}2)$$

上述公式不是线性表达式，可以根据公式$\phi_1\oplus\phi_2\equiv(\phi_1\wedge\neg\phi_2)\vee(\neg\phi_1\wedge\phi_2)$将上述资源能力约束表示成

$$(s_{ij}-d_{ij}\le s_{i'j'}\wedge\neg(s_{i'j'}-d_{i'j'}\le s_{ij}))\vee(\neg(s_{ij}-d_{ij}\le s_{i'j'})\wedge s_{i'j'}-d_{i'j'}\le s_{ij})$$

对（5-2）进行进一步转换，我们为每两个工件i，i'以及机器r定义一个布尔变量$v_{ii'r}$，则公式（5-2）可以转换成两个线性约束：

$$s_{ij}+d_{ij}-Tv_{ii'r}\le s_{i'j'} \qquad s_{i'j'}+d_{i'j'}-T(1-v_{ii'r})\le s_{ij} \qquad (5\text{-}3)$$

如果$v_{ii'r}=0[1]$则第一[二]个约束有效，第二[一]个约束无条件满足。本质上，$v_{ii'r}=0[1]$表示工件 i 的工序 j 即作业J_{ij}在工件 i'的工件 j'即作业$J_{i'j'}$之前（后）完成，作业$J_{ij},J_{i'j'}$争用机器r。

5.2 作业车间问题的 SAT 模型和 DL 模型比较

利用 Crawford & Baker 的编码方法对一个$n\times m$的作业车间问题进行 SAT 编码，布尔变量的数目是$O(mn(n+T))$，（其中T是该作业车间问题的最大可能流程时间）。用分离逻辑对一个$n\times m$的作业车间问题进行编码，变量数目为$O(mn)$。

编码最后得到的模型的大小（该模型中文字的数目）也大大减小了。Crawford & Baker 的 SAT 模型中主要考虑的是工序前后约束和资源能力约束，约束的数目是$O(mnT(m+n))$。但是约束的规模和问题的参数是独立的，所以该模型的大小是由模型中的文字来确定，SAT 模型的大小为$O(mnT(m+n))$。而在 DL 模型中除了工序前后约束和资源能力约束，还加入了加强的领域约束，约束

数量为 $O(mn(m+n))$，显然 T 是一个很大的数量，所以模型的规模大大地减小了。

5.3 领域约束

为了提高求解速度，我们在 $n \times m$（n 个工件，m 台机器）古典作业车间问题的整数线性规划模型中加入了一些其他的领域约束。

1．最大可能流程时间约束（Maximum Allowed Make-span Constraints）

在上面提到的作业车间问题的整数线性规划（ILP）模型中，T 是该作业车间问题的最大可能流程时间，所以对于任何一个工件 i，完成该工件的时间都必须小于 T，T_i 表示该工件的最大可能流程时间。可以用如下公式表示该约束，s_{i1} 是工件 i 的第一道工序 J_{i1} 的开始时间，s_{im} 是最后一道工序 J_{im} 的开始时间。

$$s_{im} - s_{i1} \le T_i - d_{im} \tag{5-4}$$

其中，$1 \le i \le n$；

2．作业开始时间约束(Start Time Constraints)

每个作业 J_{ij} 的开始时间 s_{ij} 的值域 $dom(s_{ij}) = \{0,1,\cdots T-1\}$，但是该作业的前序作业 $J_{ij'}, 1 \le j' \le j-1$ 和后继作业 $J_{ij'}, j+1 \le j' \le m$ 限制了该作业的开始时间 s_{ij} 的取值范围，可用如下的约束来表示：

$$\sum_{j'=1}^{j-1} d_{ij'} \le s_{ij} \le T - \sum_{j'=j}^{m} d_{ij'} \tag{5-5}$$

其中，$1 \le i \le n$，$2 \le j \le m$；

3．扩展的工件工序前后约束（Strong Sequencing Constraints）

在上面提到的工件的工序前后约束（Sequencing Constraints）不仅适用于同一工件 i 相邻的两个工序 J_{ij} 和 J_{ij-1}（$1 \le i \le n$，$2 \le j \le m$），而且适用于同一工件 i 中的工序对（J_{ij}，$J_{ij'}$）（$1 \le i \le n$，$1 \le j \le m$）。可用如下公式表示扩展的工件的工序前后约束（Sequencing Constraints）：

$$s_{ij'} - s_{ij} \le \sum_{k=j}^{j'-1} d_{ik} \qquad (5\text{-}6)$$

其中，$1 \le i \le n$，$1 \le j \le m$，$1 \le j' \le m$。

5.4 最大可能流程时间（Maximum Allowed Make-span）

在作业车间问题的工序前后约束（Sequencing Constraints）和我们引入的领域约束中，有一个很重要的常量：最大可能流程时间（Maximum Allowed Make-span）T。作业车间问题是以最大流程时间（Maximum Make-span）最小为目标，但是在 5.1 节的 ILP 模型中我们并没有显式地加入求解目标，在我们的作业车间的 DL 模型中，这个求解目标是通过 T 的选择来实现。因此，不管是工序前后约束（Sequencing Constraints）还是在领域约束中，T 的选择都非常重要。根据问题描述，我们可以知道 $T < \sum_{i=1}^{n}\sum_{j=1}^{m} d_{ij}$，但是 T 越接近问题的最优解的长度，根据 T 所求得的解的质量越好，但同时求解成功的机率越小。为了求得一个比较好的初始 T 值，我们采用作业车间问题的启发式算法求得一个初始值，并在利用 TSAT++求解的过程中，通过二分法使得 T 的值更加合理。

优先分配启发式方法容易实现和并且具有较小的时间复杂性，是实际求解调度问题经常使用的方法，Giffler 和 Thompson 算法被视为所有基于优先规则启发式的共同基础。Giffler 和 Thompson 提出了构造调度的两个算法：活动调度法和无延迟调度法。无延迟调度具有如下性质：如果有工件等待加工，则没有机器处于闲置状态。活动调度具有如下性质：没有工序在不延迟其他工件的情况下而能够开工。活动调度构成一个包含无延迟调度作为子集的更大的集合。Giffler 和 Thompson 构成调度的方法是树形结构方法，树中的节点对应部分调度，边对应可能选择，树的叶子是可数的调度的集合。对于一个给定的部分调度，算法主要是识别所有的加工冲突，即竞争同一机器的工序，尔后在每一阶段采取一些步骤以多种可能的方式解决这些冲突，而启发式则用优先分配规则，即在冲突的工序中按优先规则选择工序来解决这些冲突。

在每一时刻用可调度的工序的集合构造调度。可调度的工序是指迄今尚未调度而其紧前工序刚调度过的工序，可以从工序的优先关系图中确定。在每一个时

刻，选择一个工序加入到部分调度中，工序之间的冲突用优先分配规则解决。表 5-1 包含了实际中常用的一些优先规则。

表 5-1 作业车间分配规则

规则	描述
SPT	优先选择最短加工时间的工序
LPT	优先选择最长加工时间的工序
MWR	优先选择剩余总加工时间最长的工件的工序
LWR	优先选择剩余总加工时间最短的工件的工序
MOR	优先选择剩余工序数最多的工件的工序
LOR	优先选择剩余工序数最小的工件的工序
EDD	优先选择具有最早交货期的工件
FCFS	选择同一机器上工件队列中的第一道工序
RANDOM	随机选择工序

我们以 Giffler 和 Thompson 算法为基础，采用 SPT（优先选择最短加工时间的工序）优先分配规则，传统优先分配启发式方法求得作业车间问题的一个可行调度，把该调度结果的长度作为初始最大可能流程时间。

具体算法如下：

算法 5.1：优先分配启发式算法（Heuristic Algorithm）

输入：一个作业车间问题；

输出：作业车间问题的每个工件的最大可能处理时间 $T_i, i=1,2,\cdots,n$。

$PS(t)$：包含已调度工序的部分调度，（工序编号，开始时间，完工时间），其中作业顺序按照开工时间的先后排列；

$S(t)$： 对应给定的 PS_t，时刻 t 可调度工序的集合，该集合包含每个机器 m 分别构造可调度工序的集合 $S_m(t) \in S(t)$， $S_m(t)$ 中工序按照 SPT 优先分配规则排列。可调度的工序是指迄今尚未调度而其紧前工序刚调度过的工序，可以从工序的优先关系图中确定；

$\sigma(t)$：时刻 t 完工的工序 $i \in PS(t)$；

$m(t)$：时刻 t 空闲机器集合；

T=$\{T_i \mid i=1,2,\cdots n\}$：每个工件的最大可能处理时间集合。

步骤 1：令 t=0，开始时 $PS(t)$ 为空部分调度。初始的 $S(t)$ 包含无紧前工序的所有工序。初始 $\sigma(t)$ 为空；

步骤 2：根据 $\sigma(t)$ 确定当前时刻空闲机器集合 $m(t)$；

步骤 3：为 $\forall m \in m(t)$ 从 $S_m(t)$ 中选择工序，将工序 i 加入 $PS(t)$，构造一个新的部分调度 $PS(t+1)$；

步骤 4：对于 $PS(t+1)$，按如下步骤更新数据集：

1）从 $S_m(t)$ 中删除工序 i；

2）根据 SPT 优先分配规则在 $S_m(t)$ 中增加工序 i 的直接后续工序，构造 $S_m(t+1)$；

3）$t=t+1$。

步骤 5：返回步骤 2 直至构造一个完整的调度。

步骤 6：返回 $\{T_i, i=1,2,\cdots n \mid T_i = t+\max\ d_{ij}, j=1,\cdots,m\}$

图 5-1 算法 5-1 优先分配启发式算法求解最大可能流程时间

5.5 求解算法

在本文中利用分离逻辑公式对作业车间问题进行建模，然后利用分离逻辑求解器求解得到该作业车间问题的一个合理的调度方案，整个算法的框架如图 5-2。

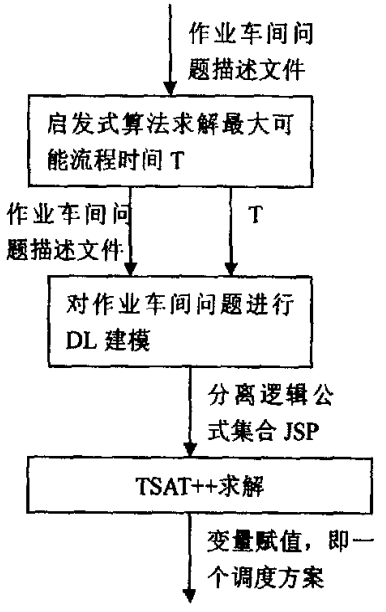


图 5-2 分离逻辑求解作业车间问题算法框图

算法 5.2：利用 TSAT++求解作业车间问题

输入：一个作业车间问题的编码 JSP，最大可能流程时间 T

输出：一个调度方案 $S(P)=\{s(t_1),s(t_2),...,s(t_{mn})\}$

通过算法 5.1 得到的 T 较大，为了使 T 的大小更加接近作业车间问题最优解的长度，我们在利用 TSAT++求解时，采用二分法寻找适当的 T 值，多次调用 TSAT++，以取得更好的解。

JSP：一个作业车间问题分离逻辑编码，即一组分离逻辑公式；

$S(P)=\{s(t_1),s(t_2),...,s(t_{mn})\}$：问题 P 一个合理的调度方案，为每个工序 $t_i, i=1,2,\cdots,mn$ 指定开工时间 s_i，该调度方案根据第五章中的算法求得，其中作业顺序按照开工时间的先后排列；

time：一个常量，在二分法查找适当的 T 值时，确定 TSAT++的调用时间限制，如果超过该时间 TSAT++还无法求得解则认为该问题不可满足；

minNum：常量，二分查找过程的结束条件；

步骤 1：从算法 5.1 求得 T，令 $up = T$, $down = 0$, $mid = (up+down)/2$；

步骤 2：如果 $up - down <= minNum$，转到步骤 6；

步骤 3：调用 TSAT++($JSP(mid)$，$time$)，如果不可满足，则转步骤 5；

步骤 4：令 $up = mid$, $mid = up + down/2$，返回步骤 2；

步骤 5：令 $down = mid$; $mid = up + down/2$，返回步骤 2；

步骤 6：调用 TSAT++ ($JSP(mid)$,)，取得调度结果 $\{s(t_1), s(t_2), ..., s(t_{mn})\}$

图 5-3　算法 5.2　TSAT++求解

5.6　实验结果

为了测试用分离逻辑方法求解作业车间问题的效率，我们对不同规模的作业车间问题进行求解，对于每个规模的作业车间问题都随机生成 10 个测试用例，最后的求解时间是 10 个用例的平均求解时间。

表 5-1 给出测试结果，测试结果是测试算法求解不同规模的作业车间问题的求解时间，从表中可以看到随着问题规模的增加，求解时间增加很快，当问题规模增加到 20*20 时无法在 500 秒内求得一个解。

这些测试结果是在 1.8GHz 的 Pentium 4 处理器，512 兆内存的硬件环境和 Linux RedHat 9 的软件环境下取得的。

表 5-1　TSAT++求解作业车间问题

问题规模	求解时间（CPU Second）
3*3	0.03
4*4	0.18
6*6	1.26
8*8	4.22
10*10	25.54
15*15	105.5

第六章 一类处理时间不确定性条件下的作业车间问题

本章主要介绍了一类处理时间不确定条件下的作业车间问题，以上一章介绍的确定性作业车间问题的分离逻辑求解方法为基础，运用适当的调度策略求解该问题。

6.1 问题描述

在第二章中我们详细介绍了古典作业车间问题，一个$n\times m$的确定的作业车间问题可以如下定义，该问题由n个工件、m个机器组成，每个工件由一定的作业构成，作业之间有一定的工序顺序，每个作业用它所要求的机器和固定的加工时间来表示。同时每台机器一次只能处理一个工作，工作一旦开始处理就不能被中断直至处理完毕。问题是确定机器上工作的处理顺序使最大流程时间（Make-span），即完成所有工作的时间，达到最小。

我们将处理时间不确定性因素引入确定的作业车间问题，在我们定义的处理时间不确定条件下的作业车间问题中作业的加工时间不是确定的，而是处于一个时间区间之内，只有在作业加工完毕之后才能确定作业的具体加工时间。具体定义如下：

定义 6.1：处理时间不确定条件下的作业车间问题（A Job Shop Problem with Uncertain Processing Time）

一个处理时间不确定条件下的作业车间问题可以表示成$P=(T,M,\prec,u,D)$，其中T是作业的集合，M是机器的集合，$\prec$是作业之间的优先关系。函数$u:T\to M$为每个作业指定机器要求，$D:T\to[\mathrm{N}\times\mathrm{N}]$为每个作业定义一个处理时间区间，$D'$，$D''$分别表示区间的上界和下界。为了表示的方便，我们用作业J_{ij}表示第i个工件的第j道工序。

一个处理时间不确定条件下的作业车间问题的实例是指该作业车间问题确定化，即作业在处理过程中处理时间确定下来后得到的一个作业车间问题。一个实例即是一个确定性的作业车间问题。

一个处理时间不确定条件下的作业车间问题的最差实例是该问题的一个实例，在该实例问题中所有作业的处理时间都是该作业的最长处理时间，即都是时间区间的上界的大小。

表 6-1 描述了2×3具有时间不确定性的作业车间问题的机器序列和加工时间表，问题可以也表示为 $J1=(m1,[4,10])\prec(m3,[2,4])\prec(m2,[1,5])$，$J2=(m2,[2,8])\prec(m1,[1,3])\prec(m3,[3,7])$

表 6-1 一个处理时间不确定条件下的 job shop 问题任务加工序列和时间表

加工时间				机器序列			
工件	工序			工件	工序		
	1	2	3		1	2	3
J_1	[4，10]	[2，4]	[1，5]	J_1	m_1	m_3	m_2
J_2	[2，8]	[1，3]	[3，7]	J_2	m_2	m_1	m_3

对于表 6-1 表示的不确定的作业车间问题，作业车间问题（表 6-2）$J1=(m1,8)\prec(m3,3)\prec(m2,5)$ 和 $J2=(m2,6)\prec(m1,2)\prec(m3,5)$ 是它的一个实例，而问题（表 6-3）$J1=(m1,10)\prec(m3,4)\prec(m2,5)$ 和 $J2=(m2,8)\prec(m1,3)\prec(m3,7)$ 则是它的最差实例。

表 6-2 表 6-1 描述的不确定处理时间的 job shop 问题的一个实例

加工时间				机器序列			
工件	工序			工件	工序		
	1	2	3		1	2	3
J_1	8	3	5	J_1	m_1	m_3	m_2
J_2	6	2	5	J_2	m_2	m_1	m_3

表 6-3 表 6-1 描述的不确定处理时间的 job shop 问题的最差实例

加工时间				机器序列			
工件	工序			工件	工序		
	1	2	3		1	2	3
J_1	10	4	5	J_1	m_1	m_3	m_2
J_2	8	3	7	J_2	m_2	m_1	m_3

定义 6.2：作业优先关系约束（Sequence Constraints）

在不确定的作业车间问题 $P=(T,M,\prec,u,D,U)$ 中，设 $s(t)$ 是作业 t 的开始时间，$d(t)$ 表示作业 t 的处理时间，作业间的优先关系由问题定义直接给出，优先约束 $s(t_i)\prec s(t_j)$ 表示作业 t_i 必须在作业 t_j 开始前完成。我们用公式（6-1）表示作业优先约束

$$s(t_j)-s(t_i)\geq d(t_i) \qquad (6\text{-}1)$$

定义 6.3：资源能力约束（Resource Capacity Constraints）

在不确定的作业车间问题 $P=(T,M,\prec,u,D,U)$ 中，设 $s(t)$ 是作业 t 的开始时间，$d(t)$ 表示作业 t 的处理时间，资源约束 $rc(t_i,t_j)$ 表示作业 t_i 和 t_j 因为争用同一个资源，所以不能并行地进行。我们用公式（6-2）表示资源约束。

$$s(t_j)-s(t_i)\geq d(t_i) \quad \oplus \quad s(t_i)-s(t_j)\geq d(t_j) \qquad (6\text{-}2)$$

该资源能力约束和确定的作业车间问题的资源约束是一致的。

在处理时间不确定条件下的作业车间问题中，一个调度方案在任何情况下都必须要满足作业之间的优先关系约束和资源能力约束。处理时间不确定条件下的作业车间问题的作业优先关系约束和确定性作业车间问题中的作业优先关系类似，不同处仅在于为了使得根据该公式求得的调度方案在任何情况下都可行，即调度方案的鲁棒性，公式（6-1）和（6-2）中的 $d(t_i)$ 是最差实例的处理时间。

对表 6-1 描述的不确定的作业车间问题的最差实例表 6-3 $J1=(m1,10)\prec(m3,4)\prec(m2,5)$，$J2=(m2,8)\prec(m1,3)\prec(m3,7)$，我们可以得到如下的约束的分离逻辑表达式：（其中 S_i 是作业 i 的开始时间）

优先约束：

$\{s_2-s_1\geq 10,\ s_3-s_2\geq 4,\ s_5-s_4\geq 8,\ s_6-s_5\geq 3\}$

资源能力约束：

$\{\quad s_5-s_1\geq 10 \ \oplus\ s_1-s_5\geq 3 \quad,\quad s_5-s_1\geq 10 \ \oplus\ s_1-s_5\geq 3 \quad,$

$s_6-s_2\geq 4 \ \oplus\ s_2-s_6\geq 7\}$

定义 6.4 合理调度方案（Feasible Schedule）

对于一个不确定的作业车间问题 $P=(T,M,\prec,u,D,U)$，设 $s(t)$ 是作业 t 的开始时间，P 的一个合理调度方案 S 可以描述为 $s:T\rightarrow R_+$，即为问题中的每个作业指定作业开始时间，S 在任何情况下（即对于该不确定的作业车间问题的任何实例）都必须满足作业优先关系约束和资源能力约束。一个合理的调度方案也称该调度方案具有鲁棒性（Robustness）。

表 6-1 表示的一个作业车间问题 $J1=(m1,[4,10])\prec(m3,[2,4])\prec(m2,[1,5])$，

$J2=(m2,[2,8])\prec(m1,[1,3])\prec(m3,[3,7])$

的一个合理调度可以表示成为图 6.1

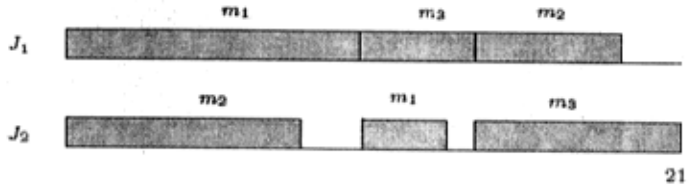


图 6-1 一个合理调度

定义 6.5 作业优先关系图（Precedence Graph）

我们在第二章中曾经介绍过古典作业车间问题的图模型 $G=(N,A,E)$，其中 N 包含代表所有工序的节点，A 包含连接同一工件的邻接工序的边，E 包含连接

同一机器上加工工序的非连接边，所谓非连接边是可以有两个可能方向的边。为了我们后面算法的求解，我们根据古典作业车间问题的图模型，一个问题的可行调度以及定义 6.2 给出的作业优先关系约束建立该作业车间问题的作业车间优先关系图 $G=(N,A,E)$，其中 N 包含代表所有工序的节点（为了表示的方便我们还加入了起始节点和终止节点）；A 包含连接同一工件的邻接工序的有向边；E 表示在可行调度方案中同一机器上工序加工先后顺序。

作业优先关系图和古典作业车间问题的图模型不同之处在于，前者是根据一个问题的描述和问题的一个可行解得到的，后者则是直接根据问题描述得到。作业优先关系图中每个节点和每个边都没有权值，该图仅仅表示了作业之间的优先关系约束。

对于表 6.1 描述的作业车间问题，根据问题定义和图 6-1 的调度方案我们可以得到如图 6-2 的作业优先关系图，在该图中实线表示作业之间的优先关系，虚线表示在同一机器上工序的加工顺序。

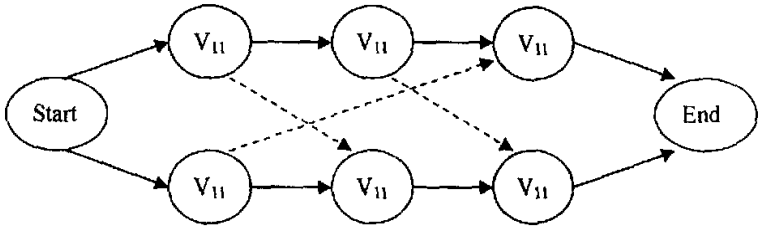


图 6-2 作业优先关系图

我们根据问题定义和一个调度方案求得一个作业优先关系图的目的在于求出对于这个调度方案，每个工序 t 的紧前工序集合 $\prod(t)$，作业必须在它的所有的紧前工序完成后才能开工。根据作业优先关系图 G，$\prod(t)=\{t'|t'\prec t\}$，其中偏序关系 $\prec$ 包括在作业优先关系图中的 E 和 A 两种偏序关系。

6.2 算法设计

为了求解处理时间不确定条件下的作业车间问题，我们以确定性作业车间问题的求解为基础，给出近似最优的调度结果，在调度过程中采用一定的调度策略（Scheduling Strategy），每个作业处理结束时，根据该调度策略调整调度结果而不是进行重调度（Re-Schedule）。整个算法分成两部分，静态调度方案求解部分和调度方案动态调整部分。

在静态调度方案求解部分，我们根据一定的原则求得一个初始的可行调度。为了得到一个初始的调度方案，而且保证该调度方案的鲁棒性，我们选择该处理时间不确定条件下的作业车间问题的最差实例，为该实例求得一个调度方案，显然该调度方案具有鲁棒性。为了求解这个静态的作业车间问题，我们采用第五章的求解方法，具体算法框图如图 5-2.

因为我们要解决的问题是处理时间不确定性条件下的作业车间问题，在该问题中处理时间是在作业加工过程中才确定下来的，为了使得在静态调度方案求解阶段求得的调度适应处理时间的动态变化，在每个工序的处理时间确定下来时根据一定的调度策略调整调度方案，使得新的调度方案既满足问题的各项约束，又可以缩短整个作业车间问题的处理时间。

6.2.1 调度策略的选择

对于一个处理时间不确定条件下的作业车间问题，处理时间是在作业处理完成后才决定的，根据我们的算法，我们将用确定值代替随机变量，得到一个合理的调度方案，为了使得该调度方案既具有鲁棒性又可以有效缩短调度时间，调度策略的选择非常重要。在我们的调度系统中我们最终将使用插空策略动态调整调度方案。在本小节我们将给出几种调度策略的说明。

为了说明的方便，我们给出一个处理时间不确定条件下的作业车间问题 $J1 = (m_1, 10) \prec (m_3, [2, 4]) \prec (m_4, 5)$ $J2 = (m_2, [2, 8]) \prec (m_3, 7)$，在该问题中只涉及到两个不确定量，工序$J_{12}$和工序$J_{21}$的处理时间，我们用 $d = (d_1, d_2) \in [2,4] \times [2,8]$ 表示该调度问题的一个实例，实例 (d_1, d_2) 表示在实际加工过程中，最后确定工件1的第二道工序的加工时间是 d_1，而工件2的第一道工序的加工时间是 d_2），比如实例（4，8）表示表示在实际加工过程中，最后确定

工件1的第二道工序的加工时间是4，而工件2的第一道工序的加工时间是8。为了求解的简单和说明的方便，在该问题中只有机器3有争用的情况出现。

为了后面的说明，我们先对于三个实例（4，8），（2，8）和（4，4），根据一定的算法可求得该问题的最优调度方案如图。

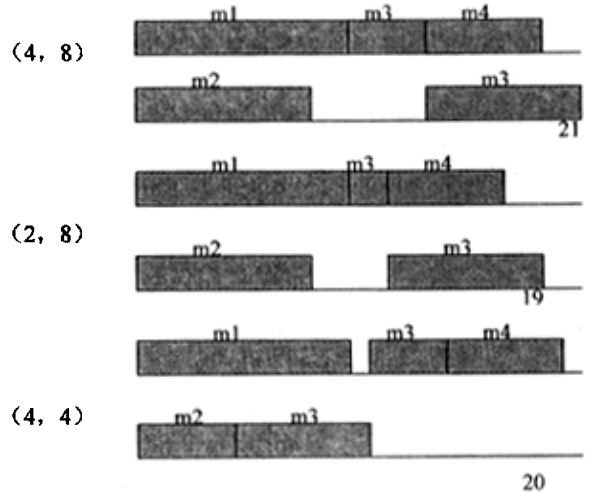


图 6-3 三个实例的最优调度

1．静态调度策略（Static Strategy）

最简单的当然是静态调度策略，即在调度执行过程中调度方案保持不变。使用静态策略初始调度方案的选择非常重要，在调度鲁棒性和调度长度之间有取得比较满意的平衡。

为了使得调度方案在任何情况都是可行的，我们根据最坏情况求出一个调度方案作为初始调度方案，然后在调度执行过程中严格按照该调度方案执行。该方案保证在任何情况下调度结果都满足优先关系约束和资源关系约束，但是在某些情况下会造成对机器时间的浪费。

首先为（4，8）的情况找出一个最优调度，然后在调度执行过程中严格按照该调度策略执行，我们可以看到根据该调度方案对于实例（2，8）和（4，4），在实例执行过程中，机器时间造成了一定的浪费。

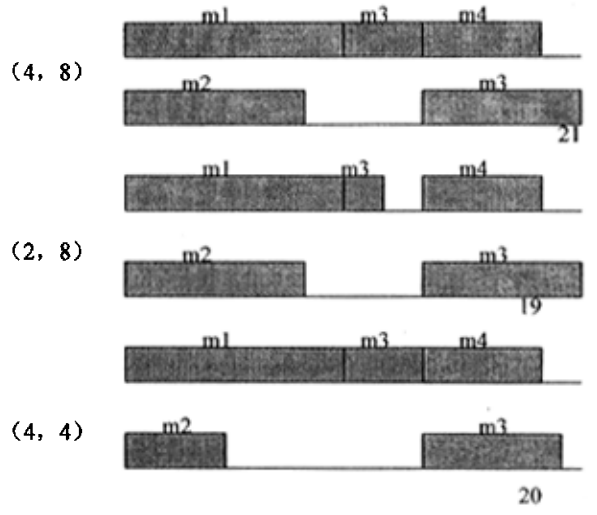


图 6-4 根据最悲观的情况（4，8）给出调度方案，
对所有其他的情况都使用同一调度方案

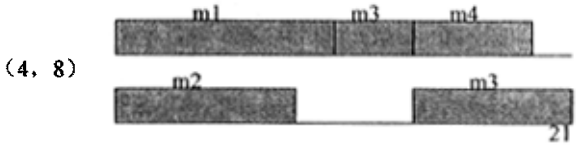
2．自适应的调度策略（Adaptive Scheduling）

自适应调度策略是首先根据一定的条件给出一个初始调度，在调度执行过程中每次一旦有一个含有不确定性因素的工序完工则将剩余的正在进行的工序和还没进行的工序重新调度。

同样对于问题 $J1 = (m_1, 10) \prec (m_3, [2, 4]) \prec (m_4, 5)$ ，

$J2 = (m_2, [2, 8]) \prec (m_3, 7)$

我们首先为（4，8）的情况找出一个最优



如果第二道作业的第一道工序在第4个时间单位的时候就完工的话，我们就得到一个剩余调度问题

$$J_1 = (m1, 6, !) \prec (m3, 4) \prec (m4, 5) \quad J_2 = (m3, 7)$$

"!" 表示第一道作业的第二道工序必须被马上调度。对这个问题而言，最

优调度将马上开始第二个作业的第二道工序。同样，如果第二道作业的第一道工序在8时间单位的时候就完工的话，我们有一个剩余调度问题

$J_1 = (m1, 2, !) \prec (m3, 4) \prec (m4, 5)$ $J_2 = (m3, 7)$

这种方法可以在任何情况下都使得调度时间最短，但是在这种方法中涉及到很多的在线计算（online computation），方法的实时性有一定的局限。

3．插空策略（Hole Filling Strategy）

根据插空策略动态调整调度方案是一个在调度时间和调度计算之间取得一定平衡的方法。首先根据某种确定性情况求解出一个调度方案，从该调度方案中抽取出每个机器上的争用该机器的工序的排列顺序。在我们的例子中先根据最坏情况（4，8）得到一个最优调度方案，得到在被争用的机器m3上J1 < J2 ，然后在调度执行的过程中，一旦一个工序完工就将它的下一道工序开始，具体过程如下图。采用插空策略可以保证调度方案的鲁棒性，即在任何情况下，调度方案都满足作业优先约束和资源约束，而且在一般情况下可以有效地缩短调度时间，而不增加过多的计算量。

当一个调度实例的最优解中机器上工序的加工顺序和我们通过静态求解求解出来的初始调度方案中的机器上的工序的加工顺序一致时，运用插空策略可以有效地缩短调度时间，但是当两者不同时，使用插空策略对最优调度而言可能并不会有效地缩短调度时间。比如对于实例（2，8）使用插空策略在调度过程中动态调整调度方案可以有效地缩短调度时间，但是对于实例(4, 4)，我们很明显可以看到，使用插空策略可以缩短调度时间，但是我们可以看到和图6-3给出的实例(4, 4)的最优调度相比，调度时间还是过长。

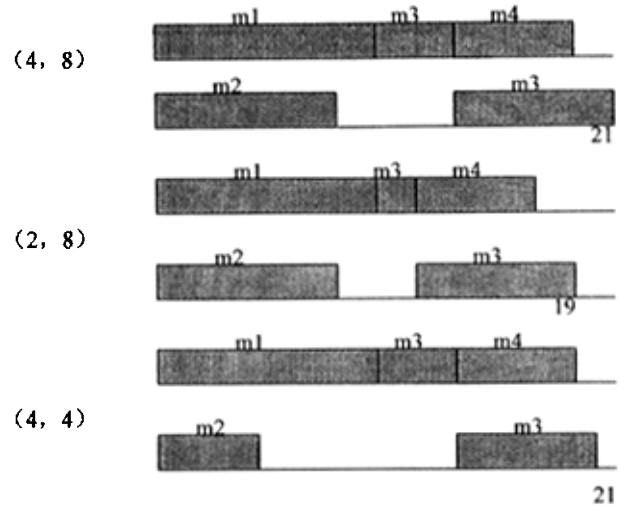


图 6-5 根据插空策略调整调度后的最终调度

6.2.2 调度算法

为了求解处理时间不确定条件下的作业车间问题，我们以确定性作业车间问题的求解为基础，给出近似最优的调度结果，在调度过程中采用一定的调度策略（Scheduling Strategy），每个作业处理结束时，根据该调度策略调整调度结果而不是进行重调度（Re-Schedule）。整个算法分成两部分，静态调度方案求解部分和调度方案动态调整部分。

1．静态调度方案求解

在静态调度方案求解部分，为了得到一个初始的调度方案，而且保证该调度方案的鲁棒性，我们选择该处理时间不确定条件下的作业车间问题的最差实例，运用第五章提出的分离逻辑求解作业车间问题的方法为该实例求得一个调度方案，将该调度方案作为初始调度方案（算法如图 5-2），显然该调度方案具有鲁棒性。静态求解具体算法框图如图 6-6。

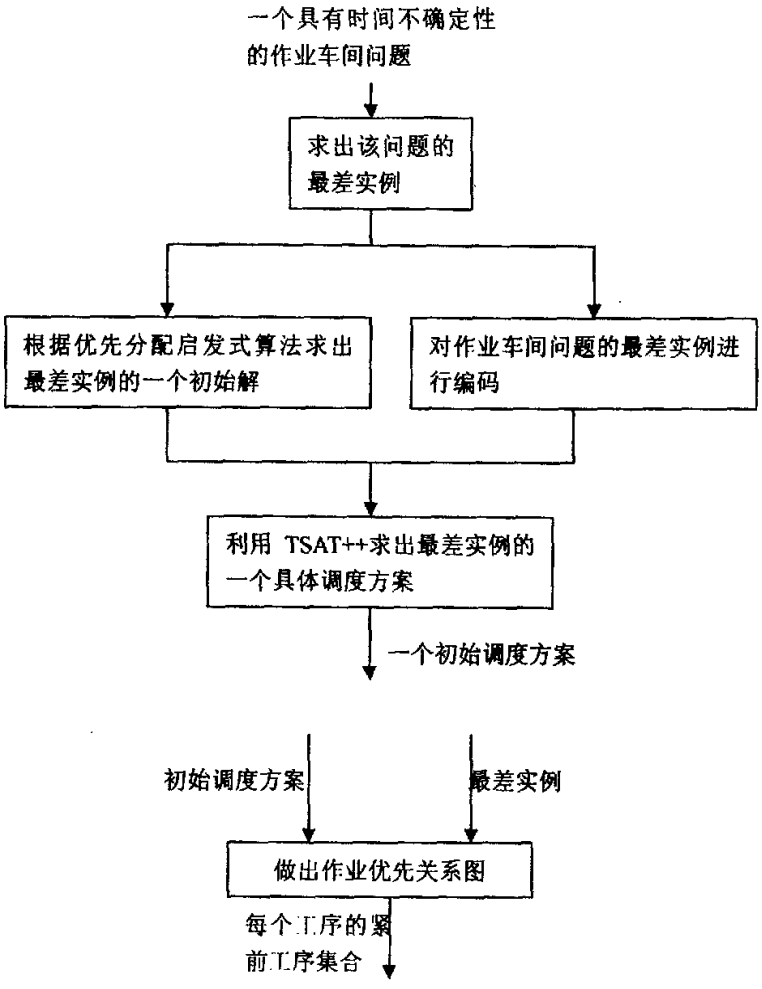


图 6-6 静态调度算法框图

2. 调度方案动态调整

因为我们要解决的问题是处理时间不确定性条件下的作业车间问题，在该问题中，处理时间是在作业加工过程中才确定下来的。为了使得在静态调度方案求解阶段求得的调度方案适应处理时间的动态变化，在每个工序的处理时间确定下来时根据插空调度策略调整调度方案，使得新的调度方案既满足问题的各项约束，又可以缩短整个作业车间问题的处理时间。调整算法框架如图 6-8。

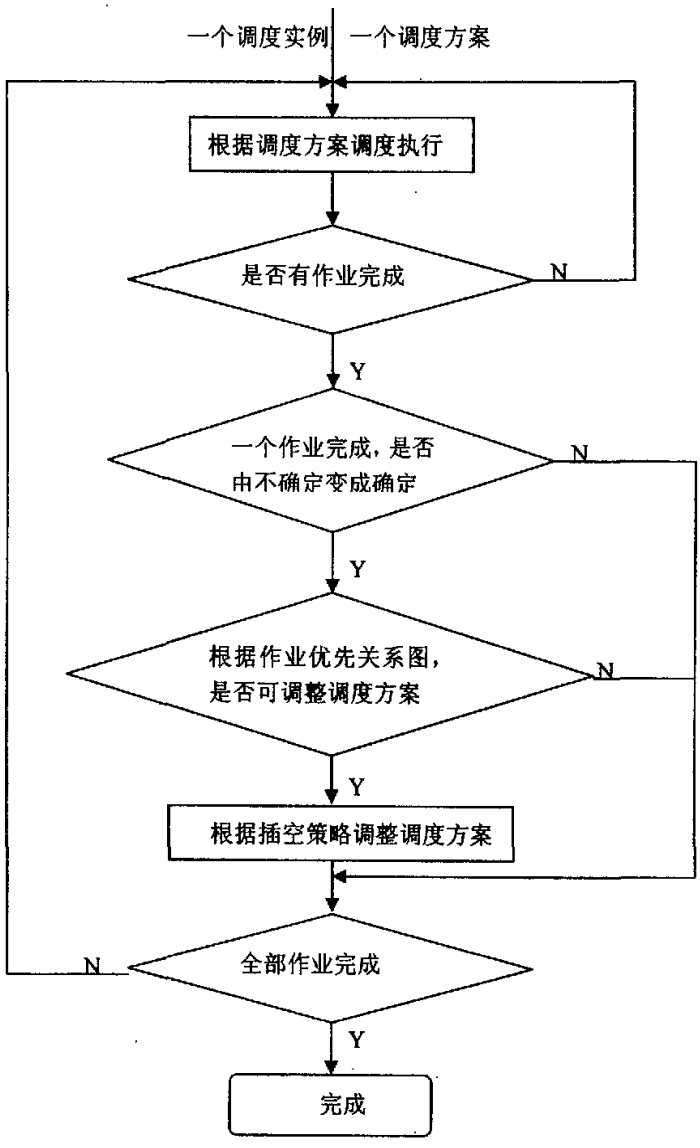


图 6-8 动态调整算法框图

算法 6.1：根据一个初始调度方案 $S(P)$ 求作业的紧前工序集合 $\prod(t)$

输入：调度结果 $S(P)=\{s(t_1),s(t_2),...,s(t_{mn})\}$，作业车间问题描述文件；

输出：工序 t 的紧前工序集合 $\prod(t)$；

$S(P)=\{s(t_1),s(t_2),...,s(t_{mn})\}$：问题 P 一个合理的调度方案，为每个工序 t_i 指定开工时间 s_i，该调度方案根据第五章中的算法求得，其中作业顺序按照开工时间的先后排列；

$\prod(t)$：时刻工序 t 的紧前工序集合；

$C_{n\times m}=\{c(i,j):s(t_i)\prec s(t_j)\}$：根据作业定义和调度结果得到的作业优先关系图中的偏序关系，矩阵表示，$c(i,j)=\begin{cases}1 & \text{作业}i\text{必须在作业}j\text{之前完成}\\ 0 & \text{两者之间没有优先关系}\end{cases}$，

$TS(m_i)$：占用机器 m_i 的作业集合，按调度结果 $S(P)$ 中作业开工顺序集合排列；

步骤 1：将作业车间问题定义中给出的作业优先关系（Sequence Constraints）加入 $C=\{c(i,j):s(t_i)\prec s(t_j)\}$

步骤 2：令 k=1；初始 $TS(m_i)$ 为空；

步骤 3：从 $S(P)=\{s(t_1),s(t_2),...,s(t_{mn})\}$ 取出 $S(k)$，将 $S(k)$ 加入 $TS(m_i)$，其中作业 $S(k)$ 占用机器 m_i；

步骤 4：如果 $k<n\times m$ 转到步骤 2；

步骤 5：根据 $TS(m_i)$ 中作业排列顺序，将作业优先关系加入 $c(i,j):s(t_i)\prec s(t_j)$；

步骤 6：根据 $c(i,j)$ 求得 $\prod(t)=\{t':t'\prec t\}$

图 6-7 算法 6.1 求作业的紧前工序集合

算法 6.2 调度方案的动态调整算法

输入：一个处理时间不确定条件下作业车间问题，该问题的一个初始调度方案 $S(P)=\{s(t_1),s(t_2),...,s(t_{mn})\}$ 和对应于该调度的每个工序的紧前工序集合。

输出：该处理时间不确定条件下作业车间问题在实际过程中的每个作业的确定执行时间

$S(P)=\{s(t_1),s(t_2),...,s(t_{mn})\}$：问题 P 的一个合理调度方案，为每个工序 t_i 指定开工时间 s_i，该调度方案根据第五章中的算法求得，其中作业顺序按照开工时间的先后排列，该调度方案根据第五章中的算法求得；

$\prod(t_i)$：工序 t_i 的紧前工序集合，由算法 6.1 求得；

$J(t)$：时刻 t 还没开工的工序集合；

$P(t)$：已开工工序集合，该集合记录每个已开工工序的剩余完工时间；

$m(t)$：时刻 t 空闲机器集合；

步骤 1：随机生成输入的处理时间不确定条件下作业车间问题的一个实例；

步骤 2：令 t=0，初始 $m(t)$ 为所有机器，初始 $J(t)$ 为所有工序；

步骤 3：根据 $S(P)$ 看是否有工序开工，如果有，则按如下步骤更新如下数据：

1）占用工序 t_i 占用的机器 m_{ti}，将 m_{ti} 从 $m(t)$ 中删除；

2）从 $J(t)$ 中删除 t_i；

步骤 4：根据调度实例看是否有工序完工，如果有工序 t_i 完工，则按如下步骤更新数据：

1）释放工序 t_i 占用的机器 m_{ti}，将 m_{ti} 加入 $m(t)$；

2）将 t_i 从 $P(t)$ 中删除；

3）将 t_i 从所有 $\prod(t_j)$ 中删除；

4）对每个还没开工的工件 t_i，如果 $\prod(t_i)=\phi$ 且 t_i 要求的机器 m_k 空闲，则 t_i 开工，做如下更新

为 t_i 指定新的开始时间，更新 $S(P)$；

从 $J(t)$ 中删除 t_i；

从 $m(t)$ 中删除 t_i 占用的机器 m_k

步骤 5 ：令 t=t+1；并做如下更新：

如果 $P(t)\neq\varnothing$ 则 $P(t)$ 中每个工序的剩余完成时间减 1；

否则 返回步骤 3；

图 6-9 算法 6.2 调度方案的动态调整算法

6.3 实验结果

为了测试在不同情况下，插空策略对调度长度的影响，我们选用的测试用例为一个 6*6 具有时间不确定性的作业车间问题，该问题具体描述如下：

$J1$ ：$(m5,[2,10]) \prec (m0,[3,8]) \prec (m1,[1,6]) \prec (m3,[10,22]) \prec (m4,[12,22]) \prec (m2,[15,30])$

$J2$ ：$(m1,[4,18]) \prec (m2,[8,12]) \prec (m0,[4,10]) \prec (m5,[9,16]) \prec (m3,[21,40]) \prec (m4,[12,41])$

$J3$ ：$(m2,[6,15]) \prec (m4,[13,24]) \prec (m0,[2,9]) \prec (m5,[2,15]) \prec (m1,[8,19]) \prec (m3,[6,10])$

$J4$ ：$(m1,[2,10]) \prec (m4,[7,25]) \prec (m2,[1,7]) \prec (m3,[21,30]) \prec (m5,[11,24) \prec (m0,[8,21])$

$J5$ ：$(m4,[15,39]) \prec (m6,[7,11]) \prec (m1,[5,7]) \prec (m5,[14 ,33]) \prec (m2,[9,21]) \prec (m3,[9,13])$

$J6$ ：$(m1,[11,23]) \prec (m3,[8,20]) \prec (m5,[8,15]) \prec (m2,[5,19]) \prec (m4,[7,14]) \prec (m0,[3,10])$

我们随机生成该处理时间不确定条件下的作业车间问题的 40 个实例问题来模拟实际情况中将会出现的实际处理时间。

针对每个问题，根据本文提出的方法使用插空调度策略动态地执行调度，得到对应于一个具体的实例问题的调度执行的长度。作为对比，我们默认 40 个实例问题都是一个具体的静态的作业车间问题，分别对 40 个实例问题都用第五章的方法求出一个好的调度结果的长度，另外还求出了静态调度策略条件下的调度执行的长度（即调度方案在调度执行过程中不作调整）。

在三种情况下调度执行的长度对比结果如表 6-4，表的第一列为实例编号；表的第二列为对于该实例问题，如果把它作为一个静态的作业车间问题可以求得的较优的调度方案的执行时间；表的第三列为使用静态调度策略，调度执行时间，第四列为和静态情况下相比，使用静态调度策略策略调度执行时间多出的百分比；表的第五列为使用插空策略的调度执行时间，第六列为和静态情况下相比使用插空调度策略策略调度执行时间长出的百分比。

对于实例 1，静态情况下一个较好的调度执行时间为 172，而对于处理时间不确定条件下使用静态策略动态调整调度结果则调度执行时间为 210，比该执行实例静态情况的调度长 22.81%；同样对于该情况使用插空策略的话则调度执行时间只为 172，只比静态情况下执行时间多出 0.58%。明显地，对于该实例，使用插空策略可以有效地缩短调度执行时间，我们在 6.2.1 中曾经分析过当一个调

度实例的最优解中机器上工序的加工顺序和我们通过静态求解求解出来的初始调度方案中的机器上的工序的加工顺序一致时，运用插空策略可以有效地缩短调度时间，该实例应该属于这种情况。同样表 6.4 中实例 6，12，16 都属于该种情况。

对于实例 40，使用静态调度策略调整调度方案得到的具体执行时间比静态调度问题的较优调度时间长 22.60%，使用插空策略则长 17.57%，两者相差不多，可见在实例 40 的情况下插空策略对于调度长度的影响不是那么大。我们在 6.2.1 中分析过当一个调度实例的最优解中机器上工序的加工顺序和我们通过静态求解求解出来的初始调度方案中的机器上的工序的加工顺序不同时，使用插空策略对最优调度而言可能并不会缩短调度时间。实例 40 属于这种情况。

平均而言，使用静态调度策略调度执行时间比静态情况下的调度长 32.70%，而使用插空策略调度执行时间比静态情况下的调度仅长 6.29%，

从表 6-4 可以看到在一般情况下使用插空策略调整调度方案都可以在一定范围内有效地缩短调度执行时间。

这些测试结果是在 1.8GHz 的 Pentium 4 处理器，512 兆内存的硬件环境和 Linux RedHat 9 的软件环境下取得的。

表 6-4 实验结果对比

No	Good	Schedule With Stacic	%	Schedule With Holefilling	%
1	171	210	22.81	172	0.58
2	145	203	40.00	148	2.07
3	178	208	16.85	189	6.18
4	156	215	37.82	168	7.69
5	146	202	38.35	154	5.48
6	169	208	23.08	170	0.59
7	166	211	27.11	174	4.82
8	173	205	18.50	180	4.05
9	167	214	28.14	181	8.38
10	164	202	23.17	169	3.05
11	145	210	44.83	156	7.59
12	164	197	20.12	165	0.61

13	165	208	26.06	174	5.45
14	161	221	37.27	177	9.93
15	173	213	23.12	179	3.47
16	187	207	10.69	188	0.53
17	151	223	47.68	169	11.92
18	156	212	35.89	163	4.49
19	162	212	30.86	169	4.32
20	159	204	28.30	170	6.92
21	144	214	48.61	154	6.94
22	158	208	31.65	172	8.86
23	143	211	47.56	158	10.48
24	168	212	26.19	175	4.17
25	154	213	38.31	164	6.49
26	156	205	31.41	167	7.05
27	141	202	43.26	147	4.26
28	164	208	26.83	184	12.19
29	153	213	39.22	159	3.92
30	151	207	37.09	157	3.97
31	160	209	30.63	176	10
32	158	221	39.87	186	17.72
33	142	218	53.52	158	11.26
34	146	223	52.74	169	15.75
35	167	213	27.55	177	5.98
36	168	218	29.55	184	9.52
37	153	208	29.76	161	5.23
38	158	213	35.95	179	13.29
39	177	217	34.81	183	3.39
40	148	219	22.60	174	17.57
Avg	159	211	32.70	169	6.29

第七章 总结和进一步研究

本文研究了一类具有时间不确定性的作业车间问题，以古典的作业车间问题为背景，加入处理时间不确定性因素——处理时间在一个闭区间的范围内变化。为了求解含有该类不确定性时间因素的作业车间问题，我们以确定性的作业车间问题的求解为基础，对一个确定性的作业车间问题进行分离逻辑建模，使用分离逻辑求解器求出一个近似最优的调度结果，在调度执行过程中使用插空策略（Hole-Filling Strategy），在每个作业处理结束时，根据该调度策略动态地调整调度方案，缩短生产时间。并通过实验证明了利用分离逻辑求解确定性的作业车间问题的有效性，也证明了在一般情况下使用插空策略动态调整调度结果可以有效地缩短调度执行时间。

在未来的工作中需要进一步解决的问题如下：

1. 利用分离逻辑对作业车间问题进行求解，有一定的局限性。当问题规模增大时，求解时间增加速度很快。甚至无法在指定时间内求出一个合理的解。
2. 对于使用插空调度策略对调度长度的影响，我们只作一个简单的分析，认为当一个调度实例的最优解中机器上工序的加工顺序和我们通过静态求解求解出来的初始调度方案中的机器上的工序的加工顺序一致时，运用插空策略可以有效地缩短调度时间，但是当两者不同时，使用插空策略对最优调度而言可能并不会有效地缩短调度时间。并没有从理论上深刻地分析这个这个问题。
3. 在我们的求解方法中，没有研究不确定性因素对调度长度的影响，所以一旦有不确定性因素的确定下来，我们只是对调度结果做一个简单的修改，并没有进行重调度，在同一机器上作业的加工顺序是保持不变的。但是在有的情况下，某些不确定性因素的确定会对机器上剩余作业的加工顺序产生影响，在这种情况下，使用插空策略并不能有效缩短调度的长度，而是需要进行重调度。所以在未来的研究中可以研究不同情况下，采取不同的调度策略，将插空策略和自适应的调度策略结合起来。

参考文献

1. 王书锋，邹益仁. 车间作业调度技术问题简明综述，系统工程理论与实践，2003 年 1 月
2. Johnson S. Optimal two-and-three stage production schedules with setup times included [J].Naval Research Logis2tics Quarterly, 1954, 1: 61-68.
3. Akers SB. A graphical approach to production scheduling problem s [J].Operations Research, 1956, 4:244-245.
4. Hefetz N, A diri I. An efficient optimal algorithm for the two-machines unit-time job-shop schedule-length problem[J].Mathematics of Operations Research, 1982,7:354-360.
5. Jackson J R. Scheduling a Production Line to Minimize Maximum Tardiness, Research Report 43, Management Science Research Projects[R]. Los Angeles, USA: University of California, 1955.
6. Smith W E. Various optimizers for single stage production [J].Naval Research Logistics Quarterly, 1956, 3:59-66.
7. Crawford, J, and Baker, A. Experimental results on the application of satisfiablity algorithms to scheduling problems. Proc. 12th National Conf. On AI. 1092-1097 1994.
8. Marco Cadoli, and Andrea Schaerf, Compiling Problem Specifications into SAT, European Symposium on Programming(ESOP 2001) , Genova, Italy, April 2-6,2001
9. Van Laarhooven P J M, A arts E H L, Lenstra J K. Job shop scheduling by simulated annealing [J]. Operations Research, 1992, 40 (1): 113- 125.
10. Kolonko M. Some new results on simulated annealing app lied to the job shop scheduling problem [J]. European Journal of Operational Research, 1999, 113: 123- 136.
11. 翁妙凤. 解 Job shop 调度问题的混合模拟退火进化规划[J]. 信息与控制, 1999, 28 (2) : 81- 84.
12. F. Glover. Tabu Search: Part1 and 2, ORSA Journal on Computing, (1989-1990)
13. Laguna M , Barnes JW , Glover F. Tabu search methods for a single machine scheduling problem [J]. Journal of Intelligent Manufacturing, 1991, 2: 63- 74.;
14. Laguna M, Barnes J W , Glover F. Intelligent scheduling with tabu search: an application to jobs with linear delay penalties and sequence dependent setup costs and times[J]. Journal of Applied Intelligence, 1993, 3: 159- 172.

15. Taillard E. Parallel taboo search techniques for the job shop scheduling problem [J]. ORSA Journal on Computing, 1994, 16 (2): 108- 117.
16. Nowicki E, Smutnick i C. A fast taboo search algorithm for the job-shop problem [J]. Management Science, 1996, 42 (6): 797- 813.
17. Taillard E. Parallel taboo search techniques for the job-shop scheduling problem [J]. ORSA Journal on Computing, 1994, 16 (2) : 108- 117.
18. 玄光男, 程润伟. 遗传算法与工程设计[M]. 北京: 科学出版社, 2000.
19. Cheng R, GenM, Tsujimura Y. A tutorial survey of job-shop scheduling problems using genetic algorithm. Representation [J]. Computers & Industrial Engineering, 1996, 30 (4): 983- 997.
20. Davis L. Job-shop scheduling with genetic algorithm [A]. Grefenstet te J J. Proceedings of the 1st International Conference on Genetic Algorithm s and Their Applications, Pittsburgh [C]. PA, USA, Lawrence, Erlbaum, 1985, 136- 140.
21. Falkenauer E, Bouffouix S. A genetic algorithm for the job shop [A]. Proceedings of the IEEE International Conference on Robotics and Automation, Sacramento [C], California, USA , 1991, 824- 829.
22. Tamak i H, Nishikaw a Y. A paralleled genetic algorithm based on a neighborhood model and its application to the job-shop scheduling [A]. Manner, R and Manderick B. PPSN ' 2 Proceedings of the 2nd International Work shop on Parallel Problem Solving from Nature, Brussels, Belgium , 1992, 573- 582.
23. Bean J. Genetic algorithms and random keys for sequencing and optimization [J]. ORSA Journal on Computing, 1994, 6 (2): 154- 160.
24. 曹承煜, 李人厚, 樊健. 车间调度算法的研究开发[J]. 控制理论与应用, 2000, 17 (1) : 31- 34.
25. 纪树新, 钱积新, 孙优贤. 遗传算法在车间作业调度中的应用[J]. 系统工程理论与实践, 1998, 5: 34- 40.
26. 王海英, 王凤儒, 柳崎峰. 用定界遗传算法解有交货期的非标准 Job2shop 调度问题[A]. Proceedings of the 3thWorld Congress on Intelligent Control and Automation[C]. China, 2000, 532- 636.
27. 顾幸生, 不确定性条件上的生产调度, 华东理工大学学报, 2000
28. Yamamoto M, Nof S F. Scheduling/rescheduling in the manufacturing operating system environment [J]. Int J Prod Res,1985, 23 (4) : 7052722.; Mignon D J ,
29. Honkomp S J, Reklaitis G V. A framework for investigating schedule robustness under uncertainty [J]. Computers and Chemical Engineering, 1995, 19 (Supp l): S6152620.

30. Pistikopoulos E N. Uncertainty in process design and operations [J]. Computers and Chemical Engineering, 1995, 19(Supp l): S5532563.

31. Honkomp S J, Reklaitis G V. Robust scheduling with processing time uncertainty [J]. Computers and Chemical Engineering, 1997, 21 (Supp l): S1 05521 060.

32. Honkomp S J, Mockus L, Reklaitis G V. A framework for schedule evaluation with processing uncertainty [J]. Computers and Chemical Engineering, 1999, 23: 5952609.

33. Schmidt CW, Grossmann I E. A mixed integer programming model for stochastic scheduling in new product development [J]. Computers and Chemical Engineering, 1996, 20 (Supp l):S1 23921 244.

34. Pistikopoulos E N. Uncertainty in process design and operations [J]. Computers and Chemical Engineering, 1995, 19(Supp l): S5532563.

35. Honkomp S J, Reklaitis G V. Robust scheduling with processing time uncertainty [J]. Computers and Chemical Engineering, 1997, 21 (Supp l): S1 05521 060.

36. Schmidt CW, Grossmann I E. A mixed integer programming model for stochastic scheduling in new product development [J]. Computers and Chemical Engineering, 1996, 20 (Supp l):S1 23921 244.

37. Yasmina Abdedda¨ım, Eugene Asarin and Oded Maler , On Optimal Scheduling Under Uncertainty (2003)

38. 李明切，间歇生产过程鲁棒调度策略的研究[D]. 上海: 华东理工大学, 1998.

39. 刘琦，顾幸生，基于模糊规划的处理时间不确定性条件下的 Job Shop 问题，华东理工大学学报，2001 年 10 月

40. 李明切，间歇生产过程鲁棒调度策略的研究[D]. 上海: 华东理工大学, 1998.

41. Pistikopoulo s E N. Uncertainty in process design and operations [J]. Computers and Chemical Engineering, 1995, 19(Supp l): S5532563.

42. Lee, E S, Reklaitis G V. Intermediate storage and operation of batch process under batch failure [J]. Computers and Chemical Engineering, 1989, 13 (4ö5): 4912498.

43. Lee, E S, Reklaitis G V. Intermediate storage and the operation of periodic process under equipment failure [J]. Computers and Chemical Engineering, 1989, 13 (11-1 2) : 1 23521 243.

44. Daeho K, Moon I. Rescheduling algorithm s in case of unitfailure fo r batch process management [J]. Computers and Chemical Engineering. 1997, 21 (Supp l):

S1 06721 072.

45. 王玮,汪定伟,王晓琦，基于模糊交货期的单件制造业准时化生产计划[J]. 系统工程学报,1998,13(2):63269.

46. 王朝晖,甘文泉,陈浩勋等.具有模糊缓冲库存约束的化工批处理过程的调度[J].系统工程理论与实践,1998,7:62268.

47. 李建更. 涂峯生, 某些调度问题区间摄动鲁棒性的研究，自动化学报，2001 年 01 期

48. 李建更，涂峯生，一类 Flow Shop 调度问题最优调度区间摄动鲁棒性，控制理论与应用，2004 年 02 期

49. 顾幸生, 李明切. Flow shop 调度问题的鲁棒性初探[J]. 信息与控制, 1999, 26 (增刊) : 3422345.

50. Honkomp S J, Reklaitis G V. Robust scheduling with processing time uncertainty [J]. Computers and Chemical Engineering,1997,21(Supp):S105521060

51. Honkomp S J, Mockus L, Reklaitis G V. A framework for schedule evaluation with processing uncertainty [J], Computers and Chemical Engineering, 1999, 23: 5952609

52. S.Smith. A methodology and architecture for reactive scheduling, In Intelligent Scheduling, Morgan Kaufman, 1994

53. 方剑，席裕庚。周期性和时间驱动的 Job shop 滚动调度策略，控制力量与应用，14(4):589-594,1997

54. Masatoshi S, Tetsuya M. An efficient genetic algorithm for job- shop scheduling problems with fuzzy processing time and fuzzy due date[J] . Computers & Industrial Engineering, 1999, 36(4): 325 —344.

55. Crawford, J, and Baker, A. 1994. Experimental results on the application of satisfiablity algorithms to scheduling problems. Twelfth National Conference on Artificial Intelligence vol.2, AAAI Press, 1994, pp. 1092–1097.

56. Marco Cadoli, and Andrea Schaerf, Compiling Problem Specifications into SAT, European Symposium on Programming(ESOP 2001), Genova, Italy, April 2-6,2001, Lecture Notes in Computer Science vol. 2028, Springer Verlag, 2001

57. Steven Prestwich and Colin Quirke, Boolean and Pseudo-Boolean Models for Scheduling;

58. R.J.Bayardo,R.C. Schrag. Using CSP Look-Back Techniques to Solve Real-World SAT Instances. Fourteenth National Conference on Artificial Intelligence, 1997, pp. 203–208.

59. C.-M. Li. Integrating Equivalency Reasoning into Davis-Putnam Procedure. Seventeenth National Conference on Artificial Intelligence, Austin, Texas, USA, 2000, pp. 291–296.

60. P. Baumgartner, F. Massacci. The Taming of the (X) OR. First International Conference on Computational Logic, Stream on Automated Deduction: Putting Theory into Practice, Lecture Notes in Artificial Intelligence vol. 1861, Springer-Verlag, 2000, pp. 508–522

61. J. Whittemore, J. Kim, K. Sakallah. SATIRE: A New Incremental Satisfiability Engine.Thirty-Eighth Design Automation Conference, 2001, pp. 542–545.

62. M. R. Dransfield, V. W. Marek, M. Truszczynski. Satisfiability and van der Waerden Numbers. Sixth International Conference on Theory and Applications of Satisfiability Testing, Portofino, Italy, 2003, pp. 325-336.

63. R. B´ejar, A. Cabiscol, C. Fernandez, F. Many`a, C. P. Gomes. Capturing Structure with Satisfiability. Seventh International Conference on Principles and Practice of Constraint Programming,Lecture Notes in Computer Science vol. 2239, Springer-Verlag, 2001, pp. 137–152.

64. M. L. Ginsberg, A. J. Parkes. Satisfiability Algorithms and Finite Quantification. Seventh International Conference on Principles of Knowledge Representation and Reasoning, Breckenridge,Colorado, USA, 2000.

65. F. A. Aloul, A. Ramani, I. Markov, K. Sakallah. PBS: a Backtrack-Search Pseudo-Boolean Solver and Optimizer. Fifth International Symposium on Theory and Applications of Satisfiability Testing, Cincinnati, Ohio, USA, 2002, pp. 346–353.

66. Some Progress in Satisfiability Checking for Difference Logic

67. Ofer Strichman, Sanjit A. Seshia, and Randal E. Bryant. Deciding separation formulas with SAT. Lecture Notes in Computer Science, 2404:209 222, 2002.

68. Alessandro Armando, TSAT++: an Open Platform for Satisfiability Modulo Theories

69. Silvio Ranise, Cesare Tinelli, et al. The SMT library (Satisability Modulo Theory). http://www.smtlib.org.

70. Altisen, K.; Gossler, G.; Pnueli, A.; Sifakis, J.; Tripakis, S.; Yovine, S.;A framework for scheduler synthesis Real-Time Systems Symposium, 1999. Proceedings. The 20th IEEE , 1-3 Dec. 1999 Pages:154 – 163

研究生期间发表的论文和参加的科研项目

发表的论文：

1. Gang Zhou，Yunfei Jiang，Sujun Sun. Planning and Scheduling for the Teams of the Software Project. International Symposium on Computing and Information (ISC&I 2004).
2. Danran Liang, Kangheng Wu, Yunfei Jiang, Sujun Sun, Zhuqiu Ye, Jianxiong Wang. Theory of Separation Predication for Scheduling under Uncertainty in Software Engineering. International Symposium on Computing and Information (ISC&I 2004).

参加的科研项目：

1. 软件研究所 VOD 项目
2. 《软件质量提升系统》，2003年广东省软件邻域关键技术突破项目

致谢

衷心感谢我尊敬的导师姜云飞教授。在研究生三年的学习过程中，他无私的传授我们学术研究的方法，协助我们解决学习、工作中碰到的困难，鞭策我们通过端正的态度、正确的途径去获得学术成果。论文从选题到完稿，都得到姜老师的悉心指导和关怀。姜老师为人谦虚、正直，有着儒雅的学者风范，他的谆谆教诲将使我终生受益。

感谢李磊教授。作为软件所所长，他为我们提供良好的科研环境。李所长风趣幽默，学识广博，对所里的学生高标准要求，永远是我学习的好榜样。

感谢班主任邓克像老师，作为我们的班主任，他热心关心我们的学习和生活，是我们的一位好家长。

感谢范昭赋、毛明志老师和潘嵘、吴康恒师兄。在参与科研究项目的过程中，得到他们的直接指导，获益良多。他们对学术严谨、执着的追求，也使我深深敬佩。

感谢我的同学叶柱秋、谷烽、梁丹然。在共同学习、生活中，从他们身上学习到不少新颖的知识和思维方法，他们的优点和积极的人生态度激励着我不断进步，超越自己。

感谢我的家人。无论在什么时候，他们一直在背后支持着我，在生活上给予了无微不致的关心和爱护，在精神上也给了我很大的鼓励和安慰，使我能够顺利完成学业。

最后，向其他关心、帮助过我的所有老师、同学和各位朋友表示深深的谢意。

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。对本文的研究作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：孙素君

日期：2005 年 4 月 15 日