



# 浙江工业大学

## 毕业论文（毕业设计说明书）

题目 多品种装配车间调度研究

专    业： 工业工程

班    级： 健行 1001

学生姓名： 陈晟恺

指导老师： 鲁建厦、董巧英

2013 – 2014 学年

# 浙江工业大学

## 学位论文原创性声明

本人郑重声明：所提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研究成果。除文中已经加以标注引用的内容外，本论文不包含其他个人或集体已经发表或撰写过的研究成果，也不含为获得浙江工业大学或其它教育机构的学位证书而使用过的材料。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。本人承担本声明的法律责任。

作者签名：

日期： 年 月 日

# 多品种装配车间调度研究

学生姓名：陈晟恺



指导老师：鲁建厦、董巧英

浙江工业大学健行学院

## 摘 要



~~汽车电子零部件企业的装配车间普遍采用主机厂对应的专线来安排生产，~~  
存在诸多弊端，突破主机厂限制是有效的解决方案，这是一个多品种多装配  
线轮番调度优化问题。

建立了 2 个适用不同管理水平的调度模型，~~提出了~~虚拟序列的概念，将  
之与调度规则及启发式策略和设计了 Cyc 和 Vtr 类多个求解算法。计算实验  
结果表明：Vtr – Tabu 算法适用于中小规模的不考虑插单情况的问题，且随着  
工期目标的重视而凸显改进效果，VVT 算法的求解结果在多种不同决策环境  
下都显示出了较高的质量和稳定性。所提算法求解该模型是有效果的，所建  
模型也适合该问题。虚拟序列的提出是本文的创新点，其应用范围很广，在  
建模方面可以进一步考虑混流生产。

**关键词：**多品种多装配线，调度模型，虚拟序列，算法设计

## **A study of multi job in assembly shop**

Author: CHEN Sheng-kai

Mentor: LU Jian-sha, Dong Qiao-ying

Jianxing Honor College, Zhejiang University of Technology

### **Abstract**

Multi-type massive production usually takes part in the assembly line of electronic parts for auto-mobile, however, a myopic way that specialize assembly line by the main factories is widely used, so that low utilization , high redundancy and unbalance production always happens. A sound solution is to despecialize the assembly line, the implementation of this solution is a so called multi-type multi-assembly-line take-turn scheduling problem. In this study, 2 mathematical models is built for different levels of managenent to apply. With proposed concept of *virtual list*,

**Keywords:** multi-type, multi-assembly-line, scheduling model, virtual list, algroithm design

# 目 录

摘 要 .....	I
Abstract .....	II
第一章 绪论 .....	1
1.1 课题研究的背景及意义 .....	1
1.2 国内外研究现状 .....	1
1.3 课题的基本思路及技术路线 .....	3
第二章 研究对象现状与分析 .....	4
2.1 公司基本情况 .....	4
2.2 装配车间作业 .....	4
2.3 产线调度现状 .....	5
2.4 生产线问题分析 .....	5
2.4.1 现行流程描述 .....	5
2.4.2 现行调度方案 .....	6
2.4.3 主要问题 .....	6
2.5 方案改进 .....	7
2.5.1 改进设计 .....	7
2.6 小结 .....	7
第三章 多品种装配车间调度建模 .....	8
3.1 建模准备 .....	8
3.1.1 基本符号说明 .....	8
3.1.2 基本 $\alpha$ 域 .....	8
3.1.3 基本 $\beta$ 域 .....	9
3.1.4 基本 $\gamma$ 域 .....	9
3.2 模型 1：多品种多装配线轮番装配调度优化模型 .....	9
3.2.1 基本假设 .....	9
3.2.2 目标函数 .....	10
3.2.3 约束条件 .....	10
3.3 模型 2：考虑插单的多品种多装配线轮番装配调度优化模型 .....	11
3.3.1 相关符号及说明 .....	11
3.3.2 相关假设 .....	12
3.3.3 目标函数 .....	12
3.3.4 约束条件 .....	14
3.4 小结 .....	14
第四章 算法设计 .....	15
4.1 分派规则 .....	15
4.1.1 基本规则 .....	15
4.1.2 复合规则 .....	15
4.2 模型 1 求解分析 .....	16
4.2.1 订单处理时间计算 .....	16
4.2.2 模型初始解构造 .....	17
4.2.3 解的改进 .....	18

4.3 模型 2 求解分析 .....	21
4.3.1 模型初始解构造 .....	22
4.3.2 解的改进 .....	23
4.4 算法设计 .....	24
4.4.1 Cyc – ATC 算法 .....	24
4.4.2 Cyc – ATCS 算法 .....	24
4.4.3 Cyc – Tabu 算法 .....	25
4.4.4 Vtr – Tabu 算法 .....	26
4.4.5 VVT 算法 .....	26
4.5 小结 .....	27
第五章 计算实验及算法评估 .....	28
5.1 实验设计 .....	28
5.1.1 生产装配信息生成 .....	28
5.1.2 相关参数确定 .....	28
5.2 结果及评估 .....	30
5.2.1 实验结果示例及说明 .....	30
5.2.2 模型 1 求解结果与分析 .....	32
5.2.3 模型 2 求解结果及分析 .....	33
5.3 小结 .....	34
第六章 总结和展望 .....	35
6.1 总结 .....	35
6.2 展望 .....	35
参考文献 .....	36
附录 A 实验结果图表 .....	38
附录 B Python 脚本 .....	51
致    谢 .....	61

# 图表目录

图 1-1 基本思路 and 关键技术路线 .....	3
图 2-1 现行订单信息流 .....	4
图 2-2 3 条生产线的现行调度 .....	5
图 2-3 3 条产线的混线装配生产示意 .....	7
图 4-1 订单的作业处理有向图 .....	17
图 4-2 4 项作业的初始调度 $S^{(1)}$ 示例 .....	18
图 4-3 $S^{(1)}$ 的 3 个相邻调度 .....	19
图 4-4 流水线的初始调度 $S_l^{(1)}$ .....	19
图 5-1 100 件订单的目标函数值和相关参数的关系 .....	29
图 5-2 20 件订单的三种算法求解模型 1 所得结果 .....	31
图 5-3 决策参数和目标函数值关系示例 .....	32
图 5-4 不同流水线数量和流水线均衡率关系 .....	33
图 5-5 决策环境和流水线均衡率关系 .....	34
图 A-1 模型 1 的 Cyc – ATC、Cyc – Tabu、Vtr – Tabu 算法求解目标函数值比较 ( $\lambda_1 = 0.4$ ) .....	39
图 A-2 模型 1 的 Cyc – ATC、Cyc – Tabu、Vtr – Tabu 算法求解目标函数值比较 ( $\lambda_1 = 0.5$ ) .....	40
图 A-3 模型 1 的 Cyc – ATC、Cyc – Tabu、Vtr – Tabu 算法求解目标函数值比较 ( $\lambda_1 = 0.6$ ) .....	41
图 A-4 模型 2 的 Cyc – ATCS、Cyc – Tabu、VVT 算法求解目标函数值比较 ( $\lambda_1 = 0.4$ ) .....	43
图 A-5 模型 2 的 Cyc – ATCS、Cyc – Tabu、VVT 算法求解目标函数值比较 ( $\lambda_1 = 0.5$ ) .....	45
图 A-6 模型 2 的 Cyc – ATCS、Cyc – Tabu、VVT 算法求解目标函数值比较 ( $\lambda_1 = 0.6$ ) .....	46
图 A-7 模型 2 的 Cyc – ATCS、Cyc – Tabu、VVT 算法求解流水线均衡率比较 ( $\lambda_1 = 0.4$ ) .....	48
图 A-8 模型 2 的 Cyc – ATCS、Cyc – Tabu、VVT 算法求解流水线均衡率比较 ( $\lambda_1 = 0.5$ ) .....	49
图 A-9 模型 2 的 Cyc – ATCS、Cyc – Tabu、VVT 算法求解流水线均衡率比较 ( $\lambda_1 = 0.6$ ) .....	50
表 2-1 制造二部装配车间信息 .....	5
表 5-1 20 件订单的实验信息数据 .....	29
表 5-2 算法相关参数确定 .....	30
表 5-3 订单量 $n = 20$ 的模型 1 3 种算法求解结果 .....	32
表 A-1 Cyc – ATC、Cyc – Tabu、Vtr – Tabu 算法求解模型 1 目标函数值 ( $\lambda_1 = 0.4$ ) .....	38
表 A-2 Cyc – ATC、Cyc – Tabu、Vtr – Tabu 算法求解模型 1 目标函数值 ( $\lambda_1 = 0.5$ ) .....	38
表 A-3 Cyc – ATC、Cyc – Tabu、Vtr – Tabu 算法求解模型 1 目标函数值 ( $\lambda_1 = 0.6$ ) .....	42
表 A-4 Cyc – ATCS、Cyc – Tabu、VVT 算法求解模型 2 目标函数值 ( $\lambda_1 = 0.4$ ) .....	42
表 A-5 Cyc – ATCS、Cyc – Tabu、VVT 算法求解模型 2 目标函数值 ( $\lambda_1 = 0.5$ ) .....	44
表 A-6 Cyc – ATCS、Cyc – Tabu、VVT 算法求解模型 2 目标函数值 ( $\lambda_1 = 0.6$ ) .....	44
表 A-7 Cyc – ATCS、Cyc – Tabu、VVT 算法求解模型 2 所得调度流水线均衡率 ( $\lambda_1 = 0.4$ ) .....	47
表 A-8 Cyc – ATCS、Cyc – Tabu、VVT 算法求解模型 2 所得调度流水线均衡率 ( $\lambda_1 = 0.5$ ) .....	47
表 A-9 Cyc – ATCS、Cyc – Tabu、VVT 算法求解模型 2 所得调度流水线均衡率 ( $\lambda_1 = 0.6$ ) .....	47

# 第一章 绪论

## 1.1 课题研究的背景及意义

汽车装配大多采用同步装配流水线方式作业，将装配过程分为多个作业单元，并安排在流水线的相应工位上，车体在移动中装配，各工位同时作业。以往，汽车装配工厂固定地生产一种或少数几种车型，采用大批量、规模化的生产。然而，随着技术的日新月异，客户需求的多样化，以及精益思想、环保节能观念的出现，汽车工业的生产模式不得不转变为面向订单的大批量、多品种的生产方式。因此，缩短交货期、提高资源利用率、降低生产成本、提高生产运作的灵活性，已成为保证企业市场竞争力的重要手段。

多品种装配是在基本不改变或较少改变现有生产设施的前提下，通过对装配生产线的合理组织与调度优化，实现多品种共线装配，以最大限度地挖掘生产线的潜能，向客户提供定制的个性化产品和服务。

生产调度就是组织执行生产进度计划的工作，作为一种决策形式，调度在制造业扮演者至关重要的角色，尤其在现今充满竞争的环境下，有效的生产调度已成为能在市场竞争下生存的必须。

从上个世纪 50 年代起，调度问题的研究就受到应用数学、运筹学、工程技术等领域科学家的重视，科学家们利用运筹学中的线性规划、整数规划、目标规划、动态规划及决策分析方法，研究并解决了一系列有代表意义的调度和优化问题<sup>[1]</sup>。如今，随着计算机的发展，信息技术的成熟，许多过去只在理论层面上的调度算法，都可以通过计算机辅助得以验证与运用，给制造业的具体实力提升带来了可能。

提高竞争力的方法有很多，对于汽车工业，产品需求多样化和市场细分化，促使越来越多的制造商将多品种装配作为增强其竞争能力的有效手段。具体来说，对于汽车零部件，装配过程主要是以零部件的安装、紧固为主，其次是联接、压装和加注冷却液、制动液等液体以及整车质量检测的工序，有时还要根据用户意向选装。因此，合理安排装配产线，优化调度作业单元，对保证汽车装配质量，快速响应需求，提高汽车装配线的生产效率有着重要的现实意义。

举例来说，随着调度问题的规模增大，人们发现即使通过计算机，有些问题的算法并不是有效的，因为它们的求解超出了可接受的运行时间。逻辑学家和计算机科学家通过研究这类问题，建立了复杂度理论，并称这些问题是  $NP$  问题，问题的复杂度是会随着问题规模增大呈现指数爆炸。

这样一来，有时得到最优调度或者最优解的成本就变得太高了，那么近似最优解便成了很好的选择。然而调度问题的算法本来就众多，求解近似最优解更是如此，不同的算法适用的情况也不尽相同。实践表明，寻找合适的调度方案对生产系统的运行有着显著的影响。因此，从多品种装配着手研究调度算法，对增加产品多样性，加快需求响应速度，加快提高企业的竞争力有着重要意义。

## 1.2 国内外研究现状

调度问题本身具有高复杂性、多约束性、多目标性，在多品种大批量生产方式下，由于产品品种多、批量大、零件多等特点，增加了调度的复杂程度。多品种的装配调度问题是一种常见的车间调度问题，对于汽车零件，又有其独有的工艺特点。所以研究这个问题需要从调度规则、目标确定、工艺过程、设施类型等方面入手，提出合适的算法。

有关调度问题的算法研究颇多，而且随着目标函数的不同及目标数量的增多而逐渐呈现多元化，与之而



来的便是难度增大,几乎不可能得到最优解,但是,许多  $NP-Hard$  问题还是相继得到解决。近来的研究热点以启发式算法为主,并呈现多种方法结合使用的趋势。

对于多品种调度的研究,国内外学者在调度的算法改造中有很多创新,在其研究的课题中体现出优良的性能,实用价值很大,对本课题的算法研究启发颇多。

将产品分类或者确定产品簇,这在多品种的装配问题来说很重要。Xie Zhiqiang<sup>[2]</sup> 等通过建立虚拟产品,将多品种问题转化为单品种问题,通过关键路径方法确定加工顺序后,根据各工位操作的特征运用不同的算法调度,然后进行整合,解决了单产品的柔性调度,最后添加作业间的约束,较好的解决了复杂的多品种调度问题,对于简单的多品种调度问题,甚至可以不用加入作业间的约束。唐勇智<sup>[3]</sup> 通过研究 RBF-LBF 串联神经网络,改进 K-means 聚类算法,提出了自适应的 SA-K-means 算法,本课题的研究可以借鉴其思想,更为有效的将产品簇分类。陈伟<sup>[4]</sup> 通过 Smith-Waterman, FASTA 和 BLAST 三个局部对比算法,较好的找到了相似性较高的 DNA 序列,对于本课题中的产品簇归类有很高的借鉴意义。

有关建模设计的研究众多,主要是有关目标函数与约束的建立,李斌<sup>[5]</sup> 等提出了车间调度 Multi-Agent 模型,以延期成本、设备利用率、综合调度性能等指标作为目标函数,并通过 Legin 软件和实例比较了不同调度规则下的效果。M.A. Adibi<sup>[6]</sup> 等研究了随机作业和有机器抛锚可能的动态作业车间调度问题,通过经学习的神经网络,更新变邻域搜索算法的参数,在常见的分配规则下,较好的解决了该目标包括制造期和延迟的调度问题,并且其适用范围很广,其特点是神经网络的应用,很大程度上提高了搜索性能。刘文平<sup>[7]</sup> 将汽车装配的多种订单产品序列优化看作约束满足的调度模型,通过邻域搜索算法中的 Memetic 算法,优化了混合品种装配线调度。杨本强<sup>[8]</sup> 运用线性规划理论,建立了汽车流水线均衡生产模型,并通过一个启发式搜索算法来探寻解,思路简单,容易实现,本课题的微观调整可以借鉴其思想。李宏霞<sup>[9]</sup> 等载荷考虑了物料配送能力,运用 FCFS 规则的相关算法<sup>[10]</sup> 提出了一种操作性较强的调度模型,较好的解决了多品种变批量的装配调度问题。B.J.V. da Silva<sup>[11]</sup> 等通过航空行业的实例研究,考虑了人力的水平等级,学习影响及作业空间的限制,将飞机装配分成 4 个不同程度的阶段,各阶建立或增改约束模型,有效解决了含有邻接约束的装配调度问题。A. Tharumarajah<sup>[12]</sup> 等考虑了基于行为的分布式控制,并将之用于装配调度模型中,有效地解决了一个 3 阶段 4 工作站的装配问题,与整数规划相比,基于行为的调度无论在运行时间或是适用规模上,都显著优于整数规划。李志娟<sup>[13]</sup> 等通过研究高校排课问题,在有效、交错、分散、固定、优先的原则下,设计了一个基于规则的算法,并在产生冲突时进行回溯,得到较好的课程表,其设计思想可以用到本课题中,尤其是在多种目标下,设计相应的规则算法。P. Chutima<sup>[14]</sup> 等考虑了学习因子,提出了基于生物地域最优方法的方法,应用于两边装配线的混合模型,并引入了适应性机制,化解了最小化生产变化率、最小化总时间利用率、最小化调度序列换线时间三个矛盾,并使它们同时最优,提高了该元启发式方法的性能,有效的解决了多样性装配的调度。

在解的搜索上,启发式方法在搜索局部最优时效果很好,而涉及到全局最优就需要考虑元启发算法。陈琳<sup>[15]</sup> 等将其撤装配车间简化成一个流水车间问题,并通过改进得到带有记忆的模拟退火算法,并引入禁忌搜索机制得到较好的近似最优解。台湾学者 Ham-Huah Hsu<sup>[16]</sup> 等研究了多机器人的装配单元,通过基于搜索算法的调度,并将之仿真。Jia Shuai<sup>[17]</sup> 等通基于禁忌搜索算法,通过路线重建和回跳追踪方法进行排序决策,解决了多目标的柔性作业车间调度问题。G. Moslehi<sup>[18]</sup> 等研究了柔性作业车间,提出了结合蚁群算法和邻域搜索算法的方法,解决了多目标的作业车间调度问题,得到了质量较高,计算时间合理的近似最优解,尤其在中、大型问题中更能体现优势,并且该方法的提升空间很大。台湾学者 Rock Lin<sup>[19]</sup> 等通过峰明机械厂的案例研究,引入作业划分概念和批量处理,提出了一个混合整数规划模型和 3 个启发式方法 FBEDD、FBFS、RFBFS,并通过计算实验证明 FBEDD 在解决小型问题上较好,而对于中大型的问题, RFBFS 兼具

有更好的求解结果。朱有产<sup>[20]</sup>等通过改进经典的 Dijkstra 搜索算法，引入空间向量，通过夹角参数，有效的快速跳出局部最优解，得到最短路径问题的全局最优，对本课题的解的搜索有启发意义，可以将之结合入粒子群算法，重新定义其方向参数以改进。高丽<sup>[21]</sup>等基于精英选择和个体迁移的多目标方法对遗传算法改进，得到了较好的多品种生产调度的 Pareto 解集。曹洪鑫<sup>[22]</sup>等将多品种产品的混流装配顺序看作是商旅问题 (TSP)，其加工和换模时间即转化为路程，并通过遗传算法进行了 100 次迭代，较好的找到了较优解。本课题主要研究对象虽然不是混流装配，但混线时可以用到这个想法。讲到 (TSP)，翁武熙<sup>[23]</sup>采用了结合蛙跳算法的新型智能算法，较好的改进了蚁群算法的搜索过程，值得借鉴。

### 1.3 课题的基本思路及技术路线

装配车间调度的关键技术是和调度相关的技术，需要有调度理论的框架和相关概念，常用的调度模型可分为确定型模型和随机型模型，它们的区别是随机型模型的相关变量不是具体值，而是一个分布，不同的情况下用到的分布，包括连续分布和离散的，使模型更接近实际情况。

本课题以某汽车电子公司的装配车间为对象，对多品种的调度安排进行研究。首先对产线生产现状分析，提出改进方案，根据方案建立相应调度的数学模型。然后根据模型的特征，设计求解算法，并通过计算实验评估算法的性能，提出该算法在调度实践的意义。

本课题采用的基本思路 and 关键技术路线如图 1-1 所示。

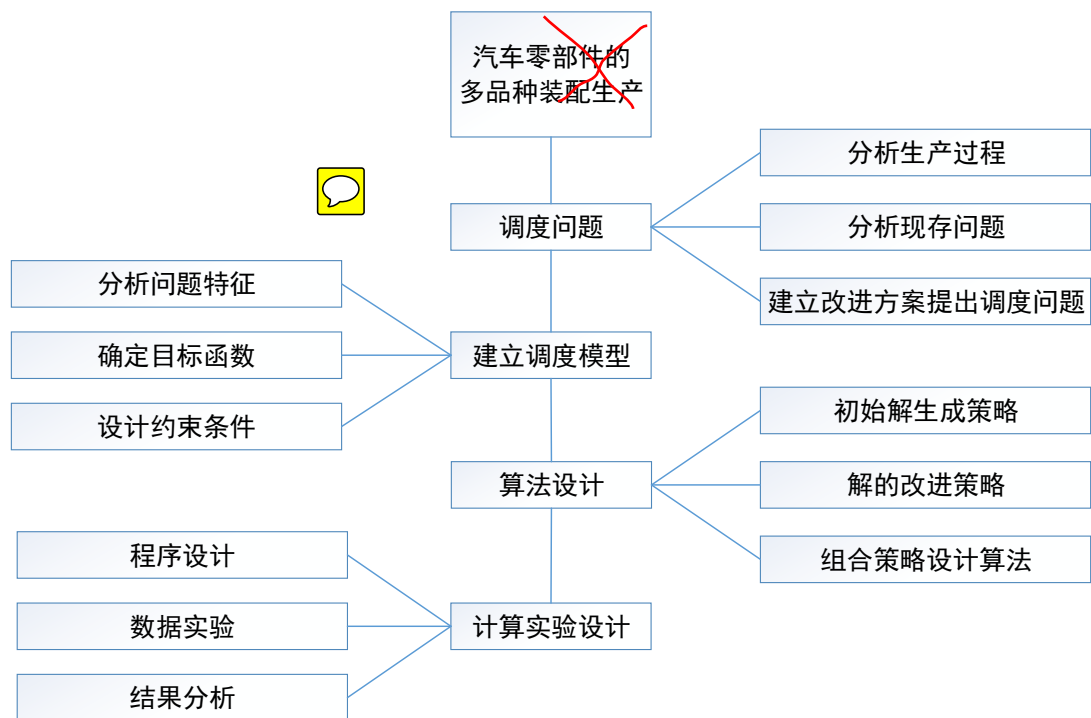


图 1-1 基本思路 and 关键技术路线

## 第二章 研究对象现状与分析

本文以某汽车电子有限公司的装配车间为研究对象，进行调研。本章将介绍该公司车间的基本情况，并对现有装配线调度情况进行初步分析，指出现有计划安排和调度存在的主要问题，然后针对这些问题，提出合理改进方案。

### 2.1 公司基本情况

该汽车电子有限公司主要产品为车用电子电器开关、控制模块、控制面板等，是美国通用、德国大众、一汽大众、上海大众等国内外 40 余家汽车主机厂的专业定点配套供应商，配套的代表性车型有奥迪 A6(L)、奥迪 A4(L)、奥迪 Q5、捷达等系列轿车，以及卡车、轻型车、微型车等，产品型号达 6000 余种。

该公司的订单特点是品种多、批量大、小型产品、工艺成熟，~~所以采用流水线生产是比较合适的~~，与其合作较多的客户（主机厂）一般有其专用线，专门负责该主机厂的订单生产。订单从接受到交付的流程如图 2-1 所示，其中虚线框内为装配车间的作业。在没有订单或者订单较少时，为了不让生产线停下来，需要进行工厂内部的计划生产，而订单较多时需要加班作业。

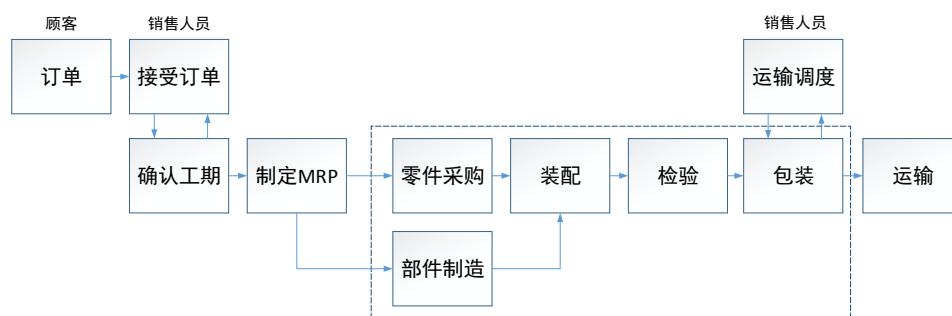


图 2-1 现行订单信息流

### 2.2 装配车间作业

该公司的制造部分为制造一部和制造二部，制造一部主要是负责零部件的生产，不是本课题的主要研究对象；制造二部主要是负责产品总装，共 8 个车间，均是流水线作业，是本课题的研究对象，其相关信息如表 2-1 所示（单位：个）。

每个总装车间均有多条生产流水线，为方便管理，每条流水线负责各自对应主机厂的装配生产。当主机厂需要多个品种的产品时，流水线根据订单顺序进行装配作业，即同一主机厂的产品在一条生产线上轮番生产。

表 2-1 制造二部装配车间信息

车间	1 车间	2 车间	3 车间	4 车间	5 车间	SGM 车间	通用车间	电子车间 <sup>a</sup>
班组数量	6	6	6	5	4	4	6	2
流水线数量	7	7	7	6	7	6	5	5
PMC 分配品种数	777	557	186	196	334	29	76	83

<sup>a</sup>包括：模块线、插件线及 SMT 线三个部分，其中，模块线是总装线，共 5 条装配线，插件线和 SMT 线是零部件生产产线。本次研究的重点是总装线，故电子车间只列出了部分信息。

## 2.3 产线调度现状

当前该公司装配车间采用专线生产的方式，即客户的订单在其专用的流水线上进行装配生产作业，当同一客户有多个订单下达时，按照先到先服务 (FCFS) 的规则进行装配生产安排，多条产线并行作业互不干扰。

订单或任务到达时，如果有流水生产线空闲可用，则立刻对其根据进行生产准备，然后开始装配生产。若产线在处理订单，那么将该订单安排入其专线队列中，等待前面的批量订单生产完毕再进行生产。现行调度的产线如图 2-2<sup>1</sup>所示。



图 2-2 3 条生产线的现行调度

现行调度方法逻辑简单，执行力强，客户满意度高，每条装配线可分时生产不同品种的产品，而且按照厂家来安排组织生产，方便了管理。然而其缺陷也是明显的，例如常常会产生有些流水线队列很长而有些流水线空闲无作业的流水线不均衡现象，造成极大浪费。下面进行具体的现有问题分析。

## 2.4 生产线问题分析

课题研究对象是该汽车电子公司的总装生产线，是典型的流水车间，每条流水线负责一家主机厂的不同品种产品总装流程。装配生产根据订单到达先后进行安排，根据先到先服务 (FCFS) 规则在相应流水线生成任务队列。

生产线分析包括上述这些差别及其产生的影响或效果，具体描述从订单、装配到交付的流程，并分析现行调度方案的一些指标。

### 2.4.1 现行流程描述

现行下达订单到交付的流程如图 2-1 所示，销售人员接到客户订单，在确认工期后，将之送达计划部门，制定主生产计划 (MRP)，随后根据产品特性安排采购与厂内加工。根据任务队列与批量进行产品总装，通过质检包装后，销售人员安排运输送达至客户。订单到达会立即安排入其对应的主机厂专用产线，当同一主机厂有多个订单同时到达时，则根据最早交货期 (EDD) 规则进行安排。

<sup>1</sup>图 2-2 中订单 a-b 表示主机厂 a 的第 b 个订单，下同

本课题的研究对象是流程中的总装调度安排，即为虚线框内的部分，其重点在订单的装配生产安排上，故要进行先行调度方案分析，发现问题。

### 2.4.2 现行调度方案

如前所述，先行调度方案十分简单，容易操作，由于客户专有各自的装配流水线，所以流水线上的生产安排只需按照客户下达订单的顺序即可，不同车间、不同流水线之间没有相互的联系。由于各流水线上没有专门的工装夹具，一般都是根据订单的内容，在其到达时快速布置流水线的各个工位，所以同一主机厂的连续订单生产时，需要考虑切换准备时间。

现行调度方案逻辑简单，执行力强，每条装配线可分时生产不同品种的产品，而且按照厂家来安排组织生产，方便了管理。

### 2.4.3 主要问题

现行调度方案存在诸多问题，例如多条装配线负荷不均衡，有的任务过重，有的任务不足，负荷不均衡，一条装配线上装配的产品工艺相似性较低，导致换线时间增加，产生更长的等待。

根据现行调度情况进行问题分析，可以归纳其主要存在问题如下：

#### (1) 产线利用率低

产线利用率低主要体现在存在大量换线时间，一方面由于产线等待队列由 FCFS 规则产生，同时到达的订单也只是根据 EDD 规则安排，没有考虑品种装配流程间的相似性。当这种差异很大时，必然会增加换线时间，进而增加了任务间的等待。另一方面，由于产线的专用性，当有多条产线都能处理某个任务时，该任务只能在订单来自的主机厂专线上生产，闲置了可用线的生产能力。

#### (2) 生产不均衡

这里的生产均衡和混流生产中的均衡生产稍有差别，此处的不均衡现象更为宏观。来自不同主机厂的任务在不同产线上进行处理，这样一来，订单较多、较频繁的主机厂产线总是会处于繁忙状态，而订单较少的产线则呈现为停线等待居多。这种不均衡现象直接导致产能的巨大浪费，同时也间接导致了换线时间增加，因为不均衡的生产表面订单较为集中。突破专线限制可以解决宏观不均衡问题，虽然细分到单条线上的混流生产可以进一步均衡化生产，但其需要较高的管理投入，可以考虑折衷。

#### (3) 工艺及设备和生产需求不匹配



各流水线需要有多品种加工的能力，故线上需要有相应加工工艺的设备，而时常不同主机厂所需产品可能有很高的相似性，这使得同样的设备需要在多条流水线上设置，尤其对于加工时间较短、处理批量较少的作业，过多的设备徒增成本与闲置。另一方面，加工时间长、处理批量大的作业，较少的设备不利于生产效率，导致在制品增多。

#### (4) 工期可控性低

工期的可控性低主要体现在应变插单的问题上，现行调度采用的是不可中断的流水作业，各流水线只能按其队列顺序进行装配作业，由于这样安排没有顾及工期的先后即订单的具体情况，导致大部分订单都需要延期交货，也存在较多订单的过早完工，增加了库存。此外，虽然插单可以较为合理安排订单加工顺序，而然会带来额外的切换时间，需要权衡考虑。

#### (5) 产线冗余度高

前面几大问题已经涉及到了一些浪费现象，除了这些之外，该厂制造二部共有 8 个装配车间，每个装配

车间有 7-8 条总装流水线，然而由于流水线是按主机厂进行分配，存在较高的冗余度，前面提到的产线间设备类似属于其中之一。产线的冗余还包括作业及管理人员，辅助设置，场地空间，相关能源等。

## 2.5 方案改进

上一节分析并提出了现有调度方案存在的为题，并对其产生原因进行分析，这为改进调度方案明确了方向。为制定合理调度方案，改善装配流水线现状，首先需要确定改进目标，进而根据目标的可行性，建立数学模型。因此，本节将结合实际情况，权衡投入和效益，建立适合本课题研究对象的多品种多装配线轮番装配生产调度模型。

### 2.5.1 改进设计

目前的生产现状的主要问题是各主机厂有其专用流水线，使得订单的调度安排为十分单一，受到很大的限制，不能合理利用流水线的生产能力，所以首要的改进是突破专用线的生产界限。如此一来，生产线可以加工多家主机厂的订单，形成所谓的混线生产，是较混流生产为宏观的均衡化生产，如图 2-3 所示。与混流生产不同的是，混流生产的最小研究单位是订单中的作业，而混线生产的最小研究单位是订单本身，即一定批量的作业。混线生产的策略是将不同订单的作业按一定比例进行生产，一般是在单一流水线上进行研究的，可使生产均衡化，但这一策略需要分析不同作业的相似程度，相差太大的作业若混流生产则会导致过多的切换准备时间，而且这个策略需要较高的管理水平。混线生产考虑的是多流水线之间的订单调度安排，使流水线的利用更为充分。

打破主机厂专用线限制后，出现了多品种多装配线的混流生产调度的问题，可以表述为，不同主机厂下达的不同订单到装配流水线车间，所有的订单可以在任意流水线上进行装配生产，需要建立合适的调度方案以合理利用各流水线的生产能力以及其他生产装配目标。



图 2-3 3 条产线的混线装配生产示意

## 2.6 小结

本章简单介绍了该汽车电子有限公司的装配车间生产现状及其问题，并提出了打破主机厂限制的解决方案，以此引导出了多品种多装配线的混流生产调度问题，需要建立合适的调度方案。调度方案的建立即该问题的求解，需要通过问题分析与建模，并设计相应的算法来求解。

## 第三章 多品种装配车间调度建模

上一章分析了现行调度存在的问题并提出了改进方案设计，而该方案的实现是一个多品种多装配线的调度问题，本章将对该问题进行具体分析建模，然后根据该问题的特点，建立相应的调度优化模型。

### 3.1 建模准备

根据上一章的改进设计，对所需生产的订单进行排列组合，较为均匀地安排在各流水线上，经行混流水线轮番装配，以期获得均衡的流水线利用率、减少换线时间浪费、缩短完工时间、降低生产成本等。需要注意的是这种生产方式和混流生产的区别，在混流生产是将不同的订单中的产品根据一定比例混合，在流水线上进行生产，使生产均衡化，其研究单位为订单中的作业。而多流水线混线装配的研究单位是各订单，其生产方式打破现行的订单在固定流水线上生产的限制的策略，从而合理利用各流水线产能资源。

产品订单确定其所需产品的数量(订单批量)，订单中的每个产品可以看作是作业，流水线上的各装配工位或机器看作处理单元，由于本课题的研究主要内容较少涉及具体的装配工艺，故产品订单也确定了作业处理所需的处理单元的数量、种类以及顺序，流水线上的订单处理可以看作是各作业按固定顺序经过线上的处理单元进行处理。

#### 3.1.1 基本符号说明

为了方便问题描述，需要说明基本符号如下，其中涉及的时间变量研究对象为系统时间：

- $n$  订单数量
- $m$  流水线数量
- $j$  订单标记,  $j = 1, 2, \dots, n$
- $N$  所有订单集合  $\{j \mid j \in \mathbb{Z}, 1 \leq j \leq n\}$
- $l$  流水线标记,  $l = 1, 2, \dots, m$
- $S_l$  流水线  $l$  上的订单调度
- $|S|$  调度  $S$  的订单数量
- $\bar{S}$  调度  $S$  中的订单集合
- $l_k$  调度  $S_l$  的第  $k$  项订单标记,  $k = 1, 2, \dots, |S_l|$
- $d_j$  订单  $j$  的交货时刻(工期)
- $t$  生产系统时间

一个调度问题可以由三元组  $\alpha | \beta | \gamma$  表示， $\alpha$  域描述单一处理单元环境， $\beta$  域包含加工特征和约束的细节， $\gamma$  域描述其目标<sup>[24]</sup>。

#### 3.1.2 基本 $\alpha$ 域

- $Pm$  同速并行机
- $Fm$  流水车间



### 3.1.3 基本 $\beta$ 域

$r_j$  订单  $j$  到达流水线系统时刻，是其最早可开始时刻

$s_j$  订单  $j$  开始之前所需的切换 (开工) 准备时间

### 3.1.4 基本 $\gamma$ 域

$\gamma$  域涉及目标函数，一般调度问题需要考虑最小化目标函数，常见的目标函数为订单  $j$  的完成时刻  $C_j$ ，为订单  $j$  离开系统的时刻。订单  $j$  的迟滞可以定义为：

$$L_j = C_j - d_j$$

进一步可定义其延迟和提前：

$$T_j = \max\{L_j, 0\}$$

$$E_j = \max\{-L_j, 0\}$$

常见基本  $\gamma$  域有：

$\sum w_j C_j$  加权订单完成时刻总和

$\sum w_j T_j$  加权订单延迟时间总和

## 3.2 模型 1：多品种多装配线轮番装配调度优化模型

根据改进调度方案——~~多流水线轮番装配~~的特点，可以建立相应的调度优化模型，在模型建立之前，需要进行一些基本假设。

### 3.2.1 基本假设

#### (1) 整数变量假设

模型中所涉及的所有变量，如订单处理时间  $p_j$ ，切换准备时间  $s_j$  等，皆在整数范围内考虑 (除了后面的决策参数  $\lambda_1, \lambda_2$ )，便于后面将问题离散化。

#### (2) 数量有限假设

假设研究对象所设计的订单数量  $n$ 、订单包含的作业数量  $n_j$ 、订单涉及的处理单元数量  $m_j$  以及流水线数量  $m$  皆有限。

#### (3) 无差别假设

对于订单来说，一般都是由一定批量的作业组成的，所以可以假设其中包含的各作业无差别，即每项作业的处理时间、工艺顺序皆相同。由于各流水线上无固定设备，在安排装配生产时，都是现场安排流程对应的工位，所以可以假设流水线之间无差别，类似于同速并行机环境，即同一订单在不同流水线上的切换准备时间、处理时间都一样。

#### (4) 无插单生产假设



模型 1 不考虑插单情况，假设所有订单在系统时刻  $t = 0$  时下达进入流水线系统，皆可开始安排生产处理，即  $\forall r_j = 0$ ，并且之后没有新的订单进入系统。

#### (5) 不可中断假设

订单在流水线上装配处理过程中，需要其中所有的作业装配完成才可以离开流水线系统，假设生产过程中没有人为或机器原因产生中断。同时，由于该厂总装厂的产品体积较小，每个工位都有足够的空间存放在制品，不会因此使生产中断。

#### (6) 无相关假设

按队列生产时，处理不同订单需要考虑切换准备时间，由于流水线上的工位都是订单到达后再根据工艺布置的，所以可以假设切换准备时间只和订单本身有关，和其前续后继订单没有相关性，并且各订单所需的切换准备时间事先已知。此外，各流水线切换准备时间不算入停线等待。

### 3.2.2 目标函数

本课题期望通过合理调度，达到提高流水线利用率、减少浪费、提高按时交货率、缩短制造期等目标，这些目标有着内在联系，所以在设计目标函数的时候，不必把每样都列入其中，比如最小化完成时刻在一定程度上相当于最大化处理单元利用率，进一步可以暗示最大化流水线利用率。

对于模型 1，可以考虑满足工期和提高流水线利用率，其中满足工期为主要目标，可用加权延迟时间和  $\sum w_t T_j$ ，流水线利用率为次要目标，可用加权完成总时刻和  $\sum w_c C_j$ 。其中  $w_t, w_c$  分别为订单  $j$  的延迟和完工权重。两目标的重要程度可以体现在目标权重系数  $\lambda_1, \lambda_2$  上。

进一步分析该问题，可以发现由不可中断假设和无差别假设，任一订单的总装配生产时间是确定的，并且与其被加工的流水线无关，所以可将各装配线看作是并行同速机环境，每条流水线看作一个可以处理所有订单的机器。该问题可以记为： $Pm | s_j | \lambda_t \sum w_t T_j + \lambda_c \sum w_c C_j$ ，那么该目标函数可表示为：

$$\min \quad \lambda_t \sum_{j=1}^n w_t T_j + \lambda_c \sum_{j=1}^n w_c C_j \quad (3.1)$$

目标权重系数的确定涉及到管理者的决策，不同的比例对应于不同的生产。根据式 (3.1) 的特性，并从流水线角度来考虑，可以改写成如下等价形式：

$$\min \quad \lambda_t \sum_{l=1}^m \sum_{k=1}^{|S_l|} w_{t_k} T_{l_k} + \lambda_c \sum_{l=1}^m \sum_{k=1}^{|S_l|} w_{c_k} C_{l_k} \quad (3.2)$$

### 3.2.3 约束条件

约束条件可以从订单间的关系中寻找，并行机环境中，可以根据流水线来考虑订单。记订单  $l_k$  的处理时间为  $p'_{l_k}$ ，由于其准备时间为定值  $s_{l_k}$ ，可以将其并入订单处理时间来简化问题而不影响结果，并记订单  $l_k$  的

整合切换准备处理时间为  $p_{l_k}$ ，那么该模型 1 的主要约束如下：

$$\left\{ \begin{array}{ll} \sum_{l=1}^m |S_l| = n & (3.3) \\ \bigcup_{l=1}^m \overline{S_l} = N & (3.4) \\ \sum_{l=1}^m \sum_{k=1}^{|S_l|} wt_{l_k} = 1 & (3.5) \\ \sum_{l=1}^m \sum_{k=1}^{|S_l|} wc_{l_k} = 1 & (3.6) \\ \lambda_c + \lambda_t = 1 & (3.7) \\ C_{l_1} = p_{l_1} & l = 1, 2, \dots, m \quad (3.8) \\ C_{l_k} = C_{l_{k-1}} + p_{l_k} & k = 2, 3, \dots, |S_l|, l = 1, 2, \dots, m \quad (3.9) \\ p_{l_k} = p'_{l_k} + s_{l_k} & k = 1, 2, \dots, |S_l|, l = 1, 2, \dots, m \quad (3.10) \\ T_{l_k} = \max\{0, C_{l_k} - d_{l_k}\} & k = 1, 2, \dots, |S_l|, l = 1, 2, \dots, m \quad (3.11) \\ p'_{l_k}, s_{l_k}, d_{l_k}, wt_{l_k}, \lambda_t, \lambda_c \geq 0 & k = 1, 2, \dots, |S_l|, l = 1, 2, \dots, m \quad (3.12) \end{array} \right.$$

式 (3.3) 表示所有流水线安排的订单数量总和为需要安排调度的订单数量总和，式 (3.4) 为式 (3.3) 的集合表达形式，式 (3.8)–(3.9) 计算各订单的完成时刻，连续的订单之间没有停顿，式 (3.10) 计算了整合订单处理时间，式 (3.11) 为订单的延迟定义，式 (3.12) 为变量的非负约束。

### 3.3 ~~模型 2:~~ 考虑插单的多品种多装配线轮番装配调度优化模型

模型 2 考虑订单插单的情况，在系统开始运行后，订单陆续进入系统，其进入系统的时刻为该订单最早可被处理时刻。也就是说各订单并不是在系统时刻  $t = 0$  时刻皆可开始进行处理，订单可以开始处理需要同时满足两个条件：① 有流水线处于闲置状态，② 系统时刻大于等于订单最早可开始时刻  $t \geq r_j$ 。但若有流水线闲置，且接下来要对这个订单进行加工处理，那么此时可以先开始该订单的切换准备，以节少流水线产能浪费。

模型 2 对工期的要求进一步提高，不仅要求订单不产生延迟，更对订单的提早完成作出相应惩罚，这是调度研究的最新热点，符合准时化生产的要求，可以很好体现一个企业的管理水平。此外，模型 2 更为深入挖掘流水线的性能，考虑了各流水线的利用率和流水线的整体均衡性。这些要求都使得模型 2 更为接近实际情况。

#### 3.3.1 相关符号及说明

模型 2 建立在模型 1 的基础上，由于有新的因素加入考虑范畴，需要补充或修改符号定义如下：

- $C_l$  流水线  $l$  的制造期，其值为  $C_{l_{|S_l|}}$
- $f_j$  订单  $j$  开始处理前，处理流水线的闲置时间
- $Rb$  流水线均衡率

$Ru_l$  流水线  $l$  的利用率

### 3.3.2 相关假设

模型 2 比模型 1 多考虑了插单的情况，所以需要增加和改变模型 1 的相关假设。

#### (1) 插单假设

模型 2 考虑了插单情况，所以模型 1 的无插单假设不在适用，而插单的情况相当于各订单陆续进入系统，即  $\exists r_k > 0$ 。

#### (2) 惩罚一致假设

由于对订单的交付准时要求有所提高，故可以将订单的延迟和提早做等价的惩罚，即  $w_e = w_t$ ，这样一来可以便于后续目标函数的改写。

#### (3) 订单最早可处理时刻假设

考虑插单的情况会导致流水线的空闲等待，而订单开始前都需要经过切换准备，假设订单的切换准备可以在其最早可处理时刻  $r_j$  之前准备，而装配生产必须在  $r_j$  之后方可开始。这样假设可以提高流水线的利用率，也是十分合理的。

### 3.3.3 目标函数

考虑订单陆续到达时，更为注重订单的按时交付，同时也关注流水线的生产均衡性。生产均衡性指的是流水线的使用均衡，不要出现某条流水线一直繁忙而有些流水线空闲居多，导致负荷不均衡，损失产能。流水线均衡率定义如下：

$$Rb = \frac{\sum_{l=1}^m C_l}{m \times \max_{1 \leq l \leq m} \{C_l\}}$$

模型 1 中，各订单没有可处理时刻的限制，各流水线除了切换准备，其余时间都在处理订单，研究流水线利用率是没有意义的。而在模型 2 中，流水线上订单间的空闲等待将会出现，其中切换准备同样不计入空闲。为了提高流水线利用率，需要对其进行定义。假设订单  $j, k$  为同条流水线上的连续处理订单，且订单  $j$  先于订单  $k$  处理，需要考虑一下 3 种情况：

#### (1) $C_j \geq r_k$

这种情况表明，订单  $k$  进入系统的时候，其前续作业  $j$  仍然在处理当中，那么显然处理他们的流水线是不存在闲置的，即  $f_k = 0$ 。

#### (2) $C_j < r_k, C_j \geq r_k - s_k$

这种情下，虽然订单  $k$  进入系统可开始处理的时间在订单  $j$  之后，然而由假设生产计划部门可以提前安排该订单的准备，所以在前续订单  $j$  处理完成后，先进行切换准备。然而该切换准备完成时，订单已处于可加工状态，所以整体来说，该流水线仍然没有闲置，即  $f_k = 0$ 。

#### (3) $C_j < r_k - s_k$

这种情况较上种情况不同，订单之间的间隔时间大于后继订单的切换准备时间，那么就会使得该流水线出现闲置等待，此时  $f_k = r_k - s_k - C_j$ 。

此外，需要人为定义首项订单的闲置  $f_{l_1} = \max\{r_{l_1} - s_{l_1}, 0\}$ , ( $l = 1, 2, \dots, m$ )，综上，可以定义订单  $l_k$  开始处理前的流水线闲置：

$$f_{l_k} = \begin{cases} \max\{r_{l_k} - s_{l_k}, 0\} & k = 1 \\ \max\{r_{l_k} - s_{l_k} - C_{l_{k-1}}, 0\} & k \geq 2 \end{cases}$$

由此，可以定义流水线  $l$  的利用率：

$$Ru_l = 1 - \frac{\sum_{k=1}^{|S_l|} f_{l_k}}{C_l}$$

表示该流水线在工期内处于生产处理状态的比重。和流水线均衡率不同，流水线利用率针对每条流水线本身，而流水线均衡率则是从全局的角度。

为了达到准时交货的目标，需要将主要目标定为加权延迟时间和与加权提早时间和的和，并有假设，这两个指标共用一个权重，即：

$$\min \sum_{j=1}^n w_j (T_j + E_j)$$

根据这个式子的特点，可将其改写成等价形式：

$$\min \sum_{j=1}^n w_j |L_j|$$

由于主要目标和工期的关联很大，容易受其牵制而安排出利用率不高的调度，故需将产线利用率考虑其中，并从流水线的角度来考虑，则主要需改写成式 (3.14)。

$$\min \sum_{l=1}^m \frac{\sum_{k=1}^{|S_l|} w_{l_k} |L_{l_k}|}{Ru_l} \quad (3.14)$$

此外，需要考虑流水线的平衡性，可以综合入次要目标加权完成总和当中，其形式可以有很多种，为了体现提高流水线平衡性的重要，可以这样设计：

$$\min e^{-Rb} \sum_{l=1}^m \sum_{k=1}^{|S_l|} w_{l_k} C_{l_k} \quad (3.15)$$

与模型 1 类似，主要和次要目标可以通过目标权重系数  $\lambda$  上。结合式 (3.14) 与 (3.15) 得到本模型的目标函数：

$$\min \lambda_1 \sum_{l=1}^m \frac{\sum_{k=1}^{|S_l|} w_{l_k} |L_{l_k}|}{Ru_l} + \lambda_2 e^{-Rb} \sum_{l=1}^m \sum_{k=1}^{|S_l|} w_{l_k} C_{l_k} \quad (3.16)$$

### 3.3.4 约束条件

模型 2 以模型 1 为基础，需要多考虑订单进入时刻，所以要对基本约束进行修改和增加：

$$\left\{ \begin{array}{ll} \sum_{l=1}^m |S_l| = n & (3.17) \\ \bigcup_{l=1}^m \overline{S_l} = N & (3.18) \\ \sum_{l=1}^m \sum_{k=1}^{|S_l|} w_{l_k} = 1 & (3.19) \\ \sum_{l=1}^m \sum_{k=1}^{|S_l|} wc_{l_k} = 1 & (3.20) \\ \lambda_1 + \lambda_2 = 1 & (3.21) \\ C_{l_1} = f_{l_1} + s_{l_1} + p_{l_1} & l = 1, 2, \dots, m \quad (3.22) \\ C_{l_k} = C_{l_{k-1}} + f_{l_k} + s_{l_k} + p_{l_k} & k = 2, 3, \dots, |S_l|, l = 1, 2, \dots, m \quad (3.23) \\ \sum_{l=1}^m \sum_{k=1}^{|S_l|} r_{l_k} > 0 & k = 2, 3, \dots, |S_l|, l = 1, 2, \dots, m \quad (3.24) \\ L_{l_k} = C_{l_k} - d_{l_k} & k = 1, 2, \dots, |S_l|, l = 1, 2, \dots, m \quad (3.25) \\ T_{l_k} = \max\{0, C_{l_k} - d_{l_k}\} & k = 1, 2, \dots, |S_l|, l = 1, 2, \dots, m \quad (3.26) \\ E_{l_k} = \max\{d_{l_k} - C_{l_k}, 0\} & k = 1, 2, \dots, |S_l|, l = 1, 2, \dots, m \quad (3.27) \\ s_{l_k}, d_{l_k}, w_{l_k}, wc_{l_k}, \lambda_1, \lambda_2, r_{l_k} \geq 0 & k = 1, 2, \dots, |S_l|, l = 1, 2, \dots, m \quad (3.28) \end{array} \right.$$

式 (3.17) 表示所有流水线安排的订单总和，式 (3.18) 为式 (3.17) 的集合表达形式，式 (3.19)–(3.21) 是权重之后为 1，式 (3.22)–(3.23) 计算了各订单的完成时刻，考虑了流水线的等待时间，式 (3.24) 确保不出现所有订单在  $t = 0$  时刻同时可以进行处理的情况，式 (3.25)–(3.27) 是订单的迟滞、延迟、提早的定义，式 (3.28) 为变量的非负定义。

至此，2 个多品种多装配线轮番装配调度优化模型建立完毕。

## 3.4 小结

本章根据上一章的现状分析和改进方案，建立了 2 个多品种多装配线轮番装配调度优化模型，模型 2 比模型 1 多考虑了插单的情况。以提高流水线利用率、减少浪费、提高按时交货率、缩短制造期等为目标，两个模型分别建立了相应的目标函数，模型 2 更进一步定义了流水线的利用率和均衡率，并将之融入了目标函数之中。这两个模型的求解将在下一章阐述。

## 第四章 算法设计



上一章构建了 2 个多品种产品调度的数学模型，需要对其检验，而这些模型求解是 NP-Hard 问题，需要设计相应算法来求解。在求解模型前，需要对订单处理时间计算，该问题也需通过算法设计求解。模型的具体求解算法可以分为构造型和改进型，前者从没有调度的情况开始，按一定分派规则逐渐安排作业，直至生成一个较好的调度；后者则是在前者的基础上，调整已有调度以改善目标。本章将针对前一章提出的 2 个模型，建立相应的调度算法。

### 4.1 分派规则

本章所设计的部分算法涉及到订单的分派规则，所以在具体算法设计之前，先要简单探讨一下分派规则。

前文已提到的 EDD 规则和 FCFS 规则是常见的调度分派规则，在具体调度作业的时候，通常会先根据分派规则进行安排，所以分派规则便是作业安排的初始策略，然后再局部调整调度方案以进一步优化。一个周详的规则可能得到最优调度的初始解，显然简化了局部调整过程，然而极可能会需要巨大的思考空间和时间（比如枚举）。同样，过于直觉的规则将会增大后续调整的难度，所以制定分派规则时要作权衡。

#### 4.1.1 基本规则

##### (1) EDD(最早交货期)

EDD 规则从工期角度出发，将作业按照交货时刻的先后进行排序，并按这个顺序进行生产处理。对于并行机环境，一旦某处理单元空闲，就可以即刻安排队列中的首个作业任务。这个规则适用于安排目标和工期相关的调度任务。

##### (2) WSPT(加权最短处理时间)

WSPT 规则是 SPT(最短处理时间) 规则的一般化，从作业的处理时间出发，对以完工时刻为目标的调度任务较为合适。这个规则将处理作业按照  $w_j/p_j$  值非增的顺序排列，处理时间较长的作业被安排在较后的位置，从一定程度上减少了排队等待时间，并且可以证明 WSPT 规则对  $1 \parallel \sum w_j C_j$  的调度是最优的<sup>[24]</sup>，然而在  $Pm \parallel \sum w_j C_j$  环境下并不一定是最优。

##### (3) MS(最小松弛)

MS 规则通过描述作业的紧迫程度来进行作业调度，与前两者最大的区别在于，这个规则是动态的，即和系统时间  $t$  相关。作业根据  $\max\{d_j - p_j - t, 0\}$  的值非减的顺序排列，显然较为紧迫的作业会被安排在前面，并且不同的系统时间会影响排列的顺序，呈现动态的调度。该规则适用于安排目标和工期相关的调度任务。

#### 4.1.2 复合规则

复合分派规则是综合了许多基本规则的一个表达式，各基本规则都有其各自的比例参数，用来确定这个规则对符合规则影响程度的比例，没有固定的形式，接下来举一例：ATC(明显滞后成本) 规则。

ATC 规则综合了 WSPT 规则和 MS 规则，每当有空闲处理单元时，所有待调度作业按式 (4.1) 计算其排

序指数，选出具有最大指数值的作业进行处理。

$$I_j(t) = \frac{wt_j}{p_j} \exp\left(-\frac{\max\{d_j - p_j - t, 0\}}{K\bar{p}}\right) \quad (4.1)$$

式中：

$K$  规则比例参数

$\bar{p}$  剩余作业平均处理时间

可以看出当  $K \rightarrow \infty$  时， $I_j(t) \rightarrow w_j/p_j$ ，此时 ATC 规则便转化为 WSPT 规则。当  $K \rightarrow 0$  时，若作业  $j$  将产生延期，即  $\max(d_j - p_j - t, 0) = 0$ ，那么 ATC 规则也转化为 WSPT 规则，若作业不会产生延期，由于  $d_j - p_j - t$  的影响超过  $w_j/p_j$ ，规则 ATC 将转化为 MS 规则。ATC 规则可以较容易地应用到  $Pm \parallel \sum w_j T_j$  问题。

## 4.2 模型 1 求解分析

模型 1 的求解可以分为两个部分：初始解的求解与解的改进。模型 1 由目标函数式 (3.2) 和约束条件式 (3.3)–(3.12) 组成，其最优调度的求解包括初始可行解与解的改进，分别在初始解构造和解的改进中详述。

### 4.2.1 订单处理时间计算

模型的求解首先需要得到合适的初始解，而初始解的求解与之后的解的改进都涉及到订单的处理时间，因此本文先对订单的处理时间进行分析计算。

前一章提到，各订单的处理时间与被处理的流水线无关，只与其所含作业数量相关，因此可将订单的处理时间看作是关于订单本身及其所含作业数量的函数，记为  $p_j = g(j, n_j)$ ，其中  $n_j$  为订单  $j$  所含的作业数量。此外，仍需要补充符号说明如下：

$n_j$  订单  $j$  所含的作业数量

$k$  作业标记， $k = 1, 2, \dots, n_j$

$m_j$  订单  $j$  的处理单元数量

$i$  订单  $j$  的处理单元标记， $i = 1, 2, \dots, m_j$

$q_{k,i}$  作业  $k$  在处理单元  $i$  上的处理时间

$c_{k,i}$  作业  $k$  在处理单元  $i$  上的完成时刻

$c_{j,\max}$  订单  $j$  中所有作业的制造期，即  $\max(c_k)$

其中不同作业在同处理单元上的处理时间相同，可以简记为  $q_i$ 。

由于订单和子单仅有作业数量的差别，故考虑订单的情况即可推广到子单。由于，产品体积小，可以假设处理单元间的缓冲空间足够用。订单在某条流水线上的作业可以看作是流水车间环境，没有换线时间，所以该问题可以记为： $Fm \parallel C_{\max}$ ，显而易见的是  $c_{j,\max} = c_{n_j, m_j}$ 。该问题的数学模型如下：

$$\begin{aligned} \min & c_{n_j, m_j} \\ \text{s.t.} & \end{aligned} \quad (4.2)$$

$$\begin{cases} c_{1,1} = q_{1,1} & (4.3) \end{cases}$$

$$\begin{cases} c_{1,i} = \sum_{i=1}^{m_j} q_{1,i} & (4.4) \end{cases}$$

$$\begin{cases} c_{k,1} = c_{k-1,1} + q_{k,1} & k = 2, 3, \dots, n_j & (4.5) \end{cases}$$

$$\begin{cases} c_{k,i} = \max\{c_{k-1,i}, c_{k,i-1}\} & k = 2, 3, \dots, n_j, i = 2, 3, \dots, m_j & (4.6) \end{cases}$$

$$\begin{cases} q_{k,i} = q_i & k = 1, 2, \dots, n_j, i = 1, 2, \dots, m_j & (4.7) \end{cases}$$

该问题可以用有向图来表示，其关键路径即为所求解的计算历程。如4-1所示，每个节点内表示作业的处理时间，横向表示作业处理顺序，纵向表示不同的处理单元，由左上角开始，按照有向弧的方向进入节点，计算出右下节点的时间即为订单处理时间。其中，式 (4.3) 和 (4.4) 可以化简得到这个等式： $c_{k,1} = k \cdot q_1, (k = 1, 2, \dots, n)$ ，便于下面算法的初始化。

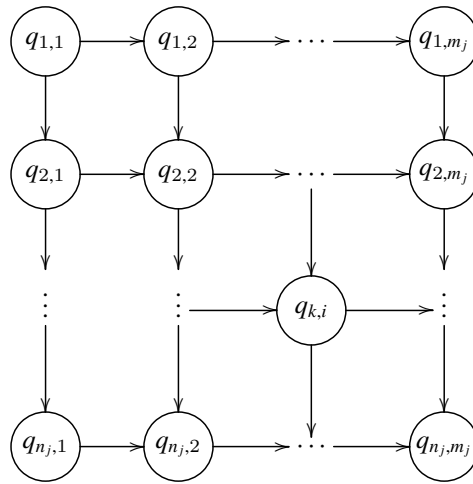


图 4-1 订单的作业处理有向图

也可以从生产节拍的角度来得到订单的装配处理时间。由于订单中的作业是相同的，所以生产节拍为订单的瓶颈工序，即  $tt_j = \max_{1 \leq i \leq m_j} \{q_i\}$ ，也就是说，除了首项作业的处理时间为各工序处理时间总和，其余作业皆按节拍完成，那么订单  $j$  的装配处理时间为：

$$p_j = \sum_{i=1}^{m_j} q_i + (n_j - 1)tt_j \quad (4.8)$$

## 4.2.2 模型初始解构造

初始解构造的主要内容是确定分派规则，虽然不能保证得到最优调度，但却有着很高的执行性，便于计划安排。模型 1 的目标函数主要成分是为加权时延迟间总和，采用 ATC 规则能得到较优调度<sup>1</sup>，关键在于比例参数的选取，相关的方法也有很多，例如回归分析、机器学习等。本课题将该规则得到的调度作为改进算法的初始解，所以不需要在该参数上过多深究，可以采用  $K = 2$  作为推荐值<sup>[25]</sup>。

针对模型 1 的实际情况，调度中的作业便是各订单，并以整合处理时间作为各订单的处理时间，这样可以不考虑切换准备时间。此外，后续的改进算法要用到构造算法的相关信息，需要记录被调度的订单序列。

<sup>1</sup>准确来说，ATC 规则较为适用于  $1 \parallel \sum w_j T_j$  问题，对于  $Pm \parallel \sum w_j T_j$  也能得到质量很高的解，虽然这个规则运用于模型 1 所涉及的问题效果不一定如这两个确切的问题，但足够作为后续改进的初始调度



使用该规则得到的初始解构造算法称为 Cyc-ATC 算法，为了方便描述这个算法以及后续算法，需要扩充符号说明如下：

$J$	待调度订单集合
$L$	已调度的订单记录序列，其集合记为 $\bar{L}$
$L_j$	订单记录序列的第 $j$ 项作业， $(j = 1, 2, \dots, n)$
$a_l$	流水线运行状态，处理订单时 $a_l = 1$ ，否则 $a_l = 0$ ， $(l = 1, 2, \dots, m)$
$t_l$	流水线 $l$ 的预计正在处理订单的完成时刻， $(l = 1, 2, \dots, m)$
$G(S)$	调度 $S$ 的目标函数
$H(S_l)$	流水线 $l$ 的调度目标函数
$h(j)$	订单 $j$ 的目标函数
$S^{(0)}$	目前为止的最佳调度
$S^{(c)}$	相邻调度，即候选调度
$\overline{S^{(c)}}$	邻域调度集合
$ S^{(c)} $	邻域容量，即相邻调度的数量
$TL$	禁忌列表
$NL$	禁忌列表长度
$A$	特赦解
$NR$	交替次数

模型 1 适合用 ATC 规则进行初始解的构造，按照系统时间  $t$  的进行，动态判断各流水线闲忙状态，若有流水线处于空闲状态，则根据式 (4.1) 选出下一个进行处理的订单，将其安排入该空闲流水线，更新流水线状态及待调度订单列表，预估该流水线的下一次空闲时刻，重复这个步骤一直到所有订单都被调度。这样便得到了 ATC 规则下的初始调度解构造，以供下面解的改进所用。

### 4.2.3 解的改进

构造算法得到的初始解需要进行改进，为了尽可能得到较优解，需要采用一些启发式 (元启发式) 算法，区域搜索是一类较为有效并且操作简便的算法，其主要的两个步骤是邻域的定义与移动选择。对于多机环境，一般采用分阶段的交替算法。此外，本文提出的一种虚拟序列算法克服了交替算法的一些劣势。

#### 4.2.3.1 邻域定义

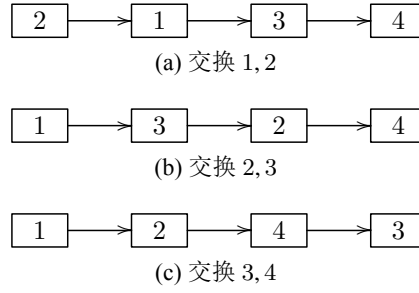
为了后面相关算法的描述，需要先定义邻域。考虑第一种情况，某机器上有共 4 项作业 (标记为 1, 2, 3, 4)，其初始调度  $S^{(1)}$  安排如图 4-2 所示，箭头方向为作业处理的顺序方向。交换  $S^{(1)}$  中相邻两项作业的顺序，可



图 4-2 4 项作业的初始调度  $S^{(1)}$  示例

以得到一个新的调度  $S^{(2)}$ ，显然此例中  $S^{(2)}$  有如图 4-3 所示的 3 种可能。这三个不同的调度构成了调度  $S^{(1)}$  的邻域，而确定其中的一个作为  $S^{(2)}$  便是一次移动选择。至此，机器内的邻域及移动定义完毕。

接下来考虑另一种情况，在多机环境下，机器间的作业移动。如果只是简单的从某个机器上挑选某个作

图 4-3  $S^{(1)}$  的 3 个相邻调度

业移动到另一个机器的某个位置，这样定义的邻域容量约为  $2 \sum \sum n_i \times n_j$ ，( $n_i, n_j$  为机器  $i, j$  上的作业数量)，不但会和前一种情况的邻域重叠，其过大的容量对搜索来说也是不合理的。所以多机环境的机器间邻域及移动要按照具体情况来设计定义。

#### 4.2.3.2 解的交替调整策略

通常并行机环境下，解的调整改进策略分为两个阶段，即机器内部的作业序列改变和机器间的作业改派，交替调整策略就是将这两个过程交替进行<sup>[26]</sup>，然而这个方法十分费时，可以简化内部改进，但这么做可能会影响解的质量。一般来说，由 ATC 规则所得到的该模型的初始解是较优解，甚至可能是最优解，如果需要对其改进，需要从两个方面来考虑。一个是流水线内部订单序列的调整，另一个是流水线间的订单调整，这两种调整涉及两种元启发式算法，需要交替使用。

##### (1) 内部调整

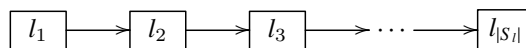
流水线内部调整是解的改进策略中的一部分，可以简单采用一些调度规则进行调整，也可以使用一些启发式方法。

##### • ATC 规则调整

流水线内部调整是  $1 \parallel \sum w_j T_j$  问题的一种扩展，可以简单运用 ATC 规则达到较好解，不同与前面的并行机环境，单机环境下，对流水线  $l$  使用 ATC 规则调度作业不用判断流水线的闲忙程度。前面的模型初始解构造完成后，各条流水线  $l$  都得到了相应的调度  $S_l$ 。进行内部调整时，只需将  $\bar{S}_l$  中的作业重新根据式 (4.1) 动态安排分配到该流水线的订单即可。内部调整需要配合流水线之间调整，并交替使用这两种调整策略，由于每次流水线之间调整至多涉及 2 条流水线的订单序列改变，故每当进行内部调整时，只需对流水线之间调整了的流水线进行重新安排即可，不需要对所有流水线都进行调整。

##### • 区域搜索调整

也可以对流水线  $l$  采用区域搜索的方法进行调整，该流水线上的调度邻域定义和前面的机器内部邻域类似，作业的序列就相当于流水线上订单的序列。举例来说，假设得到了该流水线的初始调度  $S_l^{(1)}$ ，如图 4-4 所示。这样一来，对于该流水线的内部调整来说，每次生成的调度都有  $|S_l| - 1$  个相邻的调度可供移动，并以相

图 4-4 流水线的初始调度  $S_l^{(1)}$

邻订单对  $(l_j, l_k)$  来表示解的一次移动。若每次都向着目标值改进最大的方向移动，则可能陷入局部最优解。可以采用一些机制避免此类情况发生，不同的机制原理便产生了不同的元启发式搜索算法。根据模型 1 的特征，本文将采用改进过的禁忌所搜算法来进行内部调整过程，它的机理是在移动选择时，为了不落入局部最优的循环而有意识地避开，需要设定禁忌列表  $TL$  及其长度  $NL$ ，通常禁忌列表满时候，采用 FIFO 规则将项目出栈。

使用规则调整和使用区域搜索方法调整都各有优劣，规则调整方法操作简单，运算速度快，然而具有问题适用性，不同的问题需要调整不同的规则，需要丰富的实践经验和调度问题的具体特征，而且即使规则得到了较好解，很可能陷入局部最优。区域搜索方法可以跳出一些局部最优，但不能保证找到的是全局最优解，而且一旦问题规模增大，搜索时间将会变得很长。此外由于区域搜索方法是启发式方法，每次运算的结果可能都会不同。

## (2) 之间调整

前面的部分说明了流水线内的订单调度改进，然而即使所有流水线都按照其相应的最优调度进行生产处理，从全局来看，这样的调度安排未必就是最优，这主要是由流水线使用不均衡引起的，这种不均衡性只能通过流水线之间的订单交换或重派来改变。将流水线  $a$  上的订单  $a_j$  重新安排在流水线  $b$  上的操作称为**订单重派**，两流水线间的订单交换可以看作是这两条流水线互相订单重派。因此，全局调度关于流水线之间的相邻移动可以定义为某订单重派。

为提高流水线均衡性，订单重派策略为：从具有最大目标值  $H(S_l)$  的流水线中选取具有最大函数值  $h(l_k)$  的作业，将其重派至具有最小目标值的流水线上调度末端，因此需要定义流水线  $l$  的函数贡献值与各订单的函数贡献值，根据式 (3.2) 可得两贡献值值的定义：

$$H(S_l) = \lambda_t \sum_{k=1}^{|S_l|} wt_{l_k} T_{l_k} + \lambda_c \sum_{k=1}^{|S_l|} wc_{l_k} C_{l_k} \quad (4.9)$$

$$h(l_k) = \lambda_t wt_{l_k} T_{l_k} + \lambda_c wc_{l_k} C_{l_k} \quad (4.10)$$

当然可以使用其他的策略进行流水线之间的调整，比如区域搜索，然而解的改进是由内部调整和流水线之间调整两部分组成，简单的总运算次数是两者之积。而流水线之间的区域搜索复杂程度要大大超出内部调整，而调整后解的改进效果可能差不多。所以采用简单明了的策略是比较正确的。

## (3) 交替改进

产生初始调度后，各流水线上的调度需要通过内部调整以局部改进，然后通过流水线之间的订单重派对现有解进行扰动，此时调度改变的流水线则需要再调整一次以合理安排新添入的订单。如此交替操作便是可以得到解的交替调整策略，根据内部调整策略的不同，可以得到不同的解的改进算法，如 Cyc – ATC 算法和 Cyc – Tabu 算法。模型 1 相关的交替策略算法具体参见**算法 1**和**算法 3**。

### 4.2.3.3 虚拟序列策略

本节将提出一种新的邻域结构，巧妙地避免了交替改进策略中流水线之间调整的问题，保证了解的改进调整全局性，使之更为合理，同时也不需要大量的运算次数。根据这种邻域结构的特征，本文将从另一个角度分析问题，设计出一个更为合理的虚拟序列算法。

所谓虚拟序列，即将所有流水线上的调度看作一个整体，所有订单都在这个序列上，其排列顺序由初始解的生成规则决定，也就是在调度安排时的记录序列  $L$ ，并按先后顺序记该序列上的订单为  $L_j, (j = 1, 2, \dots, n)$ 。

显而易见的是，任意流水线上两订单的先后顺序和它们在虚拟序列中的顺序是一样的。举例来说，模型的初始解生成用到一定的规则，这个规则下，所有订单根据其排序指数进入空闲的流水线，那么从全局订单的角度来说，所有的订单也有一种先后安排顺序，每当一个订单安排进入某条流水线的时候，可以将其同时安排进入一条虚拟流水线，这条流水线中订单仅仅表示订单进入的先后，不用实际处理订单。进入不同流水线的订单都会同时进入同一个虚拟流水线。这条虚拟流水线上的调度即是虚拟序列。需要注意的是虚拟序列顺序和各流水线上的调度有很大的关系，但不是一一对应的，也就是说同样的虚拟序列可能会得到不同的实际调度安排，但每个实际调度只对应于 1 个虚拟序列。

虚拟序列上只有所有订单的先后信息，其订单的一种排序称为一种**虚拟调度**。虚拟调度的邻域设计和流水线内部调度类似，一次移动改变对于虚拟序列本身来说，就是该序列中两相邻的订单处理顺序交换，由于其非一一对应的特点，它体现在各流水线上的改变则要分为以下两种情况。

(1) 相邻订单  $L_j, L_k$  安排在同一条流水线  $l$  上进行处理

该情况下，现有调度的一种可行移动就是这两个相邻订单的交换。

(2) 相邻订单  $L_j, L_k$  分别安排在不同流水线  $l, l'$  上进行处理

该情况下，现有调度关于该两相邻订单的交换可以有 3 种移动：① 将这两个订单的位置互换，即  $L_j$  安排在流水线  $l'$  上订单  $L_k$  的原有位置，订单  $L_k$  亦然；② 将订单  $L_j$  重派到流水线  $l'$  上，其位置与  $L_k$  相邻，顺序和此次移到后的虚拟序列中两订单的顺序一致；③ 将订单  $L_k$  重派到流水线  $l$  上，其位置与  $L_j$  相邻，顺序和此次移到后的虚拟序列中两订单的顺序一致。

后面两种情况在作移动时，不需要将订单交换对入栈禁忌列表，因为若再作一步该交换对的移动，不会回到之前的状态，而情况①则只对两订单交换位置的移动禁忌。虚拟序列的禁忌搜索的基本策略是以虚拟序列中相邻订单对  $(L_j, L_k)$  交换作为一次移动，可以看出，一个调度的邻域容量不是一个定值，会随着调度结构的改变而改变。

需要注意的是禁忌列表内容，尤其是以下两种情况，皆由列表长度引起：

(1) 流水线之间相邻订单过度禁忌

这种情况是这样发生的，虚拟序列中的这两相邻订单作了一次流水线互换移动，其中的一个订单在该禁忌为出列表前，被重派到其他的流水线，而这两个订单的交换仍然处于禁忌状态。这种情况的出现会使得一些可行的移动无法进行，从而浪费了迭代次数和迭代时间。

(2) 流水线之内相邻订单过度禁忌

这种情况的发生和前面一种类似，虚拟序列中相邻的订单作了一次移动，属于流水线内的互换移动，然而其中的一个订单在该移动禁忌出列表前，重派到先前的流水线。这种情况的出现常常会导致算法出错，而且不易被发现。为了防止这两种情况的发生，有甚是第二种情况，每次迭代时要根据各流水线状况更新禁忌列表，消除这些过度禁忌。

模型 1 的虚拟序列策略所得的算法称之为 Vtr-Tabu 算法，具体参见**算法 4**

## 4.3 模型 2 求解分析

模型 2 由目标函数式 (3.16) 和约束条件式 (3.17)–(3.28) 组成，与模型 1 类似，其最优调度的求解包括初始可行解与解的改进，初始解的构造将介绍 ATC 规则的推广形式，解的改进将引入变动邻域的策略以提高算法效率，这两部分将分别在模型初始解构造和解的改进中详述。

### 4.3.1 模型初始解构造

模型 1 的构造算法基于 ATC 分派规则，因为其主要目标是加权延迟时间和的变形，而模型 2 是模型 1 的更为一般的形式，主要是进一步考虑了切换准备时间<sup>2</sup>和订单进入系统时刻。可以继续使用 ATC 规则，然而本节将要使用其推广形式中的一种：ATCS<sup>3</sup> (考虑准备时间的明显滞后成本)。

ATCS 规则是 WSPT 规则、MS 规则和 SST (最小切换准备时间) 规则的复合规则，每当有空闲处理单元时，所有待调度作业按式 (4.11) 计算其排序指数，选出具有最大指数值的作业进行处理。

$$I_j(t) = \frac{w_j}{p_j} \exp\left(-\frac{\max\{d_j - p_j - t, 0\}}{K_1 \bar{p}} - \frac{s_j}{K_2 \bar{s}}\right) \quad (4.11)$$

式中：

$K_1$  与工期相关的规则比例参数

$K_2$  与切换准备时间的规则比例参数

$\bar{p}$  剩余作业平均处理时间

$\bar{s}$  剩余作业平均切换准备时间

为了确定比例参数  $K_1, K_2$ ，需要定义一些辅助变量，以作为比例参数的函数输入量。工期紧迫度可以定义为：

$$\tau = 1 - \frac{\sum d_j}{nC_{\max}}$$

$\tau \rightarrow 1$  表明工期紧迫，反之亦然。工期范围因子可以定义为：

$$R = \frac{d_{\max} - d_{\min}}{C_{\max}}$$

以及切换准备时间重要性因子：

$$\eta = \frac{\sum s_j}{\sum p_j}$$

$K_1, K_2$  在确定之前，需要对作业制造时间  $C_{\max}$  进行估计，可以简单估计如下：

$$\hat{C}_{\max} = \frac{1}{m} \sum_{j=1}^n p_j + \frac{1}{m} \sum_{j=1}^n s_j$$

然后可以根据下式确定  $K_1$ ：

$$K_1 = \begin{cases} 4.5 + R & R \leq 0.5 \\ 6 - 2R & R > 0.5 \end{cases}$$

以及  $K_2$ ：

$$K_2 = \frac{\tau}{2\sqrt{\eta}}$$

同模型 1 初始解构造类似，运用 ATCS 规则时，调度中的作业为各订单，当有流水线空闲的时候，即可根据式 (4.11) 安排订单进入相应流水线进行处理，与模型 1 不同的是，由于模型 2 考虑了插单的情况，所以订单安排入某流水线的时候，不能马上开始处理，同时模型 2 还考虑了流水线的利用率和流水线的均衡率，

<sup>2</sup>模型 1 中也考虑切换准备时间，只是由于该模型的特性，可以将其与订单处理时间合并成整合处理时间，以简化模型求解。而模型 2 需要分开考虑这两个时间

<sup>3</sup>ATCS 规则是专门为  $1 || s_j | \sum w_j T_j$  问题所设计的

所以在安排订单的时候，需要预估流水线的闲置时间，以合理安排订单并得到计算这两流水线指标的辅助变量。

### 4.3.2 解的改进

与模型 1 的改进策略类似，模型 2 的改进策略可以通过两阶段的交替策略，也可以直接使用虚拟序列策略。

#### 4.3.2.1 解的交替调整策略

解的交替调整策略分为内部调整和流水线之间调整两个部分，由于模型 2 的特点，其具体方法和模型 1 的略有不同，下面进行分述。

##### (1) 内部调整

各流水线的作业内部调整可以简单实用 ATCS 规则，但该规则最为合适的问题环境与本模型有一些差别，为了得到较好的调度效果，可以采用禁忌搜索来调整调度，也就是说使用 ATCS 规则得到的解暂时不作为后续流水线之间调整的初始解，而是先依次对流水线的调度进行区域搜索改进，然后将所得的改进解作为模型 2 的初始解。

##### (2) 之间调整

流水线之间的调整在于订单重派，而订单重派则的依据是流水线和订单对目标函数的贡献值，需要根据式 (3.16) 来重新定义这两个贡献值函数：

$$H(S_l) = \lambda_1 \frac{\sum_{k=1}^{|S_l|} w_{l_k} |L_{l_k}|}{Ru_l} + \lambda_2 e^{-Rb} \sum_{k=1}^{|S_l|} w_{c_{l_k}} C_{l_k} \quad (4.13)$$

$$h(l_k) = \lambda_1 \frac{w_{l_k} |L_{l_k}|}{Ru_l} + \lambda_2 e^{-Rb} w_{c_{l_k}} C_{l_k} \quad (4.14)$$

和模型 1 的流水线之间调整策略一致，从具有最大目标值  $H(S_l)$  的流水线中选取具有最大函数值  $h(l_k)$  的作业，将其重派至具有最小目标值的流水线上调度末端。

##### (3) 交替改进

和模型 1 类似，交替改进策略为：产生初始调度后，各流水线上的调度需要通过内部调整以局部改进，然后通过流水线之间的订单重派对现有解进行扰动，此时调度改变的流水线则需要再调整一次以合理安排新添入的订单。如此交替操作便是可以得到解的交替调整策略，根据内部调整策略的不同，可以得到不同的解的改进算法，如 Cyc – ATCS 算法和 Cyc – Tabu 算法。模型 2 相关的交替策略算法具体参见算法 2 和算法 3。

#### 4.3.2.2 虚拟序列策略

模型 1 中已经阐述并应用了虚拟序列，并设计了相应的求解策略，虚拟序列同样也适用于模型 2。对于虚拟序列，可以像模型 1 一样，直接使用禁忌搜索，不过由于虚拟序列长度要远大于任一流水线的调度序列长度，跳出局部最优需要的步骤可能会很多，可以考虑将其结合变动邻域搜索 (VNS) 来直接切换邻域结构，当然这么做也可能会跳出含有最优解的邻域分支。

变动邻域策略在于变动时机的选择，例如当某一算法在连续  $k$  次迭代都没有改进目标函数值，那么可以强行跳出当前邻域结构，选择另一个邻域开始新的搜索。所以在区域搜索开始之前，首先要确定可供选择的邻域结构，由于不是每个邻域结构都值得去搜索，还需设定变动次数。

## 4.4 算法设计

### 4.4.1 Cyc – ATC 算法

在模型 1 求解的过程中，可以采用重派订单，然后对受到影响的流水线分别使用 ATC 规则进行调整，以改进解的策略。这样的求解过程称为 Cyc – ATC (交替流水线间调整和流水线内 ATC 调整)，其具体过程如下：

#### 算法 1 Cyc – ATC 算法

- Step1** 初始化。  $J = N, \bar{L} = \emptyset, t_l = 0, \bar{S}_l = \emptyset, a_l = 0, (l = 1, 2, \dots, m)$ ，根据和式 (4.8) 计算各订单处理时间  $p'_j = g(j, n_j)$ ，进一步得到整合订单处理时间  $p_j = p'_j + s_j, (j = 1, 2, \dots, n)$ ，置系统时间  $t = 0$ ；
- Step2** 若存在  $a_l = 0$ ，记  $l^* = \min_{a_l=0}\{l\}$ ，执行 **Step3**，否则执行 **Step4**；
- Step3** 根据式 (4.1)，选取预备调度订单  $j^*$ ，使得  $I_{j^*}(t) = \max_{j \in J}\{I_j(t)\}$ ，将订单  $j^*$  安排入流水线  $l^*$  进行处理，记入调度  $S_{l^*}$ ， $\bar{S}_{l^*} = \bar{S}_{l^*} \cup \{j^*\}, J = J - \{j^*\}$ ，记录调度订单序列  $\bar{L} = \bar{L} \cup \{j^*\}$ ，更新流水线预计空闲时刻  $t_{l^*} = t + p_{j^*}$ ，修改流水线状态  $a_{l^*} = 1$ 。若  $J = \emptyset$ ，订单初始调度完毕，执行 **Step5**，否则执行 **Step2**；
- Step4** 记  $l_t$  使得  $t_{l_t} = \min_{1 \leq l \leq m}\{t_l\}$ ，修改流水线状态  $a_{l_t} = 0$ ，并更新系统时间  $t = t_{l_t}$ ，执行 **Step2**；
- Step5** 设定交替次数  $NR$ ，置  $k = 1$ ；
- Step6** 根据式 (4.9) 计算各流水线的  $H(S_l)$  值，选出具有最大值与最小值的流水线，分别记为  $l^+, l^-$ ；
- Step7** 根据式 (4.10) 计算流水线  $l^+$  的调度  $S_{l^+}$  中具有最大  $h(l_k)$  值的订单  $l_k^+$ ，并将其添入流水线  $l^-$  的调度  $S_{l^-}$  末端，更新流水线  $l^+, l^-$  的订单安排序列；
- Step8** 内部调整初始化。  $J = \bar{S}_l (l = l^+, l^-)$ ，置所选的流水线系统时间  $t_l = 0$ ，重置  $\bar{S}_l = \emptyset$ ；
- Step9** 根据式 (4.1)，选取预备调度订单  $l_k^*$ ，使得  $I_{l_k^*}(t) = \max_{l_k \in J}\{I_{l_k}(t)\}$ ，将订单  $l_k^*$  进行安排处理， $J = J - \{l_k^*\}$ ，将该订单排入该流水线的调度  $\bar{S}_l = \bar{S}_l \cup \{l_k^*\}$ 。若  $J = \emptyset$ ，该流水线上的订单调度完毕，执行 **Step11**，否则执行 **Step10**；
- Step10** 更新流水线系统时间  $t_l = t_l + p_{l_k^*}$ ，执行 **Step9**；
- Step11** 置  $k = k + 1$ ，若  $k \leq NR$ ，执行 **Step6**，否则终止算法。

上述算法步骤中，**Step1 – Step4** 的作用是利用 ATC 规则构造模型的初始解，**Step5 – Step6** 和 **Step11** 为流水线之间解的调整，内部嵌套着 **Step7 – Step10** 的流水线内部调整的过程。最终将直接得到调度结果  $S$ 。

### 4.4.2 Cyc – ATCS 算法

在模型 2 求解的过程中，与前面类似，可以采用重派订单，然后对受到影响的流水线分别使用 ATCS 规则进行调整，以改进解的策略。这样的求解过程称为 Cyc – ATCS (交替流水线间调整和流水线内 ATCS 调整)，其具体过程如下：

#### 算法 2 Cyc – ATCS 算法

- Step1** 初始化。  $J = N, \bar{L} = \emptyset, t_l = 0, \bar{S}_l = \emptyset, a_l = 0, C_{l_0} = 0, (l = 1, 2, \dots, m)^4$ ，置系统时间  $t = 0$ ；
- Step2** 若存在  $a_l = 0$ ，记  $l^* = \min_{a_l=0}\{l\}$ ，执行 **Step3**，否则执行 **Step5**；
- Step3** 根据式 (4.11)，选取预备调度订单  $j^*$ ，使得  $I_{j^*}(t) = \max_{j \in J}\{I_j(t)\}$ ，将订单  $j^*$  安排入流水线  $l^*$  进行处理，记入调度  $S_{l^*}$ ， $\bar{S}_{l^*} = \bar{S}_{l^*} \cup \{j^*\}, J = J - \{j^*\}$ ，记录调度订单序列  $\bar{L} = \bar{L} \cup \{j^*\}$ ；

<sup>4</sup> $C_{l_0}$  是人为定义的边界量，便于后续算法的初始情况

**Step4** 置  $k = |S_{l^+}|$ ，对于流水线  $l^*$ ，将订单  $j^*$  标记为  $l_k^*$ ，预估订单  $l_k^*$  开始处理前的流水线闲置时间  $f_{j^*} = f_{l_k^*} = \max\{r_{l_k^*} - s_{l_k^*} - C_{l_k^*}, 0\}$ ，更新流水线预计空闲时刻  $t_{l^*} = t + p_{j^*} + s_{j^*} + f_{j^*}$ ，修改流水线状态  $a_{l^*} = 1$ 。若  $J = \emptyset$ ，订单初始调度完毕，执行 **Step6**，否则执行 **Step2**；

**Step5** 记  $l_t$  使得  $t_{l_t} = \min_{1 \leq l \leq m} \{t_l\}$ ，修改流水线状态  $a_{l_t} = 0$ ，并更新系统时间  $t = t_{l_t}$ ，执行 **Step2**；

**Step6** 设定交替次数  $NR$ ，置  $k = 1$ ；

**Step7** 根据式 (4.13) 计算各流水线的  $H(S_l)$  值，选出具有最大值与最小值的流水线，分别记为  $l^+, l^-$ ；

**Step8** 根据式 (4.14) 计算流水线  $l^+$  的调度  $S_{l^+}$  中具有最大  $h(l_k)$  值的订单  $l_k^+$ ，并将其添入流水线  $l^-$  的调度  $S_{l^-}$  末端，更新流水线  $l^+, l^-$  的订单安排序列；

**Step9** 内部调整初始化。 $J = \overline{S}_l (l = l^+, l^-)$ ，置所选的流水线系统时间  $t_l = 0$ ，重置  $\overline{S}_l = \emptyset$ ；

**Step10** 根据式 (4.11)，选取预备调度订单  $l_k^*$ ，使得  $I_{l_k^*}(t) = \max_{l_k \in J} \{I_{l_k}(t)\}$ ，将订单  $l_k^*$  进行安排处理， $J = J - \{l_k^*\}$ ，将该订单排入该流水线的调度  $\overline{S}_l = \overline{S}_l \cup \{l_k^*\}$ 。若  $J = \emptyset$ ，该流水线上的订单调度完毕，执行 **Step12**，否则执行 **Step11**；

**Step11** 更新流水线系统时间  $t_l = t_l + p_{l_k^*}$ ，执行 **Step10**；

**Step12** 置  $k = k + 1$ ，若  $k \leq NR$ ，执行 **Step7**，否则终止算法。

该算法和 Cyc – ATC 算法类似，不同的是调度规则，其中 **Step1 – Step5** 将得到该模型的初始解，**Step6 – Step8** 和 **Step12** 为流水线之间解的调整，内部嵌套着 **Step9 – Step11** 的流水线内部调整的过程。最终将直接得到调度结果  $S$ 。

#### 4.4.3 Cyc – Tabu 算法

前面两个算法的内部调整策略是某个规则 (ATC/ATCS)，操作简单，运算速度也快，然而规则的制定却十分困难，虽然复合规则可以运用到其中，然而这样的规则缺乏普遍适用性，更为重要的是，ATCS 规则在调整模型 2 的解的时候，效果并不理想。在流水线之间内部调整改进解的时候，若采用禁忌搜索，便可以得到 Cyc – Tabu (交替流水线间调整和流水线内禁忌搜索调整) 策略：

##### 算法 3 Cyc – Tabu 算法

**Step1** 根据 ATC 或 ATCS 规则生成初始调度解  $S$ ；

**Step2** 设定交替次数  $NR$ ，置  $k_r = 1$ ；

**Step3** 根据式 (4.9) 或式 (4.13) 计算各流水线的  $H(S_l)$  值，选出具有最大值与最小值的流水线，分别记为  $l^+, l^-$ ；

**Step4** 根据式 (4.10) 或式 (4.14) 计算流水线  $l^+$  的调度  $S_{l^+}$  中具有最大  $h(l_k)$  值的订单  $l_k^+$ ，并将其添入流水线  $l^-$  的调度  $S_{l^-}$  末端，更新流水线  $l^+, l^-$  的订单安排序列；

**Step5** 初始化。设定迭代次数  $N_l$ ，清空禁忌列表  $TL$ ，设定列表长度  $NL$ ，将构造算法所得的调度作为初始调度，并记为当前最优调度， $S^{(0)} = S^{(1)} = S_l (l = l^+, l^-)$ ，并置  $k = 1$ ；

**Step6** 从  $S^{(k)}$  所有不在禁忌列表中的相邻移动  $(l_j, l_k)$  中，所得调度具有最小函数值的移动，记为  $(l_j^*, l_k^*)$ ，所得调度记为  $S^*$ ，并置  $S^{(k+1)} = S^*$ ；

**Step7** 将相邻移动  $(l_j^*, l_k^*)$  入栈禁忌列表，若列表容量已满，则按 FIFO 规则出栈最早的相邻移动；

**Step8** 若  $G(S^*) < G(S^{(0)})$ ，置  $S^{(0)} = S^*$ ；

**Step9** 置  $k = k + 1$ ，若  $k \leq N_l$ ，执行 **Step6**，否则禁忌搜索调整完成，更新调度解  $S$ ，执行 **Step10**；

**Step10** 置  $k_r = k_r + 1$ ，若  $k_r \leq NR$ ，执行 **Step2**，否则终止算法。



上述算法中，**Step2 – Step4** 和 **Step10** 为流水线之间调整步骤，内部嵌套着 **Step5 – Step9** 的使用禁忌搜索的内部调整步骤。该算法将得到调度结果  $S$  和相应的目标函数值  $G(S)$ 。

#### 4.4.4 Vtr – Tabu 算法

##### 算法 4 Vtr – Tabu 算法

**Step1** 运用调度规则 (如 ATC、ATCS) 建立流水线全局调度初始解，得到虚拟序列  $L$  及其初始调度  $S^{(0)}$ ，并将其作为目前最优调度。设定禁忌搜索迭代次数  $N_I$ ，设定列表长度  $NL$ ，并置特赦调度  $A = S^{(0)}$ ；

**Step2** 置  $S^{(1)} = S^{(0)}$ ，清空禁忌列表  $TL$ ，置  $k = 1$ ；

**Step3** 在  $L$  所生成的邻域中，按顺序选取  $(L_m, L_n)$ ，记当前调度为  $S^-$ ，若  $L_m, L_n$  当前均安排在同一流水线的调度中，则执行 **Step4**，否则执行 **Step5**；

**Step4** 交换订单对顺序，得到新的调度为  $S^+$  型；

**Step5** 将订单  $L_m$  重派入流水线  $l'$ ，得到调度为  $S^{a+}$  型，或将订单  $L_n$  重派入流水线  $l$  得到调度为  $S^{b+}$  型，或将订单  $L_m, L_n$  交换位置，得到调度为  $S^+$  型；

**Step6** 更新虚拟序列中这两个订单的位置为  $(L_m, L_n)$ 。

**Step7** 检查禁忌列表中的订单对，若它们别安排在不同的流水线，则只对其交换位置的移动禁忌；若移动后的调度为特赦调度，一样认定为可行移动。计算  $S^{(k)}$  中所有可行移动组成的邻域，选取它们中具有最小函数值调度的移动，记该订单对为  $(L_m^*, L_n^*)$ ，所得调度记为  $S^*$ ，并置  $S^{(k+1)} = S^*$ ；

**Step8** 若相邻移动所得调度属于  $S^+$  型，则将  $(L_m^*, L_n^*)$  入栈禁忌列表，若列表容量已满，则按 FIFO 规则出栈最早的相邻移动，检查禁忌列表，删除过禁忌项；

**Step9** 若  $G(S^*) < G(S^{(0)})$ ，置  $S^{(0)} = S^*$ ；

**Step10** 置  $k = k + 1$ ，若  $k \leq N_I$ ，执行 **Step3**，否则终止算法， $S^{(0)}$  为最终所得调度。

与前 Cyc 类述算法不同的是，该算法直接从全局考虑订单序列，其中 **Step3 – Step6** 的作用是确保 1 个实际调度对应于 1 个虚拟序列，以确保该算法可以正确进行。该算法将得到最终调度  $S$  和其对应的目标函数值  $G(S)$ 。

#### 4.4.5 VVT 算法

Vtr – Tabu 算法可以得到较有的结果，然而由于其邻域结构的特点，可能需要很大的迭代次数才能将解改进。采用变动邻域的策略可以人为切换邻域结构，放弃一些需要过多迭代次数的邻域结构，以减少计算时间，这样的综合策略称为 VVT (变动邻域结构的虚拟序列禁忌搜索)。

##### 算法 5 VVT 算法

**Step1** 运用调度规则 (如 ATC、ATCS) 建立流水线全局调度初始解，得到虚拟序列  $L$  及其初始调度  $S^{(0)}$ ，并将其作为目前最优调度，将其邻域集合  $\overline{S^{(c)}}$  中的调度按函数值的非减排列，记为  $S_{[1]}, S_{[2]}, \dots, S_{[|S^{(c)}|]}$ ，置  $i = 1$ 。设定禁忌搜索迭代次数  $N_I$ ，设定列表长度  $NL$ ，并置特赦调度  $A = S^{(0)}$ ；

**Step2** 若  $i \leq |S^{(c)}|$ ，置  $S^{(1)} = S_{[i]}$ ，清空禁忌列表  $TL$ ，置  $k = 1$ ，否则终止算法；

**Step3** 在  $L$  所生成的邻域中，按顺序选取  $(L_m, L_n)$ ，记当前调度为  $S^-$ ，若  $L_m, L_n$  当前均安排在同一流水线的调度中，则执行 **Step4**，否则执行 **Step5**；

**Step4** 交换订单对顺序，得到新的调度为  $S^+$  型；

**Step5** 将订单  $L_m$  重派入流水线  $l'$ ，得到调度为  $S^{a+}$  型，或将订单  $L_n$  重派入流水线  $l$  得到调度为  $S^{b+}$  型，或将订单  $L_m, L_n$  交换位置，得到调度为  $S^+$  型；

**Step6** 更新虚拟序列中这两个订单的位置为  $(L_m, L_n)$ 。

**Step7** 计算  $S^{(k)}$  中所有可行移动组成的邻域，选取它们中具有最小函数值调度的移动，记该订单对为  $(L_m^*, L_n^*)$ ，所得调度记为  $S^*$ ，并置  $S^{(k+1)} = S^*$ ；

**Step8** 若相邻移动所得调度属于  $S^+$  型，则将  $(L_m^*, L_n^*)$  入栈禁忌列表，若列表容量已满，则按 FIFO 规则出栈最早的相邻移动；

**Step9** 若  $G(S^*) < G(S^{(0)})$ ，置  $S^{(0)} = S^*$ ；

**Step10** 置  $k = k + 1$ ，若连续 50 次采用没有更新  $S^{(0)}$ ，则置  $i = i + 1$ ，执行 **Step2**，否则若  $k \leq N_I$ ，执行 **Step3**，否则终止算法， $S^{(0)}$  为最终所得调度。

VVT 算法改进了 Vtr-Tabu，使之不会陷入一个邻域花费过多运算次数，其中 **Step1** 确定了待搜索的邻域结构，**Step2** – **Step9** 的操作和 Vtr-Tabu 算法基本一致，内嵌在邻域切换 **Step1** 和 **Step10** 中，该算法将得到调度结果  $S$  和相应的目标函数值  $G(S)$ 。

## 4.5 小结

本章针对模型 1 和模型 2 一共设计了 5 个求解算法，可以分为 Cyc (交替) 类和 Vrt (虚拟序列) 类，Cyc 类算法中，使用规则的 Cyc-ATC 和 Cyc-ATCS 算法操作方便，可以很快得到结果，然而这两个规则可能不能普遍适用其他情况的问题，而且得到的结果质量不是很高。Cyc-Tabu 算法适用性比前两种强，而且得到解的质量较好。

与 Cyc 类算法不同的是，Vtr 类算法中解的改进不是采用流水线之间调整和流水线内部调整交替进行的策略，而是直接考虑全局订单，并且通过虚拟序列的设计，巧妙地设计了所搜的邻域结构，后面的实验证明这类算法可以得到质量较好的解。然而由于邻域结构的特点 Vtr-Tabu 算法运算时间普遍较长，因此设计了 VVT 算法，通过变动邻域的策略，动态得选择邻域结构，节省了运算时间，但实验表明该算法结果波动较大，改进效果不如 Vtr-Tabu 算法。

## 第五章 计算实验及算法评估

本章将对上一章的算法进行评估，通过设计相关计算实验，建立评价体系，具体评估各算法的效果及其适用规模。然后根据实验结果，为研究对象制定合适的调度方案。

### 5.1 实验设计

为了评估上一章所提出的算法的运行效果，以及给出所得调度方案，供决策者选择，所有的算法都用 Python 语言编写<sup>1</sup>，并在 2.20GHz i3 CPU，5.84GB RAM，Windows 8 的个人计算机上运行测试。

实验设计分为两个主要部分，一个是装配生产信息的生成，包括订单数量、流水线数量、各订单切换准备时间、个订单进入系统时间及其所含作业的信息；另一个是相关参数的确定，包括迭代次数、禁忌列表长度的等。

#### 5.1.1 生产装配信息生成

生产装配信息皆由随机数产生，主要包括订单及其作业的数量信息、时间信息及惩罚系数，需要给出适当的取值范围及分布。

##### (1) 数量信息

由表 2-1，设计实验中的流水线的数量  $m$  分别为 5, 6, 7，订单品种数量的范围是  $[29, 777]$ ，据此可以设计合理的订单数量  $n$  分别为 20, 30, 50, 70, 100, 150, 200, 300, 500, 750, 1000，再根据大批量的特点，可以设置订单中作业的批量范围为  $(1000, 2000)$ 。

##### (2) 时间信息

出于保护该公司的生产能力数据，实验中所采用的时间单位是经变换过的  $tu$ ，由于每个订单的作业批量都比较大，所以订单中的作业处理时间单位为  $1/500 tu$ 。订单中的作业处理时间服从参数  $\alpha = 5$  的负指数分布，订单到达系统的时间间隔服从参数  $\lambda = 3$  的泊松分布

##### (3) 惩罚系数与优先系数

多品种多装配线轮番装配调度优化模型的目标函数式 (3.2) 中，延迟完成的订单有惩罚系数  $w_t$ ，订单的完成有优先系数  $w_c$ ，考虑插单模型也有类似的这两类系数，皆由随机数产生。与模型中不同的是，由于订单中的批量增多，这些系数若满足求和为 1 的约束使得每个系数的值都很小，不利计算的准确性，而同时成倍扩大不影响调度的结果，所以这些系数不作求和为 1 的处理。

实验信息数据生成的代码见附录 (experiment\_data.py)，生成的数据在目录 (.data) 中，举例来说，20 件订单的数据所含内容<sup>2</sup>如表 5-1 所示。

#### 5.1.2 相关参数确定

使用调度规则 (ATC/ATCS) 的交替算法所含的相关参数为交替次数  $NR$ ，由于该算法收敛速度很快，该参数无需特别设定，一般取  $NR = 30$  就基本稳定在最佳的两个结果之间交替。

<sup>1</sup>版本为 Python 2.7.6

<sup>2</sup>本课题提出的两个模型使用相同的数据，考虑插单模型将用到进入系统时间  $r_j$

表 5-1 20 件订单的实验信息数据

单位: $tu$						
订单序号 ( $j$ )	装配时间 ( $p_j$ )	进入系统时刻 ( $r_j$ )	切换准备时间 ( $s_j$ )	约定工期 ( $d_j$ )	主要目标权重 ( $w_t$ )	次要目标权重 ( $w_c$ )
1	23	56	7	100	15	2
2	40	30	1	116	8	3
3	36	20	7	98	10	4
4	25	4	2	49	12	10
5	17	5	5	37	11	2
6	28	42	1	101	2	7
7	20	44	6	87	8	10
8	21	63	8	105	12	9
9	15	40	2	71	2	7
10	18	10	6	43	7	6
11	30	6	7	70	14	7
12	20	2	8	36	4	1
13	16	46	1	79	3	4
14	35	36	4	112	8	6
15	31	52	4	109	14	7
16	24	27	3	70	13	5
17	19	62	2	96	6	3
18	35	13	3	91	12	5
19	31	2	8	62	4	10
20	24	60	6	101	3	8

实验中涉及到禁忌搜索的部分，其相关参数包括迭代次数、禁忌列表长度等，迭代次数增多会增大改进解的机率，但是会使得运算时间变长，禁忌列表过长会浪费迭代次数，过短则可能跳不出局部最优。所以需要在实验前需要设定恰当的参数，以订单数量  $n = 100$  的模型 1 为例，运用 Vrt-Tabu 算法，决策权重  $\lambda_t = 0.6, \lambda_c = 0.4$ ，流水线数量  $m = 5$  时，在初始解生成迭代次数  $N_{init} = 20$  的情况下，禁忌列表长度与初始解的函数值关系如图 5-1a 所示。

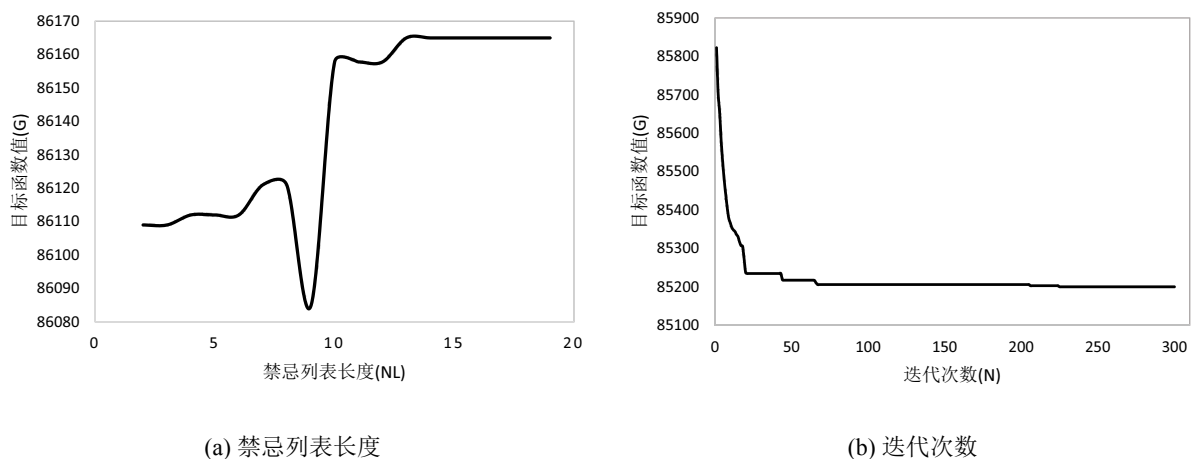


图 5-1 100 件订单的目标函数值和相关参数的关系

可以看出  $NL = 9$  是最佳列表长度，确定该值后，生成目标函数值与迭代次数的关系，如图 5-1b 所示。虽然随着迭代次数的增加，目标函数值仍然会减少，其减少幅度不大，并且运算时间增加很多，所以取迭代

次数  $N = 70$  是较为适宜的<sup>3</sup>。

以此类推，可以确定不同问题规模下的这两个参数的值<sup>4</sup>，模型 1 和模型 2 相关参数设定结果分别如表 5-2 所示。

表 5-2 算法相关参数确定

(a) 模型 1 相关参数确定

参数		订单数量										
		20	30	50	70	100	150	200	300	500	750	1000
$m = 5$	$N$	3	16	65	30	70	100	1850	1700	238	121	267
	$NL$	2	2	8	6	9	29	61	138	164	152	90
$m = 6$	$N$	14	10	25	60	130	200	850	420	350	125	440
	$NL$	2	2	2	6	20	31	52	105	190	220	312
$m = 7$	$N$	8	13	60	55	19	200	320	275	80	125	80
	$NL$	2	2	4	14	6	21	26	48	160	227	174

(b) 模型 2 相关参数确定

参数		订单数量										
		20	30	50	70	100	150	200	300	500	750	1000
$m = 5$	$N$	5	12	20	40	66	121	152	313	99	308	241
	$NL$	2	3	16	25	44	44	63	70	71	67	62
$m = 6$	$N$	20	18	25	19	53	231	165	431	157	211	153
	$NL$	2	2	28	20	33	11	8	40	18	22	31
$m = 7$	$N$	3	9	32	25	132	200	222	109	88	106	124
	$NL$	2	3	21	60	50	29	92	22	40	21	52

可以看出，迭代次数和禁忌列表长度和问题的规模有一定的正相关关系，问题规模越大，这两者的值也越大，但这两个参数主也和订单的实验信息数据特点有关。

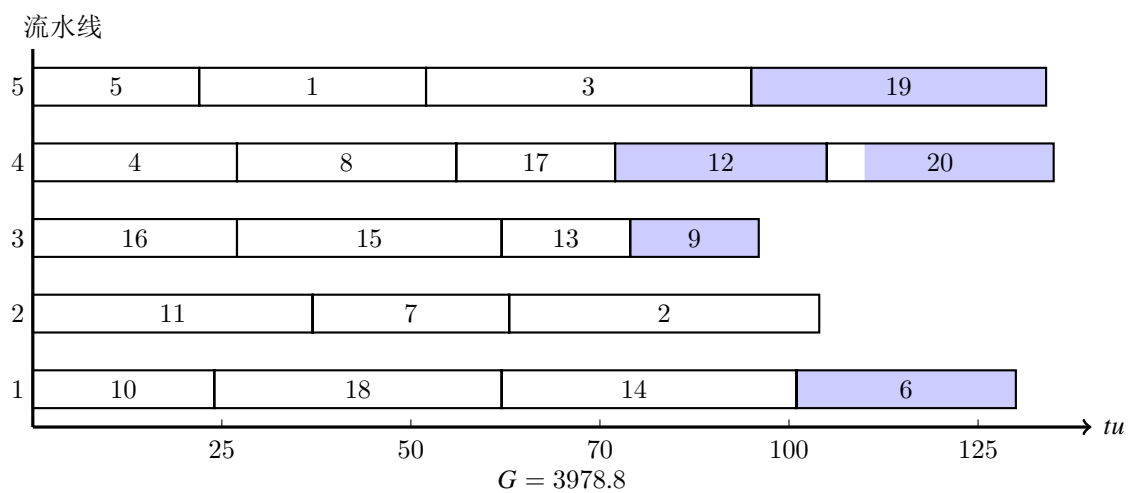
## 5.2 结果及评估

### 5.2.1 实验结果示例及说明

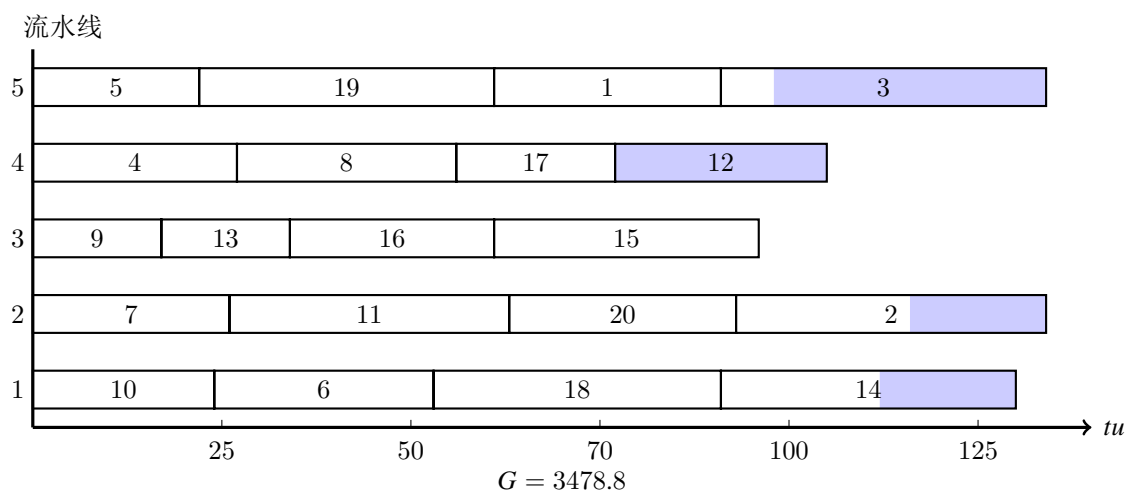
确定实验所需订单信息数据以及算法相关的参数，运行编写的算法程序，可以得到实验结果。实验结果由目标函数值  $G$  和所得调度序列  $S$  组成，以模型 1 为例，示例订单量  $n = 20$ ，其实验信息数据如表 5-1 所示，在决策参数  $\lambda_1 = 0.6, \lambda_2 = 0.4$  的环境下，分别采用 Cyc-ATC 算法、Cyc-Tabu 算法和 Vtr-Tabu 算法，根据表 5-2 选择相关参数，所得结果列入表 5-3 中，其中深色部分为订单的延迟，其跨度越长，说明延迟越大。其调度安排如图 5-2 所示。从结果上来看，就该决策环境下，模型 1 使用 Vtr-Tabu 算法可以得到较好调度结果。

<sup>3</sup>实际上  $G|_{N=70} = 85206, G|_{N=300} = 85200.2$ ，两者相差仅 0.0068%

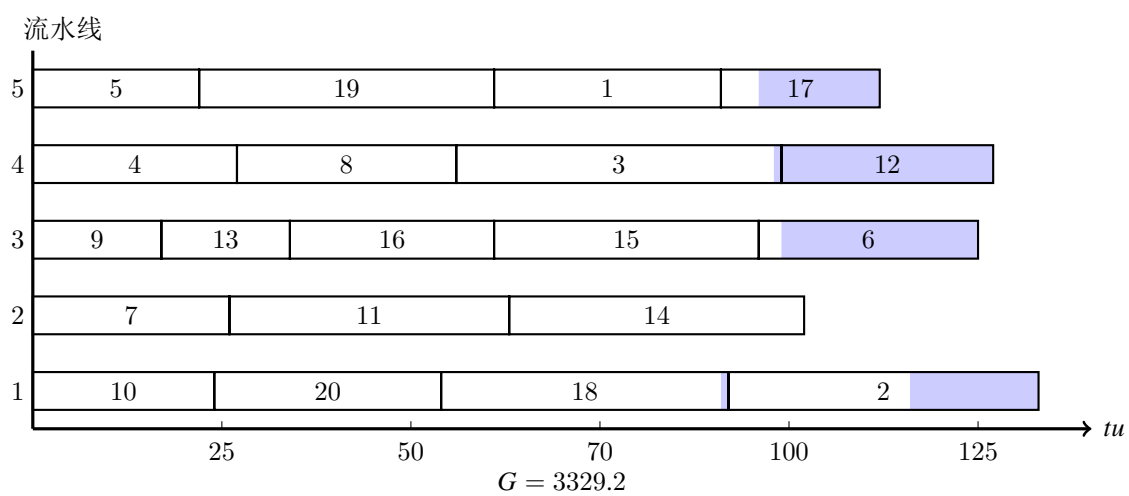
<sup>4</sup>经测试，不用决策环境  $(\lambda_1, \lambda_2)$  下，禁忌列表长度  $NL$  和迭代次数  $N$  基本不变，所以这两个参数适用于不同的决策环境下



(a) Cyc – ATC 算法调度结果



(b) Cyc – Tabu 算法调度结果



(c) Vtr – Tabu 算法调度结果

图 5-2 20 件订单的三种算法求解模型 1 所得结果

表 5-3 订单量  $n = 20$  的模型 1 3 种算法求解结果

算法	目标函数值 ( $G$ )	流水线调度安排
Cyc – ATC	3978.8	1 10 → 18 → 14 → 6
		2 11 → 7 → 2
		3 16 → 15 → 13 → 9
		4 4 → 8 → 17 → 12 → 20
		5 5 → 1 → 3 → 19
Cyc – Tabu	3478.8	1 10 → 6 → 18 → 14
		2 7 → 11 → 20 → 2
		3 9 → 13 → 16 → 15
		4 4 → 8 → 17 → 12
		5 5 → 19 → 1 → 3
Vtr – Tabu	3329.2	1 10 → 20 → 18 → 2
		2 7 → 11 → 14
		3 9 → 13 → 16 → 15 → 6
		4 4 → 8 → 3 → 12
		5 5 → 19 → 1 → 17

5.2.2 模型 1 求解结果与分析

5.2.2.1 目标函数值分析

模型 1 的求解可以使用 3 个不同的算法 Cyc – ATC、Cyc – Tabu、Vtr – Tabu 得到的结果目标函数值如表 A-1 – A-3所示，图 A-1 – A-3为该结果的图表形式。由结果目标函数值可以看出，Cyc – Tabu 和 Vtr – Tabu 算法都较 Cyc – ATC 算法能得到目标函数值小的解，并且大多数情况下 Vtr – Tabu 算法得到的目标函数值只比 Cyc – ATC 算法要小一点点，尤其在问题规模较大的时候。问题规模较小的时候，Vtr – Tabu 算法对解的改进效果比较明显。并且随着决策参数  $\lambda_1$  的增大，Vtr – Tabu 算法的改进效果也越为明显，也就是说，如果决策者对交付期较为重视的话，选择 Vtr – Tabu 算法来求解调度方案是比较好的。

5.2.2.2 不同决策环境分析

以  $m = 6, n = 200$  为例，决策参数和目标函数值的关系如图 5-3 所示。可以看出，随着决策参数  $\lambda_1$  的增大，即更为看重工期目标，三个算法所得的目标值都减少，说明模型 1 适用于强调工期的调度问题。其他的决策环境下，所得结论基本一致。

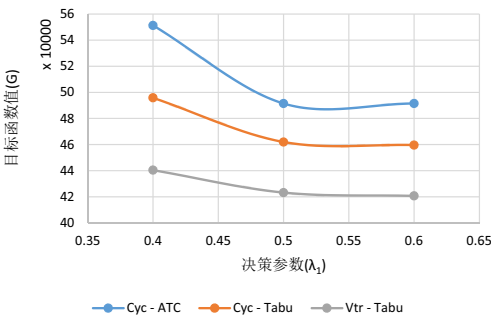


图 5-3 决策参数和目标函数值关系示例

## 5.2.3 模型 2 求解结果及分析

### 5.2.3.1 目标函数值分析

模型 2 的求解可以使用 3 个不同的算法 Cyc-ATCS、Cyc-Tabu、VVT 得到的结果目标函数值如表 A-4 – A-6 所示，图 A-4 – A-6 为该结果的图表形式。可以看出，当订单数量  $n$  较小时，Cyc-Tabu 算法的目标函数值会小于 Cyc-ATCS 算法，VVT 算法所得的目标函数值最小。当  $n$  较大的时候 ( $n \geq 200$ ) 时，Cyc-Tabu 算法则出现了不稳定现象，大多数情况其所得函数值不如 Cyc-ATCS 算法，而 VVT 算法在这种情况下，大多仍然有最小的目标函数值。此外，随着流水线数量  $m$  的增加，Cyc-Tabu 算法和 VVT 对解的改进效果和 Cyc-ATCS 相差不大 (除个别情况外)，但 VVT 算法所得结果较为稳定。

所以，Cyc-Tabu 算法较为适用于订单数量少的环境，而 VVT 算法在订单数量多和少的情况均适用，而且较 Cyc-Tabu 算法可以得到稳定的结果。

### 5.2.3.2 流水线均衡率分析

不同算法得到的调度结果下，流水线均衡率如表 A-7 – A-9 所示，图 A-7 – A-9 为流水线均衡率的图表形式。在决策参数  $\lambda_1 = 0.5$  的环境下，3 种算法所得不同流水线数所得调度的流水线均衡率如图 5-4 所示。

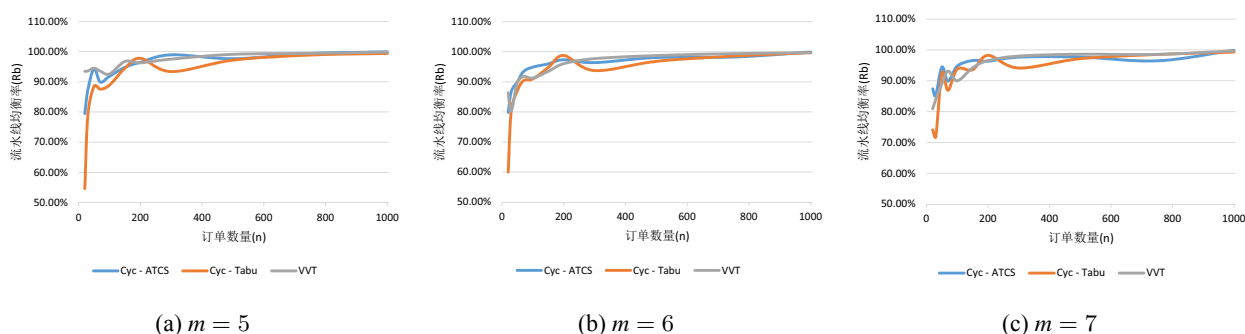


图 5-4 不同流水线数量和流水线均衡率关系

可以看出，3 个算法中，VVT 算法所得的流水线均衡率随订单数量的增加最为稳定，始终可以保持一个较高的水平，而 Cyc-Tabu 算法则会有较大波动。随着订单数量增加，各算法所得的流水线均衡率都大约成对数趋势增加趋向于 100%，一般在  $n = 500$  左右稳定在较高值，这十分符合常理，并且随着流水线数量的增加，各算法的稳定点都前移 ( $n$  减少的方向)。

### 5.2.3.3 不同决策环境分析

流水线  $m = 6$  的环境下，3 种算法所得不同决策环境下调度的流水线均衡率如图 5-5 所示。

与流水线均衡率分析类似，不同决策环境下，VVT 算法所得的流水线均衡率随订单数量的增加最为稳定，始终可以保持一个较高的水平，而 Cyc-Tabu 算法所得结果由较大波动，而且随着决策参数  $\lambda_1$  的增加，即大工期目标的重要性，各算法都会逐渐产生较为稳定的流水线均衡率，并且稳定点都前移。说明模型 2 的设计是符合工期要求的，并且能明显提高流水线生产均衡率。



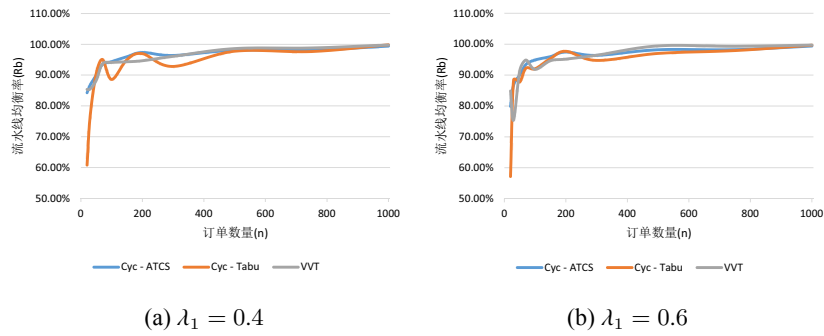


图 5-5 决策环境和流水线均衡率关系

### 5.3 小结

本章针对上一张设计的 5 个求解多品种多装配线轮番装配调度优化模型的算法，设计了相应的 Python 脚本，并根据该汽车电子有限公司装配车间的生产信息，生成了相应的订单信息实验数据，用这些数据得到了计算结果，并对这些结果进行了分析。

实验结果表明，对于模型 1，Vtr-Tabu 算法在问题规模小的时候可以得到较好的调度，当问题规模大的时候，Cyc-Tabu 算法和 Vtr-Tabu 算法都可以得到较好的调度，并且相差不大。当决策者更为注重工期目标时，选用 Vtr-Tabu 算法可以得到较好的调度。对于模型 2，无论在目标函数值还是流水线均衡率上，VVT 算法都可以得到较稳定且质量较高的调度解，而 Cyc-Tabu 算法在问题规模较大的时候，结果不稳定现象会加重，因此 Cyc-Tabu 算法较为适用于订单数量少的环境，而 VVT 算法在订单数量多和少的情况均适用。决策者若增大工期目标的重要性，各算法都会逐渐产生较为稳定的流水线均衡率，并且稳定点都前移，这说明模型 2 的设计是合理的。

## 第六章 总结和展望

### 6.1 总结

本文的前几章由某汽车电子有限公司的装配车间现存问题，提出改进方案，并将之抽象成多品种多装配线轮番调度优化模型，然后设计了多个算法对该模型进行求解，并通过 Python 脚本进行了计算实验，评价了求解结果及算法的适用性。

由第二章的装配车间生产现状及其问题分析，所提出的打破主机厂限制的解决方案，引导出了多品种多装配线的混流生产调度问题，并在第三章进行相关模型建立，其中模型 2 要比模型 1 多考虑插单的情况，第五章的实验从一些方面说明了这两个模型设计的合理性。第四章从初始解的生成和解的改进设计了 Cyc 类的算法，根据流水线之间和流水线内部调整的策略不同，可以得到不同的求解算法。值得一提的是，这一章中本文提出了虚拟序列的概念，并将之运用到了算法的设计中，并且由实验表明，使用了虚拟序列策略的算法 Vtr-Tabu 和 VVT 算法都能得到质量较高的调度结果。

### 6.2 展望

多品种装配车间调度问题有许多方面是本文没有涉及的，仅从改进方案的方面来看，本文仍可以在换线轮番装配策略基础上，考虑各流水线的混流生产情况，进一步均衡生产。由于本文在建模的时候假设了诸多限制，例如订单的不可中断假设，逐步消除这些假设的限制，可以使模型更为接近实际情况。此外，在算法设计上，调度规则的制定是一个较为复杂的过程，而且还有许多其他领域的算法可以用于车间调度问题，需要去探索。

## 参考文献

- [1] 徐俊刚, 戴国忠, 王宏安. 生产调度理论和方法研究综述 [J]. 计算机研究与发展, 2004. 41(2):257–267.
- [2] Zhiqiang Xie, Shuzhen Hao, Lei Zhang, Jing Yang. Study on integrated algorithm of complex multi-product flexible scheduling[C]//Advanced Computer Control (ICACC), 2010 2nd International Conference on. volume 1. IEEE, 2010:553–557.
- [3] 唐勇智. 基于聚类的 RBF-LBF 串联神经网络的学习算法及其应用 [D]. : 江苏: 江南大学, 2009.
- [4] 陈伟. 生物信息学中的序列相似性比对算法 [D]. : 山东: 中国海洋大学, 2006.
- [5] 李斌, 林飞龙. 多品种多工艺车间调度建模, 分析与优化 [J]. 物流技术, 2009. (8):136–139.
- [6] MA Adibi, M Zandieh, M Amiri. Multi-objective scheduling of dynamic job shop using variable neighborhood search[J]. Expert Systems with Applications, 2010. 37(1):282–287.
- [7] 刘文平. 混合品种汽车装配线平衡与排序问题研究 [D]. : 山东: 山东大学, 2009.
- [8] 杨本强. 线性规划理论在汽车装配线均衡问题中的应用研究 [D]. : 重庆: 重庆大学, 2002.
- [9] 李宏霞, 彭威, 史海波. 装配车间的多品种变批量的生产调度优化模型 [J]. 机械设计与制造, 2006. (6): 94–96.
- [10] Virginia Lo, Jens Mache. Job scheduling for prime time vs. non-prime time[C]//Cluster Computing, 2002. Proceedings. 2002 IEEE International Conference on. IEEE, 2002:488–493.
- [11] Bruno Jensen Virginio Da Silva, Reinaldo Morabito, Denise Sato Yamashita, Horacio Hideki Yanasse. Production scheduling of assembly fixtures in the aeronautical industry[J]. Computers & Industrial Engineering, 2014. 67:195–203.
- [12] A Tharumarajah, R Bemelman, P Welgama, A Wells. Distributed scheduling of an assembly shop[C]//Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on. volume 1. IEEE, 1998:433–438.
- [13] 李志娟, 王冠. 高校自动排课算法的研究与设计 [J]. 计算机与数字工程, 2008. 36(5):199–202.
- [14] Parames Chutima, Wanwisa Naruemitwong. A pareto biogeography-based optimisation for multi-objective two-sided assembly line sequencing problems with a learning effect[J]. Computers & Industrial Engineering, 2014.
- [15] 陈琳, 严洪森, 刘通, 刘霞玲. 汽车装配线生产计划与调度的集成优化方法 [J]. 计算机技术与发展 ISTIC, 2009. 19(1).
- [16] Ham-Huah Hsu, Li-Chen Fu. Fully automated robotic assembly cell: scheduling and simulation[C]//Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on. volume 1. IEEE, 1995:208–214.
- [17] Shuai Jia, Zhi-Hua Hu. Path-relinking tabu search for the multi-objective flexible job shop scheduling problem[J]. Computers & Operations Research, 2014. 47:11–26.
- [18] Ghasem Moslehi, Mehdi Mahnam. A pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search[J]. International Journal of Production Economics, 2011. 129(1):14–22.
- [19] Rock Lin, Ching-Jong Liao. A case study of batch scheduling for an assembly shop[J]. International Journal of Production Economics, 2012. 139(2):473–483.

- [20] 朱有产, 王春梅, 刘虎. 基于空间方向关系的配电网实用最佳抢修路径算法 [J]. 电气应用, 2006. 25(1): 41–43.
- [21] 高丽, 徐克林, 杨娜娜. 多品种柔性生产企业的订单调度模型及其遗传算法 [J]. 数学的实践与认识, 2012. 42(21):154–161.
- [22] 曾洪鑫, 宾鸿赞. 遗传算法在多品种装配生产排序中的应用 [J]. 现代制造工程, 2006. (7):59–62.
- [23] 翁武熙. 混合蚁群算法求解 TSP 问题 [D]. : 广西: 广西大学, 2012.
- [24] Pinedo Michael. Scheduling: theory, algorithms, and systems[M]. 2nd edition. Springer, 2002.
- [25] Ümit Bilge, Müjde Kurtulan, Furkan Kıraç. A tabu search algorithm for the single machine total weighted tardiness problem[J]. European Journal of Operational Research, 2007. 176(3):1423–1435.
- [26] 史烨, 李凯. 并行机问题的模拟退火调度算法研究 [J]. 运筹与管理, 2011. (4):104–107.

## 附录 A 实验结果图表

表 A-1 Cyc – ATC、Cyc – Tabu、Vtr – Tabu 算法求解模型 1 目标函数值 ( $\lambda_1 = 0.4$ )

流水线数量 ( $m$ )	订单数量 ( $n$ )	20	30	50	70	100	150
5	Cyc - ATC	5581	11585	35912	58236	116557	289619
	Cyc - Tabu	4945	9627	29568.6	46551.6	94332.8	229521.2
	Vtr - Tabu	4203	8003.4	25104.8	39186.6	79288.2	187143.2
6	Cyc - ATC	4756	9840.6	30910	52190.8	105418.6	262350.8
	Cyc - Tabu	4117	8133.8	25205	40733.8	81944	205849
	Vtr - Tabu	3602	6980.4	20784.6	34011.2	68408.6	167215
7	Cyc - ATC	4661	9583.6	30563.2	51475.8	104903.8	260983.4
	Cyc - Tabu	3979	7978.8	24748.6	40596.8	81936.6	204826.8
	Vtr - Tabu	3484	6828.2	20403.2	33730.6	67947.2	166778

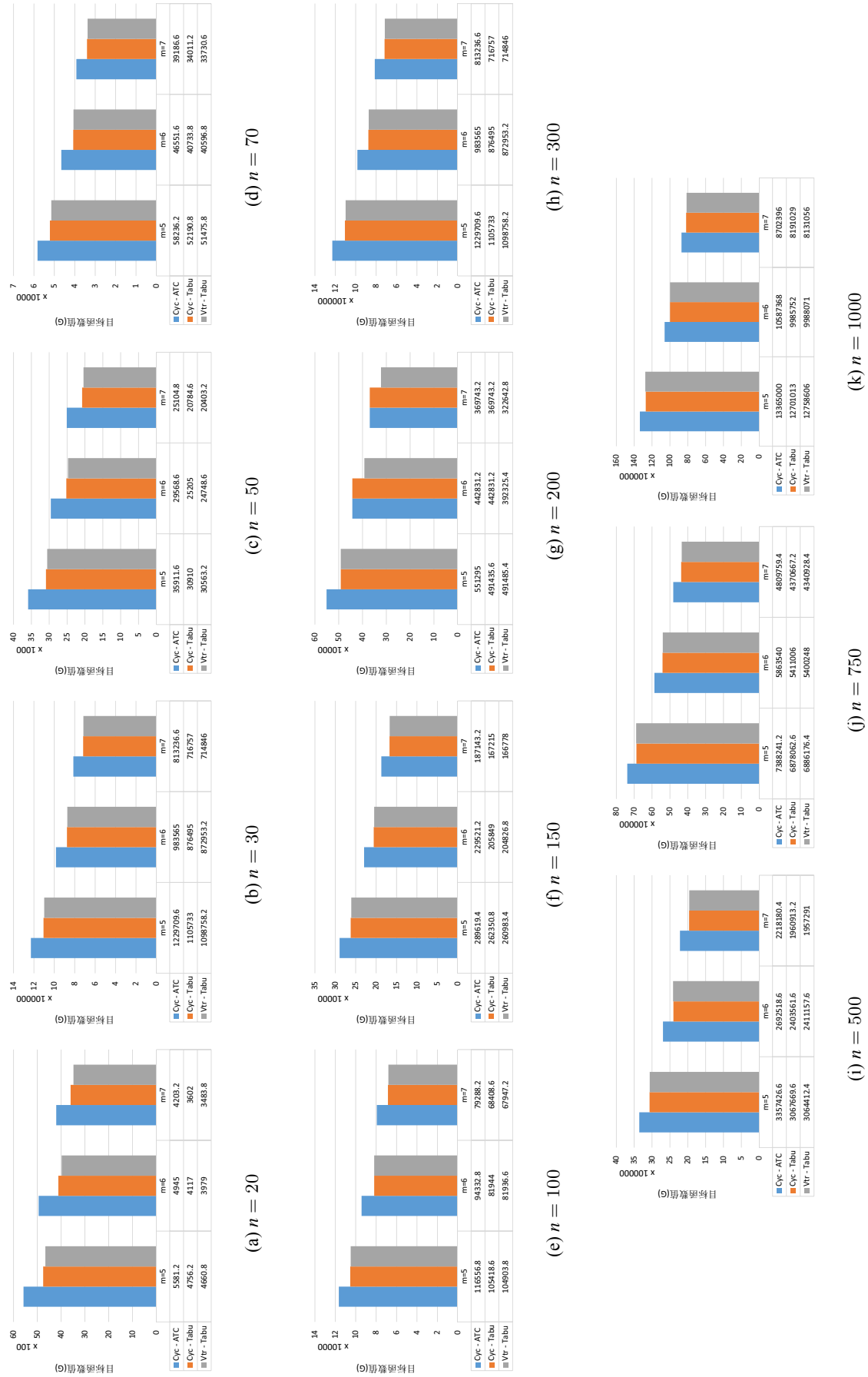
流水线数量 ( $m$ )	订单数量 ( $n$ )	200	300	500	750	1000
5	Cyc - ATC	551295	1229710	3357427	7388241	13365000
	Cyc - Tabu	442831.2	983565	2692518.6	5863540	10587368
	Vtr - Tabu	369743.2	813236.6	2218180.4	4809759.4	8702396.2
6	Cyc - ATC	491435.6	1105733	3067669.6	6878062.6	12701013
	Cyc - Tabu	442831.2	876495	2403561.6	5411006	9985752
	Vtr - Tabu	369743.2	716757	1960913.2	4370667.2	8191029.4
7	Cyc - ATC	491485.4	1098758.2	3064412.4	6886176.4	12758606
	Cyc - Tabu	392325.4	872953.2	2411157.6	5400248	9988071.2
	Vtr - Tabu	322642.8	714846	1957291	4340928.4	8131055.6

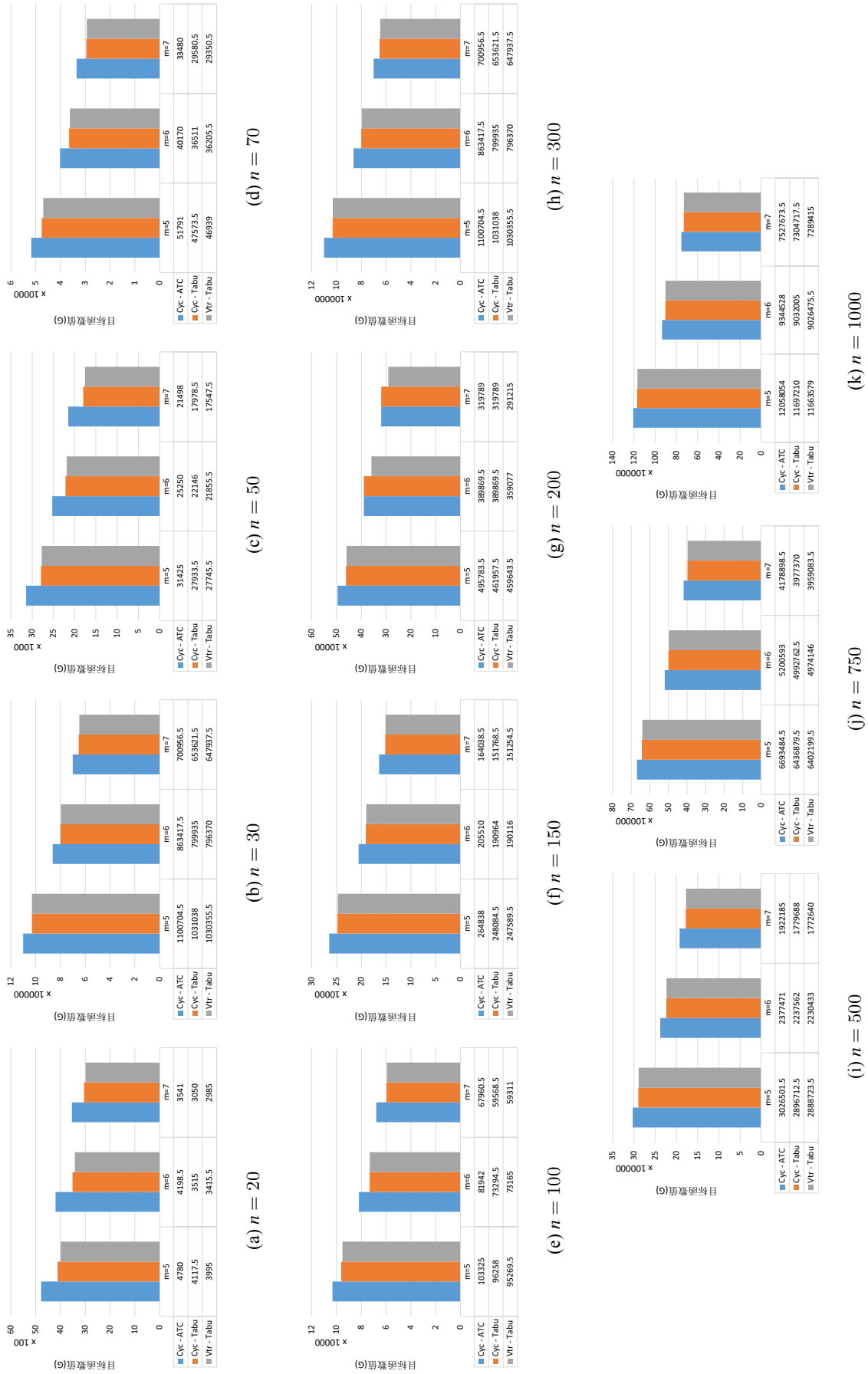
表 A-2 Cyc – ATC、Cyc – Tabu、Vtr – Tabu 算法求解模型 1 目标函数值 ( $\lambda_1 = 0.5$ )

流水线数量 ( $m$ )	订单数量 ( $n$ )	20	30	50	70	100	150
5	Cyc - ATC	4780	9978	31425	51791	103325	264838
	Cyc - Tabu	4199	8192	25250	40170	81942	205510
	Vtr - Tabu	3541	6758.5	21498	33480	67960.5	164038.5
6	Cyc - ATC	4118	8740	27933.5	47573.5	96258	248084.5
	Cyc - Tabu	3515	7129.5	22146	36511	73294.5	190964
	Vtr - Tabu	3050	6011.5	17978.5	29580.5	59568.5	151768.5
7	Cyc - ATC	3995	8527	27745.5	46939	95269.5	247589.5
	Cyc - Tabu	3416	6986.5	21855.5	36205.5	73165	190116
	Vtr - Tabu	2985	5888.5	17547.5	29350.5	59311	151254.5

流水线数量 ( $m$ )	订单数量 ( $n$ )	200	300	500	750	1000
5	Cyc - ATC	495784	1100705	3026502	6693485	12058054
	Cyc - Tabu	389869.5	863417.5	2377471	5200593	9344528
	Vtr - Tabu	319789	700956.5	1922185	4178898.5	7527673.5
6	Cyc - ATC	461957.5	1031038	2896712.5	6436879.5	11697210
	Cyc - Tabu	389869.5	799935	2237562	4992762.5	9032005
	Vtr - Tabu	319789	653621.5	1779688	3977370	7304717.5
7	Cyc - ATC	459643.5	1030355.5	2888723.5	6402199.5	11663579
	Cyc - Tabu	359077	796370	2230433	4974146	9026475.5
	Vtr - Tabu	291215	647937.5	1772640	3959083.5	7289415

图 A-1 模型 1 的 Cyc-ATC、Cyc-Tabu、Vtr-Tabu 算法求解目标函数数值比较 ( $\lambda_1 = 0.4$ )

图 A-2 模型 1 的 Cyc-ATC、Cyc-Tabu、Vtr-Tabu 算法求解目标函数值比较 ( $\lambda_1 = 0.5$ )

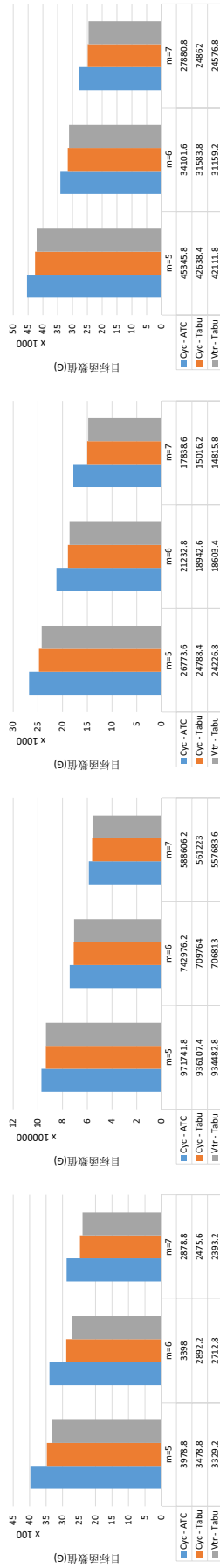
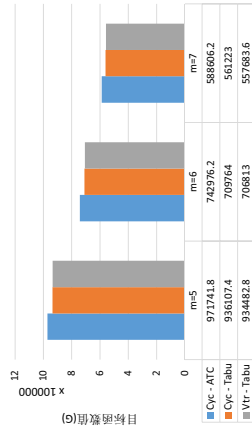
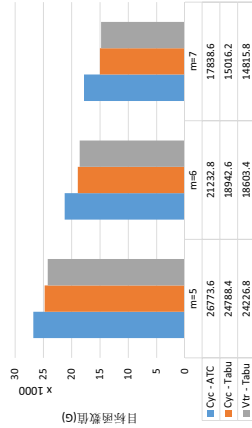
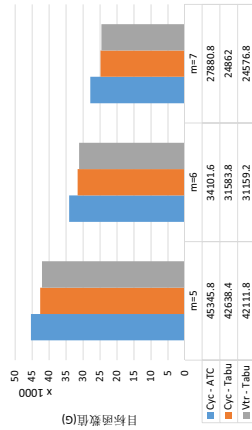
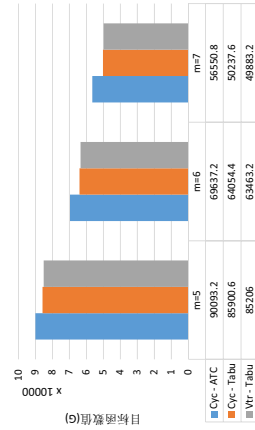
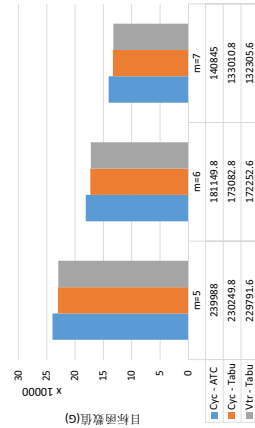
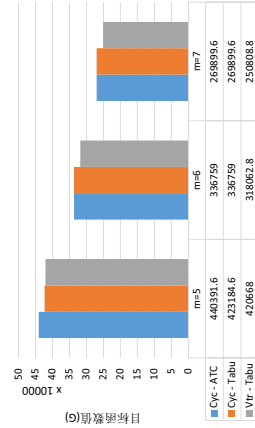
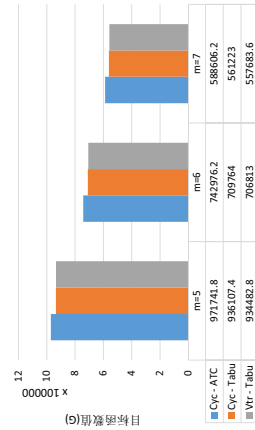
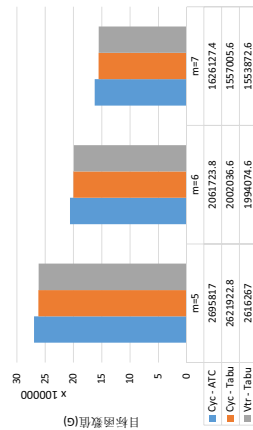
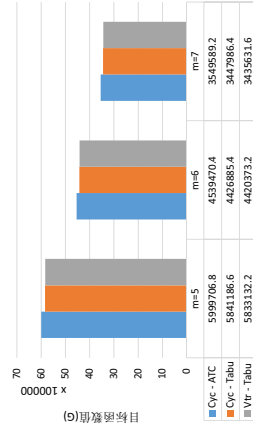
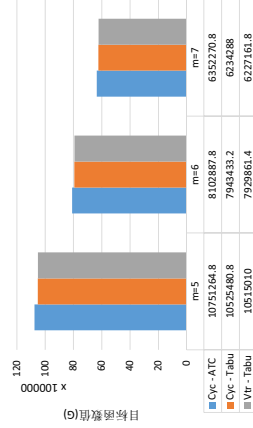
(a)  $n = 20$ (b)  $n = 30$ (c)  $n = 50$ (d)  $n = 70$ (e)  $n = 100$ (f)  $n = 150$ (g)  $n = 200$ (h)  $n = 300$ (i)  $n = 500$ (j)  $n = 750$ (k)  $n = 1000$ 图 A-3 模型 1 的 Cyc-ATC、Cyc-Tabu、Vtr-Tabu 算法求解目标函数数值比较 ( $\lambda_1 = 0.6$ )



表 A-3 Cyc - ATC、Cyc - Tabu、Vtr - Tabu 算法求解模型 1 目标函数值 ( $\lambda_1 = 0.6$ )

流水线数量 ( $m$ )	订单数量 ( $n$ )	20	30	50	70	100	150
5	Cyc - ATC	3979	8370	26774	45346	90093	239988
	Cyc - Tabu	3398	6748.4	21232.8	34101.6	69637.2	181149.8
	Vtr - Tabu	2879	5513.6	17838.6	27880.8	56550.8	140845
6	Cyc - ATC	3479	7559.6	24788.4	42638.4	85900.6	230249.8
	Cyc - Tabu	2892	5989.4	18942.6	31583.8	64054.4	173082.8
	Vtr - Tabu	2476	4950.8	15016.2	24862	50237.6	133010.8
7	Cyc - ATC	3329	7344.2	24226.8	42111.8	85206	229791.6
	Cyc - Tabu	2713	5734.2	18603.4	31159.2	63463.2	172252.6
	Vtr - Tabu	2393	4828.8	14815.8	24576.8	49883.2	132305.6

流水线数量 ( $m$ )	订单数量 ( $n$ )	200	300	500	750	1000
5	Cyc - ATC	440392	971742	2695817	5999707	10751265
	Cyc - Tabu	336759	742976.2	2061723.8	4539470.4	8102887.8
	Vtr - Tabu	269899.6	588606.2	1626127.4	3549589.2	6352270.8
6	Cyc - ATC	423184.6	936107.4	2621922.8	5841186.6	10525481
	Cyc - Tabu	336759	709764	2002036.6	4426885.4	7943433.2
	Vtr - Tabu	269899.6	561223	1557005.6	3447986.4	6234288
7	Cyc - ATC	420668	934482.8	2616267	5833132.2	10515010
	Cyc - Tabu	318062.8	706813	1994074.6	4420373.2	7929861.4
	Vtr - Tabu	250808.8	557683.6	1553872.6	3435631.6	6227161.8

表 A-4 Cyc - ATCS、Cyc - Tabu、VVT 算法求解模型 2 目标函数值 ( $\lambda_1 = 0.4$ )

流水线数量 ( $m$ )	订单数量 ( $n$ )	20	30	50	70	100	150
5	Cyc - ATCS	6487	14639	36555.939	58341	123121	293258
	Cyc - Tabu	3850	7851.1783	26889.191	49564.683	87671.223	219358.13
	VVT	3913	8657.367	28276.459	42605.208	93005.172	210875.07
6	Cyc - ATCS	5855	13784.833	33917.555	49510.443	114232.19	253435.71
	Cyc - Tabu	3510	8203.5114	25805.795	52971.678	85476.17	202066.4
	VVT	3588	6477.6687	21340.601	42056.736	82513.202	159408.82
7	Cyc - ATCS	5788	12887.76	32360.801	46228.992	106777.75	223006.92
	Cyc - Tabu	4291	11239.113	26888.348	50925.934	77160.63	203876.09
	VVT	3833	7233.7701	17850.436	35495.155	56956.974	136442.08

流水线数量 ( $m$ )	订单数量 ( $n$ )	200	300	500	750	1000
5	Cyc - ATCS	485767	1253111	3374782	6813918	13032226
	Cyc - Tabu	428398.75	1008863.9	3285292	6429213.5	12999611
	VVT	425257.79	1117715	3247024.2	6571784.7	12775684
6	Cyc - ATCS	450278.41	1158313.9	2947068.7	6029824.6	13032226
	Cyc - Tabu	425128.66	1010966.5	3156466.8	6548880.6	13017346
	VVT	366877.04	899843.51	2810177.5	5811671.2	11730364
7	Cyc - ATCS	418525.84	1041394.1	2719254.5	5478189.9	13032226
	Cyc - Tabu	427484.09	1135275.1	3239972.4	6682657.6	12939689
	VVT	311193.2	943551.23	2583663.4	5297965.8	10564891

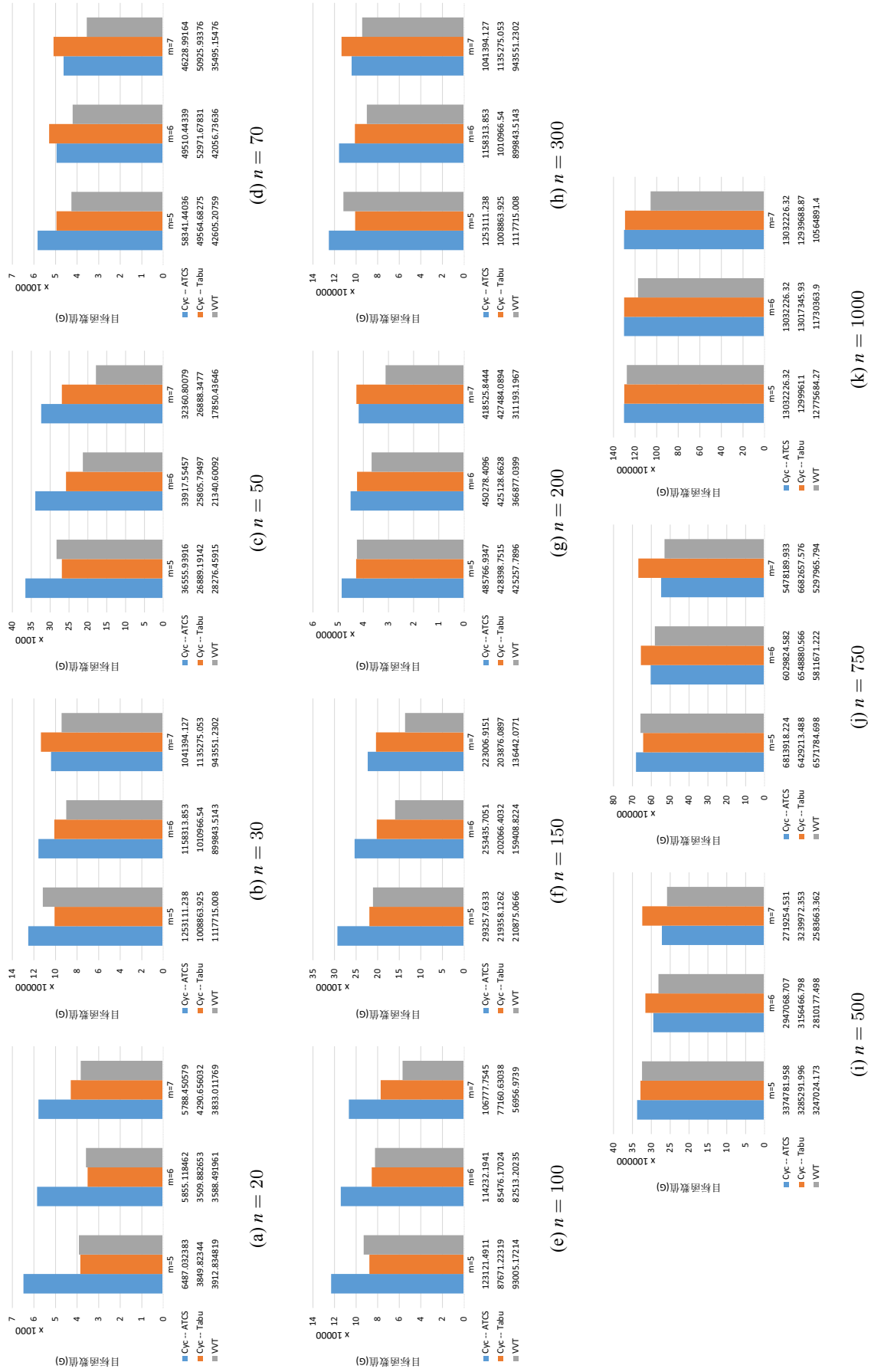
图 A-4 模型 2 的 Cyc - ATCS、Cyc - Tabu、VVT 算法求解目标函数值比较 ( $\lambda_1 = 0.4$ )

表 A-5 Cyc – ATCS、Cyc – Tabu、VVT 算法求解模型 2 目标函数值 ( $\lambda_1 = 0.5$ )

流水线数量 ( $m$ )	订单数量 ( $n$ )	20	30	50	70	100	150
5	Cyc - ATCS	6740	15186	38060	61202	133384	314488
	Cyc - Tabu	3957	7475.494	30024.952	43281.954	89535.599	230244.43
	VVT	4841	10682.449	30334.404	45382.031	91565.783	233094.79
6	Cyc - ATCS	6056	14293.191	35423.612	51775.478	122588.7	271166.69
	Cyc - Tabu	3560	7173.9686	24593.106	46707.875	85702.282	203873.55
	VVT	3185	5580.5588	22762.927	42459.636	90951.2	161589.65
7	Cyc - ATCS	5720	13337.446	34178.32	49005.801	113118.15	237750.37
	Cyc - Tabu	5033	10620.604	25618.636	48125.465	93584.444	205749.06
	VVT	3805	5781.3025	18140.016	36682.585	53703.663	134552.74

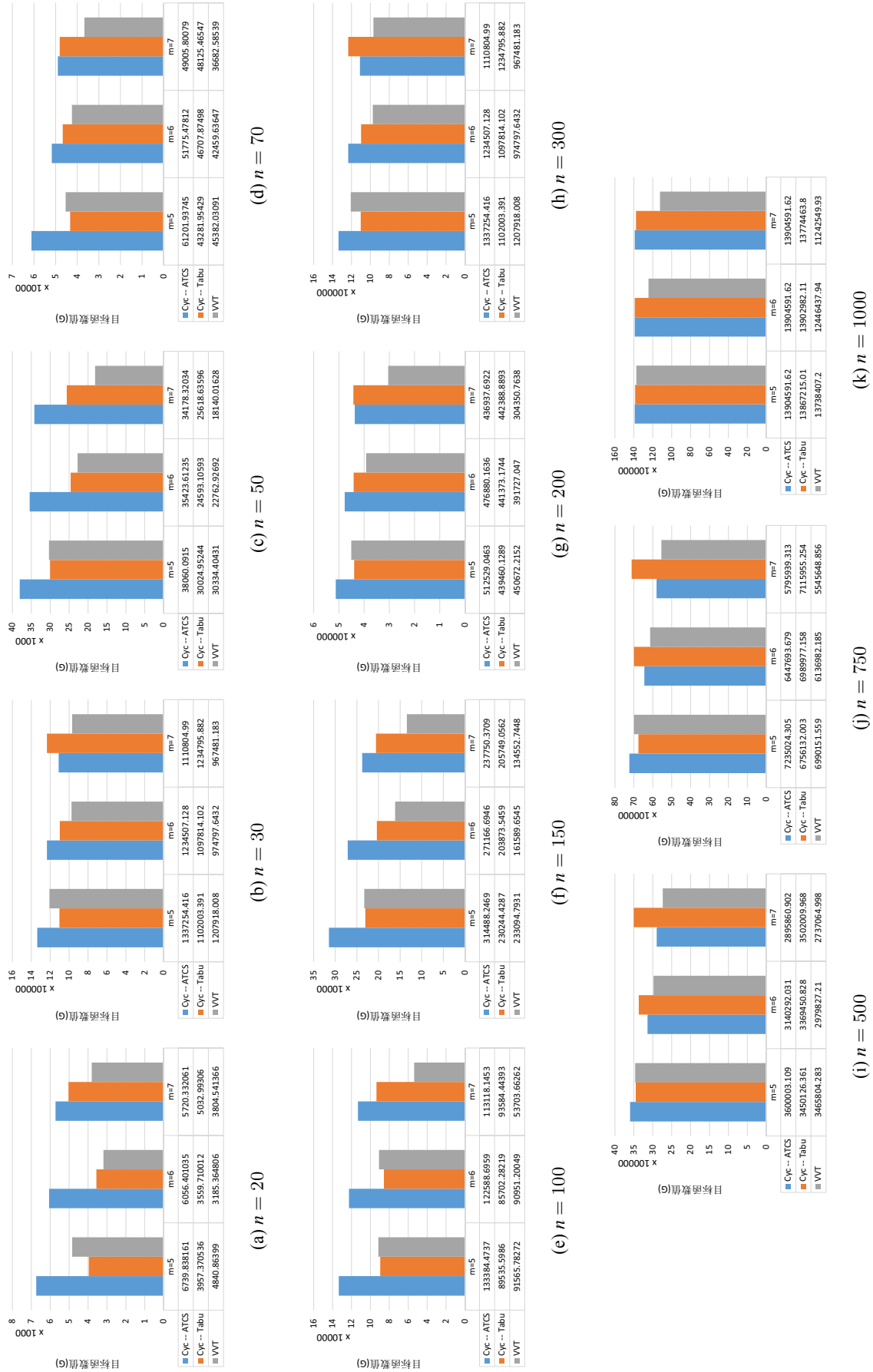
流水线数量 ( $m$ )	订单数量 ( $n$ )	200	300	500	750	1000
5	Cyc - ATCS	512529	1337254	3600003	7235024	13904592
	Cyc - Tabu	439460.13	1102003.4	3450126.4	6756132	13867215
	VVT	450672.22	1207918	3465804.3	6990151.6	13738407
6	Cyc - ATCS	476880.16	1234507.1	3140292	6447693.7	13904592
	Cyc - Tabu	441373.17	1097814.1	3369450.8	6989977.2	13902982
	VVT	391727.05	974797.64	2979827.2	6136982.2	12446438
7	Cyc - ATCS	436937.69	1110805	2895860.9	5795939.3	13904592
	Cyc - Tabu	442388.89	1234795.9	3502010	7115955.3	13774464
	VVT	304350.76	967481.18	2737065	5545648.9	11242550

表 A-6 Cyc – ATCS、Cyc – Tabu、VVT 算法求解模型 2 目标函数值 ( $\lambda_1 = 0.6$ )

流水线数量 ( $m$ )	订单数量 ( $n$ )	20	30	50	70	100	150
5	Cyc - ATCS	6536	15460	39821	64062	138328	334844
	Cyc - Tabu	4006	8412.1974	28282.738	47732.025	94457.323	226605.08
	VVT	4323	7979.0224	28277.507	46135.165	94500.141	241701.07
6	Cyc - ATCS	6276	14801.549	36929.67	54043.214	129235.53	288897.68
	Cyc - Tabu	3424	8248.3767	27369.773	47279.659	86141.272	213476.86
	VVT	3604	5725.7217	21615.213	40028.772	93898.487	164785.29
7	Cyc - ATCS	6002	13890.81	36826.504	50582.146	119156.7	252460.32
	Cyc - Tabu	4236	8735.0075	25470.642	48098.666	84992.165	210151.94
	VVT	3730	6336.405	19830.276	37488.671	57210.411	142846.99

流水线数量 ( $m$ )	订单数量 ( $n$ )	200	300	500	750	1000
5	Cyc - ATCS	539637	1421502	3842180	7681112	14821505
	Cyc - Tabu	461679.68	1190764.1	3657340.3	7177373.7	14853646
	VVT	483441.39	1240203.6	3687990.8	7344367.5	14621840
6	Cyc - ATCS	503521.09	1310700.4	3361729.1	6829840.8	14821505
	Cyc - Tabu	453662.49	1113032.7	3573224.3	7318057.6	14863523
	VVT	391036.57	993546.02	3169608.7	6478538.2	13241260
7	Cyc - ATCS	458750.04	1181963.8	3080549.8	6122988.8	14821505
	Cyc - Tabu	454908.33	1252287.2	3646277.8	7506555.2	14656063
	VVT	322579.1	1065107.2	2882401.5	5924617.8	11897569

图 A-5 模型 2 的 Cyc-ATCS、Cyc-Tabu、VVT 算法求解目标函数值比较 ( $\lambda_1 = 0.5$ )

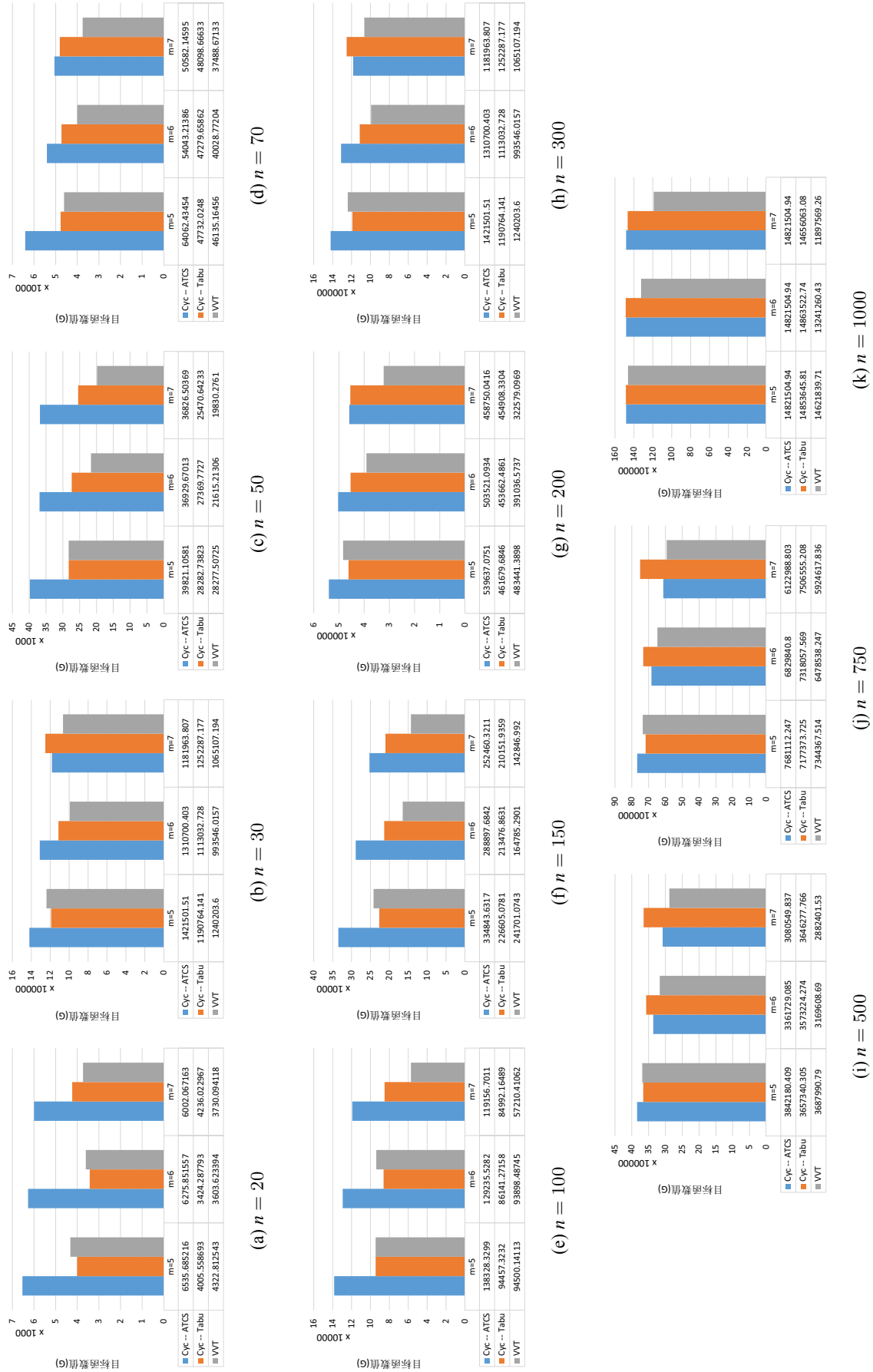
图 A-6 模型 2 的 Cyc-ATCS、Cyc-Tabu、VVT 算法求解目标函数值比较 ( $\lambda_1 = 0.6$ )

表 A-7 Cyc – ATCS、Cyc – Tabu、VVT 算法求解模型 2 所得调度流水线均衡率 ( $\lambda_1 = 0.4$ )

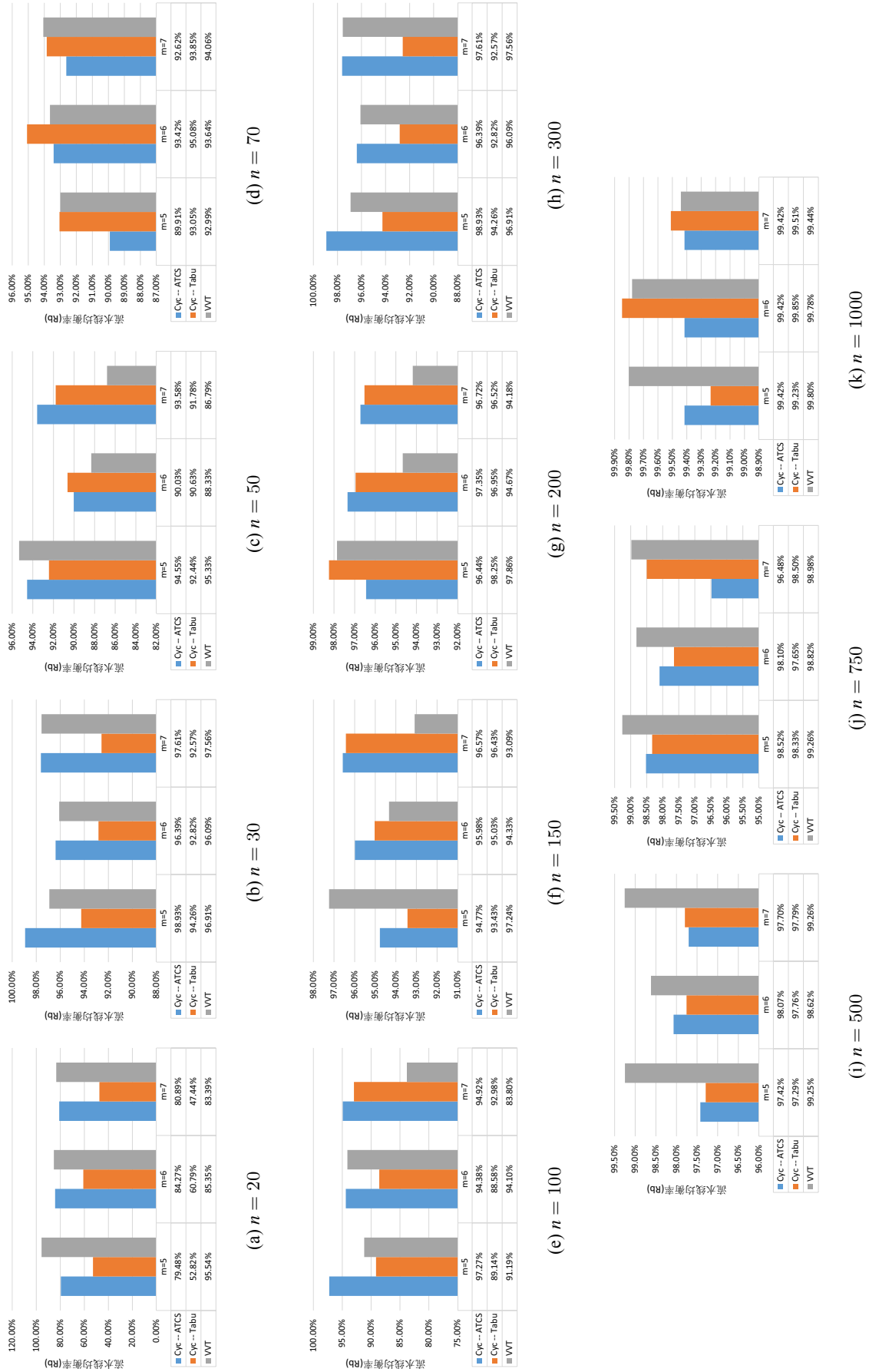
$m$	5			6			7		
$n$	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3) <sup>a</sup>
20	79.48%	52.82%	95.54%	84.27%	60.79%	85.35%	80.89%	47.44%	83.39%
30	87.39%	81.09%	90.53%	86.76%	76.63%	85.48%	85.28%	76.92%	79.37%
50	94.55%	92.44%	95.33%	90.03%	90.63%	88.33%	93.58%	91.78%	86.79%
70	89.91%	93.05%	92.99%	93.42%	95.08%	93.64%	92.62%	93.85%	94.06%
100	97.27%	89.14%	91.19%	94.38%	88.58%	94.10%	94.92%	92.98%	83.80%
150	94.77%	93.43%	97.24%	95.98%	95.03%	94.33%	96.57%	96.43%	93.09%
200	96.44%	98.25%	97.86%	97.35%	96.95%	94.67%	96.72%	96.52%	94.18%
300	98.93%	94.26%	96.91%	96.39%	92.82%	96.09%	97.61%	92.57%	97.56%
500	97.42%	97.29%	99.25%	98.07%	97.76%	98.62%	97.70%	97.79%	99.26%
750	98.52%	98.33%	99.26%	98.10%	97.65%	98.82%	96.48%	98.50%	98.98%
1000	99.42%	99.23%	99.80%	99.42%	99.85%	99.78%	99.42%	99.51%	99.44%

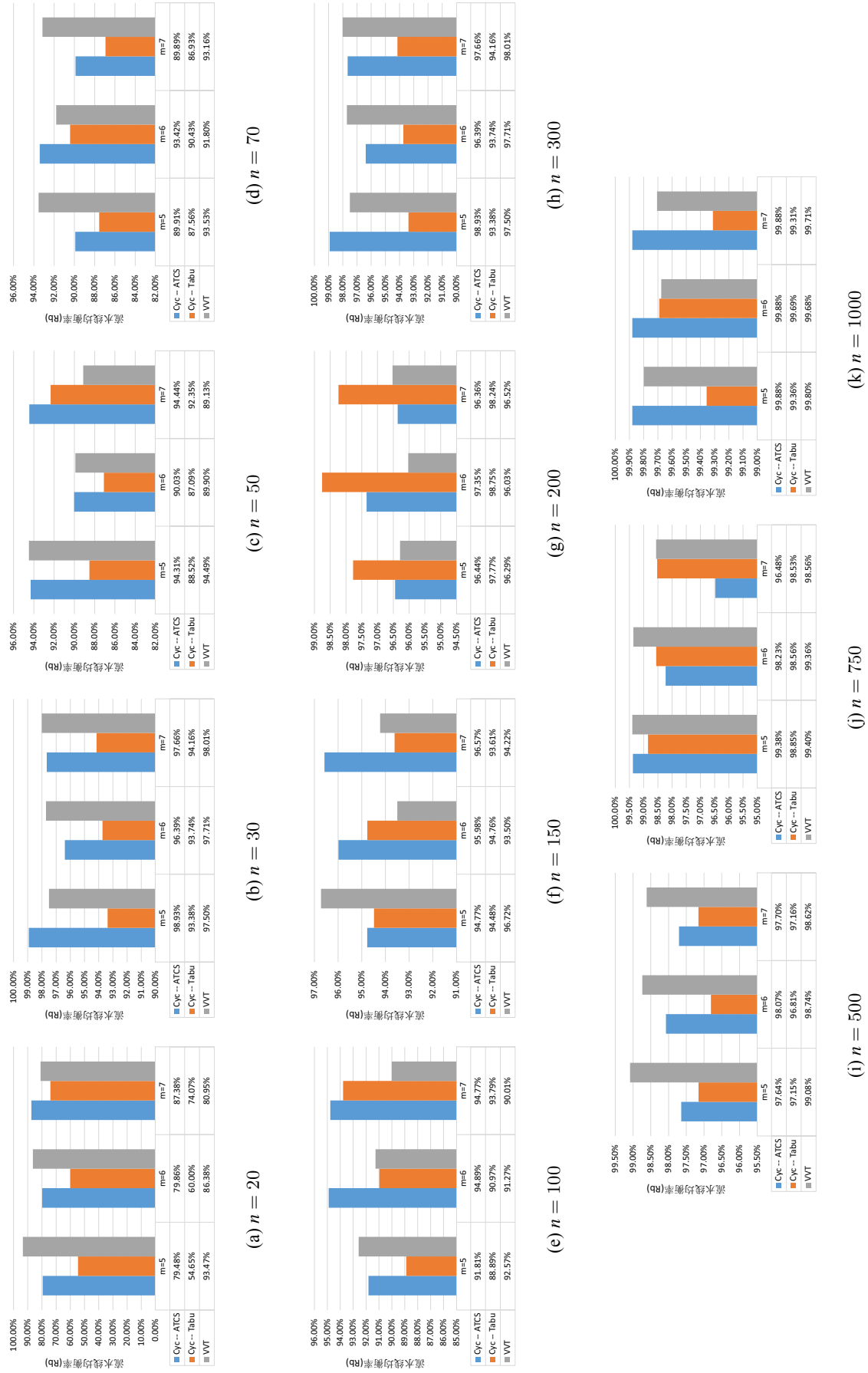
<sup>a</sup>(1): Cyc – ATCS, (2): Cyc – Tabu, (3): VVT, 表 A-8 与表 A-9 与此相同表 A-8 Cyc – ATCS、Cyc – Tabu、VVT 算法求解模型 2 所得调度流水线均衡率 ( $\lambda_1 = 0.5$ )

$m$	5			6			7		
$n$	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)
20	79.48%	54.65%	93.47%	79.86%	60.00%	86.38%	87.38%	74.07%	80.95%
30	87.39%	79.09%	93.58%	86.76%	80.45%	80.39%	85.28%	71.85%	83.89%
50	94.31%	88.52%	94.49%	90.03%	87.09%	89.90%	94.44%	92.35%	89.13%
70	89.91%	87.56%	93.53%	93.42%	90.43%	91.80%	89.89%	86.93%	93.16%
100	91.81%	88.89%	92.57%	94.89%	90.97%	91.27%	94.77%	93.79%	90.01%
150	94.77%	94.48%	96.72%	95.98%	94.76%	93.50%	96.57%	93.61%	94.22%
200	96.44%	97.77%	96.29%	97.35%	98.75%	96.03%	96.36%	98.24%	96.52%
300	98.93%	93.38%	97.50%	96.39%	93.74%	97.71%	97.66%	94.16%	98.01%
500	97.64%	97.15%	99.08%	98.07%	96.81%	98.74%	97.70%	97.16%	98.62%
750	99.38%	98.85%	99.40%	98.23%	98.56%	99.36%	96.48%	98.53%	98.56%
1000	99.88%	99.36%	99.80%	99.88%	99.69%	99.68%	99.88%	99.31%	99.71%

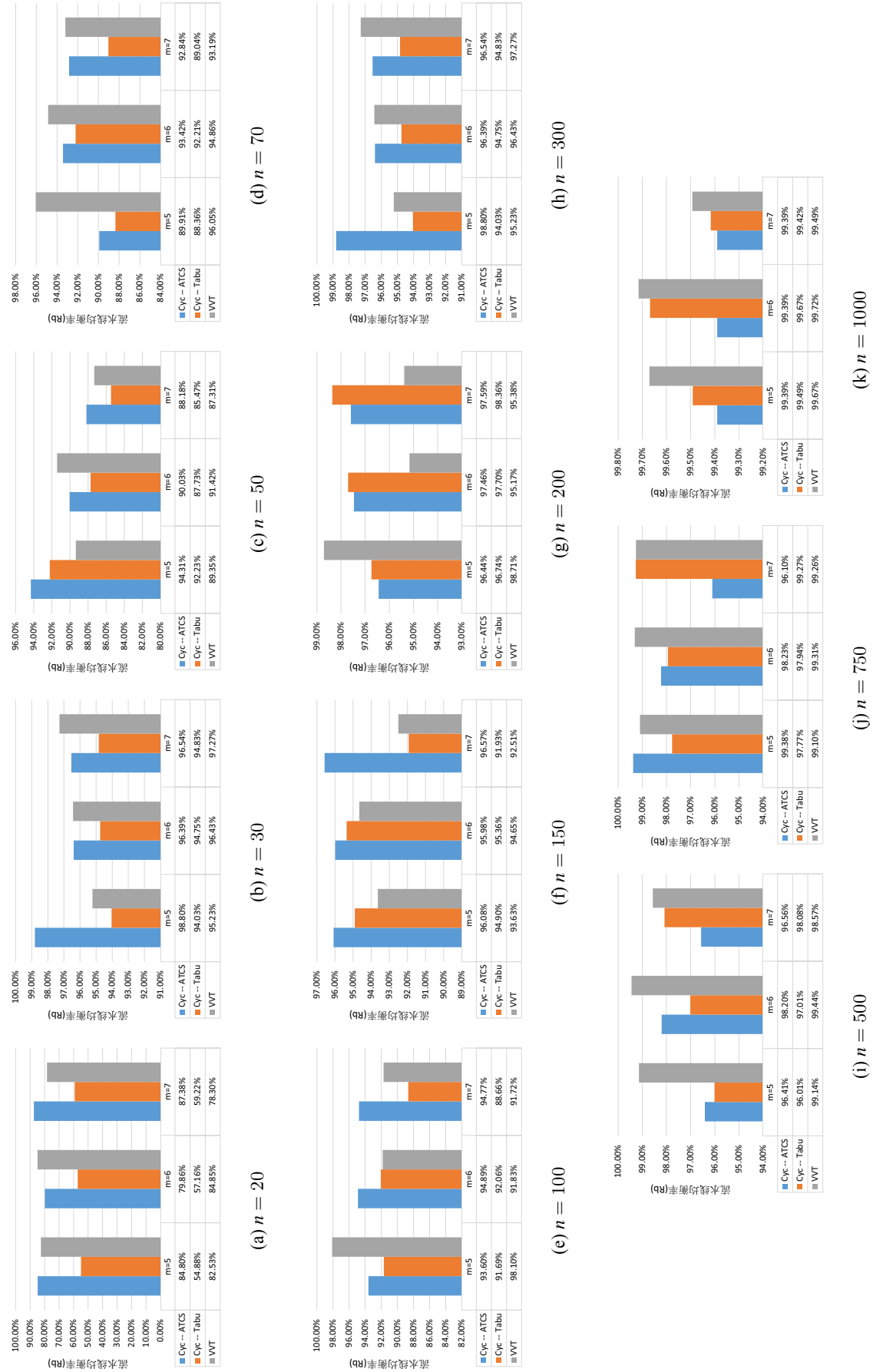
表 A-9 Cyc – ATCS、Cyc – Tabu、VVT 算法求解模型 2 所得调度流水线均衡率 ( $\lambda_1 = 0.6$ )

$m$	5			6			7		
$n$	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)
20	84.80%	54.88%	82.53%	79.86%	57.16%	84.85%	87.38%	59.22%	78.30%
30	92.27%	91.21%	82.63%	86.76%	88.40%	75.45%	87.17%	70.37%	77.28%
50	94.31%	92.23%	89.35%	90.03%	87.73%	91.42%	88.18%	85.47%	87.31%
70	89.91%	88.36%	96.05%	93.42%	92.21%	94.86%	92.84%	89.04%	93.19%
100	93.60%	91.69%	98.10%	94.89%	92.06%	91.83%	94.77%	88.66%	91.72%
150	96.08%	94.90%	93.63%	95.98%	95.36%	94.65%	96.57%	91.93%	92.51%
200	96.44%	96.74%	98.71%	97.46%	97.70%	95.17%	97.59%	98.36%	95.38%
300	98.80%	94.03%	95.23%	96.39%	94.75%	96.43%	96.54%	94.83%	97.27%
500	96.41%	96.01%	99.14%	98.20%	97.01%	99.44%	96.56%	98.08%	98.57%
750	99.38%	97.77%	99.10%	98.23%	97.94%	99.31%	96.10%	99.27%	99.26%
1000	99.39%	99.49%	99.67%	99.39%	99.67%	99.72%	99.39%	99.42%	99.49%

图 A-7 模型 2 的 Cyc-ATCS、Cyc-Tabu、VVT 算法求解流水线均衡率比较 ( $\lambda_1 = 0.4$ )

图 A-8 模型 2 的 Cyc-ATCS、Cyc-Tabu、VVT 算法求解流水线均衡率比较 ( $\lambda_1 = 0.5$ )



图 A-9 模型 2 的 Cyc-ATCS、Cyc-Tabu、VVT 算法求解流水线均衡率比较 ( $\lambda_1 = 0.6$ )

# 附录 B Python 脚本

## (1) generate.py

```

1  from __future__ import division
2  import math
3  import random
4  import basicvirtual
5
6  def Idx(time, p , due_dates, wt): # define the ordering index for
    ATC rule
7      n = len(p)
8      Idx_value = []
9      average = sum(p)/n
10     for j in xrange(n):
11         Idx_value.append(wt[j]/p[j]*math.exp(-max(due_dates[j]-p[j]-
            time,0)/2/average))
12     return Idx_value
13
14 def Idx_c(time, p, s , due_dates, wt,k_1,k_2): # define the ordering
    index for ATCS rule
15     n = len(p)
16     Idx_value = []
17     average_p = sum(p)/n
18     average_s = sum(s)/n
19     for j in xrange(n):
20         Idx_value.append(wt[j]/p[j]*math.exp(-max(due_dates[j]-p[j]-
            time,0)/k_1/average_p - s[j]/k_2/average_s))
21     return Idx_value
22
23 def estimate(m,items): # estimate parameters for k_1 & k_2
24     n = len(items)
25     p = [items[j].process for j in xrange(n)]
26     s = [items[j].setup for j in xrange(n)]
27     d = [items[j].due for j in xrange(n)]
28     c_max = (sum(p) + sum(s))/m
29     tau = 1 - sum(d)/(n*c_max)
30     eta = sum(s)/sum(p)
31     R = (max(d) - min(d))/c_max
32     if R <= 0.5:
33         k_1 = 4.5 + R
34     else:
35         k_1 = 6-2*R
36     k_2 = tau/(2*math.sqrt(eta))
37     return k_1,k_2
38
39 def poisson(alpha): # generate the poisson distribution var
40     n = 0
41     P = random.random()
42     while P >= math.exp(-alpha):
43         r = random.random()
44         P = P*r
45         n+=1
46     N = n
47     return N
48
49 def release(n,alpha): # generate the item release time
50     r = [poisson(alpha)]
51     for j in xrange(1,n):
52         a = poisson(alpha)
53         temp = a + r[-1]
54         r.append(temp)
55     random.shuffle(r)
56     return r
57
58 def process(n): # generate the job process time
59     q = [None]*n
60     for i in xrange(n):
61         q[i] = poisson(5)
62     return q
63
64 def setup(n): # generate the itme setup time
65     s = [None]*n
66     for j in xrange(n):
67         s[j] = random.randrange(1,10)
68     return s
69
70 def jobstep(n,a,b): # generate the step of a job needs
71     step = [random.randrange(a,b)]
72     for j in xrange(1,n):
73         step.append(random.randrange(a,b))
74     return step
75
76 def TL_update(TL,from_same,from_diff,S): # update the Tabu list in
    case of over-tabu
77     Same = from_same[:]
78     Diff = from_diff[:]
79     if from_same:
80         for s in from_same:
81             job = list(s)
82             a,b = job[0],job[1]
83             if find_job(a,S) != find_job(b,S):
84                 Same.remove(s)
85                 TL[TL.index(s)] = None
86     if from_diff:
87         for s in from_diff:
88             job = list(s)
89             a,b = job[0],job[1]
90             if find_job(a,S) == find_job(b,S):
91                 Diff.remove(s)
92                 TL[TL.index(s)] = None
93     return TL,Same,Diff
94
95 def itemjobs(n,a,b): # generate the number of jobs that a item
    needs
96     n_job = [random.randrange(a,b)]
97     for j in xrange(1,n):
98         n_job.append(random.randrange(a,b))
99     return n_job
100
101 def processtime(n,q): # calculate the item process time
102     m = len(q)
103     c = [None]*n
104     d = []
105     for k in xrange(n):
106         c[k] = (k+1)*q[0]
107     for i in xrange(1,m-1):
108         c[0] = c[0] + q[i]
109     for k in xrange(1,n):
110         c[k] = max(c[k],c[k-1])+q[i]
111     return int(c[-1]/500)
112
113 def due_date_r(r,p): # generate the job due dates with release
    time
114     n = len(r)
115     d = [None]*n
116     for j in xrange(n):
117         delta = int(p[j]/3)
118         d[j] = r[j] + 2*p[j] + random.randrange(-delta,delta)
119     return d
120
121 def weights(n,init): # generate the weights for Tardiness and
    Completion, just input the initial value
122     w = [init]
123     for j in xrange(1,n):
124         w.append(random.randrange(1,2*init))
125     random.shuffle(w)
126     return w

```

```

127
128 def late(complete_time,items):      # define the Lateness
129     n = len(complete_time)
130     lateness = []
131     for j in xrange(n):
132         item = items[j]
133         lateness.append(complete_time[j] - item.due)
134     return lateness
135
136 def tard(lateness): # define the Tardiness
137     if type(lateness) == int:
138         lateness = [lateness]
139     n = len(lateness)
140     tardiness = []
141     for j in xrange(n):
142         temp = max(lateness[j],0)
143         tardiness.append(temp)
144     return tardiness
145
146 def early(lateness): # define the Earliness
147     if type(lateness) == int:
148         lateness = [lateness]
149     n = len(lateness)
150     earliness = []
151     for j in xrange(n):
152         temp = max(-lateness[j],0)
153         earliness.append(temp)
154     return earliness
155
156 def initialization(items,n,m): # generate initial solution for model
157     1
158     J = range(n)
159     S = []
160     a = []
161     t1 = []
162     L = []
163     c = [None]*n
164     for l in xrange(m):
165         S.append([])
166         a.append(0)
167         t1.append(0)
168     t = 0
169     while J:
170         if 0 in a:
171             l_star = a.index(0)
172             p,d,wt = [],[],[]
173             for j in J:
174                 item = items[j]
175                 p.append(item.process)
176                 d.append(item.due)
177                 wt.append(item.wt)
178             orderidx = Idx_c(t,p,d,wt)
179             j_star = J[orderidx.index(max(orderidx))]
180             S[l_star].append(j_star)
181             J.remove(j_star)
182             L.append(j_star)
183             t1[l_star] = t + items[j_star].process
184             c[j_star] = t1[l_star]
185             a[l_star] = 1
186         else:
187             t_star = min(t1)
188             for l in xrange(m):
189                 if t1[l] == t_star:
190                     a[l] = 0
191             t = t_star
192     return S,L,c
193
194 def initialization_c(items,n,m): # generate initial solution for
195     model 2
196     J = range(n)
197     S = []
198     a = []
199     t1 = []
200     L = []
201     c = [None]*n
202     f = [None]*n
203     k_1,k_2=estimate(m,items)

```

```

202     for l in xrange(m):
203         S.append([])
204         a.append(0)
205         t1.append(0)
206     t = 0
207     while J:
208         if 0 in a:
209             l_star = a.index(0)
210             p,r,s,d,wt = [],[],[],[],[]
211             for j in J:
212                 item = items[j]
213                 p.append(item.process)
214                 r.append(item.release)
215                 s.append(item.setup)
216                 d.append(item.due)
217                 wt.append(item.wt)
218             orderidx = Idx_c(t,p,s,d,wt,k_1,k_2)
219             j_star = J[orderidx.index(max(orderidx))]
220             S[l_star].append(j_star)
221             J.remove(j_star)
222             L.append(j_star)
223             if len(S[l_star]) == 1:
224                 f[j_star] = max(items[j_star].release - items[j_star]
225                                 ].setup,0)
226             else:
227                 f[j_star] = max(items[j_star].release - items[j_star]
228                                 ].setup - c[S[l_star][S[l_star].index(j_star)
229                                 -1],0)
230             t1[l_star] = t + items[j_star].process + items[j_star].
231                 setup + f[j_star]
232             c[j_star] = t1[l_star]
233             a[l_star] = 1
234         else:
235             t_star = min(t1)
236             for l in xrange(m):
237                 if t1[l] == t_star:
238                     a[l] = 0
239             t = t_star
240     return S,L,c,f
241
242 def H(item_values,S): # define the line contribution value
243     line_values = [item_values[j] for j in S]
244     value = sum(line_values)
245     return value
246
247 def Goal(completion,items,S,lambdal,lambda2): # calculate the
248     object function value
249     lateness = late(completion,items)
250     Rb,c_max = balance_rate(completion,S)
251     Ru = idle_rate(items,completion,c_max,S)
252     line_values = []
253     l = 0
254     for s in S:
255         left = [math.fabs(items[j].wt*lateness[j]) for j in s]
256         right = [items[j].wc*completion[j] for j in s]
257         line_values.append(lambdal*sum(left)/Ru[l] + lambda2*math.exp
258                             (-Rb)*sum(right))
259         l +=1
260     return line_values,sum(line_values)
261
262 def reorder(items,S,line_values,item_values): # find the lines to
263     reorder its items
264     l_p = line_values.index(max(line_values))
265     l_m = line_values.index(min(line_values))
266     s_p = S[l_p]
267     s_m = S[l_m]
268     temp_value = [item_values[j] for j in s_p]
269     j_star = s_p[temp_value.index(max(temp_value))]
270     s_p.remove(j_star)
271     s_m.append(j_star)
272     return l_p, l_m
273
274 def innerswap(S,id1,id2): # swap items for inner line
275     K = S[id1]
276     temp = K[id1]
277     K[id1] = K[id2]
278     K[id2] = temp

```

```

272     return K
273
274 def pairsets(S):          # define the neighbor pair sets
275     pair = []
276     n = len(S)
277     for i in xrange(n-1):
278         pair.append(set([S[i],S[i+1]]))
279     return pair
280
281 def changewise(set_1,set_2):    # set_1 is the changed set while
282     set_2 is not
283     newset = (set_1|set_2) - (set_1&set_2)
284     return newset
285
286 def pairsets_update(pairs,change_set):    # renew the pair sets for
287     prograssing
288     idx = pairs.index(change_set)
289     n = len(pairs)
290     if n != 1:
291         if idx == 0:
292             pairs[1] = changewise(pairs[0],pairs[1])
293         elif idx == n-1:
294             pairs[-2] = changewise(pairs[-1],pairs[-2])
295         else:
296             pairs[idx-1] = changewise(pairs[idx],pairs[idx-1])
297             pairs[idx+1] = changewise(pairs[idx],pairs[idx+1])
298
299 def find_job(job,S):    # to find the job scheduled line
300     n = len(S)
301     for l in xrange(n):
302         if job in S[l]:
303             l_star = l
304     return l_star
305
306 def verify(S,items,lambdal,lambda2):    # this function is to verify
307     if the algorithm is right
308     n = len(items)
309     completion = [None]*n
310     for s in S:
311         t = 0
312         for j in s:
313             t += items[j].process
314             completion[j] = t
315     lateness = late(completion,items)
316     tardiness = tard(lateness)
317     item_values = []
318     for j in xrange(len(items)):
319         item = items[j]
320         wt,wc = item.wt,item.wc
321         t,c = tardiness[j],completion[j]
322         value = basicvirtual.h(t,c,wt,wc,lambdal,lambda2)
323         item_values.append(value)
324     return completion,tardiness,item_values
325
326 def balance_rate(completion,S):    # calculate the balance rate for
327     model 2
328     c_max = []
329     for s in S:
330         c_max.append(completion[s[-1]])
331     m = len(c_max)
332     Rb = sum(c_max)/(m*max(c_max))
333     return Rb,c_max
334
335 def idle(items,completion,S):    # generate item free time
336     n = len(items)
337     item_free = [None]*n
338     for s in S:
339         k = 0
340         for j in s:
341             if k == 0:
342                 item_free[j] = max(items[j].release - items[j].setup
343                                     ,0)
344             else:
345                 i = s[k-1]
346                 item_free[j] = max(items[j].release - items[j].setup
347                                     - completion[i] ,0)
348             k+=1

```

```

343     return item_free
344
345 def idle_rate(items,completion,c_max,S):    # calculate the idle rate
346     item_free = idle(items,completion,S)
347     Ru = []
348     l = 0
349     for s in S:
350         v = [item_free[j] for j in s]
351         Ru.append(l - sum(v)/c_max[l])
352         l += 1
353     return Ru

```

## (2) experiment\_data.py

```

1  import sys,pprint
2  sys.path.append("../functions")
3  import generate
4
5  def generate_data(input_data):    # generate experiment data and store
6      in the data file
7      data = input_data.split('\n')
8      N = len(data)
9      for k in xrange(N):
10         n = int(data[k])
11         s = generate.setup(n)
12         r = generate.release(n,3)
13         f = open("../data\\"+str(data[k]),'w')
14         step = generate.jobstep(n,8,20)
15         q = generate.process(step[0])
16         n_job = generate.itemjobs(n,1000,2000)
17         p = [generate.processtime(n_job[0],q)]
18         wt = generate.weights(n,8)
19         wc = generate.weights(n,6)
20         for j in xrange(1,n):
21             q = generate.process(step[j])
22             temp = generate.processtime(n_job[j],q)
23             p.append(temp)
24             d = generate.due_date_r(r,p)
25             for i in xrange(n):
26                 f.write(str(p[i])+'u' + str(r[i]) + 'u' + str(s[i]) + 'u'
27                         + str(d[i]) + 'u' + str(wt[i]) + 'u' + str(wc[i])
28                         + '\n')
29             f.close()
30             print 'data_' + str(data[k]) + 'u' + 'done!'
31
32 import sys
33 if __name__ == '__main__':
34     if len(sys.argv) > 1:
35         file_location = sys.argv[1].strip()
36         input_data_file = open(file_location, 'r')
37         input_data = ''.join(input_data_file.readlines())
38         input_data_file.close()
39         generate_data(input_data)

```

## (3) basicatc.py

```

1  from __future__ import division
2  import sys
3  sys.path.append("../functions")
4  import generate
5  from collections import namedtuple
6  Item = namedtuple("Item", ['process','due','wt','wc'])
7
8  def h(tardiness,completion,wt,wc,lambdal,lambda2):    # define the
9      contribution of one item for the obj function
10     value = lambdal*wt*tardiness + lambda2*wc*completion
11     return value
12
13 def ATC(items,S):    # use ATC rule
14     J = S[:]
15     S = []
16     t = 0
17     c = []
18     while J:
19         p = [items[j].process for j in J]

```

```

19     d = [items[j].due for j in J]
20     wt = [items[j].wt for j in J]
21     orderidx = generate.Idx(t,p,d,wt)
22     j_star = J[orderidx.index(max(orderidx))]
23     S.append(j_star)
24     J.remove(j_star)
25     t = t + items[j_star].process
26     c.append(t)
27     return S,c
28
29 def solve(input_data,m,lambdal,NR):
30     lambda2 = 1- lambdal
31     Data = input_data.split('\n') # load data
32     n = len(Data) - 1 # get the amount of items
33     items = []
34     for j in xrange(n):
35         data = Data[j]
36         parts = data.split()
37         p = int(parts[0]) # get the process time
38         s = int(parts[2]) # get the setup time
39         d = int(parts[3]) # get the due date
40         wt = int(parts[4]) # get the tardiness weights
41         wc = int(parts[5]) # get the completion weights
42         items.append(Item(p+s,d,wt,wc)) # combine those item data
43     print 'DataLoaded!'
44     S,L,completion = generate.initialization(items,n,m)
45     print 'InitializationDone!'
46
47     lateness = generate.late(completion,items) # begin the value
48     initialization
49     tardiness = generate.tard(lateness)
50     item_values = []
51     for j in xrange(n):
52         item = items[j]
53         wt,wc = item.wt,item.wc
54         t,c = tardiness[j],completion[j]
55         value = h(t,c,wt,wc,lambdal,lambda2)
56         item_values.append(value)
57     G = generate.H(item_values,L)
58     line_values = []
59     for s in S:
60         value = generate.H(item_values,s)
61         line_values.append(value)
62     print 'Initial_valuesDone!'
63     print G
64
65     for k in xrange(NR):
66         l_p,l_m = generate.reorder(items,S,line_values,item_values)
67         S[l_p],c_p = ATC(items,S[l_p])
68         S[l_m],c_m = ATC(items,S[l_m])
69         for j in S[l_p]:
70             completion[j] = c_p.pop(0)
71             c = completion[j]
72             item = items[j]
73             wt,wc = item.wt,item.wc
74             late = completion[j] - item.due
75             t = generate.tard(late)
76             item_values[j] = h(t[0],c,wt,wc,lambdal,lambda2)
77         for j in S[l_m]:
78             completion[j] = c_m.pop(0)
79             c = completion[j]
80             item = items[j]
81             wt,wc = item.wt,item.wc
82             late = completion[j] - item.due
83             t = generate.tard(late)
84             item_values[j] = h(t[0],c,wt,wc,lambdal,lambda2)
85         delta_p = generate.H(item_values,S[l_p]) - line_values[l_p]
86         delta_m = generate.H(item_values,S[l_m]) - line_values[l_m]
87         line_values[l_p] += delta_p
88         line_values[l_m] += delta_m
89         G = G + delta_m + delta_p
90     return G
91
92 if __name__ == '__main__':
93     if len(sys.argv) > 1:
94         file_location = sys.argv[1].strip()
95         input_data_file = open(file_location,'r')

```

```

95     input_data = ''.join(input_data_file.readlines())
96     input_data_file.close()
97     m = int(sys.argv[2])
98     NR = int(raw_input('iterate_time:'))
99     lambdal = float(raw_input('lambdal='))
100    f = open("result\\batac_" + str(int(file_location[7:])) + "_"
101           + str(m) + "_" + str(lambdal), 'w')
102    G = solve(input_data,m,lambdal,NR)
103    f.write(str(G) + '\n' + str(Rb) + '\n')
104    f.close()

```

#### (4) basictabu.py

```

1  import sys
2  sys.path.append("../functions")
3  import generate
4  from collections import namedtuple
5  from basicaac import h
6  Item = namedtuple("Item", ['process','due','wt','wc'])
7
8  def complete_time(S,items):
9      n = len(items)
10     c = [None]*n
11     for s in S:
12         t = 0
13         for j in s:
14             t += items[j].process
15         c[j] = t
16     return c
17
18 def value_generator(S,items,lambdal,lambda2):
19     completion = complete_time(S,items)
20     lateness = generate.late(completion,items)
21     tardiness = generate.tard(lateness)
22     item_values = []
23     for j in xrange(len(items)):
24         item = items[j]
25         wt,wc = item.wt,item.wc
26         t,c = tardiness[j],completion[j]
27         value = h(t,c,wt,wc,lambdal,lambda2)
28         item_values.append(value)
29     return completion,tardiness,item_values
30
31
32 def tabu(N,NL,S,items, completion,tardiness,lambdal,lambda2):
33     TL = [None]*NL
34     pairs = generate.pairsets(S)
35     completion_temp = completion[:]
36     tardiness_temp = tardiness[:]
37     delta_star = 0
38     Delta = 0
39     S_star = S[:]
40     for k in xrange(N):
41         delta = []
42         for s in pairs:
43             if s not in TL:
44                 job = list(s)
45                 a = S[max(S.index(job[0]),S.index(job[1]))]
46                 b = S[min(S.index(job[0]),S.index(job[1]))]
47                 delta_c_a = - items[b].process
48                 delta_c_b = items[a].process
49                 delta_t_a = - min(items[b].process,tardiness_temp[a])
50                 delta_t_b = max(completion_temp[a] - items[b].due,0)
51                             - tardiness_temp[b]
52                 wt_a,wt_b,wc_a,wc_b = items[a].wt,items[b].wt,items[a].wc,items[b].wc
53                 delta_a = h(delta_t_a,delta_c_a,wt_a,wc_a,lambdal,lambda2)
54                 delta_b = h(delta_t_b,delta_c_b,wt_b,wc_b,lambdal,lambda2)
55                 delta_h = delta_a + delta_b
56                 if delta == []:
57                     delta = delta_h
58                     delta_a_star = delta_a
59                     delta_b_star = delta_b
60                     a_star,b_star = a,b

```

```

60         elif delta_h < delta:
61             delta = delta_h
62             delta_a_star = delta_a
63             delta_b_star = delta_b
64             a_star, b_star = a, b
65
66         if delta == []:
67             break
68         a = S.index(a_star)
69         b = S.index(b_star)
70         S = generate.innerswap(S, a, b)
71         change_set = set([a_star, b_star])
72         generate.pairsets_update(pairs, change_set)
73         TL.pop(0)
74         TL.append(change_set)
75         completion_temp[a_star] -= items[b_star].process
76         completion_temp[b_star] += items[a_star].process
77         tardiness_temp[a_star] = max(completion_temp[a_star] - items[
78             a_star].due, 0)
79         tardiness_temp[b_star] = max(completion_temp[b_star] - items[
80             b_star].due, 0)
81         Delta += delta
82         if Delta < delta_star:
83             delta_star = Delta
84             S_star = S[:]
85         return delta_star, S_star
86
87 def solve(input_data, m, lambdal, N, NL, NR):
88     lambda2 = 1 - lambdal
89     Data = input_data.split('\n') # load data
90     n = len(Data) - 1 # get the amount of items
91     items = []
92     for j in xrange(n):
93         data = Data[j]
94         parts = data.split()
95         p = int(parts[0]) # get the process time
96         s = int(parts[2]) # get the setup time
97         d = int(parts[3]) # get the due date
98         wt = int(parts[4]) # get the tardiness weights
99         wc = int(parts[5]) # get the completion weights
100         items.append(Item(p+s, d, wt, wc)) # combine those item data
101
102     print 'Data loaded!'
103     S, L, completion = generate.initialization(items, n, m)
104     print 'Initialization done!'
105
106     completion, tardiness, item_values = value_generator(S, items,
107         lambdal, lambda2)
108
109     G = generate.H(item_values, L)
110     line_values = []
111     for s in S:
112         value = generate.H(item_values, s)
113         line_values.append(value)
114     print 'Initial values done!'
115
116     for l in xrange(m):
117         delta, S[l] = tabu(N, NL, S[l], items, completion, tardiness, lambdal,
118             lambda2)
119         G += delta
120         line_values[l] += delta
121         completion, tardiness, item_values = value_generator(S, items,
122             lambdal, lambda2)
123     print 'Initial Tabu Search done!'
124
125     value = G
126     S_temp = [None]*m
127     for l in xrange(m):
128         S_temp[l] = S[l][:]
129         line_values_temp = line_values[:]
130         item_values_temp = item_values[:]
131         for k in xrange(NR):
132             l_p, l_m = generate.reorder(items, S_temp, line_values_temp,
133                 item_values_temp)
134             completion_temp, tardiness_temp, item_values_temp =
135                 value_generator(S_temp, items, lambdal, lambda2)
136             line_values_temp[l_p] = generate.H(item_values_temp, S_temp[
137                 l_p])
138             line_values_temp[l_m] = generate.H(item_values_temp, S_temp[
139                 l_m])

```

```

128         delta_p, S_temp[l_p] = tabu(N, NL, S_temp[l_p], items,
129             completion_temp, tardiness_temp, lambdal, lambda2)
130         delta_m, S_temp[l_m] = tabu(N, NL, S_temp[l_m], items,
131             completion_temp, tardiness_temp, lambdal, lambda2)
132         line_values_temp[l_p] += delta_p
133         line_values_temp[l_m] += delta_m
134         value = generate.H(item_values_temp, L)
135         value = value + delta_m + delta_p
136         if value < G:
137             G = value
138             line_values = line_values_temp[:]
139             completion, tardiness, item_values = value_generator(S_temp,
140                 items, lambdal, lambda2)
141             for l in xrange(m):
142                 S[l] = S_temp[l][:]
143         print G
144         return G
145
146 if __name__ == '__main__':
147     if len(sys.argv) > 1:
148         file_location = sys.argv[1].strip()
149         input_data_file = open(file_location, 'r')
150         input_data = ''.join(input_data_file.readlines())
151         input_data_file.close()
152         m = int(sys.argv[2])
153         NR = int(raw_input('iterate time:'))
154         N = int(raw_input('tabu search iterate time:'))
155         NL = int(raw_input('tabu list volume:'))
156         lambdal = float(raw_input('lambdal:'))
157         f = open("result\\bt_" + str(int(file_location[7:])) + "_"
158             + str(m) + "_" + str(lambdal), 'w')
159         G = solve(input_data, m, lambdal, N, NL, NR)
160         f.write(str(G) + '\n' + str(Rb) + '\n')
161         f.close()

```

## (5) basicvirtual.py

```

1 from __future__ import division
2 import sys
3 sys.path.append("../functions")
4 import generate
5 import basictabu
6 import random
7 from basictac import h
8 from collections import namedtuple
9 Item = namedtuple("Item", ['process', 'due', 'wt', 'wc'])
10
11 def complete_time(S, items): # calculate the completion time
12     c = []
13     t = 0
14     for j in S:
15         t += items[j].process
16         c.append(t)
17     return c
18
19 def value_generator(S, items, lambdal, lambda2): # generate completion,
20     tardiness, and item contribution values
21     completion = complete_time(S, items)
22     item = [items[j] for j in S]
23     lateness = generate.late(completion, item)
24     tardiness = generate.tard(lateness)
25     item_values = []
26     for j in xrange(len(item)):
27         itm = item[j]
28         wt, wc = itm.wt, itm.wc
29         t, c = tardiness[j], completion[j]
30         value = h(t, c, wt, wc, lambdal, lambda2)
31         item_values.append(value)
32     return completion, tardiness, item_values
33
34 def same_delta(a, b, S, items, line_value, lambdal, lambda2): # delta
35     come from in-line swap
36     K = generate.innerswap(S, S.index(a), S.index(b))
37     c, t, v = value_generator(K, items, lambdal, lambda2)
38     new_value = sum(v)
39     delta = new_value - line_value

```

```

38     return K,c,t,v,delta
39
40 def diff_delta(S_x,items,line_value,lambdal,lambda2): # delta come
41     from between-line swap
42     c, t, v = value_generator(S_x,items,lambdal,lambda2)
43     new_value = sum(v)
44     delta = new_value - line_value
45     return c,t,v,delta
46
47 def delta1(S,l_a,l_b,a,b_idx,item_values,items,line_values,lambdal,
48     lambda2): # insert item type 1
49     S_I_1 = S[l_a][:]
50     S_I_2 = S[l_b][:]
51     S_I_2.insert(b_idx,a)
52     S_I_1.remove(a)
53     c_I_1,t_I_1,v_I_1,delta_I_1 = diff_delta(S_I_1,items,line_values[
54         l_a],lambdal,lambda2)
55     c_I_2,t_I_2,v_I_2,delta_I_2 = diff_delta(S_I_2,items,line_values[
56         l_b],lambdal,lambda2)
57     delta_I = delta_I_1 + delta_I_2
58     return S_I_1,S_I_2,c_I_1,c_I_2,t_I_1,t_I_2,v_I_1,v_I_2,c_I_2,
59         delta_I
60
61 def delta2(S,l_a,l_b,b,a_idx,item_values,items,line_values,lambdal,
62     lambda2): # insert item type 2
63     S_II_1 = S[l_a][:]
64     S_II_2 = S[l_b][:]
65     S_II_1.insert(a_idx + 1,b)
66     S_II_2.remove(b)
67     c_II_1,t_II_1,v_II_1,delta_II_1 = diff_delta(S_II_1,items,
68         line_values[l_a],lambdal,lambda2)
69     c_II_2,t_II_2,v_II_2,delta_II_2 = diff_delta(S_II_2,items,
70         line_values[l_b],lambdal,lambda2)
71     delta_II = delta_II_1 + delta_II_2
72     return S_II_1,S_II_2,c_II_1,c_II_2,t_II_1,t_II_2,v_II_1,v_II_2,
73         delta_II
74
75 def delta3(S,l_a,l_b,a,b,a_idx,b_idx,item_values,items,line_values,
76     lambdal,lambda2): # insert item type 3
77     S_III_1 = S[l_a][:]
78     S_III_2 = S[l_b][:]
79     S_III_1.insert(a_idx,b)
80     S_III_1.remove(a)
81     S_III_2.insert(b_idx,a)
82     S_III_2.remove(b)
83     c_III_1,t_III_1,v_III_1,delta_III_1 = diff_delta(S_III_1,items,
84         line_values[l_a],lambdal,lambda2)
85     c_III_2,t_III_2,v_III_2,delta_III_2 = diff_delta(S_III_2,items,
86         line_values[l_b],lambdal,lambda2)
87     delta_III = delta_III_1 + delta_III_2
88     return S_III_1,S_III_2,c_III_1,c_III_2,t_III_1,t_III_2,v_III_1,
89         v_III_2,delta_III
90
91 def tabu(N,NL,S,L,items, completion,tardiness,line_values,item_values
92     ,G,lambdal,lambda2):
93     TL = [None]*NL
94     from_same = []
95     from_diff = []
96     pairs = generate.pairsets(L)
97     completion_temp = completion[:] # backup all the var
98     tardiness_temp = tardiness[:]
99     line_values_temp = line_values[:]
100     item_values_temp = item_values[:]
101     delta_star = 0
102     Delta = 0
103     L_star = L[:]
104     m = len(S)
105     S_star = [None]*m
106     for l in xrange(m):
107         S_star[l] = S[l][:]
108     for k in xrange(N):
109         delta = []
110         issame = 0
111         isdiff = 0
112         out = 0
113         for s in pairs:
114             job = list(s)

```

```

101     a_l_index = max(L.index(job[0]),L.index(job[1]))
102     b_l_index = min(L.index(job[0]),L.index(job[1]))
103     a = L[a_l_index]
104     b = L[b_l_index]
105     l_a = generate.find_job(a,S)
106     l_b = generate.find_job(b,S)
107     a_idx = S[l_a].index(a)
108     b_idx = S[l_b].index(b)
109     if s not in TL:
110         if l_a == l_b:
111             S_k,c_k,t_k,v_k,delta_h = same_delta(a,b,S[l_a],
112                 items,line_values_temp[l_a],lambdal,lambda2
113             )
114         else:
115             # delta_I
116             S_I_1,S_I_2,c_I_1,c_I_2,t_I_1,t_I_2,v_I_1,v_I_2,
117                 c_I_2,delta_I = delta1(S,l_a,l_b,a,b_idx,
118                 item_values,items,line_values_temp,lambdal,
119                 lambda2)
120             # delta_II
121             S_II_1,S_II_2,c_II_1,c_II_2,t_II_1,t_II_2,v_II_1,
122                 v_II_2,delta_II = delta2(S,l_a,l_b,b,a_idx,
123                 item_values,items,line_values_temp,lambdal,
124                 lambda2)
125             # delta_III
126             S_III_1,S_III_2,c_III_1,c_III_2,t_III_1,t_III_2,
127                 v_III_1,v_III_2,delta_III = delta3(S,l_a,
128                 l_b,a,b,a_idx,b_idx,item_values,items,
129                 line_values_temp,lambdal,lambda2)
130             delta_h = min(delta_I,delta_II,delta_III)
131         elif s in from_same:
132             continue
133         else:
134             # delta_i
135             S_i_1,S_i_2,c_i_1,c_i_2,t_i_1,t_i_2,v_i_1,v_i_2,c_i_2,
136                 delta_i = delta1(S,l_a,l_b,a,b_idx,item_values
137                 ,items,line_values_temp,lambdal,lambda2)
138             # delta_ii
139             S_ii_1,S_ii_2,c_ii_1,c_ii_2,t_ii_1,t_ii_2,v_ii_1,
140                 v_ii_2,delta_ii = delta2(S,l_a,l_b,b,a_idx,
141                 item_values,items,line_values_temp,lambdal,
142                 lambda2)
143             delta_h = min(delta_i,delta_ii)
144         if delta == [] or delta_h < delta:
145             delta = delta_h
146             a_temp,b_temp = a,b
147             l_a_temp,l_b_temp = l_a,l_b
148             if l_a == l_b:
149                 S_temp = S_k[:]
150                 c_temp,t_temp,v_temp = c_k,t_k,v_k
151                 issame,isdiff = 1,0
152                 out = 0
153             else:
154                 if s in TL:
155                     out = 1
156                     issame,isdiff = 0,0
157                     if delta == delta_i:
158                         c1_temp,t1_temp,v1_temp = c_i_1,t_i_1,
159                             v_i_1
160                         c2_temp,t2_temp,v2_temp = c_i_2,t_i_2,
161                             v_i_2
162                         S1_temp,S2_temp = S_i_1,S_i_2
163                     elif delta == delta_ii:
164                         c1_temp,t1_temp,v1_temp = c_ii_1,t_ii_1,
165                             v_ii_1
166                         c2_temp,t2_temp,v2_temp = c_ii_2,t_ii_2,
167                             v_ii_2
168                         S1_temp,S2_temp = S_ii_1,S_ii_2
169                     else:
170                         out = 0
171                     if delta == delta_I:
172                         c1_temp,t1_temp,v1_temp = c_I_1,t_I_1,
173                             v_I_1
174                         c2_temp,t2_temp,v2_temp = c_I_2,t_I_2,
175                             v_I_2
176                         S1_temp,S2_temp = S_I_1,S_I_2
177                     issame,isdiff = 0,0

```

```

156         elif delta == delta_II:
157             c1_temp,t1_temp,v1_temp = c_II_1,t_II_1,
158                 v_II_1
159             c2_temp,t2_temp,v2_temp = c_II_2,t_II_2,
160                 v_II_2
161             S1_temp,S2_temp = S_II_1,S_II_2
162             issame,isdiff = 0,0
163         elif delta == delta_III:
164             c1_temp,t1_temp,v1_temp = c_III_1,t_III_1,
165                 v_III_1
166             c2_temp,t2_temp,v2_temp = c_III_2,t_III_2,
167                 v_III_2
168             S1_temp,S2_temp = S_III_1,S_III_2
169             issame,isdiff = 0,1
170     if delta == []:
171         continue
172     L = generate.innerswap(L,L.index(a_temp),L.index(b_temp))
173     pairs = generate.pairsets(L)
174     change_set = set([a_temp,b_temp])
175     if out:
176         TL[TL.index(change_set)] = None
177         from_diff.remove(change_set)
178         delete_set = TL.pop(0)
179         if delete_set in from_same:
180             from_same.remove(delete_set)
181         elif delete_set in from_diff:
182             from_diff.remove(delete_set)
183         if isdiff or issame:
184             TL.append(change_set)
185             if isdiff:
186                 from_diff.append(change_set)
187             if issame:
188                 from_same.append(change_set)
189         else:
190             TL.append(None)
191     if l_a_temp != l_b_temp:
192         c_temp = c1_temp + c2_temp
193         t_temp = t1_temp + t2_temp
194         v_temp = v1_temp + v2_temp
195         i = 0
196         for j in S1_temp + S2_temp:
197             completion_temp[j] = c_temp[i]
198             tardiness_temp[j] = t_temp[i]
199             item_values_temp[j] = v_temp[i]
200             i+=1
201         S[l_a_temp] = S1_temp[:]
202         S[l_b_temp] = S2_temp[:]
203         line_values_temp[l_a_temp] = sum(v1_temp)
204         line_values_temp[l_b_temp] = sum(v2_temp)
205     else:
206         S[l_a_temp] = S_temp[:]
207         line_values_temp[l_a_temp] = sum(v_temp)
208     TL,from_same,from_diff = generate.TL_update(TL,from_same,
209         from_diff,S)
210     Delta += delta
211     if Delta < delta_star:
212         delta_star = Delta
213         for l in xrange(m):
214             S_star[l] = S[l][:]
215             L_star = L[:]
216             line_values = line_values_temp[:]
217             item_values = item_values_temp[:]
218             completion = completion_temp[:]
219             tardiness = tardiness_temp[:]
220     return delta_star,S_star,L_star,line_values,item_values,
221         completion,tardiness
222
223 def solve(input_data,N,NL,m,lambdal):
224     lambda2 = 1 - lambdal
225     Data = input_data.split('\n') # load data
226     n = len(Data) - 1 # get the amount of items
227     items = []
228     for j in xrange(n):
229         data = Data[j]
230         parts = data.split()
231         p = int(parts[0]) # get the process time
232         s = int(parts[2]) # get the setup time

```

```

227     d = int(parts[3]) # get the due date
228     wt = int(parts[4]) # get the tardiness weights
229     wc = int(parts[5]) # get the completion weights
230     items.append(Item(p+s,d,wt,wc)) # combine those item data
231     print 'Data_loaded!'
232     S,L,completion = generate.initialization(items,n,m)
233     lateness = generate.late(completion,items)
234     tardiness = generate.tard(completion,items)
235     item_values = []
236     for j in xrange(len(items)):
237         item = items[j]
238         wt,wc = item.wt,item.wc
239         t,c = tardiness[j],completion[j]
240         value = h(t,c,wt,wc,lambdal,lambda2)
241         item_values.append(value)
242     print 'Initialization_done!'
243
244     G = generate.H(item_values,L)
245     line_values = []
246     for s in S:
247         value = generate.H(item_values,s)
248         line_values.append(value)
249     print 'Initial_values_done!'
250
251     for l in xrange(m):
252         delta,S[l] = basictabu.tabu(l500,NL,S[l],items,completion,
253             tardiness,lambdal,lambda2)
254         G += delta
255         line_values[l] += delta
256         completion,tardiness,item_values = generate.verify(S,items,
257             lambdal,lambda2)
258     print 'Initial_Tabu_Search_Done!'
259
260     delta,S,L,line_values,item_values,completion,tardiness = tabu(N,
261         NL,S,L,items, completion,tardiness,line_values,item_values,
262         G,lambdal,lambda2)
263     G += delta
264     completion,tardiness,item_values = generate.verify(S,items,
265         lambdal,lambda2)
266     return G
267
268 if __name__ == '__main__':
269     if len(sys.argv) > 1:
270         file_location = sys.argv[1].strip()
271         m = int(sys.argv[2])
272         input_data_file = open(file_location, 'r')
273         input_data = ''.join(input_data_file.readlines())
274         input_data_file.close()
275         N = int(raw_input('iterate_time:'))
276         NL = int(raw_input('tabu_list_volume:'))
277         lambdal = float(raw_input('lambdal:'))
278         f = open("../result/bv_" + str(int(file_location[7:])) + "_"
279             + str(m) + "_" + str(lambdal), 'w')
280         G = solve(input_data,N,NL,m,lambdal)
281         f.write(str(G) + '\n' + str(Rb) + '\n')
282         f.close()

```

## (6) continueatcs.py

```

1 import sys
2 import math
3 sys.path.append("../functions")
4 import generate
5 from collections import namedtuple
6 Item = namedtuple("Item", ['process','release','setup','due','wt','wc',
7     ''])
8
9 def h(S,completion,items,lambdal,lambda2): # redefine the
10     contribution of one item for the obj function
11     n = len(items)
12     lateness = generate.late(completion,items)
13     value = [None]*n
14     Rb,c_max = generate.balance_rate(completion,S)
15     Ru = generate.idle_rate(items,completion,c_max,S)
16     for s in S:
17         l = S.index(s)

```



```

16         for j in s:
17             value[j] = lambda1*(math.fabs(items[j].wt*lateness[j]))/
                Ru[1] + lambda2*items[j].wc*completion[j]*math.exp
                (-Rb)
18         return value
19
20 def complete_time(S,items,lambdal,lambda2):
21     n = len(items)
22     c = [None]*n
23     f = [None]*n
24     for s in S:
25         t = 0
26         for j in s:
27             item = items[j]
28             if s.index(j) == 0:
29                 f[j] = max(item.release - item.setup,0)
30             else:
31                 f[j] = max(item.release - item.setup - c[s.index(j)
                    -1],0)
32             t += item.process + item.setup + f[j]
33             c[j] = t
34         item_values = h(S,c,items,lambdal,lambda2)
35         line_values = []
36         for s in S:
37             v = [item_values[j] for j in s]
38             line_values.append(sum(v))
39         return c,line_values
40
41 def ATCS(items,S,m): # use the ATCS rule
42     J = S[:]
43     S = []
44     t = 0
45     c = []
46     n = len(items)
47     f = []
48     k_1,k_2 = generate.estimate(m,items)
49     while J:
50         p = [items[j].process for j in J]
51         r = [items[j].release for j in J]
52         d = [items[j].due for j in J]
53         s = [items[j].setup for j in J]
54         wt = [items[j].wt for j in J]
55         orderidx = generate.Idx_c(t,p,s,d,wt,k_1,k_2)
56         j_star = J[orderidx.index(max(orderidx))]
57         S.append(j_star)
58         J.remove(j_star)
59         if len(S) == 1:
60             f.append(max(items[j_star].release - items[j_star].setup
                ,0))
61         else:
62             f.append(max(items[j_star].release - items[j_star].setup
                - c[-1],0))
63         t = t + items[j_star].process + items[j_star].setup + f[-1]
64         c.append(t)
65     return S,c
66
67 def solve(input_data,m,lambdal):
68     lambda2 = 1 - lambdal
69     Data = input_data.split('\n') # load data
70     n = len(Data) - 1 # get the amount of items
71     items = []
72     for j in xrange(n):
73         data = Data[j]
74         parts = data.split()
75         p = int(parts[0]) # get the process time
76         r = int(parts[1]) # get the release time
77         s = int(parts[2]) # get the setup time
78         d = int(parts[3]) # get the due date
79         wt = int(parts[4]) # get the tardiness weights
80         wc = int(parts[5]) # get the completion weights
81         items.append(Item(p,r,s,d,wt,wc)) # combine those item data
82     print 'Data loaded!'
83     S,L,completion,item_free = generate.initialization_c(items,n,m)
84     print 'Initialization done!'
85     line_values,G = generate.Goal(completion,items,S,lambdal,lambda2)
86     print 'Initialization values done!'
87     print G

```

```

88     NR = 19
89     item_values = h(S,completion,items,lambdal,lambda2)
90     for k in xrange(NR):
91         l_p,l_m = generate.reorder(items,S,line_values,item_values)
92         S[l_p],c_p = ATCS(items,S[l_p],m)
93         S[l_m],c_m = ATCS(items,S[l_m],m)
94         completion,line_values = complete_time(S,items,lambdal,
            lambda2)
95         item_values = h(S,completion,items,lambdal,lambda2)
96         G = sum(line_values)
97         Rb,c = generate.balance_rate(completion,S)
98         return G,Rb
99
100 if __name__ == '__main__':
101     if len(sys.argv) > 1:
102         file_location = sys.argv[1].strip()
103         input_data_file = open(file_location, 'r')
104         input_data = ''.join(input_data_file.readlines())
105         input_data_file.close()
106         m = int(sys.argv[2])
107         lambdal = float(raw_input('lambdal='))
108         f = open("../result/cats_" + str(int(file_location[7:])) + "
            " + str(m) + "_" + str(lambdal), 'w')
109         G,Rb = solve(input_data,m,lambdal)
110         f.write(str(G) + '\n' + str(Rb) + '\n')
111         f.close()
112

```

## (7) continuatabu.py

```

1 import sys
2 import math
3 sys.path.append("../functions")
4 import generate
5 import continueatcs
6 import random
7 from collections import namedtuple
8 Item = namedtuple("Item", ['process','release','setup','due','wt','wc'
    ''])
9
10 def tabu(N,NL,S,l,items,G,completion,line_values,lambdal,lambda2):
11     TL = [None]*NL
12     pairs = generate.pairsets(S[l])
13     completion_temp = completion[:]
14     line_values_temp = line_values[:]
15     G_star = G
16     S_star = []
17     test_G = []
18     for s in S:
19         S_star.append(s[:])
20     for k in xrange(N):
21         test_S = []
22         s = random.choice(pairs)
23         move = 0
24         if s not in TL:
25             job = list(s)
26             a = job[0]
27             b = job[1]
28             a_idx = S_star[l].index(a)
29             b_idx = S_star[l].index(b)
30             S_temp = generate.innerswap(S_star[l],a_idx,b_idx)
31             c,v = continueatcs.complete_time(S_star[l]+[S_temp]+
                S_star[l+1:],items,lambdal,lambda2)
32             G_temp = sum(v)
33             if test_G == [] or G_temp < test_G:
34                 test_G = G_temp
35                 test_S = S_temp[:]
36                 completion_temp = c[:]
37                 line_values_temp = v[:]
38                 a_star = a
39                 b_star = b
40                 move = 1
41         else:
42             continue
43     if move == 0:
44         continue

```

```

45     change_set = set([a_star,b_star])
46     generate.pairsets_update(pairs,change_set)
47     S_star[1] = test_S[:]
48     TL.pop(0)
49     TL.append(change_set)
50     if test_G < G_star:
51         G_star = test_G
52         S[1] = test_S[:]
53         line_values = line_values_temp[:]
54         completion = completion_temp[:]
55     return G_star,S,line_values,completion
56
57 def solve(input_data,lambda1,N,NL,m):
58     lambda2 = 1 - lambda1
59     Data = input_data.split('\n') # load data
60     n = len(Data) - 1 # get the amount of items
61     m = 5
62     items = []
63     for j in xrange(n):
64         data = Data[j]
65         parts = data.split()
66         p = int(parts[0]) # get the process time
67         r = int(parts[1]) # get the release time
68         s = int(parts[2]) # get the setup time
69         d = int(parts[3]) # get the due date
70         wt = int(parts[4]) # get the tardiness weights
71         wc = int(parts[5]) # get the completion weights
72         items.append(Item(p,r,s,d,wt,wc)) # combine those item data
73     print 'DataLoaded!'
74     S,L,completion,item_free = generate.initialization_c(items,n,m)
75     print 'Initializationdone!'
76     line_values,G = generate.Goal(completion,items,S,lambda1,lambda2)
77     print 'Initialization_valuesdone!'
78     print G
79
80     NR = 10
81     item_values = continueatcs.h(S,completion,items,lambda1,lambda2)
82     G_star = G
83     S_star = []
84     for s in S:
85         S_star.append(s[:])
86     for k in xrange(NR):
87         l_p,l_m = generate.reorder(items,S_star,line_values,
88                                     item_values)
89         completion,line_values = continueatcs.complete_time(S_star,
90                                                             items,lambda1,lambda2)
91         G_star = sum(line_values)
92         G_star,S_star,line_values,completion = tabu(N,NL,S_star,l_p,
93                                                     items,G_star,completion,line_values,lambda1,lambda2)
94         G_star,S_star,line_values,completion = tabu(N,NL,S_star,l_m,
95                                                     items,G_star,completion,line_values,lambda1,lambda2)
96         item_values = continueatcs.h(S_star,completion,items,lambda1,
97                                     lambda2)
98         Rb,_ = generate.balance_rate(completion,S)
99         completion,line_values = continueatcs.complete_time(S_star,items,
100                                                             lambda1,lambda2)
101     return G_star,Rb
102
103 if __name__ == '__main__':
104     if len(sys.argv) > 1:
105         file_location = sys.argv[1].strip()
106         input_data_file = open(file_location, 'r')
107         input_data = ''.join(input_data_file.readlines())
108         input_data_file.close()
109         m = int(sys.argv[2])
110         N = int(raw_input('iterate_time=u'))
111         NL = int(raw_input('tabulist_volume=u'))
112         lambda1 = float(raw_input('lambda1=u'))
113         f = open("result\\ct_" + str(int(file_location[7:])) + "_"
114                 + str(m) + "_" + str(lambda1), 'w')
115         G,Rb = solve(input_data,lambda1,N,NL,m)
116         f.write(str(G) + '\n' + str(Rb) + '\n')
117         f.close()

```

(8) continuevirtual.py

```

1 from __future__ import division
2 import sys
3 sys.path.append("../functions")
4 import generate
5 import continueatcs
6 import continuetaabu
7 import random
8 from collections import namedtuple
9 Item = namedtuple("Item", ['process','release','setup','due','wt','wc',
10                             ''])
11
12 def same_S(S,l,a_idx,b_idx):
13     S_t = []
14     for s in S:
15         S_t.append(s[:])
16     S_t[1] = generate.innerswap(S_t[1],a_idx,b_idx)
17     return S_t
18
19 def test_G(items,S,lambda1,lambda2):
20     c_temp,v_temp = continueatcs.complete_time(S,items,lambda1,
21                                                 lambda2)
22     return sum(v_temp)
23
24 def diff1_S(S,l_a,l_b,a,b_idx):
25     S_t = []
26     for s in S:
27         S_t.append(s[:])
28     S_1 = S_t[l_a][:]
29     S_2 = S_t[l_b][:]
30     S_2.insert(b_idx,a)
31     S_1.remove(a)
32     S_t[l_a] = S_1
33     S_t[l_b] = S_2
34     return S_t
35
36 def diff2_S(S,l_a,l_b,b,a_idx):
37     S_t = []
38     for s in S:
39         S_t.append(s[:])
40     S_1 = S_t[l_a][:]
41     S_2 = S_t[l_b][:]
42     S_1.insert(a_idx + 1,b)
43     S_2.remove(b)
44     S_t[l_a] = S_1
45     S_t[l_b] = S_2
46     return S_t
47
48 def diff3_S(S,l_a,l_b,a,b,a_idx,b_idx):
49     S_t = []
50     for s in S:
51         S_t.append(s[:])
52     S_1 = S_t[l_a][:]
53     S_2 = S_t[l_b][:]
54     S_1.insert(a_idx,b)
55     S_1.remove(a)
56     S_2.insert(b_idx,a)
57     S_2.remove(b)
58     S_t[l_a] = S_1
59     S_t[l_b] = S_2
60     return S_t
61
62 def tabu(N,NL,S,L,items,G,lambda1,lambda2):
63     TL = [None]*NL
64     from_same = []
65     from_diff = []
66     pairs = generate.pairsets(L)
67     G_star = G
68     L_star = L[:]
69     S_c = []
70     S_ori = []
71     for s in S:
72         S_ori.append(s[:])
73         S_c.append(s[:])
74     G_test = 0
75     for k in xrange(N):
76         issame = 0
77         isdiff = 0

```

```

76 out = 0
77 s = random.choice(pairs)
78 move = 0
79 job = list(s)
80 a_l_index = max(L.index(job[0]), L.index(job[1]))
81 b_l_index = min(L.index(job[0]), L.index(job[1]))
82 a = L[a_l_index]
83 b = L[b_l_index]
84 l_a = generate.find_job(a, S)
85 l_b = generate.find_job(b, S)
86 a_idx = S[l_a].index(a)
87 b_idx = S[l_b].index(b)
88 if s not in TL:
89     if l_a == l_b:
90         # a <-> b
91         S_temp = same_S(S, l_a, a_idx, b_idx)
92         G_temp = test_G(items, S_temp, lambda1, lambda2)
93     else:
94         # a -> b's
95         S_temp_1 = diff1_S(S, l_a, l_b, a, b_idx)
96         G_temp_1 = test_G(items, S_temp_1, lambda1, lambda2)
97         # b -> a's
98         S_temp_2 = diff2_S(S, l_a, l_b, b, a_idx)
99         G_temp_2 = test_G(items, S_temp_2, lambda1, lambda2)
100         # a <-> b
101         S_temp_3 = diff3_S(S, l_a, l_b, a, b_idx, b_idx)
102         G_temp_3 = test_G(items, S_temp_3, lambda1, lambda2)
103         G_temp = min(G_temp_1, G_temp_2, G_temp_3)
104 elif s in from_same:
105     continue
106 elif s in from_diff:
107     # a -> b's
108     S_temp_1 = diff1_S(S, l_a, l_b, a, b_idx)
109     G_temp_1 = test_G(items, S_temp_1, lambda1, lambda2)
110     # b -> a's
111     S_temp_2 = diff2_S(S, l_a, l_b, b, a_idx)
112     G_temp_2 = test_G(items, S_temp_2, lambda1, lambda2)
113     G_temp = min(G_temp_1, G_temp_2)
114 if G_test == 0 or G_temp < G_test:
115     move = 1
116     G_test = G_temp
117     a_temp, b_temp = a, b
118     l_a_temp, l_b_temp = l_a, l_b
119     if l_a == l_b:
120         S_star = S_temp
121         issame, isdiff = 1, 0
122         out = 0
123     else:
124         if s in TL:
125             out = 1
126             issame, isdiff = 0, 0
127             if G_temp == G_temp_1:
128                 S_star = S_temp_1
129             elif G_temp == G_temp_2:
130                 S_star = S_temp_2
131         else:
132             out = 0
133             if G_temp == G_temp_1:
134                 S_star = S_temp_1
135                 issame, isdiff = 0, 0
136             elif G_temp == G_temp_2:
137                 S_star = S_temp_2
138                 issame, isdiff = 0, 0
139             elif G_temp == G_temp_3:
140                 S_star = S_temp_3
141                 issame, isdiff = 0, 1
142 if move == 0:
143     continue
144 L = generate.innerswap(L, L.index(a_temp), L.index(b_temp))
145 pairs = generate.pairsets(L)
146 change_set = set([a_temp, b_temp])
147 if out:
148     TL[TL.index(change_set)] = None
149     from_diff.remove(change_set)
150 delete_set = TL.pop(0)

```

```

151 if delete_set in from_same:
152     from_same.remove(delete_set)
153 elif delete_set in from_diff:
154     from_diff.remove(delete_set)
155 if isdiff or issame:
156     TL.append(change_set)
157     if isdiff:
158         from_diff.append(change_set)
159     if issame:
160         from_same.append(change_set)
161 else:
162     TL.append(None)
163 S = S_star
164 G = G_test
165 TL, from_same, from_diff = generate.TL_update(TL, from_same,
166     from_diff, S)
167 if G < G_star:
168     G_star = G
169     l = 0
170     for s in S:
171         S_c[l][:] = s[:]
172         l += 1
173     L_star = L[:]
174 return G_star, S_c
175
176 def solve(input_data, N, NL, lambda1, m):
177     lambda2 = 1 - lambda1
178     Data = input_data.split('\n') # load data
179     n = len(Data) - 1 # get the amount of items
180     items = []
181     for j in xrange(n):
182         data = Data[j]
183         parts = data.split()
184         p = int(parts[0]) # get the process time
185         r = int(parts[1])
186         s = int(parts[2]) # get the setup time
187         d = int(parts[3]) # get the due date
188         wt = int(parts[4]) # get the tardiness weights
189         wc = int(parts[5]) # get the completion weights
190         items.append(Item(p, r, s, d, wt, wc)) # combine those item data
191     print 'Data_loaded!'
192     S, L, completion, item_free = generate.initialization_c(items, n, m)
193     print 'Initialization_done!'
194     line_values, G = generate.Goal(completion, items, S, lambda1, lambda2)
195     print 'Initial_values_done!'
196
197     for l in xrange(m):
198         G, S, line_values, completion = continuatabu.tabu(N, NL, S, l, items,
199             G, completion, line_values, lambda1, lambda2)
200         G, S = tabu(N, NL, S, L, items, G, lambda1, lambda2)
201         c, v = continueatcs.complete_time(S, items, lambda1, lambda2)
202         Rb, _ = generate.balance_rate(completion, S)
203         return G, Rb
204
205 if __name__ == '__main__':
206     if len(sys.argv) > 1:
207         file_location = sys.argv[1].strip()
208         input_data_file = open(file_location, 'r')
209         input_data = ''.join(input_data_file.readlines())
210         input_data_file.close()
211         m = int(sys.argv[2])
212         N = int(raw_input('iterate_time_u='))
213         NL = int(raw_input('tabu_list_volume_u='))
214         lambda1 = float(raw_input('lambda1_u='))
215         f = open("result\\cv_" + str(int(file_location[7:])) + "_"
216             + str(m) + "_" + str(lambda1), 'w')
217         G = []
218         for i in xrange(3):
219             G_temp, Rb_temp = solve(input_data, N, NL, lambda1, m)
220             if G == [] or G_temp < G:
221                 G = G_temp
222                 Rb = Rb_temp
223         f.write(str(G) + '\n' + str(Rb) + '\n')
224         f.close()

```

## 致 谢

在本毕业设计完成之际，要首先感谢指导老师鲁建厦教授和董巧英老师对我的精心指导，让我受益匪浅，同时也要感谢各专业课的授课老师，让我能在毕业设计中受到启发，还要感谢一起学习的同学们，我们之间的讨论让我获得许多思考。

毕业设计是一次学习的历程，有陷入困境时的烦躁，也有解决问题时的欣喜， it's a lot of fun, and a lot of work, but a lot of fun。要感谢 CTeX 论坛里的各位大牛给我这个排版菜鸟的诸多灵感，要感谢精弘论坛里学长的参考模板以及版主 Unlucky 对我的模板所提的宝贵建议。要感谢我的室友在我忙碌忘我的时候提醒我注意休息，一起放松。

更要感谢我的父母，在背后一直默默地支持我。

由于本人的学识水平所限，对有关知识的认识还不够全面、深刻，因此，本论文难免有错漏疏忽之处，敬请各位老师同学批评指正，力求在学习工作中加以改正和提高。