流水车间与开放车间调度算法渐近分析

白丹宇 著

清华大学出版社

流水车间与开放车间调度算法渐近分析

白丹宇 著

清华大学出版社
北京

内 容 简 介

流水车间与开放车间调度在流程工业、离散制造、检测维修以及医疗管理等领域有着广泛的应用.除了极少数特殊情况之外,此类问题基本上都是 NP 难的.对于小规模问题,一般是采用基于枚举的算法进行最优求解.但是随着问题规模的增大,求得最优解所花费的时间成指数增长,在这种情况下,利用启发式算法求得问题的近似解是一种快速而有效的方法.本书针对所研究的车间调度模型,从理论的角度分析了若干典型启发式算法的性能,其中重点讨论了渐近分析方法在研究调度算法收敛性方面的应用.

本书可作为系统工程、应用数学、运筹学与控制论、计算机软件与理论、工业工程、管理科学与工程等相关专业的教师、研究生、高年级本科生以及科研人员的参考书.

图书在版编目(CIP)数据

流水车间与开放车间调度算法渐近分析/白丹宇著.--北京:清华大学出版社,2015
ISBN 978-7-302-41786-6

Ⅰ.①流…　Ⅱ.①白…　Ⅲ.①车间调度－调度模型－算法分析　Ⅳ.①F406.2

中国版本图书馆 CIP 数据核字(2015)第 246066 号

责任编辑:汪　操
封面设计:常雪影
责任校对:赵丽敏
责任印制:宋　林

出版发行:清华大学出版社
网　址:http://www.tup.com.cn, http://www.wqbook.com
地　址:北京清华大学学研大厦 A 座　　**邮　编**:100084
社总机:010-62770175　　**邮　购**:010-62786544
投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn
质　量　反　馈:010-62772015, zhiliang@tup.tsinghua.edu.cn
印 刷 者:三河市君旺印务有限公司
装 订 者:三河市新茂装订有限公司
经　　销:全国新华书店
开　　本:170mm×230mm　　**印　张**:9.5　　**字　数**:173 千字
版　　次:2015 年 12 月第 1 版　　**印　次**:2015 年 12 月第1 次印刷
印　　数:1～2000
定　　价:29.00 元

产品编号:063403-01

PREFACE 序

第二次世界大战期间运筹学(operations research)兴起,首次把运作(operation)作为研究(research)的对象.其间,研究运作的时间安排促成了排序(scheduling)概念的建立和研究的开展.20世纪60年代越民义就注意到排序问题的重要性和在理论上的难度.1963年他编写国内第一本排序论讲义.70年代初他和韩继业研究同顺序流水作业排序,开创了中国研究排序论的先河.在他们两位的倡导和带动下国内排序论的理论研究和应用研究有了较大的发展.国内自动化学科把scheduling译为"调度"始于1983年.国际著名scheduling专家Potts等指出:"排序论的进展是巨大的.这些进展得益于研究人员从不同的学科(例如,数学、运筹学、管理科学、计算机科学、工程学和经济学)所做出的贡献."国内scheduling研究始终保持蓬勃发展的势态正是得益于多学科的交叉.运筹学和组合优化中的"排序"与自动化学科的"调度"是同一个英文单词scheduling的两个译名.在此,"排序"和"调度"是内涵和外延完全相同的名词.

排序与调度(scheduling)是研究运作的时间安排,是为加工若干个工件(job),而对工件及加工所需要的机器(machine)按时间进行分配和安排,在完成所有工件加工时使得某个(些)目标为最优.排序论中的"工件"可以是任务、非圆齿轮、计算机终端、病人、降落的飞机等,"机器"可以是完成任务所需要的人财物资源、数控机床、计算机中央处理器(CPU)、医生、机场跑道等."机器"和"工件"已经不是机器制造业中的"车床"和"车床加工的螺丝",而是从"车床"和"螺丝"等具体事物中抽象出来,是抽象的概念.由于不同学科关注的出发点和角度不同,排序与调度的术语差别很大.统一排序与调度的术语是学术交流的需要,需要各个学科来关心、研究和使用.例如,shop scheduling和parallel scheduling中的机器是分别以串联和并联两种形式组成的.把shop scheduling译成"车间调度"似乎没有刻画出机器的串联形式,是否可以译成"多工序调度(排序)"?对于shop scheduling的三种作业方式:flow shop,open shop和job shop是否可以译成流水作业、自由作业和异序作业?对于平行机排序parallel scheduling的三

种机器：identical machine，uniform machine 和 unrelated machine 是否可以译成同型机、同类机和非同类机？

绝大多数排序与调度问题是 NP-完备的. 在 P≠NP 的假设下，不可能在多项式时间内得到最优解. 因此，寻找性能优良的启发式算法和近似算法是排序与调度问题在理论研究和实际应用中主要的方向. 传统评价算法优劣的方法是分析和寻找算法的最坏情况性能比. 然而，如果算法的最坏情况很少出现，用最坏情况下的性能来评价算法的性能就有一定的片面性. 之后，出现了算法的渐近性能分析，研究问题的规模趋近于无穷时，算法的目标值与离线环境下的最优值之间的接近程度. 这种方法可以用来描述算法的收敛性，而且可以为智能优化算法提供具有性能保证的初始解. 这是一个很有发展前途的新方向，目前国际上关于渐近性能分析的成果还不多.

近年来，白丹宇在排序与调度算法的渐近分析方面做了很多工作，在国际期刊上发表了多篇论文，取得了一些标志性的成果. 例如，对于 open shop（自由作业）排序问题，白丹宇等在 2013 年用渐近分析方法证明了稠密排序（dense scheduling）算法具有渐近最优性. 这是继 1993 年英国 Warwick 大学陈磚提出猜想“稠密排序算法的最坏误差紧界为 $2-1/m$（m 为机器数）”之后的又一重要结论. 本书是他科研成果的结晶，是国内外第一本介绍排序算法渐近分析的著作. 本书的问世必将有助于排序算法理论的丰富与发展.

唐国春

2015 年 8 月

FOREWORD

前言

车间调度模型源自现代化工业生产过程，是复杂的多阶段决策过程，所研究的问题是确定若干项任务在一组处理机上的开始时间和加工顺序，使得目标函数最优化. 在这些问题中，绝大多数都是 NP 难的，即此类问题无法在多项式时间内求得最优解. 对于小规模问题，一般是应用分支定界或动态规划等枚举算法进行最优求解. 但是随着问题规模的增大，求得最优解所花费的时间成指数增长，此时，人们往往放弃寻求问题的最优解，转而寻找一个可以快速求得的可行解，把它作为最优解的近似值. 通常，这样的近似解可以通过构造启发式算法而得到. 那么，如何从理论角度评价启发式的有效性（即近似解的优劣）呢？ 经典的方法就是研究算法的目标函数值与该问题（离线）最优值之间的最大误差，以此来大致描述启发式的性能，称之为最坏情况（竞争）分析. 但是，用该方法求得的算法最坏情况（竞争）比紧界，都是一些人为构造的、极其特殊的实例，在大规模工业生产环境中发生的概率为零. 为了描述算法在求解大规模问题时的性能，较为合适的方法是采用渐近性能（竞争）分析，研究极限状态下，算法的目标函数值与该问题（离线）最优值之间的接近程度. 若算法的渐近性能（竞争）比为 1，则称之为渐近最优算法，即在理想状态下（问题规模趋于无穷大）可以将其看作是最优算法.

尽管渐近分析方法在评价启发式算法的收敛性方面有着无可比拟的优势，但是国际上的相关成果却并不多见. 本书是作者近年来相关科研成果的总结，其中融合了多篇在国外期刊上发表的科研论文，重点讨论流水车间和开放车间调度算法的渐近分析，同时也对其中的一些算法进行了最坏情况（竞争）分析. 全书共分为 10 章，主要内容概括如下：第 1 章为绪论，简要介绍调度问题的描述与求解方法、算法性能分析方法以及相关问题研究现状；第 2～3 章介绍流水车间极小化最大完工时间问题，证明了基于“单工件优先”启发式算法的渐近最优性，并提出了新的问题下界；第 4～6 章介绍了流水车间极小化完工时间 k 次方和问题，证明了基于“最短加工时间优先”启发式算法的渐近最优性，并进行了最坏

情况(竞争)分析,提出了新的问题下界;第 7～8 章介绍了开放车间极小化最大完工时间问题,分别证明了“旋转排序”和“稠密排序”算法的渐近最优性,并对“旋转排序”算法进行了最坏情况分析;第 9～10 章绍了开放车间极小化完工时间 k 次方和问题,证明了“最短加工时间分块”启发式算法的渐近最优性.

本书的部分研究成果是作者与东北大学唐立新教授、清华大学张智海副教授合作完成的.感谢国家自然科学基金-青年科学基金项目(71201107)对本书的资助与支持.在本书的撰写过程中,东北大学软件学院硕士研究生唐梦倩、张强、刘冰倩、赵鹏做了大量的工作;在本书的出版过程中,得到了清华大学出版社责任编辑的支持和帮助,在此一并表示衷心的感谢.

由于作者水平有限,书中的错误和不妥之处在所难免,恳请广大读者批评指正!

作者

2015 年 9 月

目录

CONTENTS

第二部分 开放车间调度问题

绪 论

1.1 调度问题的概述

调度(scheduling)又称排序或排程,是在满足一定的约束条件下如何将资源(resource)实时地分配给不同任务(task)的决策过程,其目的在于优化一个或多个目标(objective).调度理论是组合优化中的一个重要分支,最早起源于机器制造业,现在已逐渐发展成为运筹学、系统科学、控制科学、管理科学和计算机科学等多个学科领域的交叉学科,广泛应用于工程技术和经济管理的各个领域.作为一门应用科学,它在工业生产、交通运输、物流管理等方面有着深刻的实际背景和广阔的应用前景.

调度领域内早期的工作是在制造业推动下发展起来的,尽管现在调度问题在许多非制造业领域已经取得了相当多的成果,但是制造业的术语仍然在沿用.通常把资源称为机器(machine),把任务称为工件(job).有时工件可能是由几个先后次序约束相互联系着的基本任务组成,称之为工序(operation).调度问题中的"机器"和"工件"是抽象的概念,可以代表极其广泛的实际对象.例如:机器可以是钢厂轧机、机场登机口、中央处理器等;工件可以是钢坯、降落的航班、计算机进程等.这里,工件是被加工的对象,是要完成的任务;机器是提供加工的对象,是完成任务所需要的资源.

1.2 调度问题的定义

调度问题通常由机器、工件和目标函数三个要素构成.工件的数量记作 n,机器的数量记作 m.本书中,指标 j 表示工件,指标 i 表示机器.若工件需要多道

加工步骤，则采用(i,j)表示工件j在机器i上的加工步骤. 一般地，有如下与工件j相关的参数：

(1) 加工时间(processing time) $p(i,j)$：表示工件j在机器i上的加工时间. 如果工件j只在一台机器上加工，则记为p_j.

(2) 释放时间(release date) r_j：也称为到达时间(arrival time)，表示工件j到达系统的时间，也是该工件可以开始加工的最早时间.

(3) 工期(due date)d_j：也称为交货期，表示对工件j限定的完工时间. 如果不按时完工，应受到一定的惩罚. 绝对不允许延误的工期称为截止工期(deadline)，记为$\bar{d}_j$.

(4) 权重(weight)w_j：是一个优先因子，表示工件j相对于系统内其他工件的重要性. 例如，权重可以表示将工件保留在系统中而产生的实际费用.

目前国际上通常使用 Graham 等(1979)提出的“三参数表示法”来描述调度问题. 该表示法由三个域$\alpha|\beta|\gamma$组成：α域表示机器环境，一般只包含一项；β域提供加工的特征和约束的细节，可能不包含任何一项，也可能有多个选项；γ域描述需要优化的目标函数.

α域中规定的机器环境：

(1) 单机(single machine) 1：系统中只有一台机器.

(2) 并行机(parallel machine)：所有的机器具有相同的功能，工件j只需要在其中任意一台机器上完成加工即可.

(a) 同速机(identical machine) Pm：所有的机器具有相同的加工速度.

(b) 恒速机(uniform machine) Qm：机器的加工速度不同，但每台机器的速度都是常数.

(c) 变速机(unrelated machine) Rm：机器的加工速度取决于被加工的工件.

(3) 车间作业(shop or dedicated machine)：也称多类型机，指每台机器具有不同功能. 在车间作业中，工件需要在不同的机器上加工. 通常情况下，每个工件有多道工序，每道工序需要在不同的机器上加工. 若省略m，则表示机器数为变量.

(a) 流水车间(flow shop)Fm：每个工件都必须经过m台机器加工，而且所有工件都必须遵循相同的加工路径(即它们必须首先在第一台机器上加工，然后是第二台机器，以此类推).

(b) 开放车间(open shop) Om：每个工件需要经过m台机器加工，但有些工序的加工时间可以为零. 每个工件的加工路径没有限制，不同的工件可以有不同的加工路径. 调度者需要同时确定每个工件的加工路径以及每台机器的加工

顺序.

(c) 加工车间(job shop) Jm：系统中有 m 台机器，每个工件都有其已经预先确定的加工路径.

β 域中说明的加工约束和特定限制可能包括多个选项：

(1) 释放时间 r_j：工件有不同的释放时间. 按照工件到达前对其信息的了解程度，又可分为如下两种调度环境.

(a) 离线(offline)环境：该环境中工件的所有信息(比如工件的数量、加工时间以及释放时间等)都预先知道. 离线环境通常是为了理论研究的方便而进行的假设.

(b) 在线(online)环境：在该环境中，工件的信息不完备，所有的参数只有在其达到系统之后才知道，调度者不知道下面将要到达怎样的工件或者还要到达多少工件. 在线环境更符合实际生产中在最初决策时不知道所有信息的情况. 在线环境又分为两种：(i)工件按释放时间到达(over time)，(ii)工件按列表逐个到达(over list). 本书中研究的在线环境都是指工件按释放时间到达.

如果 β 域中不出现 r_j 或者 online，则说明所有工件在零时刻都可以利用，属于离线环境.

(2) 中断(preemption) prmp：工件不必在其加工完成之前一直保留在机器上. 调度者可以在任何时刻中断某个工件的加工，而把另一个工件安排在该机器上. 中断加工的工件已经完成的部分不会丢失，当其重新返回到机器上时，只需完成余下需要加工的部分即可. 当 prmp 不在 β 域中时，表示不允许中断.

(3) 排列(permutation) prmu：只出现在流水车间中，表示所有工件按照经过第一台机器的顺序始终不变地经过所有机器. 由于本书所研究的流水车间调度模型中只考虑排列排序，故省略该项.

(4) 无等待(no-wait) nwt：只出现在车间调度问题中，表示工件相邻的两道工序不允许在机器间等待.

(5) 优先约束(precedence) prec：出现于单机或并行机环境中，在某个工件开始加工之前，另一个或多个工件必须已经完工. 有几种优先约束的特殊形式：若每个工件最多有一个先行工件和一个后继工件，则称为链式；若每个工件最多有一个后继工件，则称为入树；若每个工件最多有一个先行工件，则称为出树. 当 prec 不在 β 域中时，表示工件没有优先限制.

γ 域中表示需要优化的目标函数. 目标函数分为正则函数与非正则函数. 本书只考虑正则目标函数，它又分为两类：使最大的费用最小(minimax criteria)和使总的费用和最小(minsum criteria). 对于给定的排序，用 C_j 表示工件 j 的完工时间. 常见的目标函数主要有如下几种形式：

(1) 最大完工时间(makespan) $C_{\max}$：定义为 $\max\{C_1, C_2, \cdots, C_n\}$，等于最后一个离开系统工件的完工时间. 该目标函数旨在提高机器的利用率.

(2) 加权完工时间和(total weighted completion time) $\sum w_j C_j$：表示所有工件完工时间的加权和. 该目标函数旨在降低持有或库存成本. 若所有工件的权重都相等，则称为完工时间和(total completion time)，记为 $\sum C_j$.

(3) 完工时间 k 次方和(total k-power completion time, $k \geqslant 2$) $\sum C_j^k$：表示所有工件完工时间的 k 次方和. 该目标函数可以同时提高机器的利用率并降低持有或库存成本.

(4) 最大延迟(maximum lateness) $L_{\max}$：定义为 $L_{\max} = \max\{L_j\}$，其中 $L_j = C_j - d_j$ 是工件 j 的延迟时间.

(5) 最大拖期(maximum tardiness) $T_{\max}$：定义为 $T_{\max} = \max\{T_j\}$，其中 $T_j = \max\{L_j, 0\}$ 是工件 j 的拖期时间.

(6) 总加权拖期(total weighted tardiness) $\sum w_j T_j$：表示所有工件拖期时间的加权和. 若所有工件的权重都相等，则称为总拖期(total tardiness)，记为 $\sum T_j$.

(7) 加权拖期工件数(weighted number of tardy tasks) $\sum w_j U_j$：定义为 $\sum w_j U_j = \sum_{j=1}^{n} w_j U_j$，其中 $U_j = \begin{cases} 1, & C_j > d_j, \\ 0, & C_j \leqslant d_j. \end{cases}$ 若所有工件的权重都相等，则称为拖期工件数(number of tardy tasks)，记为 $\sum U_j$.

1.3 调度问题的求解方法

求解一个调度问题，首先要从计算复杂性的角度来判断它的难易程度. 计算复杂性是 20 世纪 70 年代在计算机科学中建立起来的理论，从算法的角度定义了问题的“难”和“易”. 在计算复杂性理论中，问题的难易程度是通过其能否由多项式时间算法求解来判定. 若某一问题总能在多项式时间内找到最优解，则称其为 P 问题；否则，称其为 NP 难问题. 对于 NP 难问题，根据其难度的不同，又可分为强 NP 难(strongly NP-hard)和一般 NP 难(ordinarily NP-hard)两类，其中任何强 NP 难问题都不存在伪多项式时间算法.

一方面，若某调度问题是 P 问题，即存在多项式时间算法，则主要任务就是尽可能地寻找那些时间复杂性小的多项式时间最优算法进行求解. 另一方面，若

某调度问题是 NP 难问题，则将该问题划分为小规模问题和大规模问题两种情况来处理.

(1) 对于小规模问题，可以利用分支定界(branch and bound)和动态规划(dynamic programming)算法来求得最优解. 分支定界和动态规划都是基于枚举的算法. 分支定界中的“分支”指的是把可行解集分为互不相交的子集，“定界”是计算目标函数在给定可行子集上的下界(极小化问题)，若该可行子集上的下界大于或等于此时最小的目标函数值，就把该可行子集剪掉，此过程称为剪支，反复利用分支、剪支和定界，直到找出目标函数值最小的可行解. 动态规划是一个多阶段决策过程，它将问题划分为一系列相互联系的阶段，每个阶段都需要决策，且会影响下一阶段的决策，最后从整个过程中，选取达到总体最好的决策.

(2) 对于大规模问题，由于输入规模的增加使得很难在允许的时间内求得最优解，因此研究的重点集中在利用智能优化算法(intelligent optimization)或者构造启发式算法(heuristic)在较短时间求得近似解. 智能优化算法是从问题的任意可行解出发，通过对其邻域的不断搜索和当前解的替换来进行迭代从而实现解的改进. 但是此类算法大多没有理论性能保证. 启发式算法是一种基于直观或经验构造的算法，是多项式时间算法，可以快速地求得问题的可行解. 由于启发式算法构造简单，便于理论分析，所以一般都有理论性能的保证.

同时，对于有些 NP 难问题，还可以给出一系列近似算法$\{A_\varepsilon\}$. 对于任意给定的$\varepsilon > 0$，$\{A_\varepsilon\}$是多项式时间算法，而且$Z(A_\varepsilon) \leqslant (1+\varepsilon)\, Z(\mathrm{OPT})$，这里$Z(A_\varepsilon)$和$Z(\mathrm{OPT})$分别表示算法目标值和问题最优值，则称$\{A_\varepsilon\}$为多项式时间近似策略(polynomial time approximation scheme)，简记为 PTAS. 进一步地，如果$\{A_\varepsilon\}$的时间复杂性是关于输入长度以及$1/\varepsilon$的某个二元多项式，则称其为全多项式时间近似策略(fully polynomial time approximation scheme)，简记为 FPTAS. 目前普遍认为，强 NP 难的问题不存在 FPTAS 算法. 因此对于强 NP 难问题，若可以设计出 PTAS 算法，则认为该问题已经解决；若不知道复杂性的某问题已经设计出了 FPTAS 算法，则此问题至少不是强 NP 难的. 而对一般 NP 难的问题，若已经设计出了 FPTAS 算法，则也认为此类问题已经解决.

1.4 求解调度问题的算法及其性能分析

1.4.1 调度算法

求解调度问题的算法(algorithm)，简称调度算法，是指对某个调度问题的任何一个具体的例子(简称为实例)，依照一个预先制定的执行程序运行后都可

以得到一个可行的排序. 按照对问题输入的了解程度,调度算法可分为两类:

(1) 离线算法:问题的所有输入预先给定,要求一次性对问题做出决策,用来求解离线问题.

(2) 在线算法:输入不是预先给定,而是随时间推移逐步得到的,算法未来可能的输入不可知,在任意时刻,当有输入时,算法必须对输入及时响应,并且算法一旦做出了决策,则在后续过程中不得改变. 可以用来求解在线问题以及带有释放时间的离线问题.

1.4.2 评价算法性能的主要方法

通过前一节的讨论,可知对于那些不存在多项式时间最优算法的 NP 难问题,往往放弃寻求最优解,转而寻找一个可以快速求得的可行解,把它作为最优解的近似值. 因此,自然希望得到的近似解与最优解能够最大可能地接近,以增强算法的有效性. 为了衡量确定性算法得到的近似解与最优解的接近程度,一般可以采用以下几种分析方法.

(1) 最坏情况分析(竞争分析). 研究离线(在线)算法的目标函数值与(离线环境下)问题最优值之间的最大误差,通常采用比值的形式来衡量. 对于极小化问题 π,记 I 是问题的实例,令 Z^A 表示算法 A 对于实例 I 的目标值,Z^* 表示实例 I(在离线环境下)的最优值,则

$$R_A(\pi) = \inf\{\gamma \geqslant 1 \mid \text{对于所有的实例 } I, \text{有 } Z^A(I)/Z^*(I) \leqslant \gamma\}$$

称为算法 A 的最坏性能(竞争)比. 如果可以找到一个实例使上式中的比值取等号,则称该比值是紧的(tight),而且无法对其做进一步改进. 最坏情况分析(竞争分析)主要是根据问题自身的特性,通过不等式的放缩求得算法目标函数值(或者上界)与问题最优值(或者下界)的比值. 通常,每个算法的分析过程都大相径庭. 对于同一个问题的不同算法,最坏情况比(竞争比)不一定相同,而且同一算法对于不同问题,最坏情况比(竞争比)也不一定相等.

对于某个问题的所有实例,上面的分析方法虽然为人们提供了一个衡量算法性能的标准,但是通过研究算法最坏情况比(竞争比)的紧界,可以发现所研究的问题都是一些人为构造、极其特殊的实例,在实际中很少发生. 特别是在工业生产环境中,通常有成千上万个工件在一台或者若干台机器上进行加工,上面提到的最坏情况发生的概率为零. 通过数值实验分析,同样也会看到,一个最坏情况很差的启发式算法,在实际中表现出的性能却非常好. 为了描述算法在求解大规模问题时的性能,可以采用下面的分析方法.

(2) 渐近性能分析(渐近竞争分析),简称渐近分析. 在问题规模趋于无穷大时,研究离线(在线)算法的目标函数值与(离线环境下)问题最优值之间的接近