第 1 章　引　　论

在工厂中, 如何利用已有的机器对要完成的工件 (订单) 给出一个可行的加工方案, 使生产成本 (或利润) 对应的一个目标函数达到最小 (或最大), 是一个排序问题。在管理科学中, 排序也称为调度。在本书中, 我们统一称为排序。对排序理论的研究最早起源于 20 世纪 50 年代人们对制造业中的生产作业研究。随着各种先进机器的发明及应用, 生产能力大幅提高, 工业生产也得到了迅速的发展。在现代的工业生产中, 工件 (订单) 数量急剧增加, 工件的加工条件也各不相同, 这使得找到一个最优解变得非常困难。一个坏的生产方案不仅会使生产成本剧增, 甚至会导致无法在规定的时间内完成所有的工件。机器大规模生产的出现, 促使了更多的工业管理人员和学术研究人员投身于排序理论的研究。因此, 工业生产的进步促进了排序理论研究的迅速发展; 而排序理论的研究和应用也反过来促进了工业生产的发展。

早在 1910 年, Gantt 就提出了 “甘特图”(Gantt chart) 来解决产品制造中出现的排序问题。Johnson (1954) 发表了他的论文 “*Optimal two-and three-stage production schedules with setup times included*”, 这被普遍认为是第一篇有关排序问题研究的论文。在这篇论文中, 作者考虑了最小化最大完工时间的两台机器流水作业排序问题 (例 1.1), 并给出了一个简单而又巧妙的多项式时间最优算法, 即著名的 Johnson 规则。此后, 关于排序研究的论文层出不穷。根据最新的 Web of Science 统计, 标题中含 “scheduling” 并且主题中含 “machine” 的论文超过 2 万篇。Muth 等 (1963) 出版的*Industrial Scheduling*是关于排序问题研究的第一本著作, 搜集了很多关于排序问题的论文。此后, 国际上一些优秀的排序著作还包括 Baker (1974) 的 *Introduction to Sequencing and Scheduling*, Baker 等 (2009) 的 *Principles of Sequencing and Scheduling*, Blazewicz 等 (2007) 的 *Handbook on Scheduling: From Theory to Applications*, Brucker (2001) 的 *Scheduling Algorithms*, Pinedo (1995) 的 *Scheduling Theory, Algorithms and Systems*以及 Pinedo (2005) 的 *Planning and Scheduling in Manufacturing and Services*等。

越民义先生是国内排序理论研究的先行者。越民义等 (1975) 的论文 “n 个零件在 m 台机床上的加工顺序问题” 开创了中国研究排序理论的先河。此外, 越民义 (2001)

出版的《组合优化导论》包含了大量多项式时间可解的排序问题。陈荣秋 (1987) 编著的《排序的原理和方法》是第一本正式出版的排序论专著。此后, 国内比较优秀的排序著作还包括林诒勋 (1997) 编著的《动态规划与序贯最优化》、唐恒永等 (2002) 出版的《排序引论》、唐国春等 (2003) 出版的《现代排序论》和万国华 (2019) 编著的《排序与调度的理论、模型和算法》等。

本章共分为 4 节。其中, 1.1 节分别从问题背景、定义与符号、研究内容三个方面介绍排序问题; 1.2 节主要介绍了一些单目标排序问题的经典结果; 1.3 节主要介绍了一些多 (双) 目标排序问题的模型; 1.4 节主要介绍了求解多目标排序问题的一些常用研究方法。

1.1 排序问题介绍

1.1.1 问题背景

假设有 m 台机器 $M_1, M_2, \cdots, M_m$ 和 n 个工件 $J_1, J_2, \cdots, J_n$, 生产者需要在一定的约束条件下将这些工件安排在机器上加工, 从而使得某个预定指标达到最优, 这就是经典的机器排序问题。随着社会的不断发展, 人们发现排序理论已经被广泛应用到制造业、服务业、计算机网络、物流运输等许多其他领域。因此, 排序理论中的“机器” 和 “工件” 已经不仅仅局限于工厂里的机器和工件。这里 “机器” 可以是数控机床、计算机中央处理器 (CPU)、医生、机场跑道等, “工件” 也可以是零件、计算机终端、病人、起飞或降落的飞机等。下面, 给出排序理论应用在不同行业中的几个例子。

例 1.1 (见文献 (Johnson, 1954)) n 个工件 $J_1, J_2, \cdots, J_n$ 要在两台机器 M_1、M_2 上加工, 每个工件 J_j 有两道不同的工序 O_{1j} 和 O_{2j} 且对应的加工时间分别为 p_{1j} 和 p_{2j}。工序 O_{1j} 首先在机器 M_1 上加工, 工序 O_{2j} 必须在 O_{1j} 完工之后才能在机器 M_2 上加工, 所有的工件按照相同的顺序分别在两台机器上加工。工序 O_{2j} 的完工时间被定义为工件 J_j 的完工时间, 追求的目标是最小化工件的最大完工时间。这是一个经典的两台机器流水作业排序问题。

例 1.2 (见文献 (Lee et al., 1992)) 若干电子芯片被成批装入一个有容量限制的烤箱中进行加热以检验它们的耐热能力。每个电子芯片的耐热时间是不一样的, 直到一批中所有的电子芯片都被烧断, 才能接着去检验下一批电子芯片。因此, 一批芯片的检验时间等于该批中芯片的最长耐热时间, 追求的目标是使用最短的时间检验完所有的电子芯片。这里电子芯片是工件, 烤箱为机器, 烤炉容量为限制条件。这就是一个有界的平行批机器排序问题。

例 1.3 (见文献 (唐恒永等, 2002))　假设某个机场有若干条不同长度的跑道, 每天有上百架飞机起飞和降落。不同飞机起飞和降落所需要的滑行距离是不同的, 每架飞机只能使用长度比其滑行距离长的跑道。一般来说, 大型飞机只能选择较长的跑道起飞和降落, 而小型飞机则有更多可供选择的跑道。如果某个飞机在它预定时间之后起飞或者降落, 则称该飞机晚点。追求的目标是最小化晚点飞机的数量, 这也是一个排序问题。在这个问题中, 跑道是机器, 飞机为工件, 飞机所需跑道长度和预定时间为限制条件。

现代社会是一个竞争的社会, 企业之间的商业竞争尤其激烈。为了提高企业的市场竞争力, 良好的生产计划和有效的成本控制已经成为企业之间竞争的关键因素。在这种情况下, 企业决策者必须设计一个好的生产计划 (即排序), 从而能够有效地利用有限的资源并尽早生产出具有竞争性的产品以满足顾客的需求。

1.1.2　定义和符号

Graham 等 (1979) 提出了排序问题中的三参数表示方法, 即使用 $\alpha|\beta|\gamma$ 来表示排序问题。其中, α 区域代表机器的环境、类型与种类; β 区域代表对工件的约束以及工件具有的一些特征; γ 区域代表我们所需要的目标函数。下面给出排序问题中常用的定义和符号。

(1) α 区域参数

- 1：单机, 表示加工生产过程中只有一台机器并且机器一次只能加工一个工件。
- Pm：m 台同速机 (或者同型机), 即有 m 台平行的 (功能一样的) 机器 $M_1, M_2, \cdots, M_m$, 并且每台机器 M_i 的加工速度 s_i 相同。不失一般性, 假设 $s_1 = s_2 = \cdots = s_m = 1$。
- Qm：m 台恒速机 (或者同类机), 即有 m 台机器 $M_1, M_2, \cdots, M_m$, 并且每台机器 M_i 的加工速度 s_i 是不变的。不失一般性, 假设 $s_1 \geqslant s_2 \geqslant \cdots \geqslant s_m$。
- Rm：m 台变速机 (或者非同类机), 即有 m 台机器 $M_1, M_2, \cdots, M_m$, 并且每台机器 M_i 的加工速度和要加工的工件 J_j 有关。
- Fm：m 台流水作业机器, 即每个工件 J_j 在机器 M_i 上有一道工序 O_{ij}, 这些工序必须依次在 $M_1, M_2, \cdots, M_m$ 上进行加工。
- Om：m 台自由作业机器, 即每个工件 J_j 在机器 M_i 上有一道工序 O_{ij}, 这些工序在机器 $M_1, M_2, \cdots, M_m$ 上加工的次序是自由的。
- Jm：m 台异序作业机器, 即每个工件 J_j 在机器 M_i 上有一道工序 O_{ij}, 这些工序在机器 $M_1, M_2, \cdots, M_m$ 上加工的次序是不同的。

(2) β 区域参数

- p_j：表示工件 J_j 的加工时间。
- p_{ij}：表示工件 J_j 在机器 M_i 上的加工时间。
- w_j：表示工件 J_j 的权重。
- r_j：表示工件 J_j 的到达时间 (就绪时间), 工件必须在 r_j 时刻或者在 r_j 时刻之后才能被加工。
- d_j：表示工件 J_j 的工期。
- $\overline{d}_j$：表示工件 J_j 的截止工期, 工件必须在 $\overline{d}_j$ 时刻或者在 $\overline{d}_j$ 时刻之前完工。
- $d_{[i]}$：表示第 i 个位置上工件的工期。
- e_j：表示工件 J_j 的拒绝费用。
- q_j：表示工件 J_j 的运输时间。
- pmtn：工件允许中断抢先加工。
- prec：工件之间有序约束。
- p-batch：平行批, 表示机器可以把多个工件作为一批同时进行加工, 并且该批的加工时间等于该批中最长工件的加工时间。
- s-batch：继列批, 表示机器可以把多个工件作为一批同时进行加工, 并且该批的加工时间等于该批中所有工件的加工时间之和。
- $c \geqslant 1$：批容量, 即每一个加工批中至多能包含 c 个工件。
- $s > 0$：批安装 (设置) 时间, 即每个加工批在加工之前必须有一个安装 (设置) 时间 s。

(3) γ 区域参数

- C_j：表示工件 J_j 的完工时间。
- $D_j = C_j + q_j$：表示工件 J_j 的运输完工时间。
- $L_j = C_j - d_j$：表示工件 J_j 的延迟时间。
- $T_j = \max\{0, C_j - d_j\}$：表示工件 J_j 的误工时间。
- $E_j = \max\{0, d_j - C_j\}$：表示工件 J_j 的提前时间。
- U_j：工件 J_j 的误工计数。$U_j = 1$ 表示工件 J_j 误工; $U_j = 0$ 表示工件 J_j 不误工。
- $C_{\max}$：表示工件的最大完工时间, 即 $C_{\max} = \max\{C_j : 1 \leqslant j \leqslant n\}$。
- $D_{\max}$：表示工件的最大运输完工时间, 即 $D_{\max} = \max\{D_j : 1 \leqslant j \leqslant n\}$。
- $L_{\max}$：表示工件的最大延误时间, 即 $L_{\max} = \max\{L_j : 1 \leqslant j \leqslant n\}$。
- $E_{\max}$：表示工件的最大提前时间, 即 $E_{\max} = \max\{E_j : 1 \leqslant j \leqslant n\}$。
- $\sum C_j$：表示工件的总完工时间, 即 $\sum C_j = \sum_{j=1}^{n} C_j$。

- $\sum w_jC_j$：表示工件的总加权完工时间, 即 $\sum w_jC_j = \sum_{j=1}^{n} w_jC_j$。
- $\sum T_j$：表示工件的总误工时间, 即 $\sum T_j = \sum_{j=1}^{n} T_j$。
- $\sum w_jT_j$：表示工件的总加权误工时间, 即 $\sum w_jT_j = \sum_{j=1}^{n} w_jT_j$。
- $\sum(E_j + T_j)$：表示工件总的误工时间与提前时间, 即 $\sum(E_j + T_j) = \sum_{j=1}^{n}(E_j + T_j)$。
- $\sum U_j$：表示误工工件的总个数, 即 $\sum U_j = \sum_{j=1}^{n} U_j$。
- $\sum w_jU_j$：表示工件的总加权误工个数, 即 $\sum w_jU_j = \sum_{j=1}^{n} w_jU_j$。

1.1.3　研究内容

对一个排序问题来说, 找到一个最优解 (即最优排序) 是至关重要的。然而, 在众多的可行解中找到最优解可能并不容易。比如, 对单机排序问题 $1||\sum C_j$ 来说, n 个工件的每一个排列都是一个可行解。因此, 该问题至少有 $n!$ 个不同的可行解。同样, 对问题 $P2||C_{\max}$, 每个工件都有两种可能, 即要么在机器 M_1 上加工, 要么在机器 M_2 上加工。因此, 即使不考虑同一台机器上工件的加工顺序, 该问题也至少有 2^n 个不同的可行解。在这种情形下, 尽管枚举所有的可行解也能找到一个最优解, 然而花费的时间特别长。当工件数 n 较大时, 即使使用当前最快的计算机, 枚举完所有的可行解可能也需要几十年甚至上百年时间。因此, 枚举并不是一个好的算法。

那么, 什么样的算法是一个好的算法呢? Cobham (1964) 和 Edmonds (1965) 建议, 一个好的、有效的算法应该是一个多项式时间算法。也就是说, 存在一个整数 k, 使得算法在 $O(n^k)$ 时间内完成求解。例如, 针对单机排序问题 $1||\sum C_j$, Smith (1955) 提出的最短处理时间优先 (shortest processing time first, SPT) 规则 (按处理时间从小到大进行排列) 可以在 $O(n\log n)$ 时间内找到最优解, 计算机只需要很短时间就能完成。

然而, 不幸的是, 对于大多数排序问题, 很难找到一个多项式时间的最优算法。因此, Karp (1972) 提出了 NP-完全的和 NP-困难的概念。给定一个判定问题, 如果 NP 类中的所有问题都可以在多项式时间内转化为该问题的一个实例, 就称这个问题是 NP-完全的。对一个排序问题, 如果一个 NP-完全的判定问题能够在多项式时间内归

结为该问题的一个实例, 那么就称该排序问题是 NP-困难的。为了证明一个排序问题是 NP-困难的, 通常将一个 NP-完全的判定问题多项式转化 (归结) 为该排序问题的判定形式。在证明一个排序问题是一般 (二元)NP-困难时, 通常用 “划分、奇偶划分、等规模划分、子集和问题、背包问题” 等问题进行归结; 在证明一个排序问题是强 (一元)NP-困难时, 通常用 “3-划分、可满足性问题、三维匹配、图的点覆盖问题、图的独立集问题、装箱问题、旅行商问题” 等进行归结。

对一个 NP-困难的排序问题, 找到一个多项式时间的最优算法是非常困难的。但是, 如果降低一些要求, 寻找一个拟多项式时间算法, 或者一个多项式时间的近似算法, 这都是有可能的。寻找一个比较有效的近似算法和拟多项式时间算法, 已经成为当前排序论研究中最活跃、最主要的一个方向。下面将介绍有关拟多项式时间算法和近似算法的一些定义。

设 P 是一个具有非负权的排序问题, 且每个实例 I 的元素都是整数。设 $\mathrm{size}(I)$ 为实例 I 中元素的个数且 $\mathrm{largest}(I)$ 为其中最大的整数, 如果算法 A 的运行时间为 $\mathrm{size}(I)$ 和 $\mathrm{largest}(I)$ 的一个多项式, 则称算法 A 是 P 的一个拟多项式时间算法。

设 P 是一个具有非负权的排序问题并且 $k \geqslant 1$, 如果 P 的一个多项式时间算法 A, 对 P 的每一个实例 I 都有 $A(I) \leqslant k\mathrm{OPT}(I)$, 则称多项式时间算法 A 是排序问题 P 的一个 k-近似算法。使得 $A(I) \leqslant k\mathrm{OPT}(I)$ 成立的最小的 k, 就称为算法 A 的近似比。

对任何一个固定的 $\epsilon > 0$, 如果算法 A_ϵ 为问题 P 的一个 $(1+\epsilon)$-近似算法, 则称算法 A_ϵ 是 P 的一个多项式时间近似方案 (polynomial time approximation scheme, PTAS)。特别地, 如果 A_ϵ 的运行时间为 $\mathrm{size}(I)$ 和 $\dfrac{1}{\epsilon}$ 的一个多项式, 则称算法 A_ϵ 是 P 的一个完全多项式时间近似方案 (fully polynomial time approximation scheme, FPTAS)。

在一个排序问题中, 如果所有的工件信息都是事先已知的, 则称这个问题为一个 “离线” 排序问题。对一个离线排序问题来说, 一般主要从三个方面进行研究: ①给出一个多项式时间算法; ②证明该问题是 (一般或者强) NP-困难的; ③对那些 NP-困难的问题, 给出一个近似算法、拟多项式时间算法或者多项式时间近似方案。

还有一种与 “离线” 排序问题相对应的问题称为 “在线” 排序问题。在在线排序模型中, 工件的信息事先不知道, 只有在它到达或者加工完毕才知道。目前, 人们主要研究两种在线排序模型。第一种模型是 “按列表在线”, 假设工件按照列表顺序依次到达, 在线算法必须立即安排当前到达的工件然后列表中的下一个工件才会出现; 第二种模型是 “按时间在线”, 假设每一个工件有一个到达时间, 工件的所有信息只有在它到达之后才被释放。除了到达时间, 更主要的区别在于, 当一个工件到达后在线算法不需要立即安排当前的工件, 可以等待一段时间再做出决定。除此之外, 还有一种排序模型介

于离线排序和在线排序之间, 即一部分信息是已知的而另一部分是未知的。这种模型称为半在线排序问题。

在线排序问题的算法称为 “在线算法”, 在线算法的近似比也称为 “竞争比”。由于信息的缺乏, 这个竞争比往往有一个下界。因此, 对一个给定的在线排序问题, 主要的研究方法有两个方面: 首先, 采用对手法获得在线算法竞争比的一个下界; 其次, 利用最优值下界估计、SPT 规则、EDD 规则、列表算法和延迟算法等, 设计一个有效的在线算法。如果一个在线算法的竞争比等于该问题竞争比的下界, 则称该在线算法是最好可能的。

1.2　单目标排序问题介绍

在早期的经典排序中, 往往只有一个目标函数, 这些问题也被称为单目标排序问题。Johnson (1954) 研究了排序问题 $F2||C_{\max}$, 并给出了著名的**Johnson 规则**: 首先将 n 个工件 $J_1, J_2, \cdots, J_n$ 分为两个集合 $A_1 = \{J_j : p_{1j} < p_{2j}\}$ 与 $A_2 = \{J_j : p_{1j} \geqslant p_{2j}\}$, 工件在两台机器上均按照先 A_1 后 A_2 的顺序加工, 其中 A_1 中的工件按照 p_{1j} 的非减顺序加工, 而 A_2 中的工件按照 p_{2j} 的非增顺序加工。Johnson 规则可以在 $O(n\log n)$ 时间内得到问题 $F2||C_{\max}$ 的一个最优排序。自此以后, 有关排序问题的研究得到了迅速的发展。下面给出单目标排序问题的一些经典结果。

对于单机排序问题 $1|r_j|C_{\max}$, Lawler (1973) 证明了**最早到达时间优先加工 (earliest release date first, ERD) 规则**在 $O(n\log n)$ 时间内得到该问题的一个最优排序, 即工件按照到达时间 r_j 非减顺序进行加工。对于排序问题 $1||L_{\max}$, Jackson (1955) 证明了**最早工期优先加工 (earliest due date first, EDD) 规则**在 $O(n\log n)$ 时间内得到该问题的一个最优排序, 即工件按照工期 d_j 非减顺序进行加工。对于排序问题 $1||\sum w_jC_j$, Smith (1955) 证明了**加权最短加工时间优先 (weighted shortest processing time first, WSPT) 规则**在 $O(n\log n)$ 时间内得到该问题的一个最优排序, 即工件按照加工时间和权重比值 $\dfrac{p_j}{w_j}$ 的非减顺序进行加工。当所有的权重 $w_j = 1$ 时, **WSPT 规则**等价于 **SPT 规则**, 即工件按照加工时间 p_j 非减顺序进行加工。Smith (1955) 证明了 **SPT 规则**在 $O(n\log n)$ 时间内得到问题$1||\sum C_j$ 和 $P||\sum C_j$ 的一个最优排序。Lawler (1973) 也证明了 **Lawler 规则**在 $O(n^2)$ 时间内得到问题 $1|\text{prec}|f_{\max}$ 的一个最优排序, 即从最后一个位置开始, 从当前可用的工件中挑选一个放在当前最后位置且使得目标函数值最小的工件进行加工。对于问题 $1||\sum U_j$, Moore (1968) 给出了一个 $O(n\log n)$ 时间的最优算法: 先把所有工件按 EDD 规则排好顺序, 即 $d_1 \leqslant d_2 \cdots \leqslant d_n$。算法从 $k = 1$ 开始, 一共迭代 n 次。令 S_k 是第 k 次迭代之前可以按时完工的工件集合并且 $S_1 = \varnothing$, 安排当前的工件 J_j 到当前工件集合 S 末尾进

行加工。如果 J_k 没有误工, 则令 $k=k+1$ 并且 $S_{k+1}=S_k\bigcup J_k$; 否则从 $S_k\bigcup J_k$ 选出加工时间最长的工件 J_j, 则令 $k=k+1$ 并且 $S_{k+1}=(S_k\bigcup J_k)\setminus J_j$。该算法也称为 **Moore 算法**。对于问题 $1||\sum w_jU_j$, Karp (1972) 证明了该问题是一般 NP-困难的并且 Sahni (1976) 给出了一个拟多项式时间的动态规划算法。对于问题 $1||\sum T_j$, Du 等 (1990) 证明了该问题是一般 NP-困难的并且 Lawler (1977; 1982) 分别给出了一个拟多项式时间最优算法和一个全多项式时间近似方案。Karp (1972) 证明了问题 $1||\sum w_jT_j$ 是强 NP-困难的。

对于平行机排序问题 $Pm||C_{\max}$, Garey 等 (1979) 证明了该问题是强 NP-困难的。对于该问题, Graham (1966) 证明了**列表算法**的近似比为 $2-\dfrac{1}{m}$, 即当有机器空闲时, 当前列表中的第一个工件被排在负载量最小的机器上进行加工。进一步, Graham (1969) 证明了**最长加工时间优先 (longest processing time first, LPT) 算法**有更好的近似比 $\dfrac{4}{3}-\dfrac{1}{3m}$, 即首先把工件按照加工时间 p_j 非增顺序进行排列形成一个列表, 然后再使用列表算法安排工件进行加工。Gonzalez 等 (1976) 对问题 $O2||C_{\max}$ 给出了一个多项式时间的最优算法, 他们也进一步证明了

$$C^*_{\max}=\max\Big\{\sum_{j=1}^{n}p_{1,j},\sum_{j=1}^{n}p_{2,j},\max_{1\leqslant j\leqslant n}(p_{1,j}+p_{2,j})\Big\}$$

当 $m\geqslant 3$ 时, 问题 $Fm||C_{\max}$ 和 $Om||C_{\max}$ 都是 NP-困难的。与流水作业排序和自由作业排序相比, 异序作业排序更难, 甚至问题 $J2||C_{\max}$ 都是 NP-困难的。

1.3 多目标排序问题介绍

在生产管理中, 决策者经常要同时对多个目标函数进行优化。因此, 必须同时对多个目标进行均衡考虑。目前, 有关多目标排序的论文多达几千篇, Lee 等 (1993), Hoogeveen (2005), T'kindt 等 (2006) 综述了大量的研究成果。

在多目标排序中, 假设有 $k\geqslant 2$ 个需要最小化的目标函数 $f^{(1)},f^{(2)},\cdots,f^{(k)}$, 可行排序 π 的目标向量记为 $(f^{(1)}(\pi),f^{(2)}(\pi),\cdots,f^{(k)}(\pi))$。若不存在其他可行排序 σ, 使得 $(f^{(1)}(\sigma),f^{(2)}(\sigma),\cdots,f^{(k)}(\sigma))\leqslant(f^{(1)}(\pi),f^{(2)}(\pi),\cdots,f^{(k)}(\pi))$ 并且 $(f^{(1)}(\sigma),f^{(2)}(\sigma),\cdots,f^{(k)}(\sigma))\neq(f^{(1)}(\pi),f^{(2)}(\pi),\cdots,f^{(k)}(\pi))$, 就称 π 是一个 Pareto 最优排序, 并称 $(f^{(1)}(\pi),f^{(2)}(\pi),\cdots,f^{(k)}(\pi))$ 为对应于 π 的 Pareto 最优点。针对 k 个目标函数 $f^{(1)},f^{(2)},\cdots,f^{(k)}$, 多目标排序的四个模型可以表示如下。

(1) $\alpha|\beta|\mathrm{Lex}(f^{(1)},f^{(2)},\cdots,f^{(k)})$ (分层目标优化排序模型)

"分层目标优化排序模型" 在实际应用中适用于多个目标的重要性程度有明显区别的情形; 此时人们将目标按照重要性程度先后排列, 而最优排序必须首先使得第一

目标 $f^{(1)}$ 达到最优, 在此前提下使得第二目标 $f^{(2)}$ 达到最优, 并依次类推。

(2) $\alpha|\beta|f^{(1)}: f^{(i)} \leqslant Q_i, 2 \leqslant i \leqslant k$ (约束目标优化排序模型)

"约束目标优化排序模型" 在实际应用中适用于多个辅助目标被限制为不超过费用预算的情形, 即在 $f^{(i)} \leqslant Q_i$ 的前提下寻求最优排序使得目标 $f^{(1)}$ 达到最优。

(3) $\alpha|\beta|\lambda_1 f^{(1)} + \lambda_2 f^{(2)} + \cdots + \lambda_k f^{(k)}$ (正组合目标优化排序模型)

"正组合目标优化排序模型" 在实际应用中适用于多个目标的权重在整体费用目标中的比例可以预测的情形, 最优排序将使得多个目标的加权和达到最优。

(4) $\alpha|\beta|\#(f^{(1)}, f^{(2)}, \cdots, f^{(k)})$(Pareto 最优化排序模型)

"Pareto 最优化排序模型" 在实际应用中适用于多个目标的重要性及权重不可区分的情形; 当生成所有 Pareto 最优点后, 管理者可依据实际需求选择理想作业。通常, Pareto 最优点的数目是有限的。当工件允许中断或拆分时, 那么 Pareto 最优点的数目可能变成无限多个。Hoogeveen (1996) 称包含所有 Pareto 最优点的曲线为**均衡曲线**。

Pareto 最优化排序, 也称为折衷排序, 具有理论难度大而应用性强的特点。当排序目标给定时, Pareto 最优化排序模型的求解会蕴含其他三个模型的求解, 而其他三个模型的求解也对 Pareto 最优化排序的求解构成支持, 有时甚至起着不可或缺的作用。此外, 前三个模型也分别有独立的应用背景, 而且在多数情形下, 相比 Pareto 最优化排序模型的求解, 前三个模型的直接求解在时间计算复杂性上会更为有效。因此, 多目标排序研究具有重要的理论意义和广阔的应用前景。

在实际应用中, 两个目标的排序问题最为常见, 因而文献中也以双目标排序的研究居多。此时, 对两个目标 f 和 g 而言, 上述四个模型可以分别表示为: $\alpha|\beta|\mathrm{Lex}(f,g)$, $\alpha|\beta|f: g \leqslant Q$, $\alpha|\beta|\lambda_1 f + \lambda_2 g$ 和 $\alpha|\beta|\#(f,g)$。为了方便叙述, 我们将关于两个目标 f 和 g 的上述四个模型统一记为 $\alpha|\beta|(f,g)$。

对问题 $1|\bar{d}_j|\sum C_j$ (与 $1||\sum C_j : L_{\max} \leqslant 0$ 等价), Smith (1955) 给出了一个 $O(n\log n)$ 时间的最优算法。该算法每次选择最长的可排工件排在最后位置。Hoogeveen 等 (1995) 首次对折衷排序 $1||\#(\sum C_j, f_{\max})$ 给出了多项式时间算法。Hoogeveen (1996) 研究了问题 $1||\#(f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)})$, 对一般情形给出了强 NP-困难性证明, 对 $m = 2, 3$ 的情形给出了多项式时间算法。后人的工作中大量沿用了两篇文献 (Hoogeveen et al., 1995; Hoogeveen, 1996) 的基本研究方法。Huo 等 (2007) 证明问题 $1||\sum C_j : \sum U_j \leqslant Q$ 和 $1||\sum T_j : \sum U_j \leqslant Q$ 是一般 NP-困难的。

在近 30 年来, 也出现了很多新的双目标排序模型, 例如加工时间可控排序、工件可拒绝排序、重新排序和多代理排序等。特别地, 多代理排序也是一种新型的多目标排序。在该模型中多个竞争的代理使用共同的机器对各自的工件进行加工, 每个代理都有一个目标函数仅仅对应各自的工件。Agnetis 等 (2014) 的专著对该领域的研究进行了全面的综述。

1.4 求解多目标排序问题的常用方法

1.4.1 最优算法设计

在实际应用中, 两个目标的排序问题最为常见, 文献中也以双目标排序的研究居多。因此, 在这里只考虑双目标排序问题。显然, 对于问题 $\alpha|\beta|f:g\leqslant Q$ 和 $\alpha|\beta|\lambda_1 f+\lambda_2 g$, 可以把它们看成一个新的单目标排序问题来求解。对于问题 $\alpha|\beta|\mathrm{Lex}(f,g)$, 可以先求出单目标问题 $\alpha|\beta|f$ 的最优解 f^*, 剩下的问题就可以转化为 $\alpha|\beta|g:f\leqslant f^*$, 也可以看成一个单目标问题来求解。这里主要介绍的问题是 Pareto 优化问题 $\alpha|\beta|\#(f,g)$, 下面给出求 $\alpha|\beta|\#(f,g)$ 的 Pareto 最优排序的两种典型方法。

1. ϵ-约束方法

该方法的求解步骤如下:

(1) 解分层优化问题 $\alpha|\beta|\mathrm{Lex}(f,g)$, 产生第 1 个 Pareto 最优序 σ_1 及对应的第 1 个 Pareto 最优点 $(x^1,y^1)=(f(\sigma_1),g(\sigma_1))$。

(2) 如果第 i 个 Pareto 最优序 σ_i 及它对应的 Pareto 最优点 (x^i,y^i) 已求出, 则解约束问题 $\alpha|\beta|f:g<y^i$(如果 g 取整数, 则解约束问题 $\alpha|\beta|f:g\leqslant y^i-1$)。

(3) 如果上述约束问题无解, 则得到所有的 Pareto 最优序 $\sigma_1,\sigma_2,\cdots,\sigma_i$, 算法停止。否则, 假设它的最优值为 x^{i+1}, 然后解约束问题 $\alpha|\beta|g:f\leqslant x^{i+1}$, 得到最优值 y^{i+1}, 从而得到它的第 $i+1$ 个 Pareto 最优序 σ_{i+1}, 以及它所对应的第 $i+1$ 个 Pareto 最优点 $(x^{i+1},y^{i+1})=(f(\sigma_{i+1}),g(\sigma_{i+1}))$; 置 $i:=i+1$, 并转到 (2)。

为了利用 ϵ-约束方法, 必须能同时解决两个约束问题 $\alpha|\beta|g:f\leqslant x$ 与 $\alpha|\beta|f:g\leqslant y$。对这两个问题, 如果只有其中的一个可有效解决, 比如 $\alpha|\beta|f:g\leqslant y$, 那么 ϵ-约束方法不再有效。但是只要对它进行一些修改, 仍可解决问题 $\alpha|\beta|\#(f,g)$。这就是所谓的单边枚举法。

2. 单边枚举法

不失一般性, 下面假设 $\alpha|\beta|f:g\leqslant y$ 可有效解决, 该方法的求解步骤如下:

(1) 解问题 $\alpha|\beta|f$, 得到最优序 π_1 及点 $(x^1,y^1)=(f(\pi_1),g(\pi_1))$。

(2) 如果第 i 个点 (x^i,y^i) 已求出, 则解约束问题 $\alpha|\beta|f:g<y^i$ (如果 g 取整数, 则解约束问题 $\alpha|\beta|f:g\leqslant y^i-1$)。

(3) 如果上述约束问题无解, 则转到 (4)。否则, 假设它的最优序为 π^{i+1} 及对应的 Pareto 最优点 $(x^{i+1},y^{i+1})=(f(\pi_{i+1}),g(\pi_{i+1}))$; 置 $i:=i+1$, 并转到 (2)。

(4) 删除所有的弱 Pareto 最优点, 剩余的点即全部的 Pareto 最优点, 算法结束。

注 ①仅当 Pareto 最优点的个数有限时, 这种算法才有效。