

加工顺序问题

加工顺序问题

- 题目背景
- 符号定义
- 状态转移方程及详解
- 标记函数



题目背景

- 有 n 个工件需要在机器 M_1 和 M_2 上加工
- 每个工件都先在 M_1 上加工，然后在 M_2 上加工
- A_i , B_i 分别表示工件 i 在 M_1 和 M_2 上所需的加工时间
- 安排工件的加工顺序，使得从第一个工件在 M_1 上加工开始到最后一个工件在 M_2 上加工完成，所需的总加工时间最短

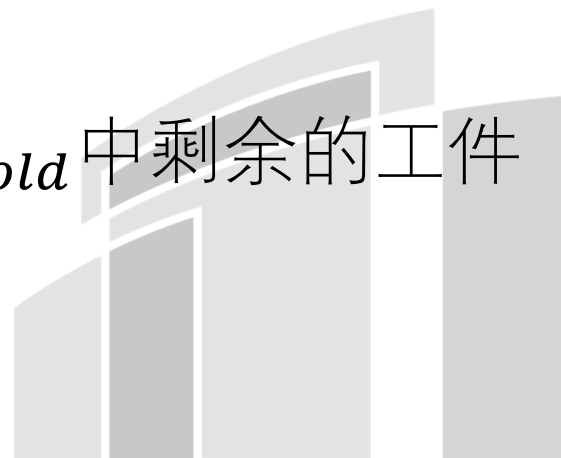
符号定义

符号	意义
T	动态规划数组
A	工件在 M_1 上加工所需的时间数组
B	工件在 M_2 上加工所需的时间数组
S	机器 M_1 未加工的工件集合
U	工件的全集
t	机器 M_2 的剩余运行时间

题设

		S_{old}
↓		
		T_{old}

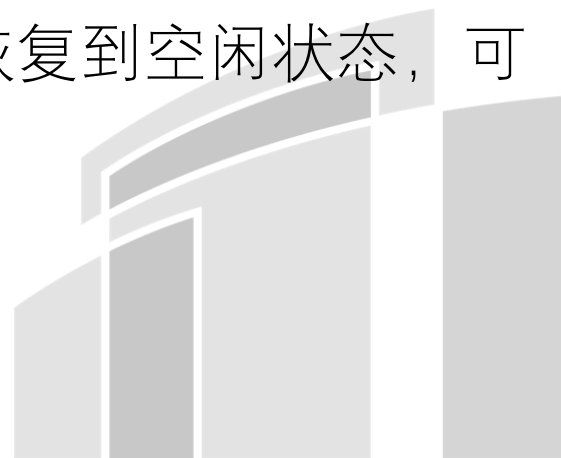
- 设集合 S_{old} 为机器 M_1 当前未加工的工件集合。
- 设集合 $U = \{1, 2, 3, \dots, n\}$ 为工件的全集。
- 初始时 $S_{old} = U$ 。
- 截至当前，两台机器加工集合 S_{old} 中剩余的工件还需花费时间 T_{old} 秒。



机器 M_1

		S_{old}
↓		
		T_{old}

- 显然，为了使加工总时间最短，机器 M_1 不能有空闲时间。
- 因此，一旦机器 M_1 空闲，立即从未加工的工件集合 S_{old} 中选取一工件 i 让机器 M_1 开始加工。当工件 i 完成加工后，时间又过去了 A_i 秒，此时机器 M_1 又恢复到空闲状态，可以开始加工下一个工件。



机器 M_1

S_{new}	$S_{\text{new}} = S_{\text{old}} - \{i\}$	S_{old}
↓		
T_{new}	$T_{\text{new}} = T_{\text{old}} - A_i$	T_{old}

- 显然，为了使加工总时间最短，机器 M_1 不能有空闲时间。
- 因此，一旦机器 M_1 空闲，立即从未加工的工件集合 S_{old} 中选取一工件 i 让机器 M_1 开始加工。当工件 i 完成加工后，时间又过去了 A_i 秒，此时机器 M_1 又恢复到空闲状态，可以开始加工下一个工件。
- 此时机器 M_1 未加工的工件集合变为 $S_{\text{new}} = S_{\text{old}} - \{i\}$ ，加工集合 S_{new} 中的工件还需花费时间 $T_{\text{new}} = T_{\text{old}} - A_i$ 秒。

机器 $M_2(1)$

S_{new}	$S_{\text{new}} = S_{\text{old}} - \{i\}$	S_{old}
t_{new}		t_{old}
		t_{old}
↓		
T_{new}	$T_{\text{new}} = T_{\text{old}} - A_i$	T_{old}

- 机器 M_1 恰开始加工工件 i 的 T_{old} 时刻，机器 M_2 可能正在加工工件，也有可能处于空闲状态。设此时机器 M_2 还需加工 t_{old} 秒到达空闲状态。若 M_2 已经处于空闲，令 $t_{\text{old}} = 0$ 。
- 机器 M_1 完成加工工件 i 的 T_{new} 时刻，机器 M_2 的剩余运行时间由于工件 i 加入 M_2 的任务队列中而产生了变化。定义工件 i 在机器 M_1 上完成加工后，机器 M_2 还需加工 t_{new} 秒到达空闲状态。

机器 $M_2(2)$

S_{new}	$S_{\text{new}} = S_{\text{old}} - \{i\}$	S_{old}
t_{new}		
↓		
T_{new}	$T_{\text{new}} = T_{\text{old}} - A_i$	T_{old}

- 对 t_{new} 的取值，分如下两种情况讨论：



机器 $M_2(2)$

S_{new}	$S_{\text{new}} = S_{\text{old}} - \{i\}$	S_{old}
t_{new}	$t_{\text{new}} = B_i$	$t_{\text{old}}, t_{\text{old}} \leq A_i$
↓		
T_{new}	$T_{\text{new}} = T_{\text{old}} - A_i$	T_{old}

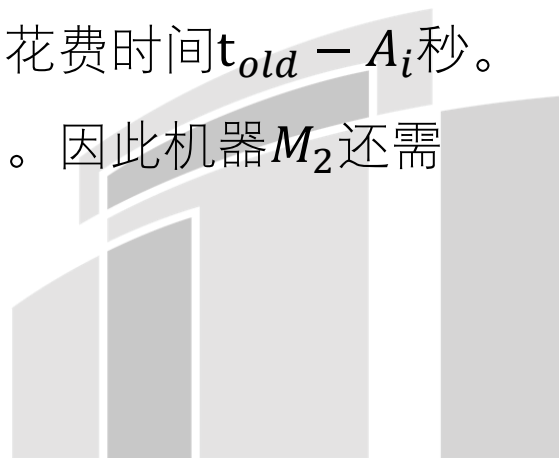
- 对 t_{new} 的取值，分如下两种情况讨论：
- 当 $t_{\text{old}} \leq A_i$ 时，即经过了 A_i 秒后，机器 M_2 处于空闲状态。此时，机器 M_2 只有任务 i 等待加工，故只需 $t_{\text{new}} = B_i$ 秒即可到达空闲状态。



机器 $M_2(2)$

S_{new}	$S_{new} = S_{old} - \{i\}$	S_{old}
t_{new}	$t_{new} = B_i$	$t_{old}, t_{old} \leq A_i$
	$t_{new} = t_{old} - A_i + B_i$	$t_{old}, t_{old} > A_i$
↓		
T_{new}	$T_{new} = T_{old} - A_i$	T_{old}

- 对 t_{new} 的取值，分如下两种情况讨论：
- 当 $t_{old} \leq A_i$ 时，即经过了 A_i 秒后，机器 M_2 处于空闲状态。此时，机器 M_2 只有任务 i 等待加工，故只需 $t_{new} = B_i$ 秒即可到达空闲状态。
- 当 $t_{old} > A_i$ 时，即经过了 A_i 秒后，机器 M_2 仍正在加工 $U - S_{new}$ 集合中的任务。对于该集合的任务，机器 M_2 还需花费时间 $t_{old} - A_i$ 秒。又多了一个新任务 i ，额外多出 B_i 的加工时间。因此机器 M_2 还需 $t_{new} = t_{old} - A_i + B_i$ 秒。



机器 $M_2(2)$

S_{new}	$S_{new} = S_{old} - \{i\}$	S_{old}
t_{new}	$t_{new} = B_i$	$t_{old}, t_{old} \leq A_i$
	$t_{new} = t_{old} - A_i + B_i$	$t_{old}, t_{old} > A_i$
↓		
T_{new}	$T_{new} = T_{old} - A_i$	T_{old}

- 对 t_{new} 的取值，分如下两种情况讨论：
- 当 $t_{old} \leq A_i$ 时，即经过了 A_i 秒后，机器 M_2 处于空闲状态。此时，机器 M_2 只有任务 i 等待加工，故只需 $t_{new} = B_i$ 秒即可到达空闲状态。
- 当 $t_{old} > A_i$ 时，即经过了 A_i 秒后，机器 M_2 仍正在加工 $U - S_{new}$ 集合中的任务。对于该集合的任务，机器 M_2 还需花费时间 $t_{old} - A_i$ 秒。又多了一个新任务 i ，额外多出 B_i 的加工时间。因此机器 M_2 还需 $t_{new} = t_{old} - A_i + B_i$ 秒。
- 因此综合两种情况，可得工件 i 完成加工后，机器 M_2 还需加工 $t_{new} = \max\{t_{old} - A_i, 0\} + B_i$ 秒。

转移公式的引入 (1)

S_{new}	$S_{new} = S_{old} - \{i\}$	S_{old}
t_{new}	$t_{new} = B_i$	$t_{old}, t_{old} \leq A_i$
	$t_{new} = t_{old} - A_i + B_i$	$t_{old}, t_{old} > A_i$
\downarrow		
T_{new}	$T_{new} = T_{old} - A_i$	T_{old}

- 综上所述，经观察，可以注意到：加工时间只与机器 M_1 未加工的工件集合 S 、机器 M_2 还需加工的时间 t 有关。因此它们需要作为状态转移方程中的变量。
- 所以，定义动态规划数组 $T_{S,t}$ 为 当机器 M_1 未加工的工件集合为 S 、机器 M_2 还需加工的时间为 t 时，将集合 S 中的全部工件加工完毕的所需时间。

转移公式的引入(2)

- 陈列前述公式如下：

S_{new}	$S_{\text{new}} = S_{\text{old}} - \{i\}$	S_{old}
t_{new}	$t_{\text{new}} = B_i$	$t_{\text{old}}, t_{\text{old}} \leq A_i$
	$t_{\text{new}} = t_{\text{old}} - A_i + B_i$	$t_{\text{old}}, t_{\text{old}} > A_i$
t_{new}	$t_{\text{new}} = \max\{t_{\text{old}} - A_i, 0\} + B_i$	
\downarrow		
T_{new}	$T_{\text{new}} = T_{\text{old}} - A_i$	T_{old}

转移公式的引入(3)

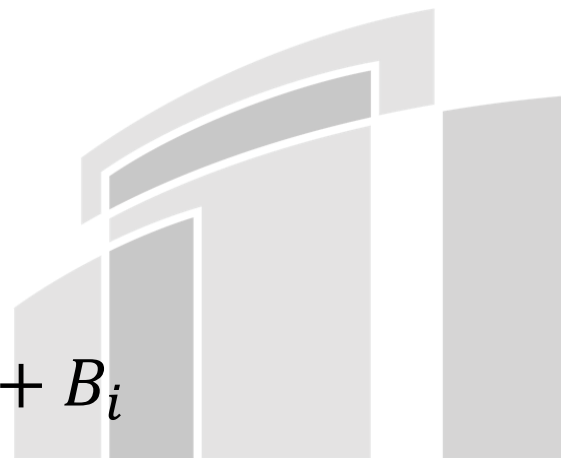
- 现展开式子

- $T_{new} = T_{old} - A_i$ (自顶向下)

- $T_{old} = T_{new} + A_i$ (自底向上)

- $T_{S_{old}, t_{old}} = T_{S_{new}, t_{new}} + A_i$

- 代入:
$$\begin{cases} S_{new} = S_{old} - \{i\} \\ t_{new} = \max\{t_{old} - A_i, 0\} + B_i \end{cases}$$



转移公式的引入(4)

- $T_{S_{old}, t_{old}} = T_{S_{old}-\{i\}, \max\{t_{old}-A_i, 0\}+B_i} + A_i$
- 化简，得
- $T_{S, t} = T_{S-\{i\}, \max\{t-A_i, 0\}+B_i} + A_i$
- 它是选取工件*i*时，所需时间的转移公式。
- $T_{S, t}$ ：当机器 M_1 未加工的工件集合为*S*、机器 M_2 还需加工的时间为*t*时，将集合*S*中的全部工件加工完毕的所需时间。

状态转移方程的边界条件

- $T_{S,t} = T_{S-\{i\}, \max\{t-A_i, 0\} + B_i} + A_i$
- 显然，转移方程的初始条件为：
- 当机器 M_1 未加工的工件集合为 \emptyset 、机器 M_2 还需加工的时间为 t 时，所需时间为 $T_{\emptyset, t} = t$ 。
- 转移方程的终止条件为：
- 当机器 M_1 未加工的工件集合为 U 、机器 M_2 还需加工的时间为 0 时，所需时间为 $T_{U, 0}$ 。



从贪心到动态规划(1)

- $T_{S,t} = T_{S-\{i\}, \max\{t-A_i, 0\} + B_i} + A_i$
- 此处的工件*i*是任意选取的。
- 如何取到最优值呢？



从贪心到动态规划(2)

- 最直接的贪心法：遍取未加工工件集合 S 中的工件 i ，使当前阶段的 $T_{S,t}$ 取得最小值。即令
- $T_{S,t} = \min_{i \in S} \{T_{S-\{i\}, \max\{t-A_i, 0\}+B_i} + A_i\}$
- 需要证明最优子结构：当前阶段的 $T_{S,t}$ 取得最小值、下一阶段的 $T_{S,t}$ 也取得最小值.....直到终止条件也取得最小值。

最优子结构的证明(1)

- 设 P 是所给作业的最优调度，它所需的加工时间为 $T_{S,t} = A_{P_1} + T'$ 。
- 其中 $T' = T_{S-\{P_1\}, \max\{t-A_{P_1}, 0\} + B_{P_1}}$ 为安排调度作业 P_2, P_3, \dots, P_n 的所需时间。
- 假设 P 不是其所给流水作业的规模缩小1的子最优调度，而 P' 是子最优调度。



最优子结构的证明(2)

- 设 $T'' = T_{S-\{P'_1\}, \max\{t-A_{P'_1}, 0\}+B_{P'_1}}$ 是安排调度作业 P'_2, P'_3, \dots, P'_n 所需的时间。
- 显然有 $T'' < T'$ 。
- 则 $T'_{S,t} = A_{P_1} + T'' < A_{P_1} + T' = T_{S,t}$ 。矛盾。
- 故作业调度问题具有最优子结构性质。

状态转移方程

- $$\begin{cases} T_{\emptyset, t} = t \\ T_{S, t} = \min_{i \in S} \{T_{S-\{i\}, \max\{t-A_i, 0\}} + B_i + A_i\} \end{cases}$$
- $T_{S, t}$: 当机器 M_1 未加工的工件集合为 S 、机器 M_2 还需加工的时间为 t 时, 将集合 S 中的全部工件加工完毕的所需时间。
- 求解 $T_{U, 0}$ 时, 采用状态压缩的技巧减少DP数组使用的空间
- 时间复杂度 $O(tn2^n)$, 空间复杂度 $O(t2^n)$



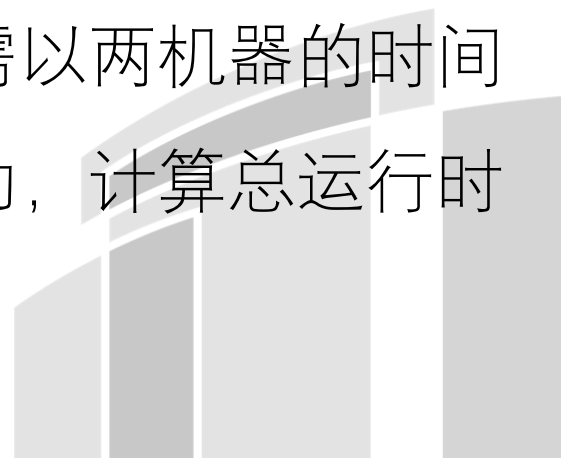
状态转移方程中状态变量的确定(1)

- 观察->理论?
- 事实上 S, t 两变量能够确定加工时间的原因如下:



状态转移方程中状态变量的确定(1)

- 观察->理论?
- 事实上 S, t 两变量能够确定加工时间的原因如下:
- 两机器是流水作业, 计算加工的总时间不能简单的将两机器的运行时间相加得到。需以两机器的时间轴之一为主, 另一时间轴作为辅助, 计算总运行时间。



状态转移方程中状态变量的确定(2)

- 机器 M_1 未加工的工件集合为 S ，它能够确定主时间轴——机器 M_1 的运作时间，但它不包含任何关于确定工件加工顺序的信息。



状态转移方程中状态变量的确定(2)

- 机器 M_1 未加工的工件集合为 S ，它能够确定主时间轴——机器 M_1 的运作时间，但它不包含任何关于确定工件加工顺序的信息。
- 机器 M_2 还需加工的时间为 t ，它能够确定辅时间轴——机器 M_1 在运作完毕后，机器 M_2 完成 M_1 递交给它的任务所需花费的额外时间。



状态转移方程中状态变量的确定(2)

- 机器 M_1 未加工的工件集合为 S ，它能够确定主时间轴——机器 M_1 的运作时间，但它不包含任何关于确定工件加工顺序的信息。
- 机器 M_2 还需加工的时间为 t ，它能够确定辅时间轴——机器 M_1 在运作完毕后，机器 M_2 完成 M_1 递交给它的任务所需花费的额外时间。
- 关于工件加工顺序的信息，在状态转移过程中进行第一维变量的决策过程中（选取下一加工的工件），被巧妙地蕴含在第二维变量里（机器 M_1 与机器 M_2 同时运行所抵消的时间）

状态转移方程中状态变量的确定(3)

- 两变量所共同组成的状态恰好涵盖了求得总运行时间的所有要素（加工工件的集合及加工的顺序），因此上述两状态能够唯一确定加工时间。
- 事实上，对于动态规划问题，几乎不可能存在一个一般性的思路，一劳永逸的解决全部此类问题。而是需要根据具体问题进行具体分析。

回溯路径的记录

- $$\begin{cases} T_{\emptyset, t} = t \\ T_{S, t} = \min_{i \in S} \{T_{S-\{i\}, \max\{t-A_i, 0\}+B_i} + A_i\} \end{cases}$$
- 站在状态 (S, t) 处，想要回溯到上一状态，需要记录哪些值？
- 为了输出工件，需要记录 i ，从而可以推导出上一状态的第一维变量。
- 记录 $\max\{t - A_i, 0\} + B_i$ ，上一个状态的第二维变量。
- 一边回溯，一边输出 i 即可。

