



浙江大学  
Zhejiang University

# 组合优化

浙江大学数学系 谈之奕



浙江大学

Zhejiang University

组合优化

# 排序问题 与装箱问题



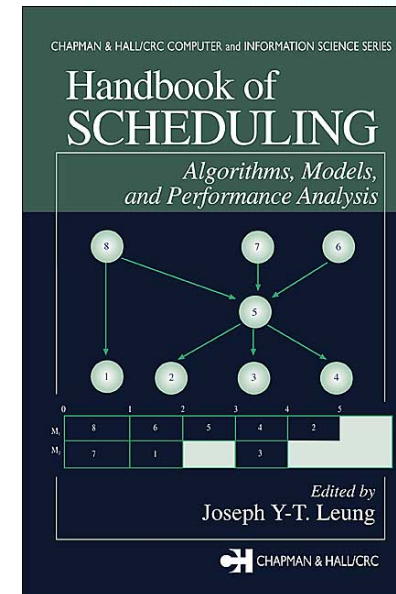


浙江大学  
Zhejiang University

组合优化

# 排序问题

- 排序 (scheduling) 主要研究如何利用有限资源, 在给定的限制条件下, 将一批任务安排在某些时间段内完成, 并使效益最大
  - 早期研究的排序问题背景源自工业生产, 习惯上把可用的资源称为机器 (machine), 需要完成的任务称为工件 (job)
  - 对部分排序问题, 可行解由工件加工的顺序决定, 这类问题最早也被称作 sequencing
  - 广义的scheduling问题也译作调度, 除排序问题外, 也包括时刻表编制、员工排班、项目管理等问题



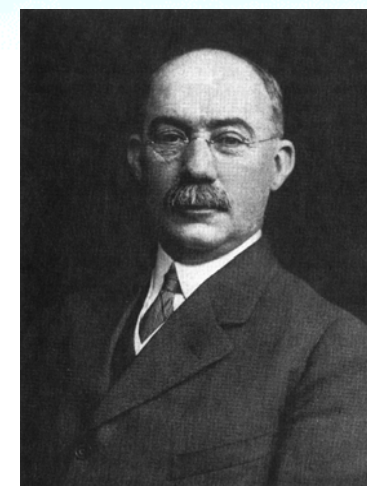
Journal of  
Scheduling

Leung, JY. (Eds.) *Handbook of scheduling: algorithms, models, and performance analysis*. CRC, 2004



# 排序问题

- 排序是组合优化中模型最丰富、应用最广泛的问题之一
- 描述排序问题及其可行解的常用工具是 **Gantt 图**



**Henry Gantt**  
(1861-1919)  
美国管理学家

# 排序问题



浙江大学

Zhejiang University

组合优化

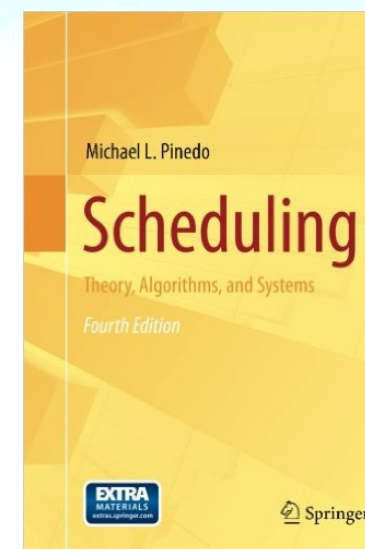
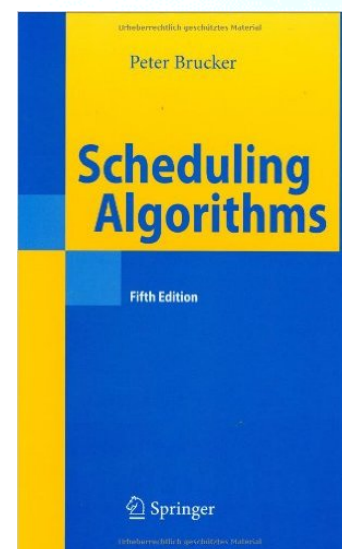
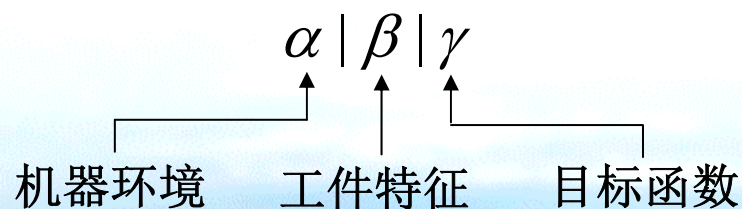
- Graham, RL, Lawler, EL, Lenstra, JK, Rinnooy Kan, AHG, Optimization and approximation in deterministic sequencing and scheduling: A survey, *Annals of Discrete Mathematics*, 5, 287–326, 1979
  - 三参数表示法 (three-field notation)  $\alpha | \beta | \gamma$
- Lawler, EL, Lenstra, JK, Rinnooy Kan, AHG, Shmoys, DB, Sequencing and Scheduling: Algorithmus and Complexity, Volume 4 of *Handbook in Operations Research and Managment Science*, North-Holland, 1993, 445-522

Sequencing and scheduling is concerned with the *optimal allocation of scarce resources to activities over time*. Of obvious practical importance, it has been the subject of extensive research since the early 1950's, and an impressive amount of literature has been created. Any discussion of the available material has to be selective. We will concentrate on the area of deterministic machine



# 三参数表示法

- 大部分经典排序问题可用机器环境，工件特征和目标函数三个要素来刻画
- 为了便于描述和规范，通常用所谓的“三参数表示法”（**three field notation**）表示一具体的排序问题



Pinedo ML. *Scheduling: theory, algorithms, and systems* (4th). Springer, 2014.

Brucker, P. *Scheduling algorithms* (5th). Springer, 2007.





# 机器环境

## 组合优化

- 单台机 (single machine, **1**) : 一台机器, 工件在其上加工一次
- 平行机 (parallel machine) : 两台以上功能相同的机器, 工件在其中任意一台机器上加工一次
  - 同型机 (identical machine, **P**) : 所有机器完全相同
  - 同类机 (uniform machine, **Q**) : 每台机器有固定的加工速度, 工件在不同的机器上加工所需时间成比例关系
  - 不同类机 (unrelated machine, **R**) : 工件在不同机器上加工需要不同的时间
- 车间作业 (shop) : 工件加工需经过多道工序, 每道工序在给定机器上加工
  - 流水作业 (Flow shop, **F**) : 所有工件有相同的工序, 按给定顺序加工
  - 工件作业 (Job shop, **J**) : 不同工件的工序可能不同, 按给定顺序加工
  - 自由作业 (Open shop, **O**) : 不同工件的工序可能不同, 且可自由决定加工顺序



浙江大学

Zhejiang University

组合优化

# 工件特征

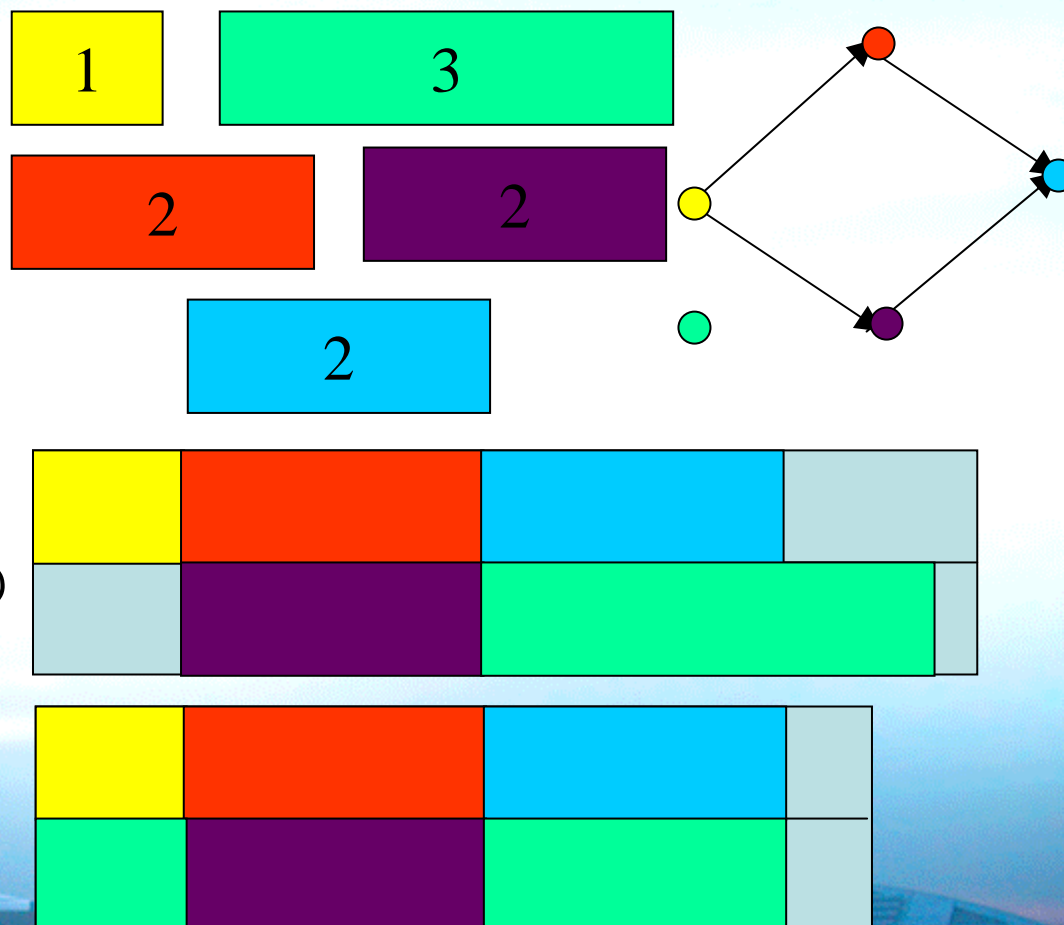
- **加工时间 (processing time)**  $p_j$  : 工件在机器上加工所需的时间
  - 不同类机  $p_{ij}$  , 同类机  $p_{ij} = \frac{p_j}{s_i}$
  - 车间排序  $(p_{j1}, p_{j2}, \dots, p_{jk})$
- **准备时间 (release time)**  $r_j$  : 准备时刻之后工件才可以开始加工
- **交工期 (due date)**  $d_j$  : 工件预定的完工交付时刻
- **权 (weight)**  $w_j$  : 工件的权重
- **序约束 (precedence constraints, prec)** : 只有当一部分工件完成加工后才能加工另一部分工件, 通常用有向图表示
  - (单、平行) 链 (chain)、(入、出) 树 (intree, outtree)
- **中断 (preemption, pmpt)** : 工件在一台机器上加工部分后可以暂停加工, 随后在该台或者其他机器上恢复加工剩余部分





# 可行排序

- 符合问题所有要求的工件加工方案称为可行排序 (feasible schedule)
  - 确定一可行排序一般需明确每个工件（或工序）加工的机器及在其上的开始时间
- 常用记号
  - 工件  $J_j, j = 1, \dots, n$
  - 机器  $M_i, i = 1, \dots, m$
  - 工件  $J_j$  的完工时间  $C_j(\sigma)$
  - 机器  $M_i$  的负载  $L_i(\sigma)$



# 目标函数

- 总完工时间 (total complete time) :  $\sum C_j = \sum_{j=1}^n C_j$
- 工件最大完工时间 (makespan) :  

$$C_{\max} = \max_{1 \leq j \leq n} C_j = \max_{1 \leq i \leq m} L_i$$
- 机器最小负载 (minimum machine load) :  

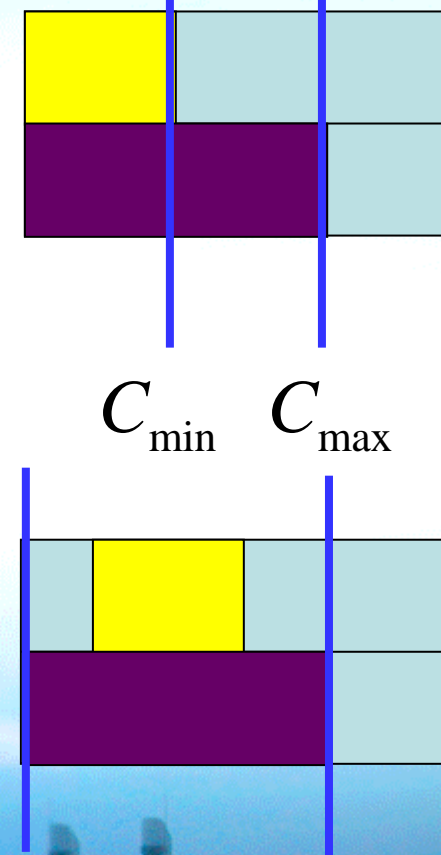
$$C_{\min} = \min_{1 \leq i \leq m} L_i$$
 极大化目标
- 最大延迟 (maximum lateness) : 不允许机器在完成该机器的所有加工任务前空闲  

$$L_{\max} = \max_{1 \leq j \leq n} (C_j - d_j)$$
- 总延误时间 (total tardiness) :  

$$\sum T_j = \sum_{j=1}^n \max \{C_j - d_j, 0\}$$
- 误工工件数 (number of tardy jobs) :  

$$\sum U_j = \sum_{j=1}^n U_j, \text{ 其中 } U_j = \begin{cases} 1 & C_j > d_j \\ 0 & \text{其他} \end{cases}$$

组合优化







浙江大学

Zhejiang University

组合优化

# Santa Claus问题

- 圣诞老人欲将  $n$  件礼物分给  $m$  位小朋友。一件礼物只能分给一位小朋友，但一位小朋友可以拿到多件礼物。第  $i$  位小朋友拿到礼物  $j$  时他的满意度为  $p_{ij}$ ,  $i = 1, \dots, m$ ,  $j = 1, \dots, n$ , 一位小朋友拿到多件礼物时的满意度为各礼物相应满意度之和。希望给出一礼物分配方案，使满意度最小的小朋友的满意度尽可能大

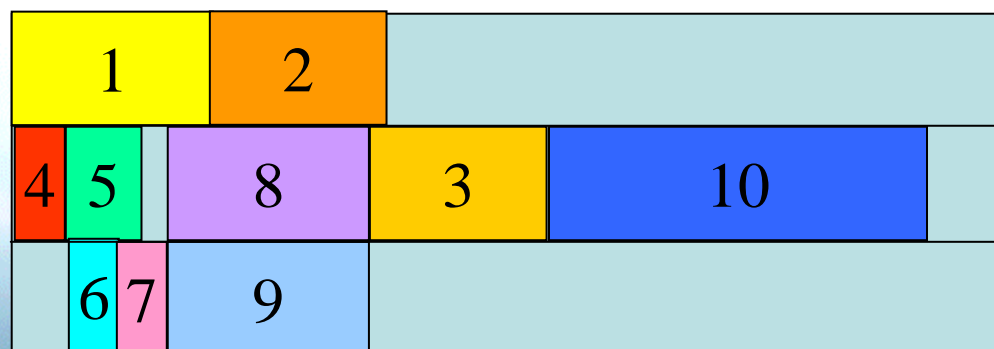
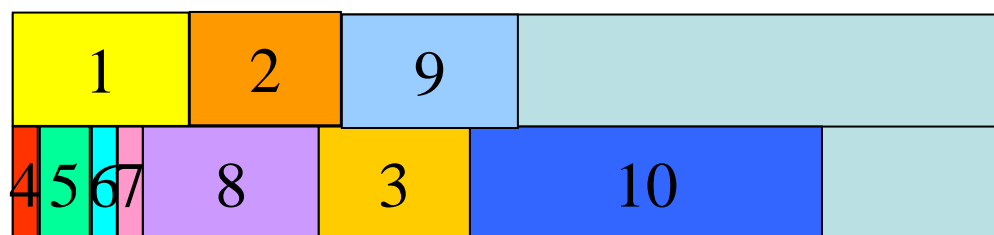
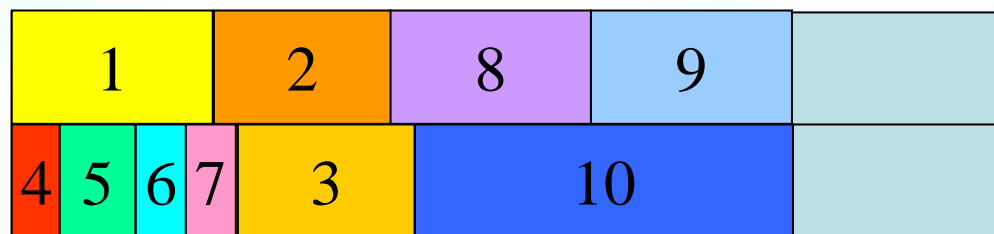


STOC

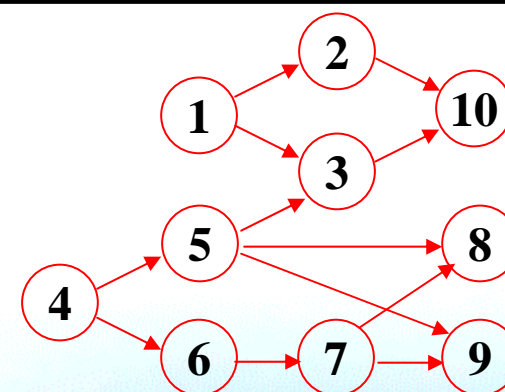
礼物 —— 工件      满意度 —— 加工时间  
小朋友 —— 机器      小朋友的满意度 —— 机器完工时间  
目标函数 —— 极大化最小机器完工时间       $Rm \parallel C_{\min}$

**Bansal, N., Sviridenko, M.,**  
**The Santa Claus problem,**  
*Proceedings of the 38th Annual*  
*ACM Symposium on Theory of*  
*computing, 31-40, 2006*

# 排序悖论



工件	1	2	3	4	5	6	7	8	9	10
加工时间	8	7	7	2	3	2	2	8	8	15
	7	6	6	1	2	1	1	7	7	14



$P2 | \text{prec} | C_{\max}$

基于贪婪思想的算法：  
可加工的工件尽早开工



$$1 \parallel \sum C_j$$

- 工件按任意顺序无空闲加工所得排序均为  $1 \parallel C_{\max}$  的最优排序
- **SPT** (Shortest Processing Time first) 规则：工件按加工时间的非减顺序排列
- **SPT** 规则给出  $1 \parallel \sum C_j$  的最优排序

排序	$p_1$	$p_2$	$p_3$	$\dots$
----	-------	-------	-------	---------

完工时间  $C_j$        $p_1$        $p_1 + p_2$        $p_1 + p_2 + p_3$

总完工时间  $\sum_{j=1}^n C_j = p_1 + (p_1 + p_2) + (p_1 + p_2 + p_3) + \dots + (p_1 + \dots + p_n)$

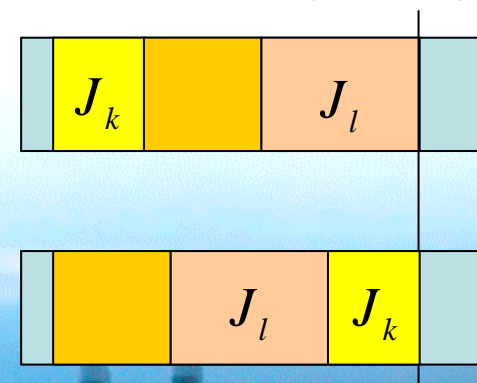
$$= np_1 + (n-1)p_2 + \dots + 2p_{n-1} + p_n$$

$$1 \parallel L_{\max}$$

- **EDD (Earliest Due Date first)** 规则：工件按预定交工期限的非减顺序排列
- **EDD**规则给出  $1 \parallel L_{\max}$  的最优排序
  - 若在最优排序  $\sigma$  中，存在工件  $J_l, J_k, d_l < d_k$ ，但  $J_k$  在  $J_l$  之前加工
  - 构造新的排序  $\sigma'$ ， $J_k$  推迟至紧接  $J_l$  之后加工，其他工件相对顺序不变
  - 除  $J_k$  外，完工时间均未增加，延迟也未增加
 
$$C_i(\sigma') - d_i \leq C_i(\sigma) - d_i \leq \max_{1 \leq j \leq n} (C_j(\sigma) - d_j), i \neq k$$

$$C_k(\sigma') - d_k = C_l(\sigma) - d_k < C_l(\sigma) - d_l \leq \max_{1 \leq j \leq n} (C_j(\sigma) - d_j)$$
  - 仿上逐步调整可得满足**EDD**规则的最优排序

$$C_k(\sigma') = C_l(\sigma)$$





$$1 \parallel \sum U_j$$

- **Moore-Hodgeson算法**

- 将工件**EDD**规则排列，即可假设  $d_1 \leq d_2 \leq \dots \leq d_n$
- 若按当前顺序加工存在误工工件，则找出第一个误工的工件  $J_i$ ，从工件  $J_1, J_2, \dots$  中删除加工时间最大的工件，得到一个新的部分工件的加工顺序，继续上述过程直至按当前顺序加工不存在误工工件为止
- 将历次删除的工件以任意顺序安排在一直未被删除的工件之后加工，得到所有工件的一个加工顺序

**Moore JM. An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management science*, 15, 102-109, 1968**

**Author's Supplement<sup>4</sup>**

It has been brought to the attention of the author that the “compliment” of the presented algorithm should be computationally more efficient. This algorithm is given below:

<sup>4</sup> The algorithm given here was proposed by Mr. T. J. Hodgson, Department of Industrial Engineering, The University of Michigan.



# 延迟、延误与误工

## 组合优化

1|| $L_{\max}$  最优排序：最大延迟为5，误工工件数为5  
总延误时间为21

$p_j$	4	3	9	5	3
$d_j$	1	3	12	16	19

1|| $\sum U_j$  最优排序：最大延迟为14，误工工件数为2  
总延误时间为26

1|| $\sum T_j$  最优排序：最大延迟为14，误工工件数为2  
总延误时间为18





$$1 \parallel \sum T_j$$



浙江大学  
Zhejiang University

组合优化

- $1 \parallel \sum T_j$  存在伪多项式时间算法
- $1 \parallel \sum T_j$  存在FPTAS
- $1 \parallel \sum T_j$  是普通意义下  $\mathcal{NP}$ -完全的

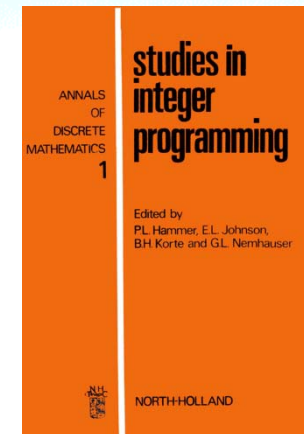
Lawler EL. A “pseudopolynomial” algorithm for sequencing jobs to minimize total tardiness.

*Annals of Discrete Mathematics*, 1, 331-342, 1977.

Lawler EL. A fully polynomial approximation scheme for the total tardiness problem.

*Operations Research Letters*, 1, 207-208, 1982.

Du J, Leung JYT. Minimizing total tardiness on one machine is NP-hard. *Mathematics of Operations Research*, 15, 483-495, 1990.



# $Pm \parallel C_{\max}$

- $Pm \parallel C_{\max}$  是  $\mathcal{NP}$  - 完全的      划分问题  $\leq_m^P P2 \parallel C_{\max}$  的判定形式
  - 任取划分问题的实例  $I_p: A = \{a_1, a_2, \dots, a_n\}$ , 记  $S = \frac{1}{2} \sum_{j=1}^n a_j$ 。构造  $P2 \parallel C_{\max}$  判定形式实例  $I_s$ : 机器  $M_1, M_2$  与  $n$  个工件, 工件  $J_j$  的加工时间为  $a_j$ , 阈值为  $S$
  - 若  $I_p$  答案为“是”, 则存在子集  $A_1, A_2$ , 使得  $A = A_1 \cup A_2, A_1 \cap A_2 = \emptyset$ , 且满足  $\sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i = S$ , 将集合  $\mathbf{J}_i = \{J_j \mid a_j \in A_i\}$  中的工件安排在机器  $M_i$  上加工,  $i=1, 2$ , 即得一可行排序, **makespan** 为  $S$
  - 若  $I_s$  答案为“是”, 则存在一可行排序, **makespan** 不超过  $S$ , 设在两台机器上加工的工件集为  $\mathbf{J}_1, \mathbf{J}_2$ , 则  $\sum_{J_i \in \mathbf{J}_i} a_i \leq S, i=1, 2$ , 又由于  $\sum_{J_i \in \mathbf{J}_1} a_i + \sum_{J_i \in \mathbf{J}_2} a_i = 2S$ , 故  $\sum_{J_i \in \mathbf{J}_i} a_i = S, i=1, 2$ 。令  $A_i = \{a_j \mid J_j \in \mathbf{J}_i\}$ ,  $A_1, A_2$  即为  $A$  的一个划分



# 近似算法



浙江大学  
Zhejiang University

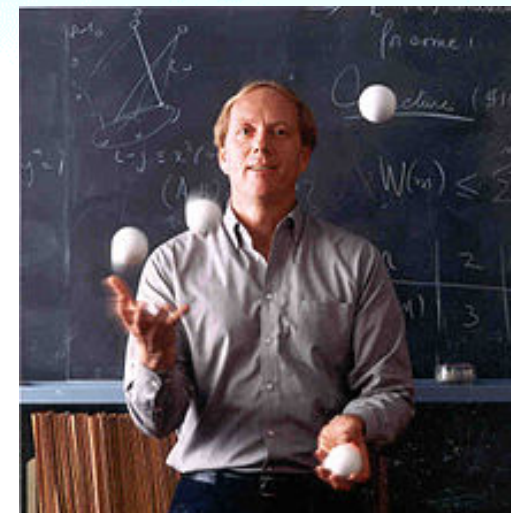
组合优化



**Fan Chung (金芳蓉)**  
华裔数学家  
(1949- )



**American Mathematical Society  
Mathematical Association of America  
International Jugglers Association**  
前主席



**Ronald Lewis Graham**  
美国数学家  
(1935- )





# LS 算法

- 算法 **List Scheduling (LS)**
  - 将工件按给定顺序安排在能使其最早开始加工的机器上加工
- **LS 算法求解  $Pm \parallel C_{\max}$  的最坏情况界为  $2 - \frac{1}{m}$** 
  - 若存在实例  $I$ , 使得  $\frac{C^{LS}(I)}{C^*(I)} > 2 - \frac{1}{m}$ , 记  $I_0$  是所有反例中工件数最少的那个, 称为 **最小反例 (minimal counterexample)**
  - 在  $I_0$  中, 最后一个工件  $J_n$  决定 **makespan**
    - 若不然, 设  $J_k, k < n$  决定 **makespan**
    - 删去  $I_0$  中  $J_k$  之后所有工件得到一新实例  $I'$ 

$$C^{LS}(I') = C^{LS}(I_0) = C_k \quad C^*(I') \leq C^*(I_0)$$

$$\frac{C^{LS}(I')}{C^*(I')} \geq \frac{C^{LS}(I_0)}{C^*(I_0)} > 2 - \frac{1}{m}$$
**与最小反例的假设矛盾**





# LS 算法

- 记  $s_n$  为  $J_n$  的开工时间。由算法规则，在时刻  $s_n$  之前，所有机器都不会完成加工，故  $\sum_{j=1}^n p_j - p_n \geq s_n m$

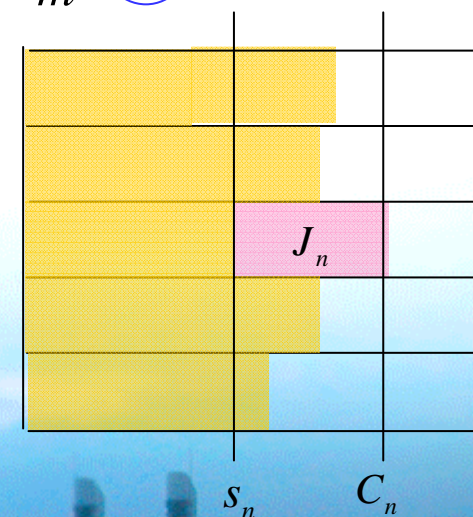
- $C^{LS}(I_0) = s_n + p_n \leq \frac{1}{m} \left( \sum_{j=1}^n p_j - p_n \right) + p_n = \frac{1}{m} \sum_{j=1}^n p_j + \frac{m-1}{m} p_n$

- 由于  $C^*(I_0) \geq \frac{1}{m} \sum_{j=1}^n p_j$  ,  $C^*(I_0) \geq \max_{1 \leq j \leq n} p_j \geq p_n$  ,

$$\frac{C^{LS}(I_0)}{C^*(I_0)} \leq \frac{\frac{1}{m} \sum_{j=1}^n p_j}{C^*(I_0)} + \frac{\frac{m-1}{m} p_n}{C^*(I_0)} \leq 1 + \frac{m-1}{m} = 2 - \frac{1}{m}$$

紧例？

矛盾





# 从LS到LPT

- 算法 **Longest Processing Time first (LPT)**

- 将工件按加工时间从大到小顺序排列，即假设  $p_1 \geq p_2 \geq \dots \geq p_n$
- 将工件按重排后的顺序用LS算法加工

Graham, R. L., Bounds for certain multiprocessing anomalies, *The Bell System Technical Journal*, 45, 1563-1581, 1966.

Graham, R. L., Bounds on multiprocessing finishing anomalies, *SIAM Journal on Applied Mathematics*, 17, 416-429, 1969.

$$m \left\{ \begin{array}{c|c|c|c|c} 1 & 1 & \cdots & 1 & m \\ \hline 1 & 1 & \cdots & 1 & \\ \hline \cdots & \cdots & & & \\ \hline 1 & 1 & \cdots & 1 & \\ \hline 1 & 1 & \cdots & 1 & \\ \hline \end{array} \right.$$

$m-1$

1	1	...	1	1	$C^{LS} = 2m - 1$
1	1	...	1	1	$C^* = m$
...	...				$\frac{C^{LS}}{C^*} = 2 - \frac{1}{m}$
1	1	...	1	1	
$m$					



# LPT 算法

- LPT算法求解  $Pm \parallel C_{\max}$  的最坏情况界为  $\frac{4}{3} - \frac{1}{3m}$

$$\frac{C^{LS}(I_0)}{C^*(I_0)} \leq \frac{\frac{1}{m} \sum_{j=1}^n p_j}{C^*(I_0)} + \frac{\frac{m-1}{m} p_n}{C^*(I_0)} \leq 1 + \frac{m-1}{m} \frac{p_n}{C^*(I_0)} \leq 1 + \frac{m-1}{3m} = \frac{4}{3} - \frac{1}{3m}$$

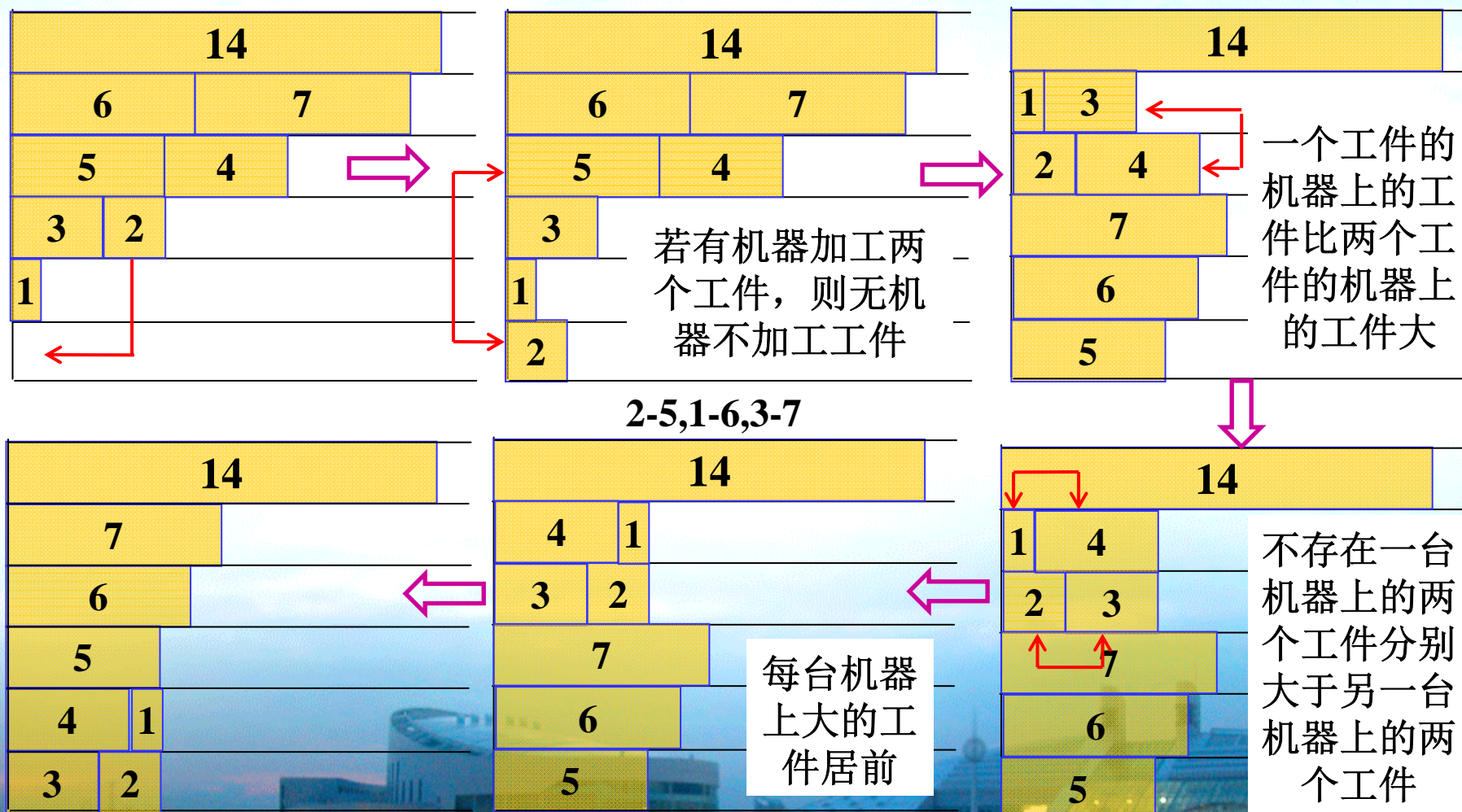
- 若实例  $I$  满足  $C^*(I) \geq 3p_n$ ，则  $\frac{C^{LPT}(I)}{C^*(I)} \leq \frac{4}{3} - \frac{1}{3m}$
- 若实例  $I$  满足  $C^*(I) < 3p_n$ ，则在最优排序中每台机器至多加工两个工件
  - LPT算法给出的排序也为最优排序

某个最优排序可在保持最优性的前提下转化为LPT算法给出的排序



# 最优排序与LPT排序

## 组合优化





# LPT 算法

## 组合优化

$2m-1$	$m$	$m$	
$2m-1$	$m$		
$2m-2$	$m+1$		
$2m-2$	$m+1$		
$\vdots$			
$\frac{3m}{2}+1$	$\frac{3m}{2}-2$		
$\frac{3m}{2}+1$	$\frac{3m}{2}-2$		
$\frac{3m}{2}$	$\frac{3m}{2}-1$		
$\frac{3m}{2}$	$\frac{3m}{2}-1$		

$$C^{LPT}(I_m) = 4m-1 \quad C^{LPT}(I_m) = 3m$$

$m$  为偶数

$2m-1$	$m+1$	
$2m-1$	$m+1$	
$2m-2$	$m+2$	
$2m-2$	$m+2$	
$\vdots$		
$\frac{3m}{2}+1$	$\frac{3m}{2}-1$	
$\frac{3m}{2}+1$	$\frac{3m}{2}-1$	
$\frac{3m}{2}$	$\frac{3m}{2}$	
$\frac{3m}{2}$	$\frac{3m}{2}$	
$m$	$m$	$m$

$2m-1$	$m$	$m$	
$2m-1$	$m$		
$2m-2$	$m+1$		
$2m-2$	$m+1$		
$\vdots$			
$\frac{3m+1}{2}$	$\frac{3m-3}{2}$		
$\frac{3m+1}{2}$	$\frac{3m-3}{2}$		
$\frac{3m-1}{2}$	$\frac{3m-1}{2}$		

$$C^{LPT}(I_m) = 4m-1 \quad C^{LPT}(I_m) = 3m$$

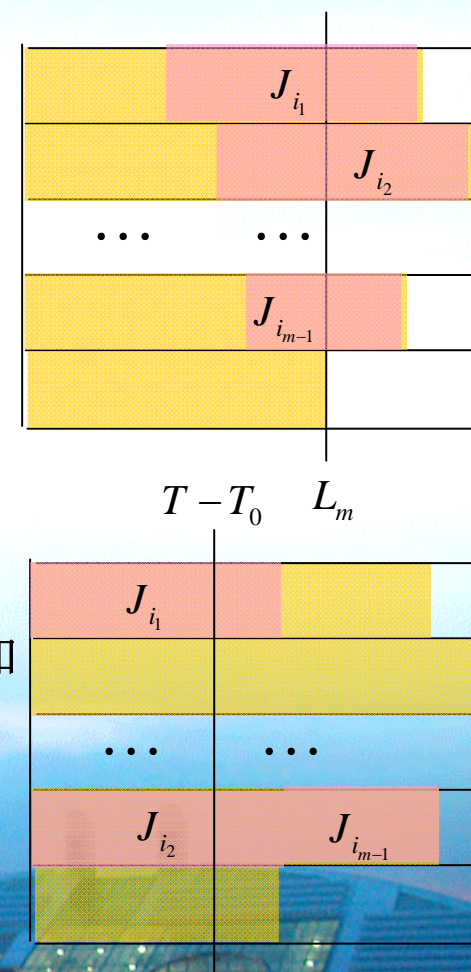
$m$  为奇数

$2m-1$	$m+1$	
$2m-1$	$m+1$	
$2m-2$	$m+2$	
$2m-2$	$m+2$	
$\vdots$		
$\frac{3m+1}{2}$	$\frac{3m-1}{2}$	
$\frac{3m+1}{2}$	$\frac{3m-1}{2}$	
$m$	$m$	$m$

$$\frac{C^{LPT}(I)}{C^*(I)} = \frac{4}{3} - \frac{1}{3m}$$

# $Pm \parallel C_{\min}$

- LS算法求解  $Pm \parallel C_{\min}$  的最坏情况界为  $m$ 
  - 不妨设  $C^{LS}(I) = L_m \leq L_i, i = 1, \dots, m-1$
  - 设在  $M_i, i = 1, \dots, m-1$  上加工的最后一个工件为  $J_{j_i}$   
令  $\mathcal{J}_0 = \{J_{j_1}, J_{j_2}, \dots, J_{j_{m-1}}\}$ ,  $T = \sum_{j=1}^n p_j, T_0 = \sum_{i=1}^{m-1} p_{j_i}$
  - $L_i - p_{j_i} \leq L_m, i = 1, \dots, m-1$
  - $C^*(I) \leq T - T_0$ 
    - 若不然, 在最优排序中所有机器的负载均大于  $T - T_0$
    - $\mathcal{J}_0$  中的  $m-1$  个工件至多在  $m-1$  台机器上加工
    - 未加工  $\mathcal{J}_0$  中工件的机器的负载与  $\mathcal{J}_0$  中工件的加工时间之和大于  $T - T_0 + T_0 = T$





$$Pm \parallel C_{\min}$$

- LS算法求解  $Pm \parallel C_{\min}$  的最坏情况界为  $m$

- $$\sum_{i=1}^{m-1} L_i - \sum_{i=1}^{m-1} p_{j_i} \leq (m-1)L_m \quad L_i - p_{j_i} \leq L_m, i = 1, \dots, m-1$$

- $$mL_m \geq L_m + \sum_{i=1}^{m-1} L_i - \sum_{i=1}^{m-1} p_{j_i} = \sum_{j=1}^n p_j - \sum_{i=1}^{m-1} p_{j_i} = T - T_0$$

- $$\frac{C^*(I)}{C^{LS}(I)} = \frac{T - T_0}{L_m} \leq m \quad C^*(I) \leq T - T_0$$

- LPT算法求解  $Pm \parallel C_{\min}$  的最坏情况界为  $\frac{4m-2}{3m-1}$

Deurmeyer BL, Friesen DK, Langston MA. Scheduling to maximize the minimum processor finish time in a multiprocessor system. *SIAM Journal on Algebraic Discrete Methods*, 3, 190-196, 1982

权函数法

Csirik J, Kellerer H, Woeginger G. The exact LPT-bound for maximizing the minimum completion time. *Operations Research Letters*, 11, 281-287, 1992.

$$m \left\{ \begin{array}{|c|c|c|} \hline 1 & m & \\ \hline 1 & m & \\ \hline \dots & \dots & \\ \hline 1 & m & \\ \hline 1 & & \\ \hline \end{array} \right.$$

$m$			$C^{LS} = 1$
$m$			
$\dots$			$C^* = m$
$m$			$\frac{C^*}{C^{LS}} = m$
1	$\dots$	1	

# $C_{\max}$ 与 $C_{\min}$

- 对任意  $(\alpha, \beta) \in S = \left\{ \left( \frac{5}{4+s}, \frac{6+s}{6} \right) \mid 0 \leq s \leq 1 \right\}$ ,  
不存在（多项式时间或指数时间）算法，使得对任意实例  $I$ ,  $\frac{C_{\min}^*(I)}{C_{\min}^A(I)} < \alpha, \frac{C_{\max}^A(I)}{C_{\max}^*(I)} < \beta$

Gu MY, Tan ZY, Xia BZ, Yan YJ. A new approach for bicriteria partitioning problem. *Optimization Letters*, 9, 1025-1037, 2015.

对任意实例，是否总存在某个排序，它同时是  $Pm \parallel C_{\min}$  和  $Pm \parallel C_{\max}$  的最优排序  
是否存在更好的不可同时近似性结论

9	4	
6	4	
6	4	

$Pm \parallel C_{\min}$  最优解

9		
6	6	
4	4	4

$Pm \parallel C_{\max}$  最优解



$$Fm \parallel C_{\max}$$

- $F2 \parallel C_{\max}$  的**Johnson**算法

- 将工件集分为两个子集  $\mathcal{J}_1, \mathcal{J}_2$ ，其中

$$\mathcal{J}_1 = \{J_j \mid p_{j1} < p_{j2}\}, \mathcal{J}_2 = \{J_j \mid p_{j1} \geq p_{j2}\}$$

- 将  $\mathcal{J}_1$  中工件按它们在第一台机器上的加工时间的非减顺序排列，将  $\mathcal{J}_2$  中工件按它们在第二台机器上的加工时间的非增顺序排列
- 两台机器上的工件加工顺序相同：先按重排后顺序加工  $\mathcal{J}_1$  中的工件，再按重排后顺序加工  $\mathcal{J}_2$  中的工件

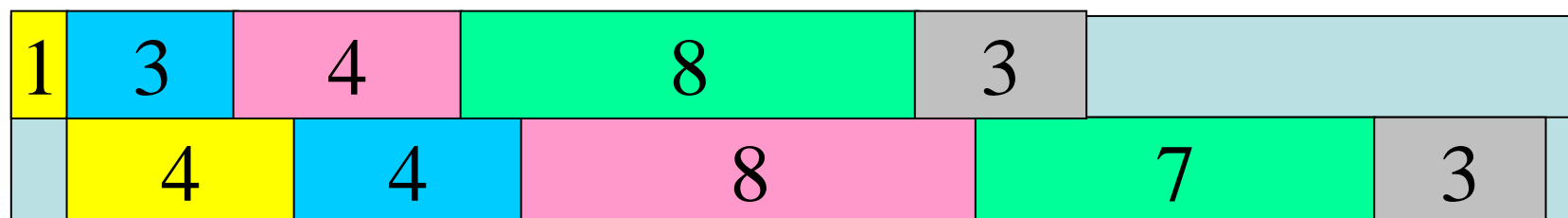
Johnson SM. Optimal two- and three- stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1, 61-68, 1954.

# Johnson 算法

$J_j$	1	2	3	4	5
$p_{j1}$	4	3	3	1	8
$p_{j2}$	8	3	4	4	7

	$\mathcal{J}_1$			$\mathcal{J}_2$	
$p_{j1}$	4	3	1	3	8
$p_{j2}$	8	4	4	3	7

	$\mathcal{J}_1$			$\mathcal{J}_2$	
$p_{j1}$	1	3	4	8	3
$p_{j2}$	4	4	8	7	3

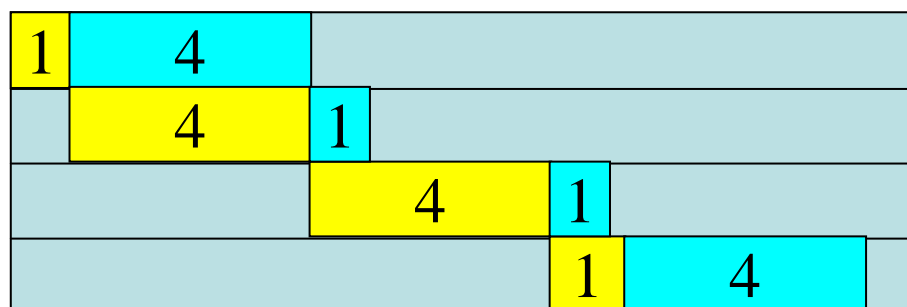


对  $Fm \parallel C_{\max}$ ，存在一个最优排序，其中前两台机器上工件的加工顺序相同，最后两台机器上工件的加工顺序也相同

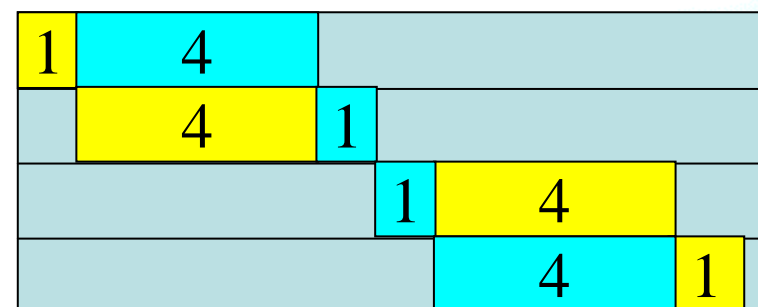


$$Fm \parallel C_{\max}$$

- 当  $m \geq 4$  时,  $Fm \parallel C_{\max}$  的最优排序中各台机器上工件加工的顺序未必相同



makespan为14



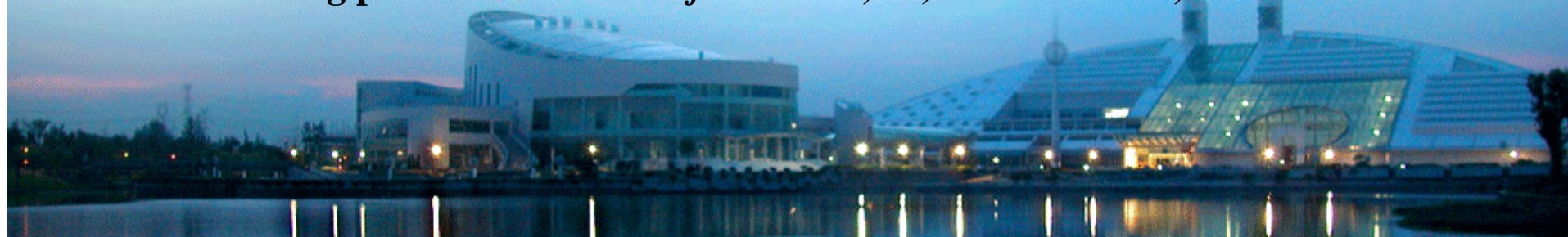
最优makespan为12

- 限定各台机器上工件加工顺序必须相同的Flowshop问题称为**Permutation Flowshop**, 记为  $Fm | \text{prmu} | C_{\max}$

# 下界

- 记  $D$  (**dilation**) 为各工件在所有工序上的加工时间之和的最大值,  $C$  (**congestion**) 为各机器所需加工的所有工序的加工时间之和的最大值,  $LB = \max\{D, C\}$ , 则  $C_{\max}^* \geq LB$  一般情况下已知的下界
  - 对  $Fm \parallel C_{\max}$  和  $Om \parallel C_{\max}$ ,  $LB = \max\left\{\max_{i=1,\dots,m} \sum_{j=1}^n p_{ji}, \max_{j=1,\dots,n} \sum_{i=1}^m p_{ji}\right\}$
  - 对  $F2 \parallel C_{\max}$  和  $O2 \parallel C_{\max}$ ,  $C_{\max}^* \geq \max\left\{\sum_{j=1}^n p_{j1}, \sum_{j=1}^n p_{j2}, \max_{1 \leq j \leq n} \{p_{j1} + p_{j2}\}\right\}$
- 对  $Fm \parallel C_{\max}$ , 存在一实例使得  $C_{\max}^* = \Omega\left(LB \frac{\log LB}{\log \log LB}\right)$

Mastrolilli M, Svensson O. Hardness of approximating flow and job shop scheduling problems. *Journal of the ACM*, 58, Article No. 20, 2011





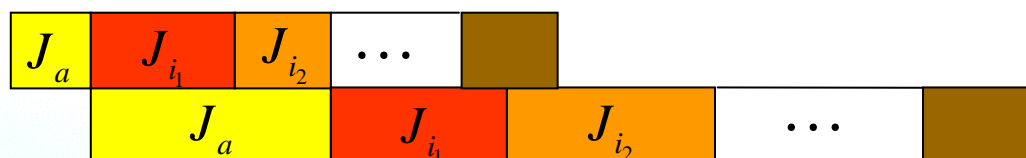
# $O2 \parallel C_{\max}$

## • 双向排序法

- 将工件集分为两个子集  $\mathcal{J}_1, \mathcal{J}_2$ ，其中

$$\mathcal{J}_1 = \{J_j \mid p_{j1} < p_{j2}\}, \mathcal{J}_2 = \{J_j \mid p_{j1} \geq p_{j2}\}$$

- 设  $J_a$  是  $\mathcal{J}_1$  中在  $M_2$  上加工时间最大的工件，即  $p_{a2} = \max_{J_j \in \mathcal{J}_1} p_{j2}$ ， $J_b$  是  $\mathcal{J}_2$  中在  $M_1$  上加工时间最大的工件，即  $p_{b1} = \max_{J_j \in \mathcal{J}_2} p_{j1}$



排序是否可行?  $p_{i1} \leq p_{i2} \leq p_{a2}$

$$p_{i1} + p_{i2} \leq p_{i2} + p_{i2} \leq p_{i2} + p_{a2}$$

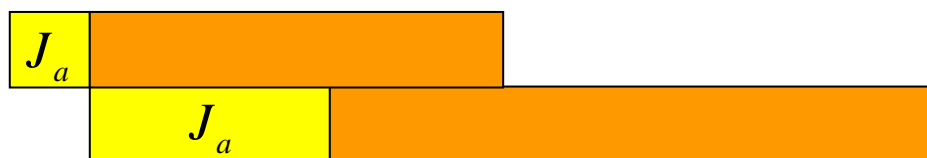
$$p_{j2} \leq p_{j1} \leq p_{b1}$$

$$p_{j2} + p_{j2} \leq p_{j1} + p_{j2} \leq p_{j1} + p_{b1}$$

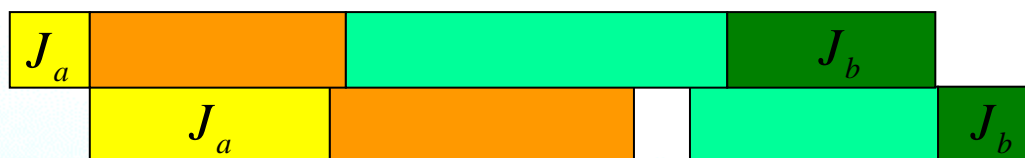


# $O2 \parallel C_{\max}$

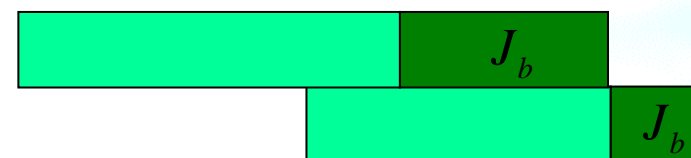
- makespan为  $\sum_{j=1}^n p_{j1}$ ,  $\sum_{j=1}^n p_{j2}$ , 或  $\max_{1 \leq j \leq n} \{p_{j1} + p_{j2}\}$  的排序必为最优排序



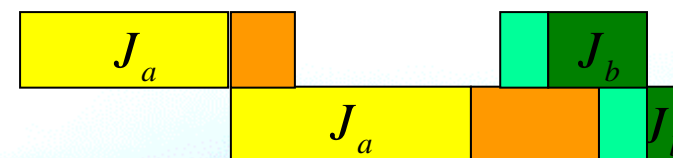
移动过程不改变排序的可行性



$$C_{\max} = \sum_{j=1}^n p_{j1}$$



$$C_{\max} = \sum_{j=1}^n p_{j2}$$









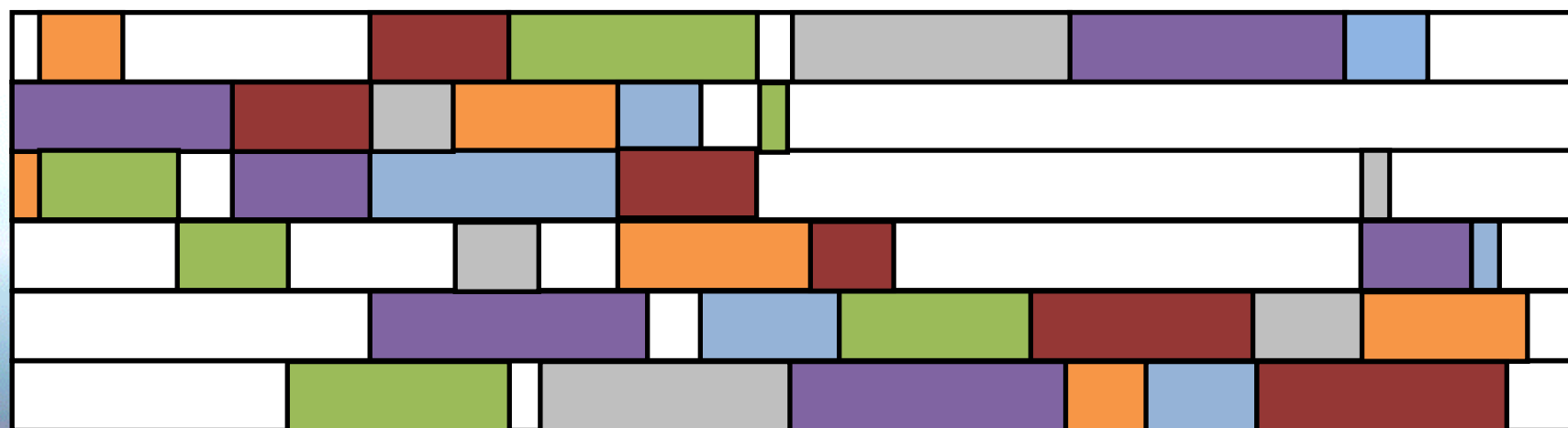
$$C_{\max} = p_{a1} + p_{a2}$$



# 车间作业

- 实例FT06: 6机器6工件6工序  $C_{\max} = 55$

	$J_1: (M_3,1) (M_1,3) (M_2,6) (M_4,7) (M_6,3) (M_5,6)$
	$J_2: (M_2,8) (M_3,5) (M_5,10) (M_6,10) (M_1,10) (M_4,4)$
	$J_3: (M_3,5) (M_4,4) (M_6,8) (M_1,9) (M_2,1) (M_5,7)$
	$J_4: (M_2,5) (M_1,5) (M_3,5) (M_4,3) (M_5,8) (M_6,9)$
	$J_5: (M_3,9) (M_2,3) (M_5,5) (M_6,4) (M_1,3) (M_4,1)$
	$J_6: (M_2,3) (M_4,3) (M_6,9) (M_1,10) (M_1,4) (M_3,1)$



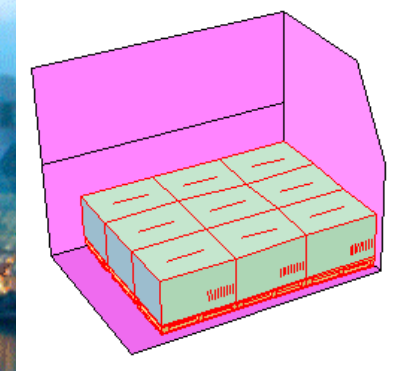


浙江大学  
Zhejiang University

组合优化

# 装箱问题

- **装箱问题 (bin-packing problem)**: 将一系列物品放入容量一定的若干箱子中, 放入每个箱子中的物品大小之和不超过箱子容量, 目标为所用箱子数尽可能少
  - **下料问题 (cutting-stock problem)**: 给定生产一批产品所需的某种材料的大小与数量列表, 如何从一定规格的原料中下料, 使所用的原料最少
  - 一维装箱: 箱子容量和物品大小均为有理数
  - 二维装箱: 箱子和物品均为长、宽一定的矩形
  - 二维条状装箱 (**strip packing**): 将若干长、宽一定的矩形物品无重叠地放入长无限、宽固定的矩形区域, 目标为物品占用长度最小
  - 三维装箱: 箱子和物品均为长、宽、高一一定的长方体
  - 物品可以/不可以旋转、倒置, 考虑/不考虑稳定性等



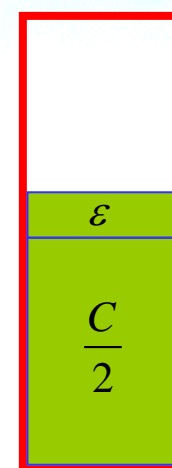
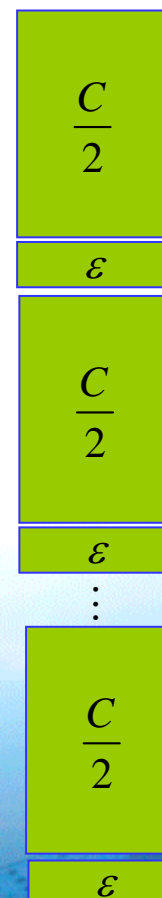




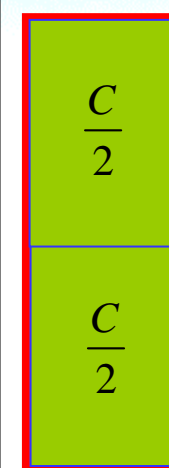
# 一维装箱

- 箱子容量为  $C$ ，物品  $j$  的大小为  $w_j$ ， $0 \leq w_j \leq C$ ， $j = 1, \dots, n$ 
  - 用  $B_i$  表示按顺序第  $i$  只启用的箱子，它同时表示当前放入该箱子中物品的大小之和
- Next Fit 算法**：设当前装箱的物品为  $j$ ，最后启用的箱子为  $B_i$ 
  - 若  $w_j + B_i \leq C$ ，则将  $j$  放入  $B_i$  中
  - 若  $w_j + B_i > C$ ，则将  $j$  放入  $B_{i+1}$  中，并且  $B_i$  自此不再装物品
- Next Fit 算法的最坏情况界为 2**

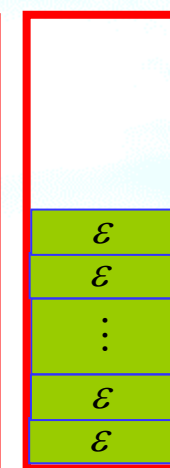
4M 个物品



$\times 2M$



$\times M$



$$C^{NF} = 2M \quad C^* = M + 1$$

$$\frac{C^{NF}}{C^*} = \frac{2M}{M+1} \rightarrow 2(M \rightarrow \infty)$$

# Next Fit算法

- **Next Fit 算法的最坏情况界为 2**

- 若存在实例  $I$ ，使得  $\frac{C^{NF}(I)}{C^*(I)} > 2$ 。记  $C^*(I) = k$ ，则  $C^{NF}(I) > 2k$
- 对任意  $i$ ，考虑相继启用的两个箱子  $B_{2i-1}$  和  $B_{2i}$ 。由于放入  $B_{2i}$  中的第一个物品  $l$  无法放入  $B_{2i-1}$  中， $w_l + B_{2i-1} > C$
- 箱子  $B_{2i}$  的最终容量至少为  $w_l$ ， $B_{2i} + B_{2i-1} \geq w_l + B_{2i-1} > C$
- $\sum_{i=1}^k (B_{2i}^n + B_{2i-1}^n) > kC$
- 由于  $C^*(I) = k$ ，所有物品必能放入  $k$  个箱子中， $\sum_{i=1}^k (B_{2i}^n + B_{2i-1}^n) < \sum_{j=1}^n w_j \leq kC$

箱子不能装下当前物品，仍可能装下后面的物品  
NF算法在某些特定背景下仍有应用价值

矛盾





# First Fit 算法



组合优化

- **First Fit 算法**: 将当前物品放在按箱子启用顺序**第一个**能放下的箱子中

- First Fit 算法的最坏情况界为  $\frac{17}{10}$

Simchi-Levi D. New worst case results for the bin-packing problem.

*Naval Research Logistics*, 41, 579–585, 1994

$\frac{7}{4}$

Xia BZ, Tan ZY, Tighter bounds of the First Fit algorithm for the bin-packing problem. *Discrete Applied Mathematics*, 158, 1668–1675, 2010

Boyar J, Dósa G, Epstein L. On the absolute approximation ratio for First Fit and related results. *Discrete Applied Mathematics*, 160, 1914–1923, 2012

$\frac{12}{7}$

Dósa G, Sgall J. First Fit bin packing: A tight analysis. *Proceeding of 30th Symposium on Theoretical Aspects of Computer Science*, 538–549, 2013

$\frac{17}{10}$





# First Fit算法

组合优化



$$C = 101$$

$$C^* = 10$$

$$C^{FF} = 17$$

$$\frac{C^{FF}}{C^*} = \frac{17}{10}$$

Johnson, D.S., Demers, A., Ullman, J. D., Garey, M. R., Graham, R. L.,  
Worst-case performance bounds for simple one-dimensional packing  
algorithms, *SIAM Journal on Computing*, 3, 299-325, 1974





# First Fit Decreasing 算法

- **First Fit Decreasing**算法 (FFD)

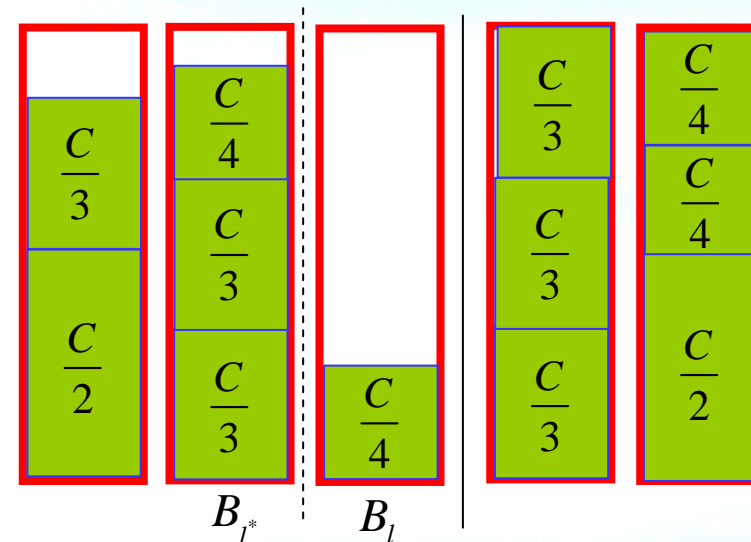
- 将物品按大小从大到小的顺序重新排序, 即有  $w_1 \geq w_2 \geq \dots \geq w_n$
- 按上述顺序用**First Fit** 算法装箱

- **FFD**算法的最坏情况界为  $\frac{3}{2}$

- 记  $C^{FFD}(I) = l$ ,  $C^*(I) = l^*$ . 不妨设  $l^* \geq 2$
- 记被**FFD**算法放入  $B_{l^*+1}, B_{l^*+2}, \dots, B_l$  中的物品集合为  $\mathcal{J}$

- $\mathcal{J}$  中物品大小均不大于  $\frac{C}{3}$
- $\mathcal{J}$  中物品数不超过  $l^* - 1$

- $l - l^* \leq \left\lceil \frac{l^* - 1}{3} \right\rceil$ ,  $\frac{C^{FFD}(I)}{C^*(I)} = \frac{l}{l^*} = 1 + \frac{l - l^*}{l^*} \leq 1 + \frac{\left\lceil \frac{l^* - 1}{3} \right\rceil}{l^*} \leq 1 + \frac{\frac{l^* - 1}{3} + 1}{l^*} = \frac{4}{3} + \frac{1}{3l^*} \leq \frac{3}{2}$



$$C^{FFD} = 3 \quad C^* = 2 \quad \frac{C^{FFD}}{C^*} = \frac{3}{2}$$

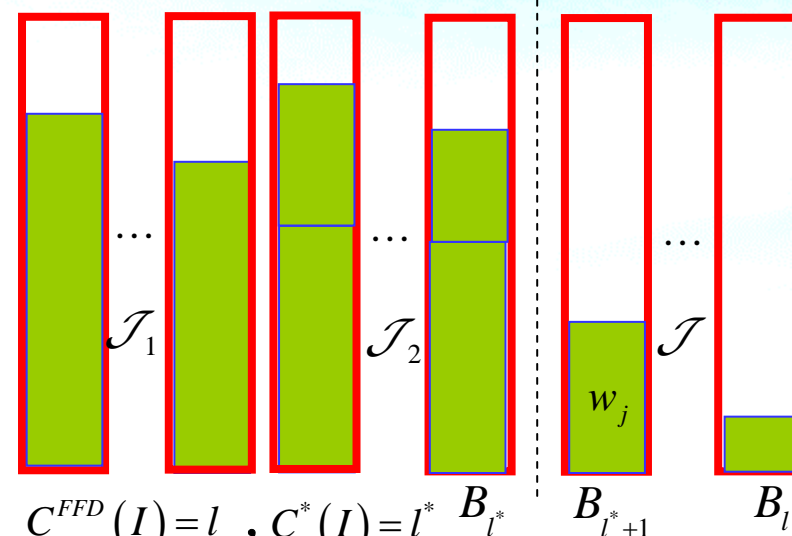


# FFD算法

## 组合优化

- $\mathcal{J}$  中物品大小不大于  $\frac{C}{3}$ 
  - 若  $w_j > \frac{C}{3}$ , 装入  $j$  时前  $l^*$  个箱子中每个箱子至多装入一个物品
  - 物品  $j$  未放入前  $l^*$  个箱子中
    - 最优解中  $\mathcal{J}_1$  中任一物品不能和前  $j$  个物品中的任一个同装一个箱子
    - 最优解中任一箱子至多只能装入两个  $\mathcal{J}_2$  中的物品
    - 最优解中无法用  $l^*$  个箱子放入前  $j$  个物品
- $\mathcal{J}$  中物品数不超过  $l^* - 1$ 
  - 若  $\mathcal{J}$  中至少有  $l^*$  个物品
  - $\mathcal{J}$  中最小的  $l^*$  个物品未放入前一个箱子中,  $B_i + w_{j_i} > C, j=1, \dots, l$

$$\sum_{i=1}^{l^*} (B_i + w_{j_i}) > l^* C \geq \sum_{i=1}^n w_i \quad \text{矛盾}$$



$j$  为装入后  $l-l^*$  个箱子中的最大物品, 它也是前  $j$  个物品中的最小物品  
 $\mathcal{J}_1, \mathcal{J}_2$  分别为装入  $j$  时前  $l^*$  个箱子中已装有 1 个和 2 个物品的箱子中所装入的物品集合

$\mathcal{J}$  中大小最小的  $l^*$  个物品为  $j_1, \dots, j_{l^*}$





# 难近似性

- 对一维装箱问题, 不存在最坏情况界小于  $\frac{3}{2}$  的多项式时间近似算法, 除非  $P=NP$ 
  - 若一维装箱存在最坏情况界小于  $\frac{3}{2}$  的多项式时间近似算法  $H$
  - 任取划分问题的实例  $I_P: A = \{a_1, a_2, \dots, a_n\}$ 。构造装箱问题实例  $I_B: n$  个物品, 物品  $j$  的大小为  $a_j$ , 箱子容量为  $C = \frac{1}{2} \sum_{j=1}^n a_j$ 
    - 若  $I_P$  的答案为“是”, 则存在子集  $A_1, A_2$ , 使得  $A = A_1 \cup A_2, A_1 \cap A_2 = \emptyset$ , 且满足  $\sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i = C$ , 子集  $\mathcal{J}_i = \{j \mid a_j \in A_i\}$  中的物品可装入一个箱子, 故  $C^*(I_B) = 2$
    - 若  $C^*(I_B) = 2$ , 由于所有物品大小之和为  $2C$ , 两个箱子中所装物品大小之和均为  $C$ 。记  $\mathcal{J}_1, \mathcal{J}_2$  为装入两个箱子的物品集, 令  $A_i = \{a_j \mid j \in \mathcal{J}_i\}$ , 则  $A_1, A_2$  为  $A$  的一个划分,  $I_P$  的答案为“是”
  - 用算法  $H$  求解实例  $I_B$ 
    - 若  $C^H(I_B) \geq 3$ , 则  $C^*(I_B) > 2$ ,  $I_P$  的答案为“否”; 若  $C^H(I_B) = 2$ , 则  $C^*(I_B) = 2$ ,  $I_P$  的答案为“否”, 由  $H$  可给出划分问题的多项式时间算法

矛盾



浙江大学

Zhejiang University

# FFD的紧性

组合优化

FFD算法是否为一维装箱问题最坏情况界意义下的最好算法?

不存在最坏情况界小于  $\frac{3}{2}$  的多项式时间近似算法

$\frac{C^{FFD}}{C^*} \leq \frac{3}{2}$  且存在实例  $I$ ,  $\frac{C^{FFD}(I)}{C^*(I)} = \frac{3}{2}$

是否存在实例  $I$ ,  $C^{FFD}(I) = 6, C^*(I) = 4$

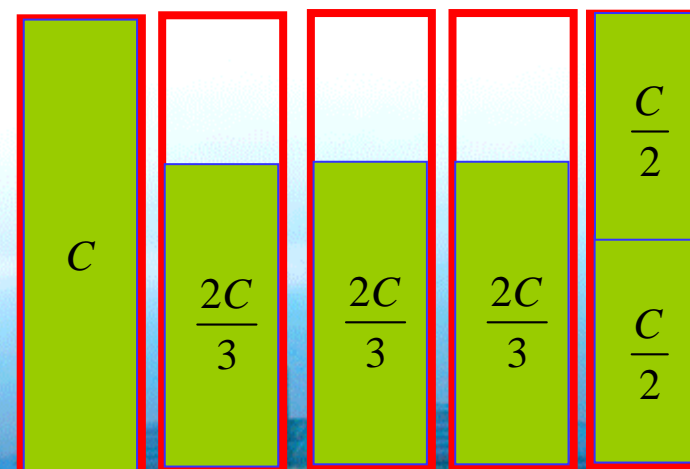
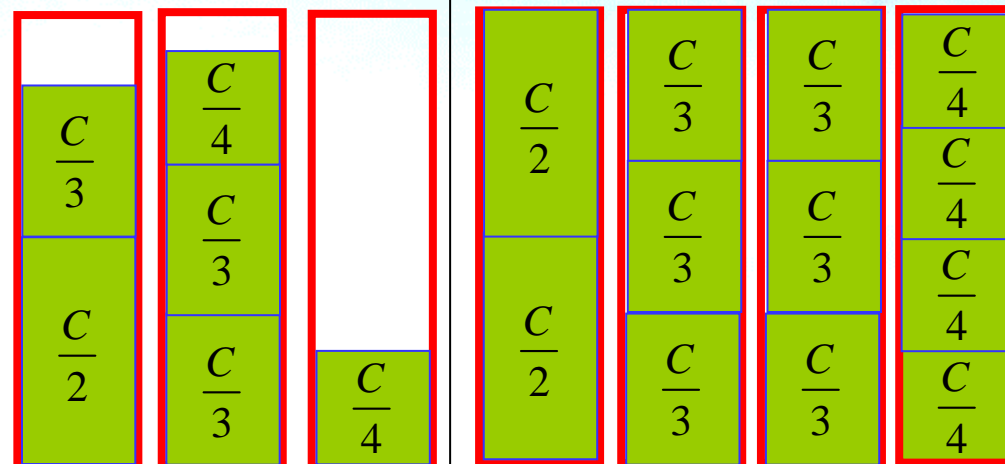
LS 算法求解  $P2 \parallel C_{\max}$  的最坏情况界为  $\frac{3}{2}$

1	2
1	

$$C^{LS}(I) = 3, C^*(I) = 2$$

2	4
2	

$$C^{LS}(I) = 6, C^*(I) = 4$$





# 渐近性能比

- 算法  $A$  的渐近性能比 (asymptotic performance ratio) 为

$$r_A^\infty = \inf \left\{ r \geq 1 \mid \text{存在整数 } N, \text{使得 对任意满足 } C^*(I) \geq N \text{ 的实例 } I, \frac{C^A(I)}{C^*(I)} \leq r \right\}$$
$$= \inf_N \inf \left\{ \frac{C^A(I)}{C^*(I)} \leq r, \text{对任意满足 } C^*(I) \geq N \text{ 的实例 } I \right\}$$

- 若存在与实例无关的常数  $c$ , 使得对所有的实例  $I$ ,  $C^A(I) \leq kC^*(I) + c$ , 则  $A$  的渐近性能比不超过  $k$
- 相应称  $r_A = \inf \{ r \geq 1 \mid C^A(I) \leq rC^*(I), \forall I \}$  为算法  $A$  的绝对性能比 (absolute performance ratio)。对任意算法  $A$ ,  $r_A^\infty \leq r_A$ 。  $r_A^\infty < r_A$

的情况确实在某些问题或某些问题的某个算法中存在

绝对性能比与渐近性能比均是最坏情况意义下算法性能的度量,

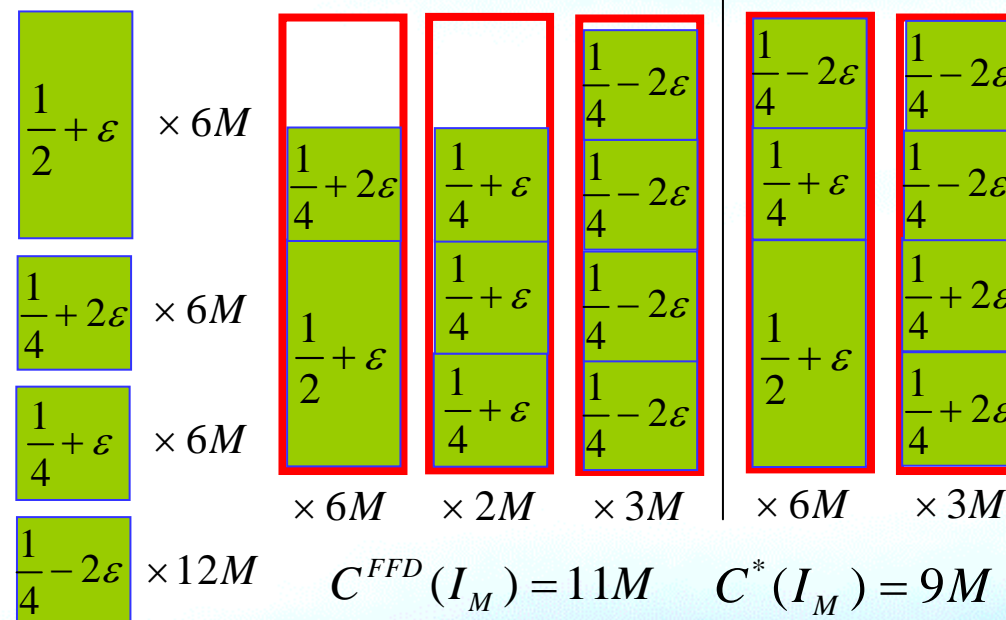
渐近性能比允许忽略算法在最优值较小的那些实例上的性能

# 渐近性能比的紧性

- 为证明渐近性能比的紧性，需给出包含有最优值趋于无穷的实例，以确保渐近性能比不能通过增加右端常数项而减小

- 仅有  $C^A(I) = 20, C^*(I) = 10$  的实例  $I$  不足于说明  $A$  的渐近性能比至少为 2，如可能为  $C^A(I) \leq \frac{9}{5}C^*(I) + 2$  或  $C^A(I) \leq \frac{7}{5}C^*(I) + 6$

- FFD**算法的渐近性能比至少为  $\frac{11}{9}$



Johnson DS, Demers A, Ullman JD, Garey MR, Graham RL, Worst-case performance bounds for simple one-dimensional packing algorithms, *SIAM Journal on Computing*, 3, 299-325, 1974



# FFD算法的渐近性能比



浙江大学  
Zhejiang University

组合优化

- FFD 算法的渐近性能比为  $\frac{11}{9}$   $C^{FFD} \leq \frac{11}{9} C^* + 4 \rightarrow 3 \rightarrow 1 \rightarrow \frac{6}{9}$  FF 算法的渐
- FF 算法的渐近性能比为  $\frac{17}{10}$   $C^{FF} \leq \frac{17}{10} C^* + 3 \rightarrow 2 \rightarrow \frac{9}{10} \rightarrow \frac{7}{10} \rightarrow 0$  近性能比等于  
绝对性能比

Johnson DS, *Near-optimal bin packing algorithms*, Doctoral Thesis, Massachusetts Institute of Technology, 1973

Baker BS, A new proof for the first-fit decreasing bin-packing algorithm, *Journal of Algorithms*, 6, 49-70, 1985

Yue, M, A simple proof of the inequality  $FFD(L) \leq \frac{11}{9} OPT(L) + 1, \forall L$  for the FFD bin-packing algorithm, *Acta Mathematica Applicata Sinica*, 7, 321-331, 1991

Dósa, G, The tight bound of first fit decreasing bin-packing algorithm is  $FFD(L) \leq \frac{11}{9} OPT(L) + \frac{6}{9}$ , *Lecture Notes in Computer Science*, 4614, 1-11, 2007

Ullman JD, The performance of a memory allocation algorithm, Technical Report 100, Princeton University, 1971

Johnson DS, Demers A, Ullman JD, Garey MR, Graham RL, Worst-case performance bounds for simple one-dimensional packing algorithms, *SIAM Journal on Computing*, 3, 299-325, 1974

Garey MR, Graham RL, Johnson DS, Yao ACC, Resource constrained scheduling as generalized bin packing, *Journal of Combinatorial Theory(A)*, 21, 257-298, 1976

Xia BZ, Tan ZY, Tighter bounds of the First Fit algorithm for the bin-packing problem. *Discrete Applied Mathematics*, 158, 1668-1675, 2010

Dósa G, Sgall J. First Fit bin packing: A tight analysis. *Proceeding of 30th Symposium on Theoretical Aspects of Computer Science*, 538-549, 2013



# 渐近近似方案

- 渐近多项式时间近似方案 (Asymptotic Polynomial Time Approximation Scheme, APTAS)

$C^{A_\varepsilon} \leq (1 + \varepsilon)C^* + 1$  时间复杂性为  $n$  的多项式, 但为  $\frac{1}{\varepsilon}$  的指数函数

- 渐近完全多项式时间近似方案 (Asymptotic Fully Polynomial Time Approximation Scheme, AFPTAS)

$C^{A_\varepsilon} \leq (1 + \varepsilon)C^* + O\left(\frac{1}{\varepsilon^2}\right)$  时间复杂性为  $n$  与  $\frac{1}{\varepsilon}$  的多项式

Fernandez de la Vega W, Lueker GS, Bin packing can be solved within in linear time, *Combinatorica*, 1, 349-355, 1981

Karmarkar N, Karp RM, An efficient approximation scheme for the one-dimensional bin-packing problem, *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, 312-320, 1982



# 绝对近似与有限类型

- 存在多项式时间算法，使得  $C^A \leq C^* + O(\log C^* \log \log C^*)$ 
  - 是否存在多项式时间算法，使得  $C^A \leq C^* + \text{const}$  仍是未知的
- 若物品大小共有  $d$  种，大小为  $s_i$  的物品有  $a_i$  个，则存在时间复杂性为  $\left( \log \left( \max_{i=1, \dots, d} \{s_i, d_i\} \right) \right)^{2^{O(d)}}$  的最优算法

**Rothvoß T, Approximating bin packing within  $O(\log OPT \cdot \log \log OPT)$  bins. *Proceedings of 54th IEEE Annual Symposium on Foundations of Computer Science*, 20-29, 2013.**

**Goemans MX, Rothvoß T, Polynomiality for bin packing with a constant number of item types, *Proceedings of the 25<sup>th</sup> Annual ACM-SIAM Symposium on Discrete Algorithms*, 830-839, 2014.**





浙江大学

Zhejiang University

组合优化

谢 谢