# 18 - Approx Algs & Load Ballancing

## 为什么近似?

- 对于某些问题（例如背包问题），我们不期望找到多项式时间算法
- 我们应该如何处理这些问题?
- 我们可以设计近似算法，这些算法:
  - 运行在多项式时间内
  - 计算一个接近最优解的解

## 挑战

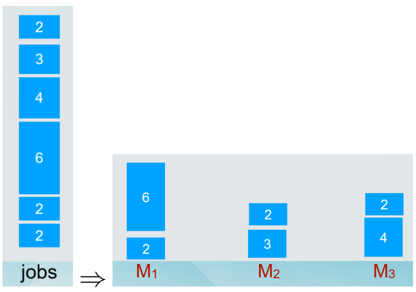
- “接近”最优解是什么意思？我们如何衡量?
  - 使用近似比率
- 如果我们不能真正找到最优解，我们如何提出这样的论证?
  - 通过上下界约束最优解
- 我们如何知道我们的算法是最好的？我们能否“更接近”最优解?

## 近似算法的方法

- 贪心算法（Greedy algorithms）
- 定价方法（Pricing method，也称为原始-对偶方法（Primal-Dual method））
- 线性规划和取整（Linear Programming and Rounding）
- 在四舍五入输入上的动态规划（Dynamic Programming on rounded inputs）

# 负载均衡（Load Balancing）

- 我们有一组 $m$ 个相同的机器 $M_1, \ldots, M_m$
- 我们有一组 $n$ 个作业，每个作业 $j$ 的处理时间为 $t_j$
- 我们希望将每个作业分配给某台机器
- 设 $A(i)$ 为分配给机器 $i$ 的作业集
- 机器 $i$ 的负载为 $T_i = \sum_{j \in A(i)} t_j$
- 目标是最小化最大完成时间，即使得 $T = \max_i T_i$




makespan = 8（负载最大的那个机器的负载叫做 makespan）

- 在相同机器上的负载均衡问题是 NP-hard 的
- 我们将为此设计贪心近似算法

# 贪心算法

- 选择任何作业
- 将其分配给负载最小的机器
- 从作业堆中移除它

Greedy-Balance 算法：

1. 开始时没有分配任何作业
2. 对所有机器 $M_i$ 设定 $T_i = 0$ 且 $A(i) = \emptyset$
3. 对于 $j = 1, \ldots, n$：
    - 令 $M_i$ 为实现最小 $T_k$ 的机器
    - 将作业 $j$ 分配给机器 $M_i$
    - 设 $A(i) = A(i) \cup \{j\}$
    - 设 $T_i = T_i + t_j$
4. 结束循环

我们设：

- $T$ 是 Greedy-Balance 算法达到的 makespan
- $T^*$ 是最优的 makespan

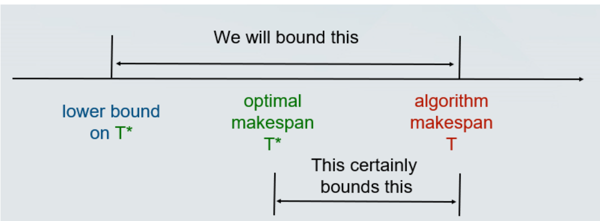
## 最优性

挑战：我们不知道 $T^*$，应该如何论证：

- 我们想证明 $T$ 离 $T^*$ 不远
- 我们将要证明 $T$ 离某个小于 $T^*$ 的东西不远
- 那么它显然离 $T^*$ 不远

近似算法分析中的基本技术：

- 从下界约束最优解（对于最小化问题）和从上界约束最优解（对于最大化问题）




## 对最优解的下界进行约束

### 第一个约束

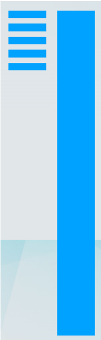
我们可以使用什么边界来约束最优解?

- 考虑所有作业的总处理时间（处理时间 $t_j$ 的总和）
- $m$ 台机器中的一台必须分配至少 $1/m$ 份的总工作量
- 因此，我们有：

$$T^* \geq \frac{1}{m} \sum_{j=1}^{n} t_j$$

这是一个好的边界吗？

- 这个边界在作业处理时间相当相似的情况下可能是好的
  - 取处理时间的平均就意味着假设这个作业可以被拆分
  - 实际上，OPT 会将这个作业分配给某台机器，且最大完成时间将是它的处理时间
  - 我们的边界假设 OPT 大约比它好 m 倍


（图中，其中一个作业比其他作业的处理时间多非常多）

## 第二个约束

我们能想到另一个边界吗？

- 每个作业必须被分配到某台机器
- 最大完成时间肯定至少是任何作业的最大处理时间 $t_j$
- 我们有：

$$T^* \geq \max_j t_j$$

这是一个好的边界吗？

- 在上个例子中，这是一个好的边界，因为最大处理时间非常大
- 在其他情况下，这可能不是一个很好的边界（任务很多，每个机器被分配多个任务）
- 但是我们实际上会使用两个边界

## 边界与证明

算法 Greedy-Balance 产生的作业分配使得最大完成时间：

$$T \leq 2T^*$$

## 第一个边界

假设 $M_i$ 是最后一个被分配的机器，$j$ 是最后一个被分配的作业，$T_i$ 是机器 $M_i$ 的负载
那么在最后一个分配发生之前，$M_i$ 的负载为 $T_i - t_j$
而其他的每台机器的负载至少为 $T_i - t_j$（如果有机器的负载比这个小，那么 $j$ 会被分配被这个机器）
因此我们得到

$$\sum_k T_k \geq m(T_i - t_j) \Rightarrow T_i - t_j \leq \frac{1}{m} \sum_k T_k$$

根据第一个约束，我们得到

$$T_i - t_j \leq T^*$$

## 第二个边界

假设 $M_i$ 是最后一个被分配的机器，$j$ 是最后一个被分配的作业，$T_i$ 是机器 $M_i$ 的负载
那么在最后一个分配发生之前，$M_i$ 的负载为 $T_i - t_j$

添加最后一个作业之后，负载变为 $T_i - t_j + t_j$
显然

$$t_j \leq \max_k t_k \leq T^*$$

$t_j$ 显然小于等于最大的一个负载，而最大的负载显然小于等于最优 makespan

## 考虑两个边界

考虑到

$$T_i - t_j \leq T^*$$
$$t_j \leq T^*$$

我们便得到了

$$T_i \leq 2T^*$$

根据我们的假设，$T_i$ 是拥有最大负载的机器，因此

$$T \leq 2T^*$$

## 我们的边界是紧的吗?

- 我们已经证明了 Greedy-Balance 解决方案的最大完成时间至多比最优解多一个因子2
- 我们能否证明在最坏情况下它至少也比最优解多一个因子2?
  - 换句话说，是否存在负载均衡问题的一个实例，使得算法实际上生成的最大完成时间是最优解的两倍?
  - 换句话说，我们对算法的分析是否是紧的?

**考虑一个最坏的情况:**
我们拥有 $m(m-1)$ 个小作业，每个作业的处理时间都是 1
我们还拥有一个大作业，大作业的处理时间是 $m$

Greedy-Balance 算法会首先将这些小作业分配给每一个机器，然后才将大作业分配给其中一台机器
因此这 $m$ 台机器，每台的运行时间在分配最后一个大作业前，都是 $m-1$
因此其 makespan 是 $2m-1$

而最优的分配显然是将最大的作业分配给一个机器，然后将小作业分配给剩余机器
这样做的 makespan 是 $m$

所以我们的边界是紧的

## Approximation Ratio 近似比率

考虑一个最小化问题 $P$ 和一个目标函数 $\text{obj}$

- 在这里：在理想机器上进行负载均衡与 makespan

- 考虑一个近似算法 $A$
- 考虑问题 $P$ 的一个输入 $x$
- 令 $\text{obj}(A(x))$ 为算法 $A$ 在输入 $x$ 上求解所得的目标值
- 令 $\text{opt}(x)$ 为输入 $x$ 上目标函数的最小可能值

那么，算法 $A$ 的近似比率定义为

$$\max_x \frac{\text{obj}(A(x))}{\text{opt}(x)}$$

即，算法在所有可能输入下的目标值与最优目标值的最坏情况下的比率

这意味着：

- 为了证明近似比率的上界，我们必须以某种方式论证问题的所有输入
- 为了证明近似比率的下界（对于特定算法），我们必须论证问题的一个输入

因此，对于最大化问题，我们可以定义

$$\max_x \frac{\text{opt}(x)}{\text{obj}(A(x))}$$

对于最小化问题，我们可以定义

$$\max_x \frac{\text{obj}(A(x))}{\text{opt}(x)}$$

习惯上，近似比率总是大于等于 1

## 更好的贪心算法（Sorted-Balance）

- 将作业按处理时间的非递增顺序排序（其实就是递减）
- 按照这个顺序选择一个作业
- 将其分配给当前负载最小的机器
- 从作业堆中移除它

## 第三个边界

假设我们有多于 $m$ 个作业，那么 $T^* \geq 2t_{m+1}$：

- 我们现在处于排序平衡算法的前提下
- 考虑前 $m+1$ 个作业，每个作业至少需要 $t_{m+1}$ 时间（不然这些作业就不会排在前 $m+1$ 了）
- 由于我们有 $m$ 台机器，那么必然有一台机器至少接受两个作业
  - 那么这台机器的负载至少是 $2t_{m+1}$ （它肯定至少得比最小的大吧）

于是我们得到了

$$T^* \geq 2t_{m+1}$$

## 新的边界

对于算法 Sorted-Balance 产生的作业分配，其最大完成时间是

$$T \leq \frac{3}{2}T^*$$

## 证明

假设

- $M_i$ 是最后一个被分配的机器
  - 那么 $M_i$ 被分配了至少两个作业
  - （因为如果只被分配了一个，那么这个作业就是最优的）
- $j$ 是最后一个被分配的作业，$j \geq m+1$
- $T_i$ 是机器 $M_i$ 的负载

因此，$t_j \leq t_{m+1} \leq \frac{1}{2}T^*$
即

$$t_j \leq \frac{1}{2}T^*$$

于是我们根据第一个边界，可以得到

$$T \leq \frac{3}{2}T^*$$

# 算法提供的近似比率

Sorted-Balance 算法实际上通过更好的分析提供了 $4/3$ 的近似比率

对于相同机器上的负载均衡问题，存在一个多项式时间近似方案（PTAS）:一个给定输入和常数参数 $\epsilon$ 的算法，在多项式时间内运行，并产生一个与最优解相差不超过 $(1+\epsilon)$ 的结果