



近似算法

2020年6月8日

目录

Part1 NP-完全性理论

Part2 近似算法

1 NP-完全性理论

计算机科学的局限性

可解性：问题及其可解性可用函数和可计算性来代替

可计算性理论：研究计算的一般性质的数学理论，它通过建立计算的数学模型（例如抽象计算机），精确区分哪些是可计算的，哪些是不可计算的。

可计算函数：能够在抽象计算机上编出程序计算其值的函数。这样就可以讨论哪些函数是可计算的，哪些函数是不可计算的。

Church-Turing论题：若一函数在某个合理的计算模型上可计算，则它在图灵机上也是可计算的。

不可计算性：很多问题和函数是无法用具有有穷描述的过程完成计算

停机问题

停机问题：能否写一个程序正确判定输入给它的任何一个程序是否会停机？

设程序halts(P,X)总是正确地判定程序P在其输入X上是否停机：若停机，则返回yes；否则死循环，返回no。设另有一程序：

```
diagonal(Y){  
    a: if halts(Y,Y) then  
        goto a;  
    else halt;  
}
```

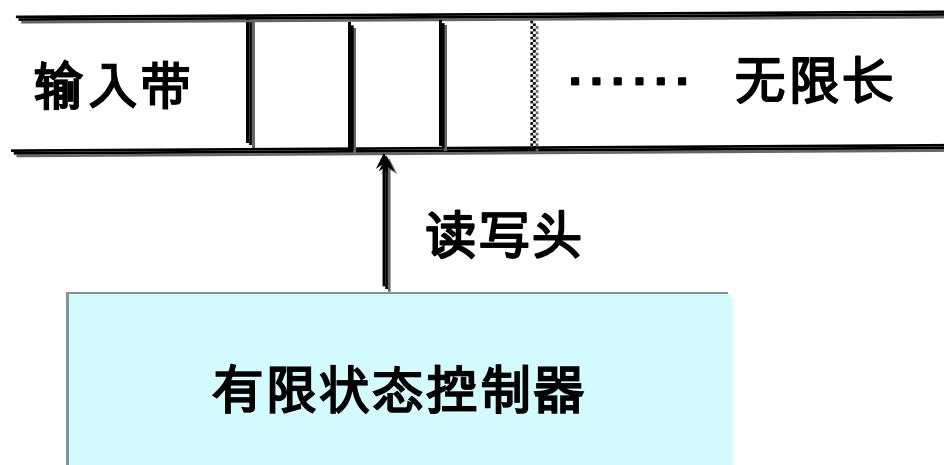
功能：若halts断定当程序Y用其自身Y作为输入时Y停机,则diagonal(Y)死循环；否则它停机

diagonal (diagonal) 是否停机？ **不可判定**

它停机当且仅当halts(diagonal,diagonal)返回否，也就是：
diagonal停机当且仅当它自己不停机，矛盾！

即：halts(P,X)并不存在，停机问题是不可解的！

图灵机



有单带、多带等变种，计算能力等价

依据控制器的状态和读写头所读字符，决定执行以下3个操作之一、之二或全部：

- 1) 改变有限状态控制器中的状态；
- 2) 读写头在相应的方格中写一符号；
- 3) 读写头左、右移一格或不动。

确定型图灵机DTM：若对任一个状态和符号，要执行的动作都是唯一的

非确定型图灵机NTM：若执行的动作是有穷多个可供选择

P、NP及NPC类问题

P类问题：一类问题的集合，对其中的任一问题，都存在一个**确定**型图灵机 M 和一个多项式 p ，对于该问题的任何（编码）长度为 n 的实例， M 都能在 $p(n)$ 步内，给出对该实例的回答。即：**多项式时间内可解的问题**

NP类问题：一类问题的集合，对其中的任一问题，都存在一个**非确定**型图灵机 M 和一个多项式 p ，对于该问题的任何（编码）长度为 n 的实例， M 都能在 $p(n)$ 步内，给出对该实例的回答。

若NTM在每一步都恰有2步可供选择，则回答实例需考察 $2^{p(n)}$ 种不同的可能性。

存在多项式时间的算法吗？

多项式时间内可验证问题（指验证其解的正确性）

P、NP及NPC类问题

多一归约

假设 L_1 和 L_2 是两个判定问题， f 将 L_1 的每个实例 l 变换成 L_2 的实例 $f(l)$ 。若对 L_1 的每个实例 l ， l 的答案为“是”当且仅当 $f(l)$ 是 L_2 的答案为“是”的实例，则称 f 是从 L_1 到 L_2 的多一归约，记作： $L_1 \leq_m L_2$ (传递关系)

直观意义：将求解 L_1 的问题转换为求解 L_2 的问题，而问题 L_1 不会难于 L_2

多项式时间多一归约：若 f 是多项式时间可计算，则上述归约称为多项式时间多一归约，也称多项式时间变换。记作：

$$L_1 \leq_m^p L_2$$

P、NP及NPC类问题

NPC问题：对于一个(判定性)问题 q ，若

(1) $q \in \text{NP}$

(2) NP中任一问题均可多项式时间多一归约到 q

则称问题 q 为NP-完全的(NP-complete, NPC)

NP-hard问题：若问题 q 仅满足条件(2)而不一定满足条件(1)，则问题 q 称为NP-难的(NP-hard, NPH)。显然： $\text{NPC} \subseteq \text{NP-hard}$

NPC和NP-hard关系

NP-hard问题至少跟NPC问题一样难。

NPC问题肯定是NP-hard的，但反之不一定

例：停机问题是NP-hard而非NPC的。

∴ 该问题不可判定，即无任何算法(无论何复杂度)求解该问题

∴ 该问题 $\notin \text{NP}$ 。但是

可满足问题 $\text{SAT} \leq_p \text{停机问题}$

P、NP及NPC类问题

NP = ? P

∵ 确定型图灵机是非确定型图灵机的特例, ∴ $P \subseteq NP$

是否有 $NP \subseteq P$? 即是否 $NP = P$?

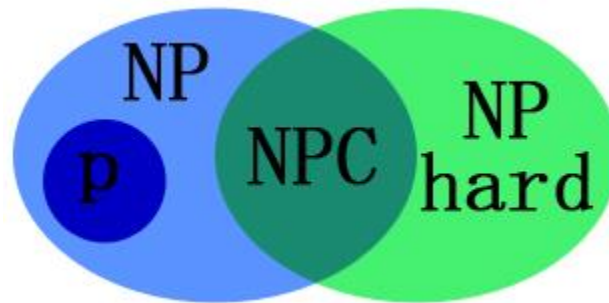
美国麻省Clay数学研究所于2000年5月24日在巴黎法兰西学院宣布: 对七个“**千年数学难题**”中的每一个均悬赏100万美元, 而问题 $NP = ? P$ 位列其首:

1. **P问题对NP问题**
 2. 霍奇猜想
 3. 庞加莱猜想 (2002. 11–2003. 7, 俄罗斯数学家佩雷尔曼在3篇论文预印本中证明了几何化猜想, 2006被授予菲尔兹奖)
 4. 黎曼假设
 5. 杨-米尔斯存在性和质量缺口
 6. 纳维叶-斯托克斯方程的存在性与光滑性
 7. 贝赫和斯维纳通-戴尔猜想
-

P、NP及NPC类问题

P、NP、NPC和NP-hard之关系

NPC是NP中最难的问题，但是NP-hard至少与NPC一样难



如何证明问题q是NP-hard或是NPC的？

若已知 $q' \in \text{NPC}$ 或 $q' \in \text{NPH}$ ，且 $q' \leq_p q$ ，则 $q \in \text{NPH}$ ；若进一步有 $q \in \text{NP}$ ，则 $q \in \text{NPC}$ 。

即：要证q是NPH的，只要找到1个已知的NPC或NPH问题 q' ，然后将 q' 多项式归约到q即可。若还能验证 $q \in \text{NP}$ ，则q是NPC的。

\therefore NP中任意问题均可多项式归约到 q' ，由于 \leq_p 有传递性

\therefore 他们也都能多项式归约到q，由定义可知q是NPH的

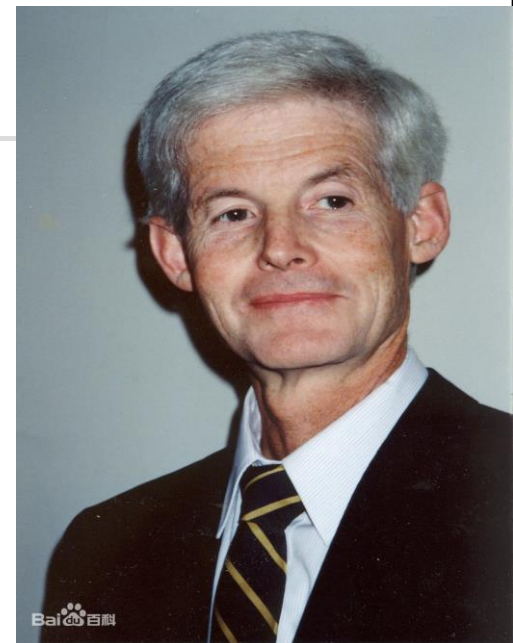
NP-完全性理论

Cook的贡献：第一个NPC问题

史提芬·库克(Stephen Arthur Cook, 1939 -) NP完全性理论的奠基人，他在1971年论文“The Complexity of Theorem Proving Procedures”中，给出了第一个NP完备的证明，即**Cook定理**：可满足性问题 (Satisfiability problem) 是NP完全问题，亦即 $SAT \in NPC$ 。且证明了：

$SAT \in P$ 当且仅当 $P=NP$

Cook于1961年获Michigan大学学士学位，1962和1966年分获哈佛大学硕士与博士学位。1966-1970，他在UC Berkeley担任助教授；1970年加盟多伦多大学，现为该校CS和数学系教授，他的论文开启了NP完备性的研究，令该领域于之后的十年成为计算机科学中最活跃和重要的研究。因其在计算复杂性理论方面的贡献，尤其是在奠定NP完全性理论基础上的突出贡献而荣获1982年度的图灵奖。



NP-完全性理论

Karp的贡献

理查德·卡普 (Richard Karp, 1935-) 1972年论文 "Reducibility among Combinatorial Problems" 发展和加强了由库克提出的 "NP完全性" 理论。尤其是库克仅证明了命题演算的可满足问题是NP完全的，而卡普则证明了从组合优化中引出的**大多数经典问题** (背包问题、覆盖问题、匹配问题、分区问题、路径问题、调度问题等) **都是NP完全问题**。只要证明其中任一个问题是属于P类的，就可解决计算复杂性理论中最大的一个难题，即 $P=?NP$ 。

Karp于1955、1956和1959年分别获哈佛大学文学学士、理学硕士和应用数学博士学位，现任UC Berkeley计算机科学讲座教授，美国科学院、美国工程院、美国艺术与科学院、欧洲科学院院士。因其在计算机科学领域的基础贡献曾获图灵奖(1985)、冯诺依曼奖、美国国家科学勋章、哈佛大学百年奖章等奖项。



NP-完全性理论

NP-完全性理论的局限性

易解问题：可多项式时间内求解的问题

难解问题：需超多项式时间求解的问题

NP-完全性理论既没有找到第二类问题的多项式时间的算法，也没有证明这样的算法就不存在，而是证明了这类问题计算难度之等价性（彼此间困难程度相当）。因此，NPC具有如下性质：若其中1个问题多项式可解当且仅当其他所有NPC问题亦多项式可解

难解问题与易解问题之相似性

1) 最短/最长简单路径

单源最短路径问题：对有向图G，时间 $O(VE)$ ，P问题

两点间最长路径：NPC问题，即使所有边上权为1

2) 欧拉环/哈密尔顿圈（G为无向图或有向图）

欧拉环：G中有通过每条边恰好一次的环？P，多项式时间可解

哈氏圈：G中有通过每个顶点恰好1次的圈？NPC

NP完全问题的求解

减少搜索量

简单算法是穷举搜索，时间为指数

减少搜索量：分枝限界法，隐枚举法、动态规划等
可以提高效率，但时间复杂度不变

优化问题

降低优化要求，求近似解，以得到一个多项式时间的算法。即：找寻在容许的时间内得到容许精度的近似最优解的算法

2 近似算法



Ch.1 导论

现实中许多优化问题是NP-hard的，由复杂性理论知：若 $P \neq NP$ (很可能为真)，就不可能找到**多项式时间**的算法来对问题的**所有输入实例**求出**最优解**。但若放松要求，就可能存在有效求解算法。

实用中可考虑3种放宽要求的可能性：

1. 超多项式时间启发

不再要求多项式时间算法，有时求解问题存在超多项式时间算法，实用中相当快。例如，0/1背包问题是NPC问题，但存在1个伪多项式时间算法很容易解决它。

缺点：该技术只对少数问题有效。

Ch.1导论

2. 启发式概率分析

不再要求问题的解满足所有的输入实例。在某些应用中，有可能输入实例的类被严格限制，对这些受限实例易于找到其有效算法。而这种结果往往归功于对输入实例约束的概率模型。

缺点： 选取一个特殊的输入分布往往是不易的。

3. 近似算法

不再要求总是找到最优解。在实际应用中有时很难确定一个最优解和近似最优解(次优解)之间的差别，因问题的输入实例数据本身就可能是近似的。

设计一个算法能够求出所有情况下的次优解来解NP-hard问题往往是真正有效的手段。

Ch.1 导论

优化问题近似解分类

1) 容易近似

Knapsack, Scheduling, Bin Packing等

2) 中等难度

Vertex Cover, Euclidean TSP, Steiner Trees等

3) 难于近似

Graph Coloring, TSP, Clique等

这类问题即使找到很差的近似解也是NP-hard

§1.1 预备知识和基本定义

Def1.1 一个优化问题 Π 由三部分构成：

- n 实例集 D ：输入实例的集合
- n 解集 $S(I)$ ：输入实例 $I \in D$ 的所有可行解的集合
- n 解的值函数 f ：给每个解赋一个值， $f: S(I) \rightarrow \mathbb{R}$

一个最大值优化问题 Π 是：

对于给定的 $I \in D$ ，找一个解 $\sigma_{opt}^I \in S(I)$ 使得：

$$\forall \sigma \in S(I), f(\sigma_{opt}^I) \geq f(\sigma)$$

此最优解的值称为 $OPT(I)$ ，即： $OPT(I) \triangleq f(\sigma_{opt}^I)$
有时不太严格地可称其为最优解

§1.1 预备知识和基本定义

装箱问题 (BP)

非形式地，给定一个size在0,1之间的项的集合，要求将其放入单位size的箱子中，使得所用的箱子数目最少。故有最小优化问题：

1)实例： $I=\{s_1, s_2, \dots, s_n\}$, 满足 $\forall i, s_i \in [0,1]$

2)解： $\sigma=\{B_1, B_2, \dots, B_k\}$ 是I的一个不相交的划分，使得

$$\forall i, B_i \subset I \text{ 且 } \sum_{j \in B_i} s_j \leq 1$$

3)解的值：使用的箱子数, 即 $f(\sigma)=|\sigma|=k$

最小优化问题是求最小的k

§1.1 预备知识和基本定义

约定

1) f 的值域和 I 里的所有数是整数

易于扩展至有理数

2) 对任何 $\sigma \in S(I)$, $f(\sigma)$ 受囿于多项式(对 I 中数的size)

只讨论NP完全的优化问题, 因为它易被转换为判定问题

Def1.2 若一个NPH判定问题 Π_1 是多项式可归约为计算一个优化问题 Π_2 的解, 则 Π_2 是NPH的。

典型情况是问题 Π_1 是问题 Π_2 的判定版本。若 Π_2 是最大值问题, 则 Π_1 的形式为:

“是否存在 $\sigma \in S(I)$, 使得 $f(\sigma) \geq k$?”

§1.1 预备知识和基本定义

Def1.3 一个近似算法A，是一个**多项式时间**求解优化问题 Π 的算法，使得对一个给定的 Π 的输入实例I，它输出某个解 $\sigma \in S(I)$ 。通常，我们用A(I)表示算法A所获得的解的值 $f(\sigma)$ 。（有时不太严格地也可表示其解）

近似算法的性能

算法质量(measure of goodness)是在最优解和近似解之间建立某种关系，这种度量也称为**性能保证**(Performance guarantees)。

§1.2 绝对性能保证

在可行的时间内，求装箱问题的最优解是不可能的，但可求次最优解是多少？显然，比最优解多使用1个箱子的解是次最优的。一般地，我们希望找到1个近似解，其值与最优解的值相差某一小的常数。

Def1.4：绝对性能度量 一个绝对近似算法是优化问题 Π 的多项式时间近似算法 A ，使得对某一常数 $k>0$ 满足：

$$\forall I \in D, |A(I) - \text{OPT}(I)| \leq k$$

显然,我们期望对任何NP-hard问题都有一个绝对近似算法，但是对于大多数NP-hard问题，仅当 $P=NP$ 时，才能找到绝对近似算法(指多项式时间)。

§1.2.1 绝对近似算法

1、图的顶点着色

使用**最少**的颜色数来为图G的顶点上色，使得所有相邻的顶点均有不同的颜色，即使G是平面图，该问题的判定版本也是NP-hard的，它有1个绝对近似算法。

Th1.1: 判定一个平面图是否可3着色的问题是NPC的。

近似算法A(G) { //对任意平面图G染色

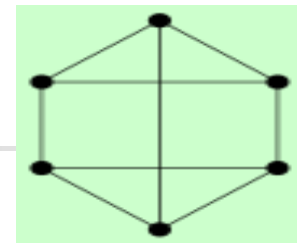
- 1) 检验G是否可2染色(即G是二部图?), 若是则G可2染色;
- 2) 否则, 计算5染色; //可在多项式时间内, \because 任何平面图G是
//可5染色的(实际上四色定理告诉我们G是可4染色的)

} 这就证明了算法A比最优解多用的颜色数不会超过2:

Th1.2: 对给定的任意平面图G, 近似算法A的性能满足:

$$|A(G) - \text{OPT}(G)| \leq 2$$

§1.2.1 绝对近似算法



2、图的边着色

使用**最少**的颜色为图的边上色，使得所有相邻的边有不同的颜色。如下定理 (Vizing) 说明最大度数 Δ 与边着色数之关系：

Th1.3 : 任一图至少需要 Δ ，至多需要 $\Delta+1$ 种颜色为边着色

Vizing定理的证明给出了一个多项式时间的**算法A**找到 $\Delta+1$ 边着色，但令人惊奇的是边着色问题即使是很特殊的情况也是NP-hard的，正如下述的Holyer定理：

Th1.4 : 确定一个3正则平面图所需的边着色数问题是NPH.

综上所述：存在绝对近似算法A，使得下述定理成立：

Th1.5 : 近似算法A有性能保证： $|A(G)-OPT(G)| \leq 1$

§1.2.2 绝对近似算法之否定

前面的例子似乎说明只有很特殊的一类优化问题可能有绝对近似算法：**已知最优解的值**或值所在的小范围。但最优解的值不易确定时是否有绝对近似算法仍然是open.

对于大多数NP-hard问题，存在绝对近似算法(多项式时间)当且仅当存在多项式的精确算法（即：**可在多项式时间内找到最优解**）？

否定结果：证明问题的绝对近似算法的不存在性！

§1.2.2 绝对近似算法之否定

0/1背包问题 问题实例：

| 项 集： $I=\{1,2,\dots,n\}$

| 大 小： s_1, s_2, \dots, s_n

| 利 润： p_1, p_2, \dots, p_n

| 背包容量： B

问题的一个可行解是子集 $I' \subseteq I$, $\sum_{i \in I'} s_i \leq B$

最优解是使得 $f(I') = \sum_{i \in I'} p_i$ 最大的可行解

该问题(0/1背包)是NP-hard的, 除非存在多项式时间算法能够找到最优解, 否则不存在绝对近似算法。

§1.2.2 绝对近似算法之否定

Th.1.6 若 $P \neq NP$ ，则对任何确定的 k ，找不到近似算法 A 可解背包问题使得： $|A(I)-OPT(I)| \leq k$

Pf: 使用扩放法反证。

假定存在算法 A 具有性能保证 k （注意 k 是正整数）

设 $\forall I \in D$ ，可构造新实例 I' 使得

$$s_i = s_i' \quad p_i' = (k+1)p_i \quad \text{for } i \in [1, n]$$

即：除了利润扩放 $k+1$ 倍之外，其余参数不变。故 I 的可行解也是 I' 的可行解，反之亦然。只是解的值相差 $k+1$ 倍。

在 I' 上运行算法 A 获得解 $A(I')$ ，设 A 在实例 I 上的解是 σ ：

$$|A(I')-OPT(I')| \leq k \Rightarrow |(k+1)f(\sigma)-(k+1)OPT(I)| \leq k$$

$$\Rightarrow |f(\sigma)-OPT(I)| \leq k/(k+1) \Rightarrow |f(\sigma)-OPT(I)|=0$$

§1.2.2 绝对近似算法之否定

Th.1.6 若 $P \neq NP$ ，则对任何确定的 k ，找不到近似算法 A 可解背包问题使得： $|A(I) - OPT(I)| \leq k$

Pf: 使用扩放法反证。

假设存在算法 A 具有性能保证： $|A(I) - OPT(I)| \leq k$ (注意： k 是常数)

即：我们已经找到了一个**多项式时间**的算法 A ，它对背包问题的任一输入实例 I ，均能找到最优解。

即：除了利润扩放 $k+1$ 倍之外，其余参数不变。故 I 的可行解也是 I' 的可行解，反之亦然。只是解的值相差 $k+1$ 倍。

在 I' 上运行算法 A 获得解 $A(I')$ ，设 A 在实例 I 上的解是 σ ：

$$|A(I') - OPT(I')| \leq k \Rightarrow |(k+1)f(\sigma) - (k+1)OPT(I)| \leq k$$

$$\Rightarrow |f(\sigma) - OPT(I)| \leq k/(k+1) \Rightarrow |f(\sigma) - OPT(I)| = 0$$

§1.2.2 绝对近似算法之否定

上述证明之关键是Scaling性质，输入实例中数字间线性相关很易使其成立。对非数字问题Scaling性质是否仍然成立？

团(Clique)问题 找图 G 中最大团（最大完全子图），该问题是NP-hard的。

Th.1.7 若 $P \neq NP$ ，则对于团问题不存在绝对近似算法

Pf: 定义图的 m 次幂 G^m ：取 G 的 m 个拷贝，连接位于不同副本里的任意两顶点。

Claim: G 中最大团的size为 α 当且仅当 G^m 里最大团的size是 $m\alpha$ (Ex.)

§1.2.2 绝对近似算法之否定

反证： 设 G 是任意的无向图，近似算法 A 给出的绝对误差是 k 。在 G^{k+1} 上运行 A ，若 G 中最大团size为 α ，则我们有

$$|A(G^{k+1}) - \text{OPT}(G^{k+1})| \leq k \Rightarrow |A(G^{k+1}) - (k+1)\text{OPT}(G)| \leq k$$

对于任给的 G^m 中体积为 β 的团，易于用**多项式时间**在 G 中找到一个体积为 β/m 的团。因此我们能够在 G 中找到一个团 C ，使得：

$$||C| - \text{OPT}(G)| \leq k/(k+1)$$

因为 $|C|$ 和 $\text{OPT}(G)$ 均是整数，故 C 是最优团。

§1.3 相对性能保证

虽然我们渴望得到绝对性能保证，但是较难的优化问题很难找到绝对近似算法。因此，需要放松对“好的近似算法”的要求。

§1.3.1 多机调度

考虑简单的多机调度问题：输入 n 个作业 J_1, J_2, \dots, J_n ，相应的运行时间为 P_1, P_2, \dots, P_n ，设每个 P_i 是**有理数**。将 n 个作业分配到 m 台同样的机器上，以使得完成时间最短。**完成时间**定义为：所有机器上作业运行总时间最长的那一台机器的运行时间。

可行解集合： n 个作业被划分为 m 个子集，一个解的值是所有子集中总运行时间最长的子集的运行时间。该问题即使在 $m=2$ 时也是NP-hard的。

§1.3.1 多机调度

List调度算法(Graham): 将n个作业依次以online的方式分配到m台机器中的某一台上, 规则是将当前作业分配到当时负载最小的机器上, 而机器负载是分配给它的所有作业的总的运行时间。

Th.1.8 设A表示List调度算法, 则对于所有输入实例I

$$\frac{A(I)}{OPT(I)} \leq 2 - \frac{1}{m}$$

界是紧致的, 因为存在一个实例 I^* , 使得上式相等:

$$A(I^*)/OPT(I^*) = 2 - 1/m$$

Pf: 先证近似比上界。不失一般性, 假定所有作业分配完毕后, 机器 M_1 的负载最大。设L表示 M_1 上所有作业总运行时间,

§1.3.1 多机调度

Pf(续): 设 J_j 是 M_1 上最后一个分配到的作业, 则其它机器上的负载均大于等于 $L - P_j$ 。这是因为当 J_j 被分配到 M_1 时, M_1 是负载最小的机器, 其负载为 $L - P_j$ 。于是可得:

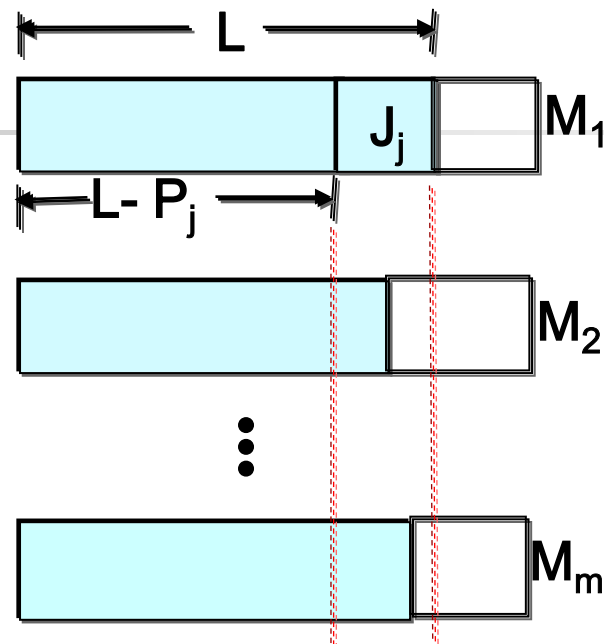
$$\sum_{i=1}^n P_i \geq m(L - P_j) + P_j$$

但 I 的最优解的值显然满足: $OPT(I) \geq \frac{\sum_{i=1}^n P_i}{m}$

由于 $A(I) = L$, 故有: $OPT(I) \geq (L - P_j) + P_j/m = A(I) - (1 - 1/m)P_j$

\because 必须有某台机器执行作业 J_j , $\therefore OPT(I) \geq P_j$

于是: $A(I)/OPT(I) \leq 2 - 1/m$



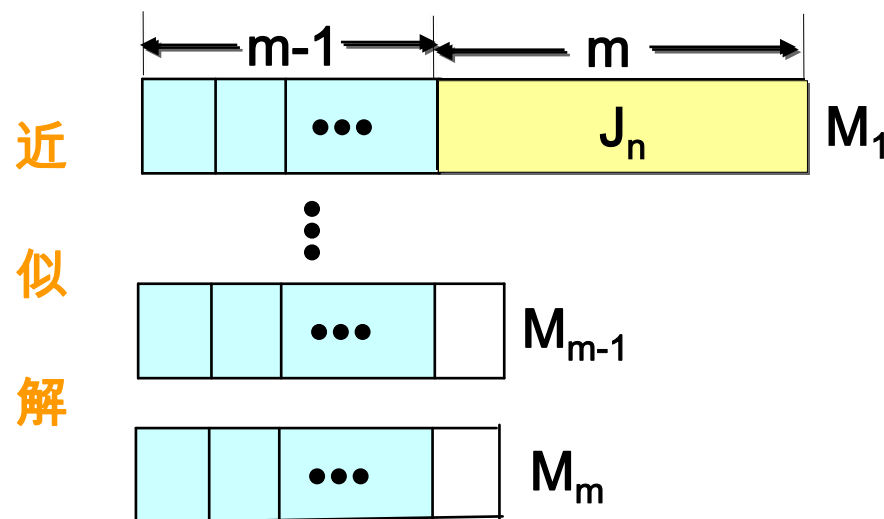
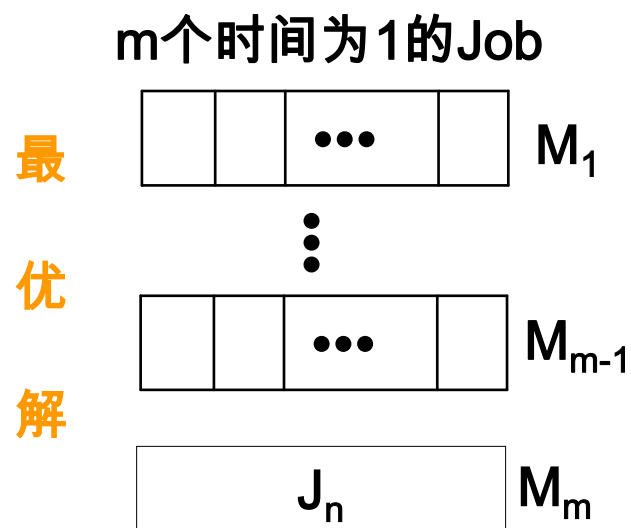
§1.3.1 多机调度

Pf(续): 现证近似比的紧确界。考虑输入实例 I^* , 设

$$n = m(m-1)+1$$

设前 $n-1$ 个作业每个均有运行时间1, 而最后1个作业 J_n 有 $P_n=m$ 。易于证明: $OPT(I^*)=m$, 而 $A(I^*)=2m-1$

故有: $A(I^*)/OPT(I^*)=2-1/m$



§1.3.1 多机调度

上述结果导出了近似算法质量的另一种度量方法：**相对性能度量**。其形式化定义如下

Def.1.5 设A是优化问题 Π 的一个近似算法，算法A在一个输入实例I上的**性能比** $R_A(I)$ 被定义为：

$$R_A(I) = \begin{cases} \frac{A(I)}{OPT(I)} & \text{if } \Pi \text{ 是最小化问题} \\ \frac{OPT(I)}{A(I)} & \text{if } \Pi \text{ 是最大化问题} \end{cases}$$

此定义统一使近似比(性能比) $R_A(I) \geq 1$ ，越接近1越好

若 $R_A(I) \leq (1 + \epsilon)$ ，则称A是 $(1 + \epsilon)$ - 近似算法

§1.3.1 多机调度

Def1.6 对于优化问题 Π ，近似算法 A 的绝对性能比 R_A 是

$$R_A = \inf \{ r \mid R_A(I) \leq r, \forall I \in D \}$$

即： R_A 是性能比上界集合中的下确界(最大下界)

例如：对于 List 调度算法 A ，我们有： $R_A = 2 - 1/m$

更好的调度是 **LPT** (Longest Processing Time)：将作业按其运行时间递减序排序，然后用 List 策略调度，其结果：

Th.1.9：LPT 算法的性能（近似）比： $R_{LPT} = \frac{4}{3} - \frac{1}{3m}$

Pf：当 $m=1$ 时，

$$\because A(I) = \text{OPT}(I) \quad \therefore A(I)/\text{OPT}(I) = 4/3 - 1/3$$

设对某个 $m > 1$ ，该定理不成立。

§1.3.1 多机调度

Pf(续): 则可设违反该定理具有最少作业数的实例 I 是 J_1, J_2, \dots, J_n , 不妨设 $P_1 \geq P_2 \geq \dots \geq P_n$

LPT的调度次序是 J_1, J_2, \dots, J_n , 其完成时间是 $A(I)$ 。设其中最迟完成的作业是 J_k , 则 $k=n$ 。否则, 若 $k < n$, 算法 A 运行作业 J_1, J_2, \dots, J_k (记为实例 $I' \subset I$) 的完成时间仍是 $A(I)$, 即 $A(I) = A(I')$, 而对最优解显然 $OPT(I') \leq OPT(I)$, 故有:

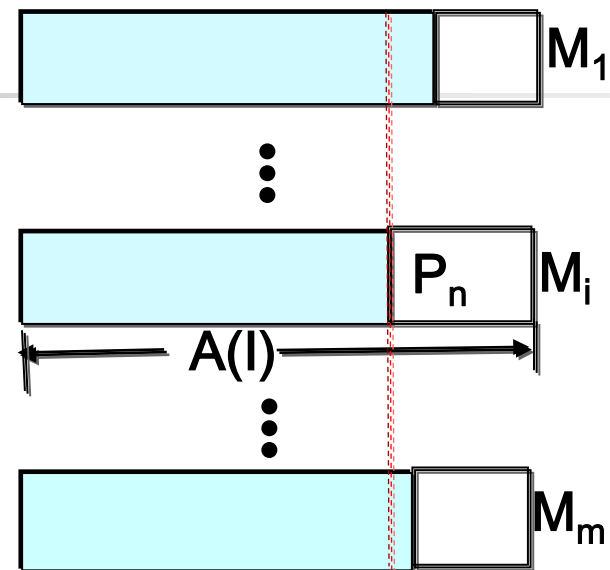
$$\frac{A(I')}{OPT(I')} \geq \frac{A(I)}{OPT(I)} > \frac{4}{3} - \frac{1}{3m}$$

这与 I 是违反定理的最少作业数的实例 **矛盾**, $\therefore |I'| < |I|$

§1.3.1 多机调度

现证明：对于上述实例I的最优调度，
在任何机器上分配的作业数不超过2，
因此 $n \leq 2m$

$\because J_n$ 是LPT调度A中最迟完成的作业，
 \therefore 在A中它开始于时刻 $A(I) - P_n$ ，且此刻
其它机器均无空余时间，即：



$$A(I) - P_n \leq \frac{1}{m} \sum_{i=1}^{n-1} P_i \quad \Rightarrow \quad A(I) \leq \frac{1}{m} \sum_{i=1}^n P_i + \frac{m-1}{m} P_n$$

另一方面，因为 $OPT(I) \geq \frac{1}{m} \sum_{i=1}^n P_i$

$$\frac{4}{3} - \frac{1}{3m} < \frac{A(I)}{OPT(I)} \leq \frac{OPT(I) + \frac{m-1}{m} P_n}{OPT(I)} = 1 + \frac{m-1}{m} \frac{P_n}{OPT(I)}$$

§1.3.1 多机调度

$$\frac{4}{3} - \frac{1}{3m} < \frac{A(I)}{OPT(I)} \leq \frac{OPT(I) + \frac{m-1}{m} P_n}{OPT(I)} = 1 + \frac{m-1}{m} \frac{P_n}{OPT(I)}$$

$$4m - 1 < 3m + 3(m-1) \frac{P_n}{OPT(I)} \Rightarrow OPT(I) < 3P_n$$

∴ P_n 是时间最短的作业

∴ 实例 I 的最优调度中任何机器上的作业数 ≤ 2

当最优调度在任何机器上至多包含 2 个作业时，LPT 也是最优的。证明如下：

不妨设 $n=2m$ ，若 $n < 2m$ ，则令 J_{n+1}, \dots, J_{2m} 的时间均为 0，将其加入 I，不妨设 $P_1 \geq P_2 \geq \dots \geq P_{2m}$

§1.3.1 多机调度

设最优调度使得每台机器恰有2个作业： J_i 和 J_j ，则必有 $i \leq m, j > m$ 。否则若某最优调度 O 有 $i, j \leq m$ ，则定有某台机器上有 J_s 和 J_t ，使得 $s, t > m$ 。

$\because P_i, P_j \geq P_s, P_t$ ，交换 P_j 和 P_t ，则

$$P_i + P_t \leq P_i + P_j \quad P_s + P_j \leq P_i + P_j$$

交换后的调度 O' 的最迟完成时间只可能减少，故 O' 也是最优调度。对于 $i, j > m$ 可类似证明。

\therefore 必有最优调度使 J_1, \dots, J_m 分别分配到 M_1, \dots, M_m 上，当将 J_{m+1}, \dots, J_{2m} 分配到 M 台机器上时，LPT是将长时间的作业分配到轻负载上，必与该最优调度结果相同。

$$\frac{A(I)}{OPT(I)} = 1 \leq \frac{4}{3} - \frac{1}{3m} \quad (m \geq 2)$$

Ex. 完善此证明

§1.3 相对性能保证

有时，绝对性能比可能并不是好的性能保证。因为可能存在输入实例使得最优解的值很小，而近似算法的性能也仅与最优值稍有不同，但此时其近似比仍然会过大。为此可定义：

Def1.7：一个优化问题 Π 的近似算法 A ，其**渐近性能比**为：

$$R_A^\infty = \inf\{r \mid \exists N \in \mathbb{Z}^+, R_A(I) \leq r \text{ for } \forall I \in D_\Pi \text{ with } OPT(I) \geq N\}$$

显然有： $1 \leq R_A^\infty \leq R_A < \infty$ 但未必有： $R_A = R_A^\infty$

对于有Scaling性质的问题，近似算法的绝对性能比和渐近性能比是相同的，为什么？但多数优化问题无此性质！

§1.3 相对性能保证

对于一个最小化问题，如何求性能比的上界？

- 1) 证明 $A(I)$ 关于某个参数 x 的上界；
- 2) 证明 $OPT(I)$ 关于 x 的下界

然后从这两个不等式中消除 x 即可得性能比。

§ 1.3.2 装箱问题 (BP)

装箱定义如前。即：设有 n 件物品，每件物品大小 $S_i \in [0,1] (1 \leq i \leq n)$ ，按某种策略将其装入大小为1的若干箱子中，使箱子数尽可能小。

§ 1.3.2 装箱问题 (BP)

首次适应(First Fit)算法

FF策略：依次将物品装箱，设当前要装第 i 件， B_1, B_2, \dots, B_j 是当前已开过的箱子，则从头依次扫描箱子，将物品 i 放入第1个适合(指大小够放)的箱子中；若不存在适合的箱子，则新开箱子 B_{j+1} ，将物品 i 放入其中。

Claim：对所有实例， $R_{FF}(I) \leq 2$

Pf：在整个装箱过程中至多只有1个箱子是大于半空的。若否，不妨设 B_i 和 B_j 均大于半空且 $i < j$ ，则第1件放入 B_j 的物品大小至多是0.5，按照FF策略该物品应该放入 B_i 而不应该打开新箱子 B_j 。

§ 1.3.2 装箱问题 (BP)

近似算法FF的上界

所有物品总size至少为FF使用的箱子总数的一半：

$$\lceil \sum S_i \rceil \geq FF(I)/2, \text{ 即 } FF(I) \leq 2 \lceil \sum S_i \rceil$$

最优解的下界

所有物品总size是最优解的下界： $OPT(I) \geq \lceil \sum S_i \rceil$

$$\text{故有 : } R_{FF}(I) = \frac{FF(I)}{OPT(I)} \leq \frac{2 \lceil \sum S_i \rceil}{\lceil \sum S_i \rceil} = 2$$

可以证明上述比可达7/4

Th.1.10 $R_{FF}^{\infty} = 1.7$ 且更精确地有：

$$\forall I, FF(I) \leq 1.7 OPT(I) + 2$$

$$\exists I, FF(I) \geq 1.7(OPT(I) - 1)$$

渐近性能比不同于绝对性能比！

§ 1.3.2 装箱问题 (BP)

递减首次适合 (First Fit Decreasing FFD)

Th1.11 $R_{\text{FFD}} \geq 3/2$, 但是: $R_{\text{FFD}}^{\infty} = \frac{11}{9}$

渐近性能比不同于绝对性能比 !

§ 1.3.3 旅行商问题 (TSP)

TSP问题 (Travelling Salesman Problem)

设 G 是具有 n 个城市的地图，旅行商从其中某一城市出发周游，遍历每个城市1次恰好1次后回到出发地，求其最短的周游路线。亦即，求一条最短的哈密顿回路 (Hamiltonian Cycle，简称HC)。

因为边的长度可以为无限大，因此不妨设 G 为边加权的无向完全图。下面只考虑TSP的特殊版本：满足三角不等式的 Δ TSP (Metric TSP) 问题。

§ 1.3.3 旅行商问题 (TSP)

Def1.8: Δ TSP是TSP的特例，所有输入满足三角不等式，即对所有顶点 i ， j 和 k ，边的长度满足：

$$d(i,k) \leq d(i,j) + d(j,k)$$

几何意义：去掉途中1个中转站不会使路径长度增加

近邻法(NN)

从任一顶点开始，在每一步从当前顶点出发去访问一个最近的尚未访问过的顶点，由此构造哈密尔顿圈。但这种方法性能差。

Th.1.12 设 Δ TSP的顶点数为 n ，则 $R_{NN}^{\infty} = \theta(\lg n)$

§ 1.3.3 旅行商问题 (TSP)

MST启发：有几种启发已达到渐近性能比2，大多数好的启发式是基于找到一个**欧拉环** (Euler Tour, 简称**ET**)，然后使用“Short-cut”来获得一个哈密尔顿圈。

Def1.9 设 G 是一个多重图， G 的一条欧拉路是一条经过每个顶点至少一次、经过每条边恰好一次的周游路线。

Th.1.13 一个多重图 G 有一个欧拉环当且仅当 G 连通且所有顶点的度数为偶数(易在多项式时间内构造一个欧拉环)。

基于MST启发的 Δ TSP的**近似算法**：求 G 的任一MST T ；重复 T 的边构造一欧拉环ET；从ET产生一哈密尔顿圈。

§ 1.3.3 旅行商问题 (TSP)

Alg. MST:

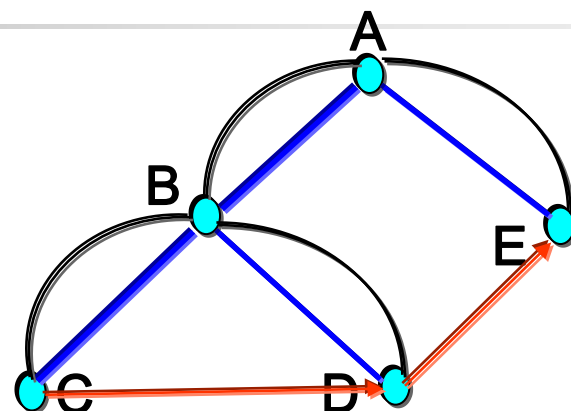
Input: $G(V,E)$ 和距离函数 d

Output: G 里的一个哈密尔顿圈

- 1.在 G 里找一棵最小生成树 T ;
- 2.通过将 T 的每条边复制一copy构造一多重图 T' ;
- 3.在 T' 中找一欧拉圈 ET ;
- 4.通过短路欧拉路的方法构造哈密尔顿圈 HC :

从任一顶点出发，沿着欧拉圈前进，遇新顶点则访问，遇到重复顶点则直接跳到下一未访问过的顶点，最终回到出发点即可。

ET: ABCBDBAEA; **HC:** ABCDEA



§ 1.3.3 旅行商问题 (TSP)

Th.1.14 用MST启发解 Δ TSP时, 有: $R_{MST}^{\infty} = 2$

Pf: 正确性: $\because T'$ 连通且每个顶点度为偶数, 故可找到欧拉环; 在完全图的假定下, step4产生一个哈密尔顿圈。

若H是G的边集, 则d(H)表示H里所有边的长度之和。

\because 任何哈密尔顿圈去掉1条边后都是1棵生成树

$\therefore \text{OPT}(G) \geq d(\text{某生成树}) \geq d(T)$

$d(ET) = d(T') = 2d(T) \leq 2\text{OPT}(G)$

短路过程确保 $A_{MST}(G) \leq d(ET) \leq 2\text{OPT}(G)$: 因为短路是用1条非欧拉边取代2或多条欧拉边, 三角不等式保证其成立。故有:

$$\frac{A_{MST}(G)}{\text{OPT}(G)} \leq 2$$

§ 1.3.3 旅行商问题 (TSP)

CH启发(Christofides) 避免从MST T到欧拉环的双边，为T的每对奇度顶点加一条边使所有顶点度数之和为偶数。
∵ T中所有顶点度数之和为 $2|E|$ ， ∴ 奇度顶点总数为偶数

$V(G)$ 的子集S的一个**匹配**是 $E(G)$ 的一个子集，其中所有边的端点集恰为S，S中每个顶点恰好依附匹配集中的一条边。∵ G是完全图，故任一S均存在一个匹配($|S|$ 为偶数)。已证明，可在多项式时间内找到S的一个最小权匹配。

Th.1.15 $A_{CH}^{\infty} = 1.5$

Pf: 设M是T中奇数点集O上的最小权匹配集，则可断言：

$$d(M) \leq \frac{OPT(G)}{2}$$

§ 1.3.3 旅行商问题 (TSP)

在1个最优解中做短路操作以排除不在O中的顶点，所得的圈X(O上的圈)必满足：

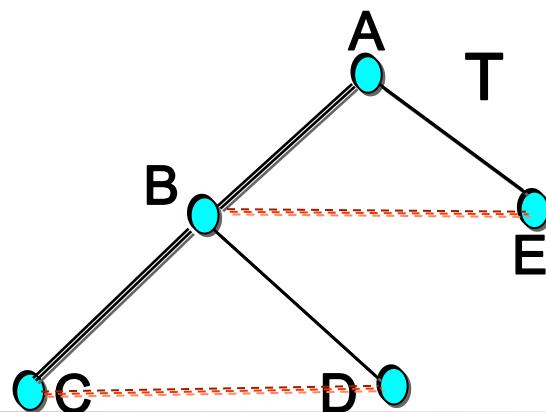
$$d(X) \leq \text{OPT}(G) \quad (\text{由}\Delta\text{不等式决定})$$

点集O上的圈必是O的两个匹配集的并。 \because T中奇度顶点是偶数个， $\therefore |O|=m$ 为偶数。O上圈有m条边，但是O上的一个匹配集只有m/2条边，故该圈是2个匹配集之并。

这两个匹配集之一的权必定至多是圈X的一半，而M是O的最小权匹配集，故有：

$$d(M) \leq d(X)/2 \leq \text{OPT}(G)/2$$

例： $O=\{B,C,D,E\}$. 匹配 $M_1=\{(B,E),(C,D)\}$,
 $M_2=\{(B,C),(D,E)\}$. $M_1 \cup M_2$ 是O上圈



§ 1.3.3 旅行商问题 (TSP)

∵ 图 $T \cup M$ 中所有顶点度为偶数, ∴ 可从其构造欧拉环 ET

$$d(ET) = d(T \cup M) \leq 1.5OPT(G)$$

用短路法从 ET 构造 HC 不会使权增加, 故有:

$$A_{CH}(G) \leq d(ET) \leq 1.5OPT(G), \text{ 即:}$$

$$\frac{A_{CH}(G)}{OPT(G)} \leq 1.5$$

近似比1.5是目前最好的结果, 但是要找一个小权值匹配需要 $O(n^3)$ 时间, 而MST启发的运行时间几乎是线性的(求MST除外)。

§ 1.3.4 相对近似之否定结果

为什么有的问题易于近似，而有的问题难于近似？遗憾的是问题的近似算法之间似乎没有什么联系，尽管这些问题的最优算法是紧密相关的(NPC问题彼此相关)。

例：顶点覆盖(VC)和最大独立集(MIS)。设 $G(V,E)$ 是图。

VC问题：G的顶点覆盖是一顶点集 $C \subseteq V$ 使得E中每条边至少有一端点在C中，VC问题是在图中找一个顶点数**最小的覆盖**。

MIS问题：G的独立集是一顶点集 $I \subseteq V$ 使得I里任何点对之间无边，MIS问题是在G中找一个**最大的独立集**。

设G是任意图，C是一顶点覆盖当且仅当 $I = V \setminus C$ 是一独立集，且C是VC的最优解当且仅当I是MIS的最优解

§ 1.3.4 相对近似之否定结果

VC和MIS的最优解是如此相关，他们的近似解是相关问题吗？**否！**

∵对于VC问题有一个近似比为2的近似算法；但是对于MIS问题，我们并未找到性能比好于 $O(n)$ 的近似算法。为什么VC的近似无助于MIS的近似？

设VC对图G的1个最优解 $OPT_{VC}(G)=n/2-1$ (即最小覆盖的顶点数)，∵近似比为2，∴近似解 $A_{VC}(G) \leq 2OPT_{VC}(G)=n-2$. 解 A_{VC} 给出的顶点覆盖集的补集(即独立集 A_{MIS})的size为2；而 $OPT_{VC}(G)$ 的补集(即最优的独立集 OPT_{MIS})的size为 $n/2+1$ 。
故：

$$\frac{OPT_{MIS}(G)}{A_{MIS}(G)} = \frac{\frac{n}{2} + 1}{2} = \frac{n}{4} + \frac{1}{2} = O(n)$$

§ 1.3.4 相对近似之否定结果

尽管NP-hard归约对优化问题的近似行为揭示的不够，但是我们仍可使用NPC理论来证明：除非 $P=NP$ ，某类近似算法不存在。

Def.1.10 对于一个优化问题 Π ，最佳可达性能比 $R_{\text{MIN}}(\Pi)$ 定义为：

$$R_{\text{MIN}}(\Pi) = \inf\{r \geq 1 \mid \exists \Pi \text{ 的多项式时间算法 } A \text{ 使 } R_A^\infty \leq r\}$$

$R_{\text{MIN}}=1$ ：最希望的结果，这类问题是容易近似的

R_{MIN} 有界：如 Δ TSP问题

R_{MIN} 无界：难于近似。下面先考虑此类问题

§ 1.3.4 相对近似之否定结果

Th.1.16 若 $P \neq NP$ ，则 $R_{\text{MIN}}(\text{TSP}) = \infty$

Pf：假定有一算法A对某常数k满足： $R_A^\infty = k$

基本思想：使用A构造一个多项式时间的算法解决HC问题，因为HC是NPC的，若 $P \neq NP$ ，则矛盾！

假定 $G(V, E)$ 是无向无权图，可为TSP构造**实例I**：设H是V上的完全图，H中来自于E的边长度置为1，其余边长置为kn， $n = |V|$ 。

Claim：若G有HC则 $\text{OPT}(I) = n$ ；否则 $\text{OPT}(I) \geq (k+1)n - 1$

将I作为A的输入实例。若G有HC，则 $A(I) \leq k \text{OPT}(I) = kn$ ；否则， $A(I) \geq \text{OPT}(I) \geq (k+1)n - 1$ 。

因此，可通过A找到的解之值来判定G中是否存在HC。
即：将HC问题多项式时间归约到A，与HC是NPC矛盾！

§ 1.3.4 相对近似之否定结果

绝对近似之否定：很多问题找不到绝对近似算法

相对近似之否定：有些问题的有界性能比近似算法不存在

两类近似算法之间的近似性能（相对误差）：A是一个 $f(n)$ -近似算法当且仅当对每个size为n的输入实例I，有：

$$\frac{|OPT(I) - A(I)|}{OPT(I)} \leq f(n) \quad \text{假定 } OPT(I) > 0$$

BP的一个近似算法A满足： $|A(I) - OPT(I)| \leq O(\lg^2 OPT(I))$)
蕴含着渐近性能比为1。

$$\frac{|A(I) - OPT(I)|}{OPT(I)} = \frac{A(I)}{OPT(I)} - 1$$

$$\frac{A(I)}{OPT(I)} = 1 + \frac{|A(I) - OPT(I)|}{OPT(I)} \leq 1 + \frac{O(\lg^2 OPT(I))}{OPT(I)} \rightarrow 1 \text{ (当 } OPT(I) \rightarrow \infty \text{)}$$

§ 1.4 其他

近似方案(Approximation Scheme)

一个优化问题 Π 的近似方案是一个算法 A ，它以实例 I 和误差界 ε 作为输入，且有性能保证： $R_A(I, \varepsilon) \leq 1 + \varepsilon$ 。

对于最小化问题， ε 是相对误差。实际上，算法 A 对应一个算法族 $\{A_\varepsilon : \varepsilon > 0\}$ 使得 $R_{A_\varepsilon} \leq 1 + \varepsilon$

多项式近似方案(PAS: Polynomial Approximation Scheme)

是一近似方案 $\{A_\varepsilon\}$ ，对任一确定的 ε ，每个算法均以其 $\text{size}[I]$ 的多项式时间运行。

完全多项式近似方案(FPAS: Fully PAS)

是一近似方案 $\{A_\varepsilon\}$ ，对任一确定的 ε ，每个算法均以其 $\text{size}[I]$ 和 $1/\varepsilon$ 的多项式时间运行

§ 1.4 其他

理想的近似方案

近似方案实际上研究近似算法的**运行时间**和**近似质量**之间的关系（二者间如何折衷？）。希望：近似算法的运行时间并不随着性能比的减少而增长太快！

理想情况： ε 减小1个常数因子，为获得预期的近似质量，所增加的运行时间不应超过1个常数因子（尽管两常数因子不一定相同）

PAS VS FPAS

背包问题的PAS中，算法 A_ε 的运行时间一般是 $n^{O(1/\varepsilon)}$ ；多机调度的PAS中，算法 A_ε 的运行时间为 $O(m^{1/\varepsilon})$ 。显然， ε 减小一点将会引起算法时间的急剧增加。为此，引进FPAS可克服此缺点。

§ 1.4 其他

例子：调度问题的近似方案

假定 $n > m$ ，运行时间为降序($i < j$ 蕴含 $P_i > P_j$)，对每一整数 $k \in [0, n]$ 定义算法 A_k ：

Algorithm A_k :

Input: n 个作业的运行时间 $\{P_1, \dots, P_n\}$ 和机器数 m ;

Output: 一个可行的调度;

Step1: 最优调度前 k 个作业 J_1, \dots, J_k ;

Step2: 从前一步所获得的部分调度开始，使用LPT规则调度剩余作业。

LPT策略：取下一运行时间最长尚未调度的作业，将其分配到当前负载最轻的机器上。该算法显然是多项式时间的

§ 1.4 其他

Th. 算法 A_k 的绝对性能比：

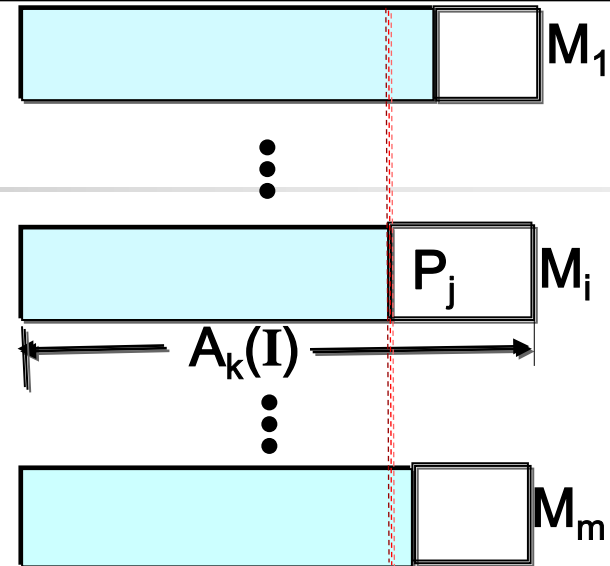
$$R_{A_k} \leq 1 + \frac{1 - 1/m}{1 + \lfloor k/m \rfloor}$$

Proof:

设 K 表示 step 1 中找到的调度的完成时间，显然：

- 1) 若 $A_k(I) = K$ ，则该算法已找到一个最优调度；
- 2) 若 $A_k(I) > K$ (总的调度完成时间 $> K, n > k$)，则必有某个作业 $J_j (j > k)$ 是在 $A_k(I)$ 时间完成的。这蕴含着其它所有机器在时间区间 $[0, A_k(I) - P_j]$ 都在忙，否则作业 J_j 将在此前被调度 (注意：一旦某机器变为空闲，它直到调度结束仍然是空闲的)。设 $T = \sum P_i$ 是 n 个作业总的运行时间，则：

$$T \geq m(A_k(I) - P_j) + P_j$$



§ 1.4 其他

要证性能比：1)先证 $A_k(I)$ 的上界；2)再证 $OPT(I)$ 的下界

1) $A_k(I)$ 的上界//可能用到 $OPT(I)$ 和某参数 X

$$T \geq mA_k(I) - (m-1)P_j \geq mA_k(I) - (m-1)P_{k+1}$$

// \because 作业是按运行时间降序, $j \geq k+1, \therefore P_{k+1} \geq P_j$

$$A_k(I) - ((m-1)/m)P_{k+1} \leq T/m$$

$$\because OPT(I) \geq T/m \quad \therefore A_k(I) \leq OPT(I) + (1-1/m)P_{k+1}$$

2) $OPT(I)$ 的下界//可能用到 $A(I)$ 和 X

$\because P_i \geq P_{k+1} \ (1 \leq i \leq k+1)$, 必有某台机器至少执行这 $k+1$ 个作业中的 $1 + \lfloor k/m \rfloor$ 个作业

$$\therefore OPT(I) \geq (1 + \lfloor k/m \rfloor)P_{k+1}$$

上面的 P_{k+1} 是可以消除的参数 X

§ 1.4 其他

由 $A_k(I) \leq OPT(I) + (1 - 1/m)P_{k+1}$ 和 $OPT(I) \geq (1 + \lfloor k/m \rfloor)P_{k+1}$ 可得性能比:

$$\begin{aligned} \frac{A_k(I)}{OPT(I)} &\leq \frac{OPT(I) + (1 - 1/m)P_{k+1}}{OPT(I)} = 1 + \frac{(1 - 1/m)P_{k+1}}{OPT(I)} \\ &\leq 1 + \frac{(1 - 1/m)P_{k+1}}{(1 + \lfloor k/m \rfloor)P_{k+1}} = 1 + \frac{1 - 1/m}{1 + \lfloor k/m \rfloor} \end{aligned}$$

使用上述结果, 可构造一个PAS, 该方案用 ε 作为输入变量, 对任意输入的 ε , 它计算一个整数 k 使得

$$R_{A_k} \leq 1 + \varepsilon \Rightarrow \frac{1 - 1/m}{1 + \lfloor k/m \rfloor} \leq \varepsilon \Rightarrow k \geq \frac{(1 - \varepsilon)m - 1}{\varepsilon}$$

§ 1.4 其他

A_k 的时间分析:

Step1中, 在 m 台机器上对 k 个作业做最优调度的时间, 穷举法 $O(m^k)$, 这只有当 m 较小(比如为常数)才是多项式的; step2中, 对于其余 $n-k$ 个作业, LPT调度时间是 $O(n \lg n)$ 。 A_k 的总运行时间为:

$$O\left(n \lg n + m^{\frac{(1-\varepsilon)m-1}{\varepsilon}}\right)$$

结论: 对于固定的 m , 对 m 台机器的调度问题有一个多项式近似方案