近似算法

肖明军

基础知识

现实中许多优化问题是NP-hard的，由复杂性理论知：若P≠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。

基础知识

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。

基础知识

优化问题近似解分类

1）容易近似

Knapsack，Scheduling，Bin Packing等

2）中等难度

VertexCover，EuclideanTSP，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

基础知识

定义 1 一个优化问题Π由三部分构成：

实例集D：输入实例的集合

解集S(I)：输入实例I∈D的所有可行解的集合

解的值函数f：给每个解赋一个值，f：S(I)→R

一个最大值优化问题Π是：

对于给定的I∈D，找一个解σ_{opt}^{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

基础知识

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

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

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

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

基础知识

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

定义 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$f(\sigma) \geq k$？"

基础知识

定义.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σ∈S(I)。通常，我们用A(I)表示算法A所获得的解的值f(σ)。（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)。

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。

定义 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：

$$\forall I \in D,\ |A(I)-OPT(I)| \le k$$

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)。

绝对性能保证

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。

定理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,∵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：

定理2：对给定的任意平面图G，近似算法A的性能满足:

$$|A(G)-OPT(G)|\le 2$$

绝对性能保证

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：

定理 3：任一图至少需要Δ，至多需要Δ+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Δ+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：

定理 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：

定理 5：近似算法A有性能保证：$|A(G)-OPT(G)|\leq 1$

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！

绝对近似算法之否定

□ 0/1背包问题问题实例：

- 项集：I={1,2,…,n}
- 大小：$s_1,s_2,\ldots,s_n$
- 利润：$p_1,p_2,\ldots,p_n$
- 背包容量：B

问题的一个**可行解**是子集I'⊆I，$\sum_{i\in I'} s_i \le B$

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。

绝对近似算法之否定

定理 6 若P≠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：$|A(I)-OPT(I)|\le k$

证明：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：

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

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

即：我们找到了一个多项式时间的算法A，它对背包问题的任意一个输入实例I，均能找到最优解。

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:

定理 7 若P≠NP，则对于团问题不存在绝对近似算法

证明：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mα (Ex.)

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

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

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：

$$|\,|C|-OPT(G)\,|\le k/(k+1)$$

因为|C|和OPT(G)均是整数，故C是最优团。

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. 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，…，J_n，相应的运行时间为P_1，P_2，…，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。

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。

定理 8. 设A表示List调度算法，则对于所有输入实例I

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

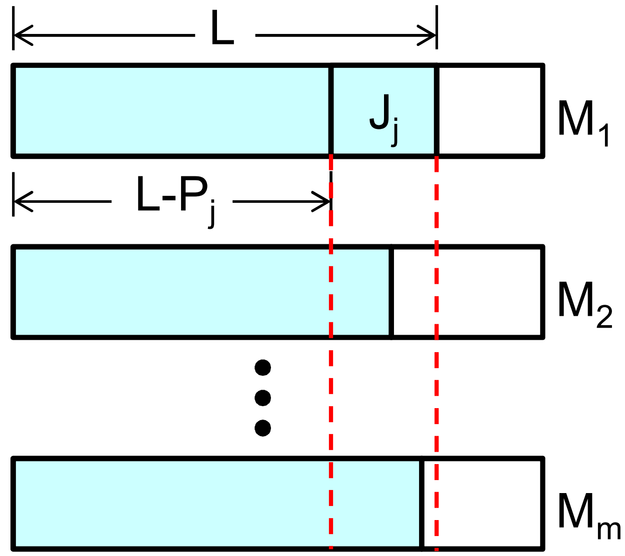
界是紧致的，因为存在一个实例I*，使得上式相等：

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

证明：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负载最大。设L表示M_1上所有作业总运行时间,

多机调度

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

∵必须有某台机器执行作业Jj，∴$OPT(I) \geq P_j$

于是：$A(I)/OPT(I) \leq 2-1/m$

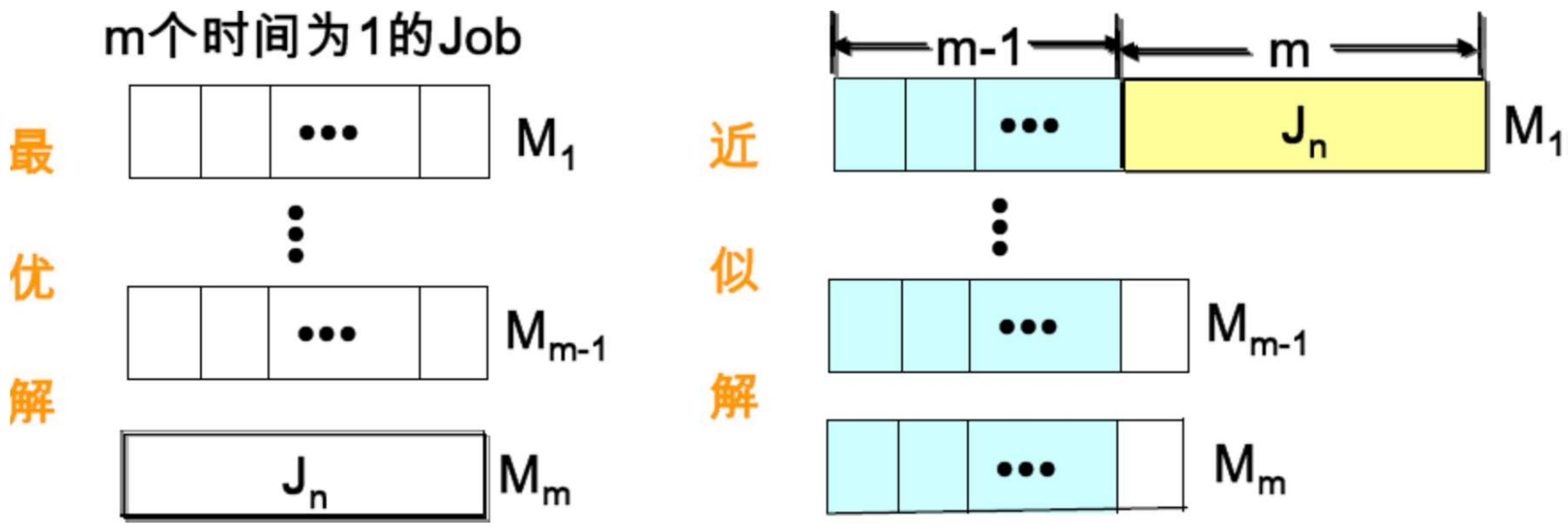
多机调度

证明(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



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

定义 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：

$$R_A(I)=\begin{cases}\dfrac{A(I)}{OPT(I)} & if\ \Pi\text{是一个最小化问题}\\[2ex] \dfrac{OPT(I)}{A(I)} & if\ \Pi\text{是一个最大化问题}\end{cases}$$

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+ε），则称A是**(1+ε) - 近似算法**

多机调度

定义 6. 对于优化问题Π，近似算法A的绝对性能比R_A是

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

即： R_A是性能比上界集合中的**下确界**

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:

定理 9：LPT算法的性能（近似）比：$R_{LPT} = \frac{4}{3} - \frac{1}{3m}$

证明：当m=1时，

$\because$ A(I)=OPT(I)　　　$\therefore$ A(I)/OPT(I)=4/3-1/3

设对某个m > 1，该定理不成立。

多机调度

证明(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, … , J_n, 不妨设$P_1 \geq P_2 \geq \ldots \geq P_n$

LPT的调度次序是J_1, J_2, … , 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, … , J_k (记为实例 I’ ⊂ I)的完成时间仍是A(I)，即A(I) = A(I’)，而对最优解显然OPT(I’) ≤OPT(I)，故有：

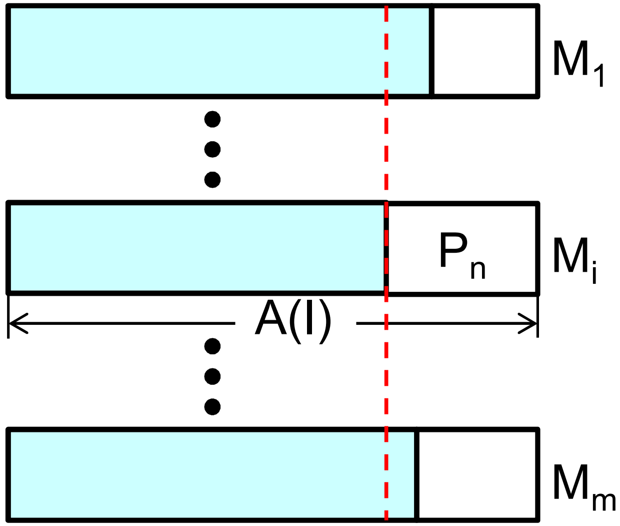
$$\frac{A(I')}{OPT(I')} \geq \frac{A(I)}{OPT(I)} \geq \frac{4}{3} - \frac{1}{3m}$$

这与I是违反定理的最少作业数的实例矛盾, ∵ | I’ | < | I |

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≤2m**

∵J_n是LPT调度A中最迟完成的作业，∴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:



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

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

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

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}，…，J_{2m}的时间均为0，将其加入I，不妨设$P_1 \geq P_2 \geq \ldots \geq P_{2m}$

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$i \le m$, $j > m$。否则若某最优调度O有$i, j \le m$，则定有某台机器上有J_s和J_t，使得$s,t > m$.

$\because P_i, P_j \ge P_s, P_t$，交换$P_j$和$P_t$，则 $P_i+P_t \le P_i+P_j$ $P_s+P_j \le P_i+P_j$

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$J_1, \ldots, J_m$分别分配到$M_1, \ldots, M_m$上，当将$J_{m+1}, \ldots, J_{2m}$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。

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

Ex. 完善此证明

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：

定义 7：一个优化问题Π的近似算法A, 其**渐近性能比**为:

$$R_A^\infty = \inf\{r \mid \exists N \in Z^+, R_A(I) \le r \; for \; \forall I \in D_\Pi \; with \; OPT(I) \ge N\}$$

显然有: $1 \le R_A^\infty \le R_A < \infty$ 　　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！

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。

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。

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，$B_1, B_2, \ldots, B_j$ 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，$R_{FF}(I) \le 2$

证明: 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。

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：

$[\,\Sigma\, s_i\,] \geq FF(I)/2$ ， 即$FF(I) \leq 2\,[\,\Sigma\, s_i\,]$

□ 最优解的下界

所有物品总size是最优解的下界： $OPT(I) \geq [\,\Sigma\, S_i\,]$

故有：

$$R_{FF}(I) = \frac{FF(I)}{OPT(I)} \leq \frac{2\sum S_i}{\sum S_i} = 2$$

可以证明上述比可达7/4

定理10： $R_{FF}^{\infty} = 1.7$ 且更精确地有：

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

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

渐近性能比不同于绝对性能比！

装箱问题（BP）

递减首次适合(First Fit Decreasing FFD)

定理 11：$R_{FFD} \geq 3/2$, 但是： $R_{FFD}^{\infty} = \dfrac{11}{9}$

渐近性能比不同于绝对性能比！

相对性能保证

3. 旅行商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ΔTSP（MetricTSP）问题。

旅行商问题 (TSP)

定义8： ΔTSP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：

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

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。

定理 12： 设ΔTSP的顶点数为n，则 $R_{NN}^{\infty} = \theta(\lg n)$

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“短路(Short-cut)”来获得一个哈密尔顿圈。

定义 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。

定理 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）。

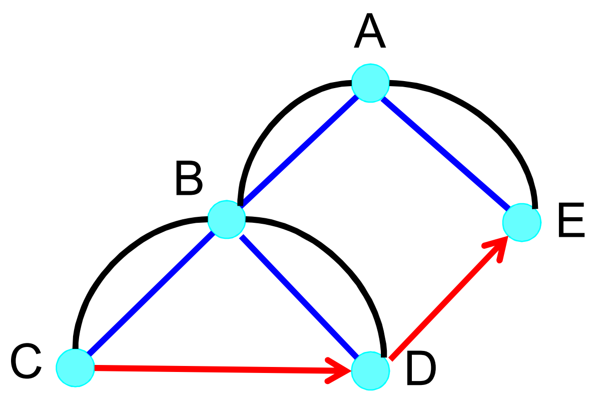
基于MST启发的ΔTSP的**近似算法**：求G的任一MST T；重复T的边构造一欧拉环ET；从ET产生一哈密尔顿圈。

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

旅行商问题 (TSP)

定理 14. 用MST启发解ΔTSP时，有：$R_{MST}^{\infty} = 2$

证明：正确性：∵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。

∵任何哈密尔顿圈去掉1条边后都是1棵生成树

∴OPT(G) ≥ d(某生成树) ≥ d(T)

d(ET) = d(T') = 2d(T) ≤ 2OPT(G)

短路过程确保**A_{MST}(G) ≤ d(ET)** ≤ 2OPT(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：

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

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之度数为偶数。

∵T中所有顶点度数和为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。

定理 15. $A_{CH}^{\infty} = 1.5$

证明： 设M是T中**奇数点集O**上的最小权匹配集，则可断言：

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

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：

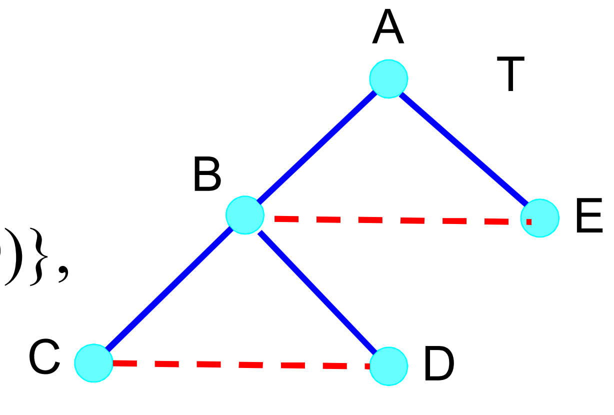
$$d(X) \le OPT(G)(\text{由}\Delta\text{不等式决定})$$

点集O上的圈必是O的两个匹配集的并。∵T中奇度顶点是偶数个,∴|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：

$d(M) \le d(X)/2 \le OPT(G)/2$

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



旅行商问题 (TSP)

∵图T∪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：

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

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

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

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⊆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⊆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\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

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:

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

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，但是我们仍能使用NPC理论来证明：除非P=NP，某类近似算法不存在。

定义 10. 对于一个优化问题Π，**最佳可达性能比**$R_{MIN}(\Pi)$定义为：

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

R_{MIN}=1：最希望的结果，这类问题是容易近似的

R_{MIN}有界：如ΔTSP问题

R_{MIN}无界：难于近似。下面先考虑此类问题

相对近似之否定结果

定理 16. 若P≠NP，则$R_{MIN}(TSP)=\infty$

证明： 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≠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OPT(I)=n；否则OPT(I)≥(k+1)n-1

将I作为A的输入实例。若G有HC，则A(I)≤kOPT(I)=kn；否则，A(I)≥OPT(I)≥(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！

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：

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

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

由于$\frac{|A(I)-OPT(I)|}{OPT(I)} = \frac{A(I)}{OPT(I)} - 1$，于是当$OPT(I) \to \infty$，我们有

$$\frac{A(I)}{OPT(I)} = 1 + \frac{|A(I)-OPT(I)|}{OPT(I)} \le 1 + \frac{O(\lg^2 OPT(I))}{OPT(I)} \to 1$$

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_ε：ε>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_ε}，对任一确定的ε，每个算法均以其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_ε}，对任一确定的ε，每个算法均以其size[I]和1/ε的多项式时间运行

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。

其他

□ 例子：多机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$\{P_1,\ldots,P_n\}$和机器数m；

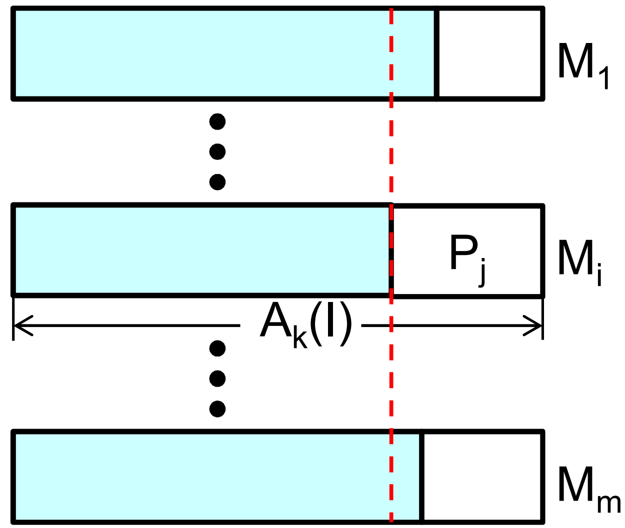
Output：一个可行的调度；

Step1：最优调度前k个作业$J_1,\ldots,J_k$；

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

其他



定理. 算法A_K的绝对性能比：

$$R_{A_K} \le 1 + \frac{1 - 1/m}{1 + \lfloor k/m \rfloor}$$

证明：设K表示step1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$T=\Sigma P_i$是n个作业总的运行时间，则：

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

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

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

//∵作业是按运行时间降序, $j \geq k+1$, ∴ $P_{k+1} \geq Pj$

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

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

2)OPT(I)的下界//可能用到A(I)和X

∵ $P_i \geq P_{k+1}$ $(1 \leq i \leq k+1)$，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

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

上面的P_{k+1}是可以消除的参数X

其他

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

$$\frac{A_k(I)}{OPT(I)} \le \frac{OPT(I) + (1-1/m)P_{k+1}}{OPT(I)} = 1 + \frac{(1-1/m)P_{k+1}}{OPT(I)}$$

$$\le 1 + \frac{(1-1/m)P_{k+1}}{(1+\lfloor k/m \rfloor)P_{k+1}} = 1 + \frac{1-1/m}{1+\lfloor k/m \rfloor}$$

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

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

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(nlgn)。A_k的总运行时间为：

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

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

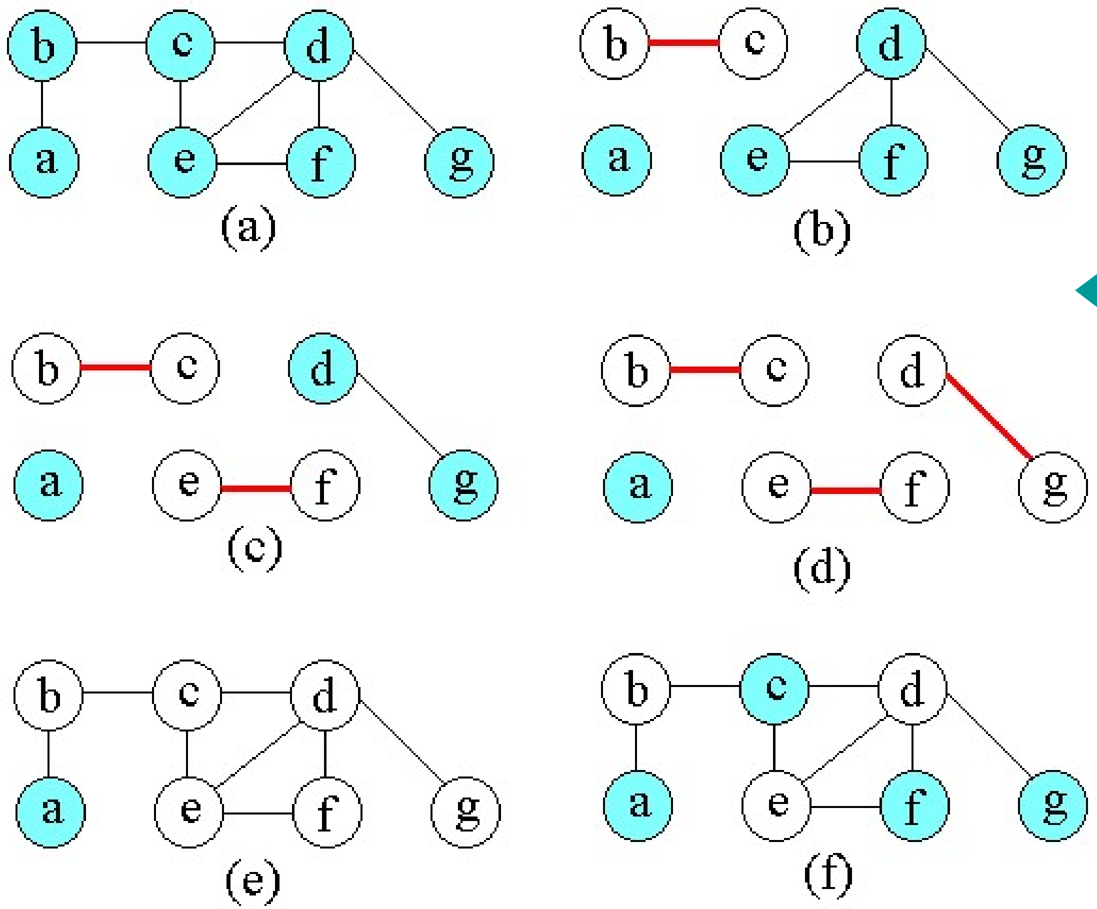
顶点覆盖问题的近似算法

问题描述：无向图G=(V,E)的顶点覆盖是它的顶点集V的一个子集V'⊆V，使得若(u,v)是G的一条边，则v∈V'或u∈V'。顶点覆盖V'的大小是它所包含的顶点个数|V'|。

```
VertexSet approxVertexCover ( Graph g )
{   cset=∅;
     e1=g.e;
     while (e1 != ∅) {
       从e1中任取一条边(u,v);
       cset=cset∪{u,v};
       从e1中删去与u和v相关联的所有边;
     }
     return c
}
```

Cset用来存储顶点覆盖中的各顶点。初始为空，不断从边集e1中选取一边(u, v)，将边的端点加入cset中，并将e1中已被u和v覆盖的边删去，直至cset已覆盖所有边。即e1为空。

顶点覆盖问题的近似算法



图(a)～(e)说明了算法的运行过程及结果。(e)表示算法产生的近似最优顶点覆盖cset，它由顶点b, c, d, e, f, g所组成。(f)是图G的一个最小顶点覆盖，它只含有3个顶点：b, d和e。

算法approxVertexCover的性能比为2。