近似算法

黄刘生

2013年9月16日

目录

1 NP-完全性理论

计算机科学的局限性

- **问题的可解性**：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。可计算函数对应的问题是**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

可计算性与计算复杂性

- **可计算性问题**：其计算复杂性如何度量?
  - **多项式时间算法**：设n是输入的size，k是某个常数，其最坏情况下的运行时间是 $O(n^k)$ 的算法。
  - **超多项式时间算法**：问题可解，但并非对任意的常数k其算法的时间均是 $O(n^k)$ 。

  通常认为：

  多项式时间可解的问题是易解的

  超多项式时间可解的问题是不易解的

- **不可计算性问题**：无论使用何种计算机，无论使用多少时间都无法解决的问题。例如，图灵停机问题。

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halt(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  halt(Y,Y)  then
           goto a;
       else  halts;
}
```

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

即：halt(P,X)并不存在，停机问题是不可解的!

问题形式化的定义

- **抽象问题**

抽象问题 Q 是一个输入实例集合I到问题的解集合S 的二元关系。

例： Shortest-Path

实例：<G,u,v>， 解：G的顶点序列（可能为空）

最短路径不一定唯一，故1个给定的实例可能有多个解

判定问题 f： 实例集I→解集{0,1}

优化问题转化 最优Shortest-Path ⇒ 判定Path

Path={ <G,u,v,k>: u和v之间是否存在1条至多包含k条边的路径，u,v是顶点，k是某整数 }

编码

- **编码**：抽象问题必须编码为具体问题才能被程序理解和求解

  S(抽象对象集合)的**编码**是从S到二进制串集合的映射e:

  即： $e: S \rightarrow \{0,1\}^*$

- **具体问题**：将实例集编码为二进制串集合的问题称为具体问题
- **算法的时间复杂度**：是对具体问题定义的，输入实例的size是其编码长度

设算法的时间是$\Theta(k)$, 整数k是唯一的输入，若k是一元编码（串为k个1），则对编码长度为n的输入，算法的时间是$O(n)$

若k采用二进制编码，当编码长度为n时，$n=\lfloor \lg k \rfloor+1$，则算法的时间是$\Theta(k) = \Theta(2^n)$

形式语言框架

- **字母表**：字母表∑是符号的有限集
- **语言 L**：字母表∑上的语言L是由表中符号组成的串的任意集合，ε表示空串，Φ表示空语言
- **闭包∑***：∑上所有串构成的语言，∑上每个语言均是∑*的一个子集
- **接受/拒绝串**：设输入$x \in \{0, 1\}^*$, 若算法输出 $A(x) = 1$，则称算法A**接受**串x；若 $A(x) = 0$，则算法A**拒绝**串x。
- **接受语言**：算法A接受的串的集合，即

$$L = \{ x \in \{0, 1\}^* : A(x) = 1 \}$$

- **算法A接受语言L**：∀x∈L，A(x)=1

形式语言框架（续）

- **判定语言L**

  若算法A接受L中的每个串，拒绝每个不在L中的串，则称L是由A判定的语言

- **算法A判定语言L**

  $\forall x \in L$, A(x)=1; $\forall x \notin L$, A(x)=0;

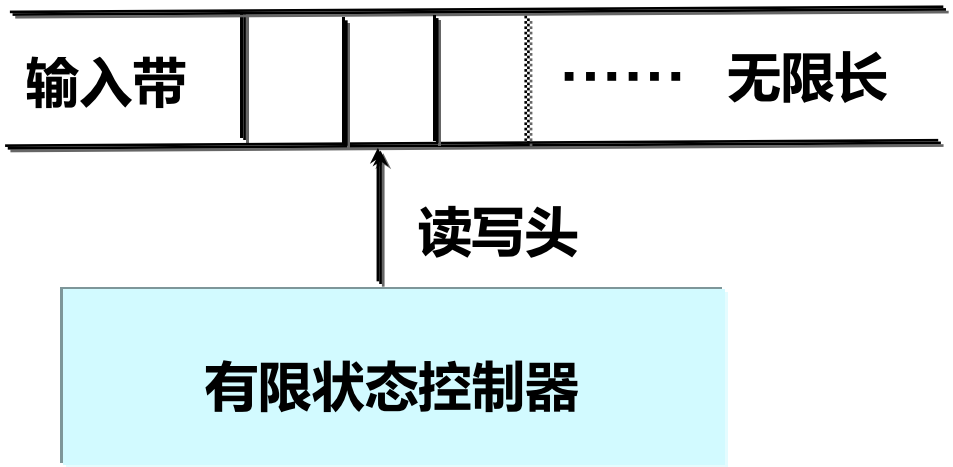
- **多项式时间接受的语言L**

  若存在某个常数k，对任意长度为n的串 $x \in L$, 算法A在$O(n^k)$时间内接受x， 则称L是算法A在多项式时间内**接受**的语言。

- **多项式时间判定的语言L**

  若存在某个常数k，对任意长度为n的串 $x \in \{0, 1\}^*$, 算法A可在$O(n^k)$时间内正确地**判定**是否有$x \in L$， 则称L是算法A在多项式时间内可判定的语言。

图灵机（TM）



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（yes）。

形式化定义：

P={ L：L是一个能在多项式时间内被一台DTM所接受的语言}

非形式定义：

P是所有用**确定性算法**在多项式时间内可解的(判定)问题的集合

P、NP及NPC类问题

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（yes）。

形式化定义： NP={ L：L是一个能在多项式时间内被一台NTM所接受的语言}

非形式定义：NP是所有用非确定性算法在多项式时间内可解的(判定)问题的集合

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（确定性）算法吗?

P、NP及NPC类问题

- **NP类问题（续）：** 通常，其求解可分为两个阶段：
  - **猜测阶段**

    对规模为n的输入实例x，产生一个输出y。

    通常是用多项式时间的**非确定算法**产生解y，可以通过增加1条非确定性选择语句实现之：

    choice(S)： 任意选择集合S的1个元素。

    对同一个实例x，下次输出的解就不一定是y。
  - **验证阶段**

    检验y是否满足解的形式，是否是问题的解？

    验证阶段是使用多项式时间的**确定性**算法。

P、NP及NPC类问题

非确定算法执行

可以设想非确定机器在执行choice(S) 语句时，不是随机选择，而是并行地计算，可以想象是算法的多个副本同时执行，若正确值存在，则它具有主动选择正确值（导致success）的能力。

非确定机器在现实技术条件下是不存在的，是为了分析计算机复杂度难题而虚构的理论模型。

非确定的判定算法

输出0：当且仅当没有一种选择可导致success

输出1：当且仅当至少存在一种选择导致success

P、NP及NPC类问题

非确定算法

查找： 在数组A[1..n]（ n≥1)中查找x，即确定j使A[j]=x，若x∉A, 则j=0。

```
Search（A）{
    j = choice( A );
    if  A[ j ]=x then {
        print( j ); success//若x∈A，必定成功
    }
    print(0); failure // 当且仅当A中没有j满足A[j]=x
}
```

时间：O（1）

P、NP及NPC类问题

■ 非确定算法

命题演算公式可满足： 设$E(x_1,x_2,\ldots,x_n)$是由布尔变量x_i组成的命题演算公式，可满足性问题是：是否存在一组对$(x_1,x_2,\ldots,x_n)$的真值赋值，使得公式$E(x_1,x_2,\ldots,x_n)$为真。

```
Eval  (E, n)  {
    for i=1 to n do
        xi = choice( ture,false);
    if  E(x1,x2,…,xn) =ture then success//可满足
    else  failure
}
```

时间：O（n+|E|）

P、NP及NPC类问题

- **验证算法A**

  A有2个参数：x是待验证的输入串，y是称为“证书”的二进制串。算法A验证了输入串x是指：

  若 $\forall x \in L$，$\exists$证书y，使得A(x, y)=1。

- **算法A验证的语言**

  $L=\{ x \in \{0,1\}^*: \exists$证书$y \in \{0,1\}^*$, 满足$A(x, y)=1\}$

  直观上，若对任意串$x \in L$，都存在一个证书y，算法A能用y来证明$x \in L$，则算法A就验证了语言L 。

- **多项式时间验证**

  若存在一个多项式时间算法A和常数c，满足：

  $L = \{ x \in \{0,1\}^* : \exists$证书$y$，$|y| = O(|x|^c)$ 满足 $A(x,y)=1 \}$

  则称算法A在多项式时间内验证了语言L。

P、NP及NPC类问题

NP类问题（续）

多式时间内可**验证**的问题（指验证其解的正确性）

验证：给定一个问题的**实例**和**证书**（Certificate，证书相当于**证据**），验证该证书是否是问题的正确答案。

例如哈密尔顿回路问题：实例G=(V,E)，证书 $V_h=\{v_1,v_2,\ldots,v_n\}$

验证：顶点序列V_h(证书)是一个汉密尔顿回路吗？若是，则该序列就是实例G是哈密尔顿图的**证书/证据**。

验证算法：

1）遍历V_h，每个点仅出现1次?

2）$\forall(v_i,v_{i+1})\in V_h$ for $1\le i\le n-1$，检验$(v_i,v_{i+1})\in E$及$(v_n,v_1)\in E$?

可在多项式时间$O(n^2)$完成，其中n是图G的**编码长度**。

P、NP及NPC类问题

- **NP类问题的另一种定义**
  - **定理**：语言L可在多项式时间内被**非确定性**地**接受**，当且仅当L可在多项式时间内被**确定性**地**验证**

    非形式地：多式时间内**可验证**的问题即为NP问题
- **NP类问题的两个等价定义**

  NP= {NDTM能在多项式时间内接受的语言类}

  NP= {DTM能在多项式时间内验证的语言类}

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I变换成L_2的实例f(I)。若对L_1的每个实例I，I的答案为“yes”当且仅当f(I)是L_2的答案为“yes”的实例，即：

  $\forall I$，I是L_1输出“yes“的实例 $\Leftrightarrow$ f(I)是L_2输出”yes“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，L_2至少比 L_1难。

- **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)$q \in NP$

  (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：$NPC \subseteq NP\text{-}hard$

- **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$\notin$NP。但是

  可满足问题SAT$\leqslant_p$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⊆NP

是否有NP⊆P？即是否NP=P？

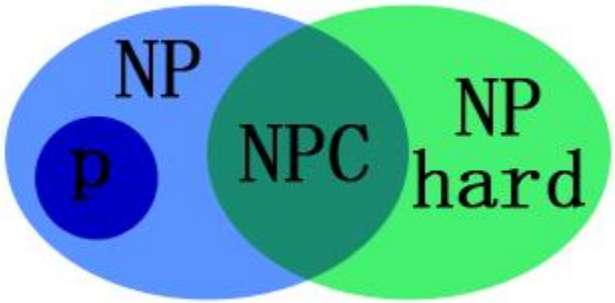
美国麻省的Clay数学研究所于2000年5月24日在巴黎法兰西学院宣布：对七个“千年数学难题”中的每一个均悬赏100万美元，而问题NP=？P位列其首：

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. 贝赫和斯维讷通－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?

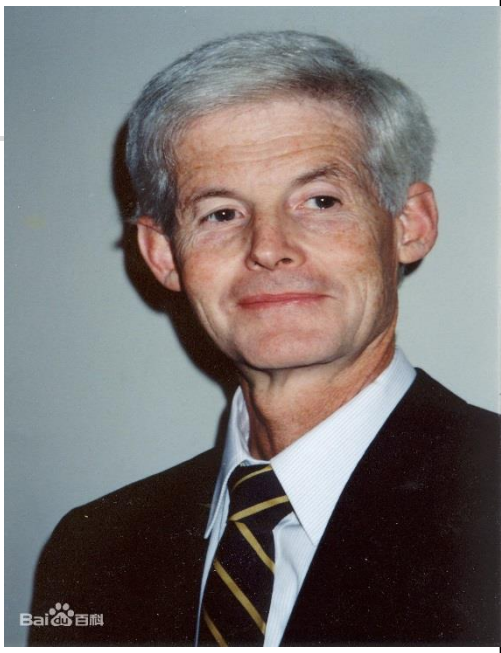
若已知q'∈NPC或q'∈NPH，且q'≤$_p$q，则q∈NPH；若进一步有q∈NP，则q∈NPC。

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 ∈NP，则q是NPC的。

∵ NP中任意问题均可多项式归约到q'，由于≤p有传递性

∴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(Satisfiablity problem)是NP完全问题，亦即SAT∈NPC。且证明了：

SAT∈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。

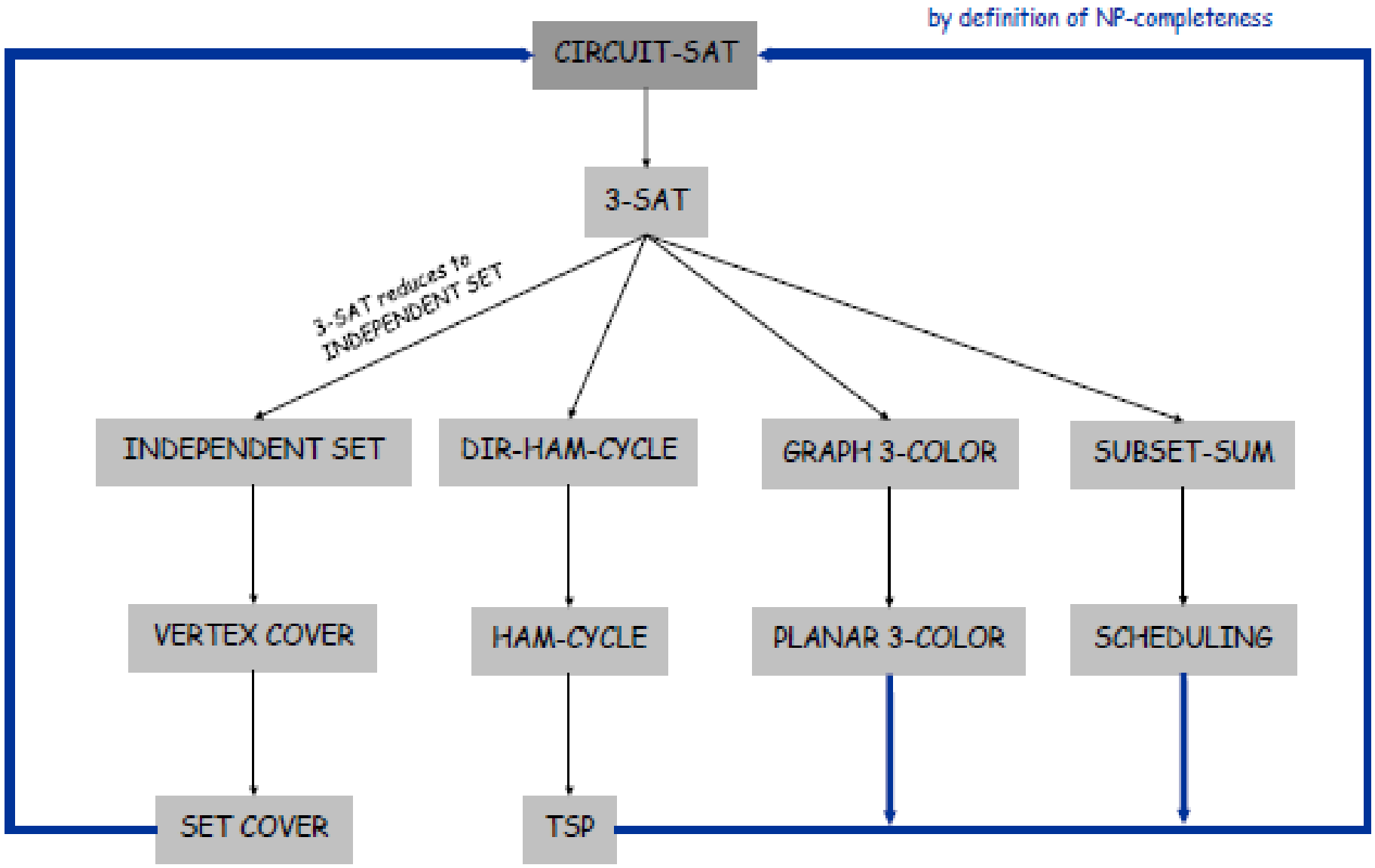
NP-完全性理论

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.

部分NPC问题图


by definition of NP-completeness
CIRCUIT-SAT
3-SAT
3-SAT reduces to INDEPENDENT SET
INDEPENDENT SET
DIR-HAM-CYCLE
GRAPH 3-COLOR
SUBSET-SUM
VERTEX COVER
HAM-CYCLE
PLANAR 3-COLOR
SCHEDULING
SET COVER
TSP

NPC问题分类

1.包装问题：SET-PACKING，INDEPENDENT SET

2.覆盖问题：SET-COVER， VERTEX-COVER

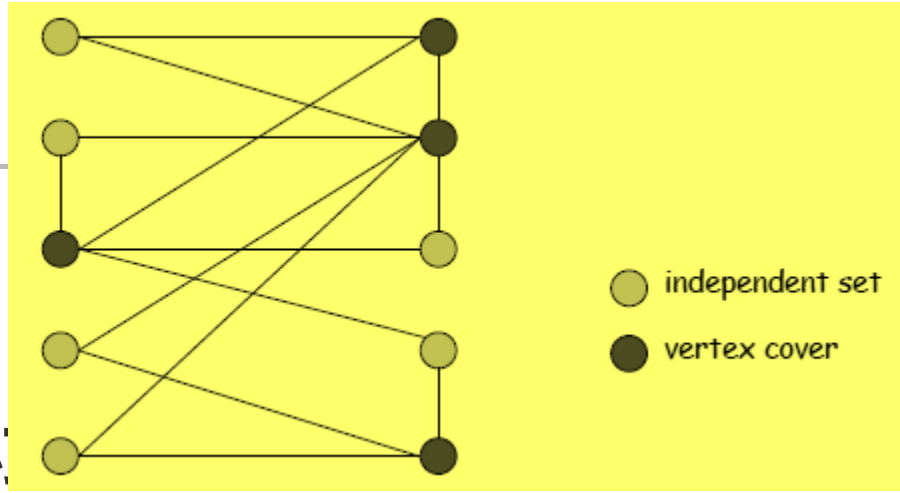
3.约束满足问题：SAT， 3-SAT

4.序列问题：HAMILTONIAN-CYCLE， TSP

5.分区问题：3D-MATCHING, 3-COLOR

6.数值问题：SUBSET-SUM， KNAPSACK

三种规约策略



1、简单等价规约（续）

若$X \leq_p Y$和$Y \leq_p X$，则$X \equiv_p Y$。

例1：最大独立集问题 $\equiv_p$ 最小

- **独立集I**：G=<V,E>，设I⊆V，I中任何点对之间无边，即E中每条边**至多**有1个端点在I中。

  Independent-Set={ <G,k>: G有一个size≥k的独立集}

- **顶覆盖C**：G=<V,E>，设C⊆V，E中每条边**至少**有1个端点在C中。

  Vertex-Cover={ <G,k>: G有一个size≤k的顶点覆盖}

两个集合互补：I是一个独立集当且仅当V-I是一个顶点覆盖集，即C=V-I。

三种规约

1、简单等价规约（续）

- Independent-Set $\leq_p$ Vertex-Cover （式1）

令I为任意独立集，$\forall (u,v) \in E$

∵ I是独立集，I中任何点对间无边，每边至多有1端点在I中

∴ $u \notin I$ 或 $v \notin I \Rightarrow u \in V - I$ 或 $v \in V - I$

即：u，v至少有1个顶点在 C=V-I 中

遍历E中所有边，检查其端点是否在I中，将不在I中的端点放入点集C中。此过程在多项式时间$O(n^3)$内完成

三种规约

1、简单等价规约（续）

- Vertex-Cover $\leq_p$ Independent-Set （式2）

设C为任意的顶点覆盖集，令I=V-C，则I是独立集

Pf: 要证明I中任意点对之间无边，即：$\forall$ u,v $\in$ I，(u,v) $\notin$ E(G)

反证: 若 (u,v) 是E(G)中的1条边，因为C=V−I是顶点覆盖集，则(u,v)至少有1个端点在V-I中。即：

u $\in$ V - I 或 v $\in$ V – I $\Rightarrow$ u $\notin$ I 或 v $\notin$ I

与 u $\in$ I and v $\in$ I 矛盾!

此过程亦可在多项式时间内完成

由式1和2可得：Vertex-Cover $\equiv_p$ Independent-Set

三种规约

$S=\{1,2,3,4,5,6,7\}$，$k=2$

$S_1=\{3,7\}$　　$S_4=\{2,4\}$

$S_2=\{3,4,5,6\}$　　$S_5=\{5\}$

$S_3=\{1\}$　　$S_6=\{1,2,6,7\}$

2、从特殊到一般

例2：Vertex-Cover ≤$_p$ Se

- 集合覆盖S’

非空集S 的一个覆盖S’ 是S 的若干非空子集的集合，他们的并集等于S。即：

$S_i \subseteq S$, $S_i \neq \Phi$ ($1 \le i \le m$), 若$J \subseteq \{1,2,\ldots,m\}$ 且$\bigcup_{j\in J} S_j = S$ ，

则称 $\{S_j\}_{j\in J}$ 为S 的一个覆盖。注意：子集的交未必为空!

图例：最小集合覆盖

Set-Cover={ <S,k>, S有一个size≤k的集合覆盖 }

图中S’ = { S_2, S_6 }是S的一个最小的集合覆盖

三种规约-从特殊到一般（续）

例2：Vertex-Cover $\leq_p$ Set-Cover（续）

Pf：设C={ <G,k> }是1个顶点覆盖集，我们构造1个集合覆盖，其大小与C相等。

令集合U为G中的边集合，即U=E(G)

∀v∈C，构造U的子集 S_v= { e∈E(G)： 所有关联顶点v的边e }

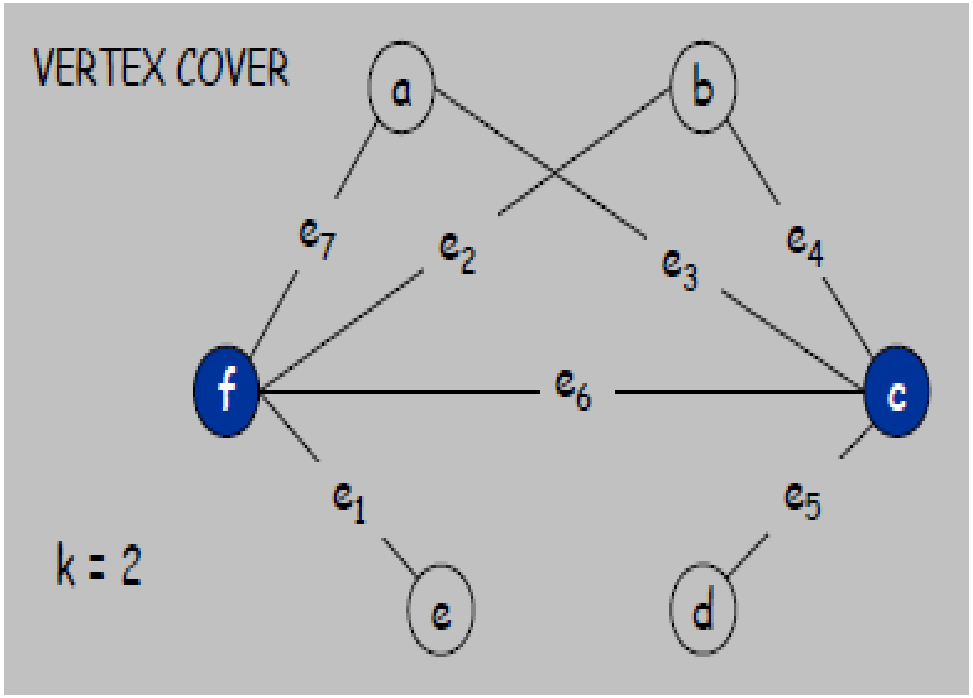
顶覆盖C中有k个顶点，将每个顶点构造U的1个子集S_v，下面证明这k个构造的子集之并集等于G的边集U。

设：S={ S_v： ∀v∈C }，则 |S|=|C|=k

∀e∈E(G), 设e=(u,v), 由顶覆盖定义知，u∈C or v∈C，则:

若u∈C 则e ∈S_u，若v∈C 则e ∈S_v 即E中任一条边至少在S中的某个子集中，S中子集之并集=E(G)=U。S是U的size为k的集合覆盖

三种规约-从特殊到一般（续）



SET COVER

U = {1, 2, 3, 4, 5, 6, 7}
k = 2

S_a = {3, 7}　　S_b = {2, 4}
S_c = {3, 4, 5, 6}　　S_d = {5}
S_e = {1}　　S_f = {1, 2, 6, 7}

这里：U=E(G)={e_1, e_2, e_3, e_4, e_5 , e_6, e_7 }, C={ c, f }

S_c={ e_3, e_4, e_5, e_6 }，　S_f={ e_1, e_2, e_6, e_7 }　S={ S_c, S_f }

U= S_c ∪ S_f , S是U的最小集合覆盖。即：C ≤$_p$ S

三种规约

3、通过编码（Encoding with gadgets）

■**3-SAT**：**3-CNF的可满足性**

- **字面量(literal)** **布尔变量或其“非”：** x_i， $\overline{x}_i$
- **子句(clause)** **若干个字面量的析取：** $C_j = x_1 \vee \overline{x}_2 \vee x_3$
- **合取范式(Conjunction Normal Form, CNF)**

  所有子句的合取(“与”)的布尔公式
- **3合取范式(3-CNF)**

  每个子句中恰有3个不同字面量的合取范式
- **3-SAT** **3-CNF的可满足性问题是指3-CNF形式的布尔公式 ϕ 是否存在一组变量的真值赋值，使得公式 ϕ 的值为真。**

三种规约

3、通过编码（Encoding with gadgets）

- 3-SAT $\leq_p$ Independent-Set

设$\Phi=C_1 \wedge C_2 \wedge \cdots \wedge C_k$，构造G的独立集S

对Φ中每个子句$C_i=z_{i1} \vee z_{i2} \vee z_{i3}$，在G中创建3个顶点$z_{i1}, z_{i2}, z_{i3}$和3条边$(z_{i1}, z_{i2})$, (z_{i2}, z_{i3}), (z_{i3}, z_{i1})的三角形，由此形成点集V_i和边集E_i。

若在2个不同子句中有 x_i和$\overline{x}_i$ ，则在这2点间连1条边，称其为冲突边，表示他们不可能同时赋值1，故对应的2点不可能同时入选S。

Φ可满足 $\Leftrightarrow$ S是G的size为k的独立集（S由每个V_i选1点组成）

$\Rightarrow$设有1个Φ=1赋值,对每个C_i, 若z_{ij}使得C_i=1, 则$S=S \cup \{z_{ij}\}$。

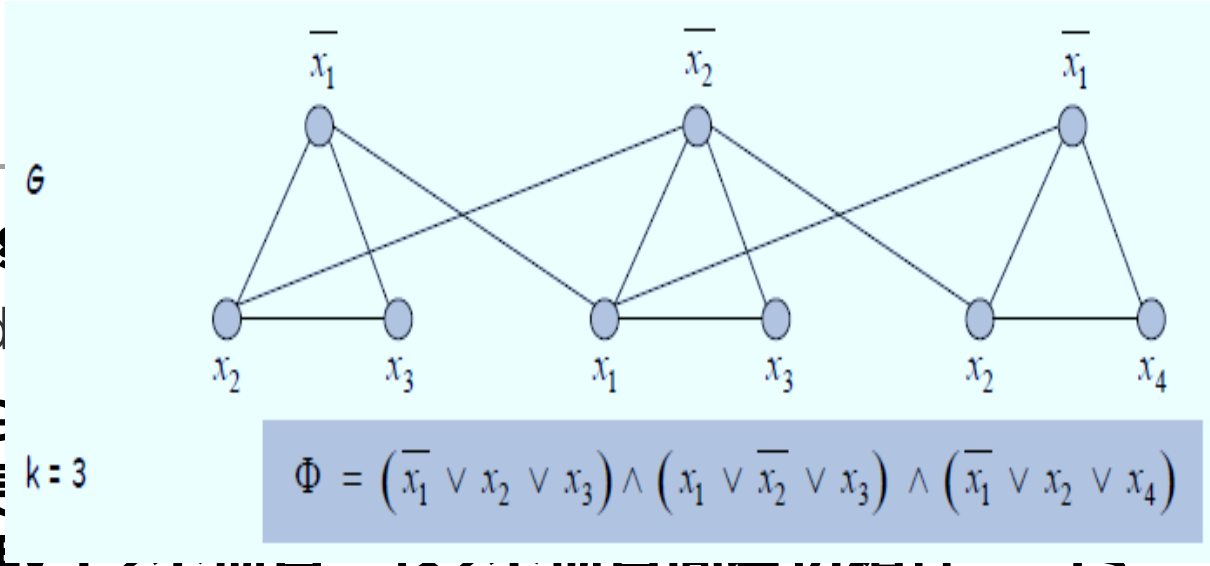
∵每个V_i中只选了1个顶点加入S，且冲突边的2个顶点中至多只有1个顶点被选入S，即：S中任意点对之间不可能有边。

∴S是独立集

三种规约

3、通过编码（Encoding

- 3-SAT $\leq_p$ Independ



⇐在如前构造的图G
三角形中， S一定是
会在某个三角形中取了2个顶点，这2个顶点间有边相连，与S
是独立集矛盾！

∀i∈[1,k]，若在V_i中是选择z_{ij}加入S，则令z_{ij}=1，因此与V_i对应的子句C_i=1， 故有：$\Phi=C_1 \wedge C_2 \wedge \cdots \wedge C_k = 1$

因此，存在一个真值赋值，使得Φ是可满足的。

图例：

G的独立集S={$z_{12}(x_2)$, $z_{21}(x_1)$, $z_{33}(x_4)$ }

真值赋值 C1: x_2=ture, C2: x_1=ture, C3: x_4=ture ⇒ Φ=ture

NPC问题证明举例

例：证明顶点覆盖问题是NP完全的。

1）先证Vertex-Cover∈NP

假设已知一个图G=（V,E）和整数k，选顶点覆盖集V’作为G的证书。验证算法先验证|V’|=k，然后对每条边（u，v）∈E，检查是否有u ∈V’或v∈ V’？很容易在多项式时间内完成上述验证。

2） Independent-Set $\leq_p$ Vertex-Cover

见前面证明。因为NP完全问题Independent-Set可在多项式时间内规约到顶点覆盖问题 ，故顶点覆盖问题亦是NPC问题。

Ex.：给出与图的独立集问题相关的判定问题的形式化描述，并证明它是NP完全的（提示：根据团问题进行规约）。

NP-完全性理论的局限性

■ NP-完全性理论的局限性

易(难)解问题：多项式（超多项式）时间内求解的问题

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 vs. 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 vs. 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中有通过每个顶恰好1次的圈？NPC

3）2-CNF vs. 3-CNF 可满足性

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

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（弱NPC问题）。

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。

■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：

- **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∏是：

对于给定的I∈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)。

§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：

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

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;

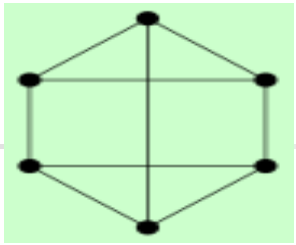
2)否则,计算5染色;//可在多项式时间内,∵任何平面图G是
//可5染色的(实际上四色定理告诉我们G是可4染色的)

} 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：

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

§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：

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：$I=\{1,2,\ldots,n\}$
- 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$I'\subseteq I$，$\sum_{i\in I'} s_i \le B$

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。

§1.2.2 绝对近似算法之否定

■**Th.1.6** 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：$|A(I)-OPT(I)| \le k$

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）

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。

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：

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

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

§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≠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mα（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

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

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。

§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，…，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。

§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

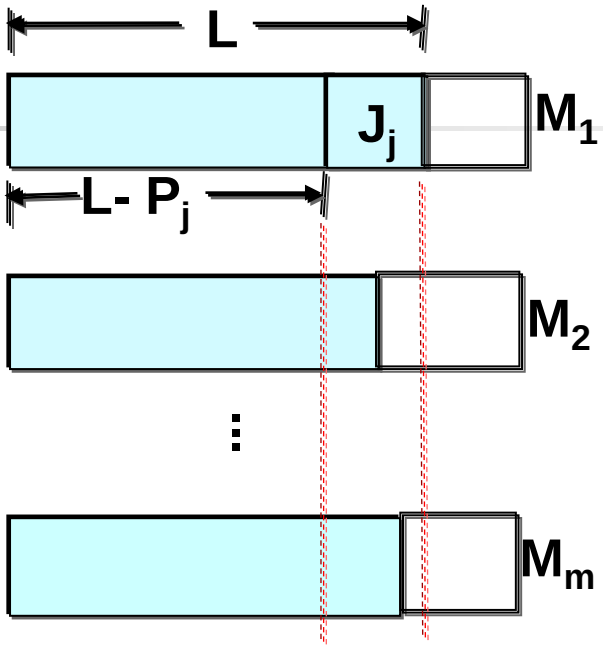
$$\frac{A(I)}{OPT(I)} \le 2 - \frac{1}{m}$$

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：

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

但I的最优解的值显然满足：$$OPT(I) \geq \frac{\sum_{i=1}^{n} P_i}{m} \quad (2)$$

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J_j，∴$OPT(I) \geq P_j$

于是：$A(I)/OPT(I) \leq 2-1/m$

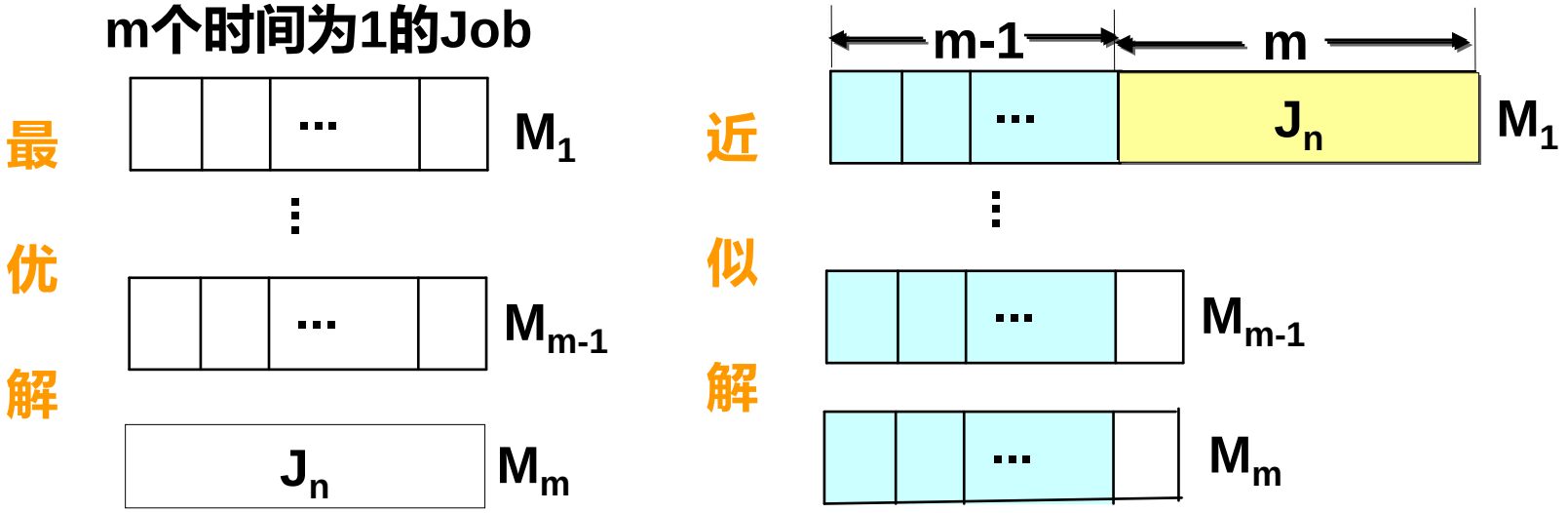
§1.3.1 多机调度

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：

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

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

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

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，

∵ A(I)=OPT(I)　∴A(I)/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

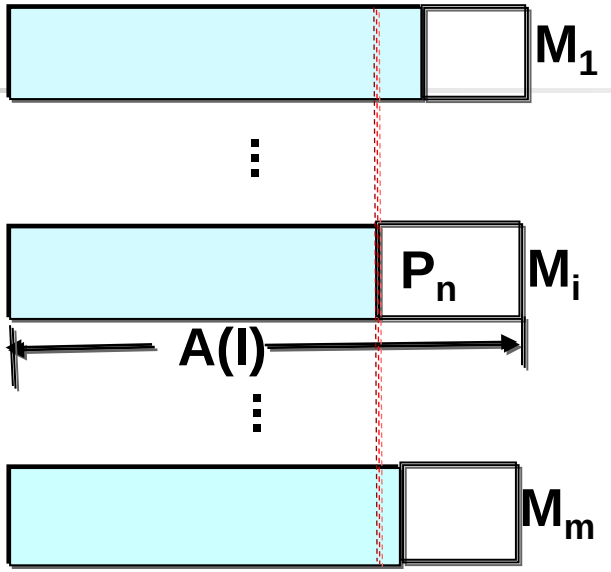
Pf(续)：则可设违反该定理具有最少作业数的实例I是$J_1,J_2,\ldots,J_n$，不妨设$P_1 \geq P_2 \geq \ldots \geq P_n$

LPT的调度次序是$J_1,J_2,\ldots,J_n$,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$J_1,J_2,\ldots,J_k$(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，∵ |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≤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：

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

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

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

§1.3.1 多机调度

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

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

∵ P_n是实例I中时间最短的作业

∴ 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$J_{n+1},\ldots,J_{2m}$的时间均为0，将其加入I，不妨设$P_1\ge P_2\ge\ldots\ge P_{2m}$

§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$i \le m, j > m$。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$s,t > m$.

∵ $P_i, P_j \ge P_s, P_t$，交换P_j和P_t，则

$$P_i + P_t \le P_i + P_j \qquad P_s + P_j \le P_i + P_j$$

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。

∴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

§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:

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

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！

§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$S_i \in [0,1](1 \le i \le n)$，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，$B_1,B_2,\ldots,B_j$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，$R_{FF}(I)\leq 2$

  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：

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

■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：$R_{FF}(I) = \dfrac{FF(I)}{OPT(I)} \leq \dfrac{2\lceil \sum S_i \rceil}{\lceil \sum S_i \rceil} = 2$

可以证明上述比可达7/4

Th.1.10 R_{FF}^{∞} =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！

§1.3.2 装箱问题（BP）

■递减首次适合 (First Fit Decreasing FFD)

将物品按照size从大到小排序，然后用FF策略

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

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ΔTSP（Metric TSP）问题。

§1.3.3 旅行商问题（TSP）

Def1.8: 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。

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

§1.3.3 旅行商问题（TSP）

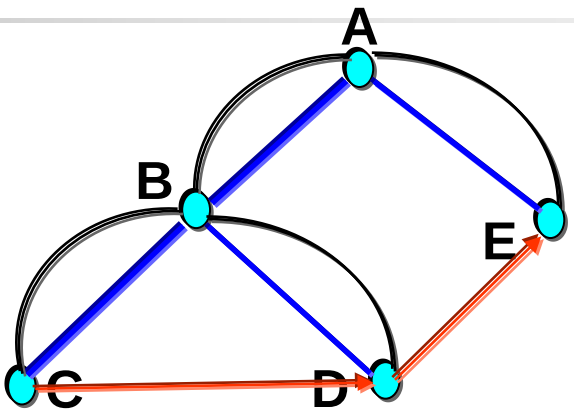
■**MST启发：** 有几种启发已达到渐近性能比2，大多数好的启发式是基于找到一个**欧拉环**(Euler Tour,简称**ET**)，然后使用“Short-cut”来获得一个哈密尔顿圈。

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。

§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ΔTSP时，有：$R_{MST}^{\infty} = 2$

Pf： 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。

设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$A_{MST}(G) \le d(ET) \le 2OPT(G)$: 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：

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

§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M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，故任一顶点数为偶数的S均存在一个匹配。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中奇度点集O上的最小权匹配集，则可断言：

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

§1.3.3 旅行商问题（TSP）

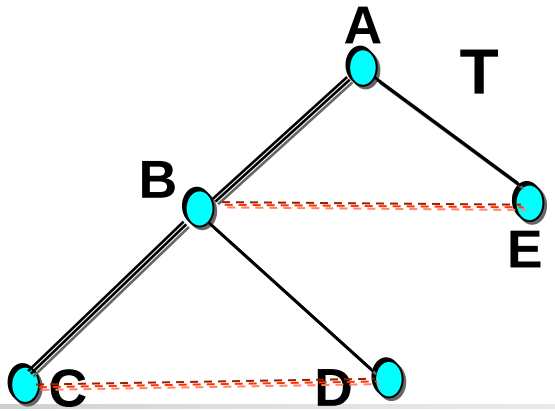
在1个最优解中做短路操作以排除不在O中的顶点，所得的圈X(O上的圈)必满足：

$$d(X) \le OPT(G)$$ (由Δ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

§1.3.3 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)。

§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⊆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（最小）当且仅当I是MIS的最优解（最大）

§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得不够，但是我们仍能使用NPC理论来证明：除非P=NP，否则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**$R_{MIN}(\Pi)$定义为：

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

- R_{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

§1.3.4 相对近似之否定结果

Th.1.16 若P≠NP，则$R_{MIN}(TSP)=\infty$

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≠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！

§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：

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

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

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

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

§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：$R_A(I,\varepsilon) \leqslant 1+\varepsilon$。

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size[I]和1/ε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小而增长太快！

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$\{P_1,\ldots,P_n\}$和机器数m；

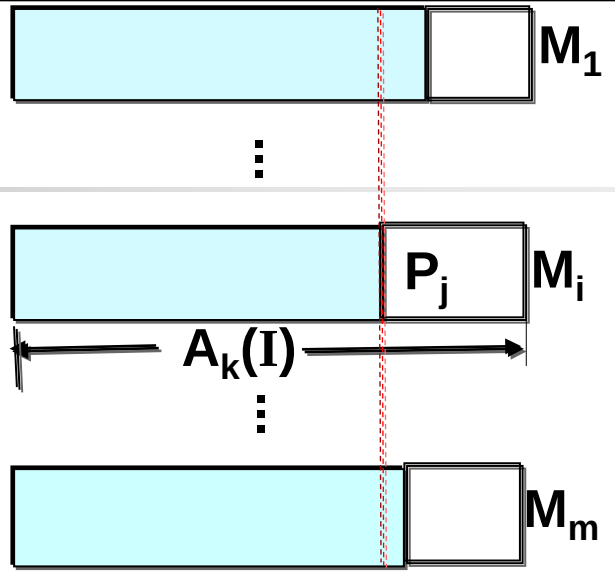
Output：一个可行的调度；

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

§1.4 其他



Th.算法A_K的绝对性能比：

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

Proof：

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$T=\sum P_i$是n个作业总的运行时间，则：

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

§1.4 其他　　$T \geq m(A_k(I)-P_j)+P_j$

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, $j \geq k+1$, ∴ $P_{k+1} \geq P_j$

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

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

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

§1.4 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

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

§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(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