第五讲

复杂性理论

主讲教师： 朱建明

Email: jmzhu@ucas.ac.cn

Homepage: http://people.gucas.ac.cn/~jianming

- ✓ 问题规模
- ✓ 计算复杂性
- ✓ **NP-complete**

问题规模

- **问题（problem）**：要回答的一般性提问，通常含有若干个满足一定条件的参数（或自由变量）。可以从两方面描述：

  （1）对所有参数的一般性描述；

  （2）答案（或解）必须满足的性质。

- **实例（instance）**:给问题的所有参数指定具体值，得到问题的一个实例。这些具体值称为**数据**；这些数据输入计算机所占的空间称为**实例的长度**（size）.

最优化问题

- 一类最优化问题是由一些类似的具体问题（实例）组成的，每一个具体问题可表达成二元组（F,C）.F为可行解集合;C是费用函数，是由F到R（实数集）的映像。问题是在F中找到一个点f*，使对F中任意的f，有C(f*)≤C(f),称f*为这一具体问题的最优解（或全局最优解）.

- ✓ **算法计算量的度量：**

  加、减、乘、除、比较的总运算次数与实例的计算机计算时的二进制输入数据的大小关系。

- ✓ **正整数x的二进制位数是**:(整数到二进制的转换)

记x的输入规模（编码长度）为$l(x)$，则

$$l(x)=\lceil \log_2(|x|+1)\rceil+2$$

其中2是考虑了1个符号位和1个数据分隔位。

✓ 算法计算量的度量之例——TSP枚举法

计算量的统计：

城市数n

第一城市为始终点，计算一条路径$(1, i_2, \cdots, i_n, 1)$长度的基本运算为两两城市间距离的n个数求和, 共有$(n-1)!$条路径,

求和运算次数为:$\{(n-1)!\}n = n!;$

枚举所有路径进行$(n-1)!$次比较可得最优路径, 基本计算总次数为

总计算量:$(n! + (n-1)!)$

✓ 实例的输入长度：

设对$\forall i, j$有$d_{ij} \le K$,则

$$L \le n(n-1)(\lceil \log_2(K+1) \rceil + 2) + \lceil \log_2(n+1) \rceil + 2$$

- 实例的输入长度是n的多项式函数
- 枚举法的基本计算量是n的阶乘函数，随n的增加，比指数函数增加得还快.

- ✓ 评价算法的好坏——计算时间的多少、解的偏离程度
- ✓ 例　非对称距离TSP问题的算法实现：所有路径枚举。

计算时间：n个城市，固定1个为起终点需要(n-1)!个枚举，设计算机1秒能完成24个城市的枚举，则城市数与计算时间的关系如下表：

城市数	24	25	26	27	28	29	30	31
计算时间	1 sec	24 sec	10 min	4.3 hour	4.9 day	136.5 day	10.8 year	325 year

随城市增多，计算时间增加很快。到31个城市时，要计算325年。

描述算法的好坏——计算复杂性——讨论计算时间与问题规模之间的关系。以目前二进制计算机中的存储和计算为基础，以理论的形式系统描述，是评估算法性能的基础。

实例I，实例规模：$l(I)$，算法A

基本计算总次数：$C_A(I)$

存在函数$g(x)$和一个常数α，使得对于该问题的任意实例 I 都满足

$$C_A(I) \le \alpha g(l(I)) \qquad (X X)$$

则二者关系表示为：$C_A(I) = O(g(l(I)))$

$g(x)$的性质决定了算法的性能。

计算复杂性

- **时间复杂性和空间复杂性概念**

  算法的时间复杂性：算法对时间的需要量（加、减、乘、除、比较、读、写等操作的总次数）；

  算法的空间复杂性：算法对空间的需要量（存储空间的大小，二进制位数）；

  问题的时间复杂性：所有算法中时间复杂性最小的算法时间复杂性；

  问题的空间复杂性：所有算法中空间复杂性最小的算法空间复杂性；

- **复杂性问题分类**

  P类、NP类、NP完全类

- **复杂性表示方法**

  复杂性表示为问题规模n（如TSP的n）的函数，

  时间复杂性$T(n)$，关键操作的次数；

  空间复杂性$S(n)$，占用的存储单元数量；

- **复杂性表示方法**

若算法A的时间复杂性为$T_A(n)=O(p(n))$，$O(p(n))$为复杂性函数$p(n)$主要项的阶，且$p(n)$为n的多项式函数，则称算法A为多项式算法。

当不存在多项式函数$p(n)$时，称相应的算法为非多项式时间算法或指数时间算法；

随着变量的增加，多项式函数增长的速度比指数函数和非多项式函数增长的速度要慢得多。

- **P类问题（ deterministic polynomial ）**

  具有多项式时间求解算法的问题类

  迄今为止，许多组合优化问题都没有找到求最优解的多项式时间算法。

- **NP类问题(Nondeterministic polynomial)**

  定义1 实例是问题的特殊表现，所谓实例就是确定了描述问题特性的所有参数的问题，其中参数值称为数据，这些数据占有计算机的空间称为实例的输入长度。

- **NP类问题(Nondeterministic polynomial)**

  定义2 若一个问题的每个实例只有“是”或“否”两种回答，则称该问题为判定问题。

  例，TSP的判定问题：给定z，是否存在n个城市的一个排列W，使得$f(W)\leq z$。满足$f(W)\leq z$的一个排列W称为判定问题的“是”答案（可行解）。

- **NP类问题(Nondeterministic polynomial)**

若存在一个多项式 $g(x)$ 和一个验证算法 H，对一类判定问题 A 的任何一个“是”的判定实例 I 都存在一个字符串 S 是 I 的“是”回答，满足其输入长度 $d(S)$ 不超过 $g(d(I))$（其中 $d(I)$ 为 I 的输入长度），且验证算法验证 S 为 I 的“是”回答的计算时间不超过 $g(d(I))$，则称判定问题 A 为非多项式确定问题，简称 **NP问题**。

NP类问题是比P问题更为广泛的问题类。

- **简单地理解**

  如果计算工作量在计算数据规模n的多项式（如n，n^2，n^3等）的范围内，则计算花费的机时是可以接受的，该问题称为P问题；

  有些问题尚未证实是否属于P类，又未找到有效算法，则称NP问题。

- **NP－C （NP-Complete）和NP－hard类问题**

Cook在1971年给出并证明了有一类问题具有下述性质（1）这类问题中任何一个问题至今未找出多项式时间算法；（2）如果这类问题中的一个问题存在有多项式时间算法，那么这类问题都有多项式时间的算法，这类问题中的每一个问题称为NP完全问题，这个问题的集合简记NPC。

多项式转换与多项式归约

- **定义** 给定问题A_1和A_2，称A_1可多项式转换为A_2，若存在一个多项式函数g (x)，满足：(1)对A_1的任何一个实例I_1，在其输入长度的多项式时间$g(d(I_1))$内构造A_2的一个实例I_2，使其长度不超过$g(d(I_1))$；(2)由此构造使得实例I_1和I_2的解一一对应，且d_1为I_1的“是”回答的充要条件为d_1对应的解是I_2的一个“是”回答。
- **定义** 给定判定问题A_1和A_2，称A_1可多项式归约为A_2，若存在多项式函数$g_1(x)$和$g_2(x)$，使得对A_1的任何一个实例I，在多项式时间内构造A_2的一个实例，其输入长度不超过$g_1(d(I))$，并对A_2的任何一个算法H_2，都存在问题A_1的一个算法H_1，使得H_1调用H_2且计算时间$f_{H_1}(d(I)) \le g_2(f_{H_2}(g_1(d(I))))$。

- ✓ 由此可见，若问题A_2存在多项式时间算法，则问题A_1一定存在多项式时间算法。

$A_1 \propto_p A_2$： A_1可多项式归约为A_2

NP-Complete

- **NP－C（NP-Complete）和NP－hard类问题**

  定义： 称判定问题A∈NP-C，若A∈NP 且NP中的任何一个问题可多项式归约为问题A。称判定问题A为NP－hard，只要上述第二个条件成立。

定理： 设A是NP完全的，则

(1)A∈P当且仅当P＝NP；

(2)若$A \propto_p L_1$，且$L_1 \in NP$，则L_1是NP完全的。

定理的(2)可用来证明问题的NP完全性。但前提是：要有第一个NP完全问题L。

Cook定理

定理（Cook定理）：布尔表达式的可满足性问题SAT是NP完全的。

证明：SAT的一个实例是k个布尔变量x_1，…，x_k的m个布尔表达式A_1，…，A_m若存在各布尔变量x_i(1≤i≤k)的0，1赋值，使每个布尔表达式A_i(1≤i≤m)都取值1，则称布尔表达式$A_1\ A_2 \ldots A_m$是可满足的。

SAT∈NP是很明显的。对于任给的布尔变量x_1，…，x_k的0，1赋值，容易在多项式时间内验证相应的$A_1\ A_2\ \ldots A_m$的取值是否为1。因此，SAT∈NP。

现在只要证明对任意的L∈NP有$L \propto_p$SAT即可。

合取范式的可满足性问题（CNF-SAT）

问题描述：给定一个合取范式α，判定它是否可满足。

如果一个布尔表达式是一些因子和之积，则称之为合取范式，简称CNF(Conjunctive Normal Form)。这里的因子是变量x或$\bar{x}$。例如：$(x_1 + x_2)(x_2 + x_3)(\bar{x}_1 + \bar{x}_2 + x_3)$ 就是一个合取范式，而 $x_1 x_2 + x_3$ 就不是合取范式。

要证明CNF-SAT∈NPC，只要证明在Cook定理中定义的布尔表达式A，...，G或者已是合取范式，或者有的虽然不是合取范式，但可以用布尔代数中的变换方法将它们化成合取范式，而且合取范式的长度与原表达式的长度只差一个常数因子。

3元合取范式的可满足性问题（3-SAT）

问题描述：给定一个3元合取范式α，判定它是否可满足。

证明思路：

3-SAT∈NP是显而易见的。为了证明3-SAT∈NPC，只要证明CNF-SAT$\propto_p$ 3-SAT，即合取范式的可满足性问题可在多项式时间内变换为3-SAT。

团问题CLIQUE

问题描述：给定一个无向图G=(V，E)和一个正整数k，判定图G是否包含一个k团，即是否存在，$V'\subseteq V$，$|V'|=k$，且对任意u，$w\in V'$有$(u，w)\in E$。

证明思路：

已经知道$CLIQUE\in NP$。通过$3\text{-}SAT\propto_p CLIQUE$来证明CLIQUE是NP难的，从而证明团问题是NP完全的。

顶点覆盖问题（VERTEX-COVER）

问题描述：给定一个无向图G=(V，E)和一个正整数k，判定是否存在$V'\subseteq V$，$|V'|=k$，使得对于任意$(u，v)\in E$有$u\in V'$或$v\in V'$。如果存在这样的V'，就称V'为图G的一个大小为k顶点覆盖。

证明思路：

首先，VERTEX-COVER$\in$NP。因为对于给定的图G和正整数k以及一个“证书”V'，验证$|V'|=k$，然后对每条边$(u，v)\in E$，检查是否有$u\in V'$或$v\in V'$，显然可在多项式时间内完成。

其次，通过$\text{CLIQUE}\propto_p\text{VERTEX-COVER}$来证明顶点覆盖问题是NP难的。

- ✓ VERTEX-COVER
- ✓ CLIQUE
- ✓ Independent set

子集和问题（SUBSET-SUM）

问题描述：给定整数集合S和一个整数t，判定是否存在S的一个子集$S' \subseteq S$，使得S'中整数的和为t。例如，若S={1，4，16，64，256，1040，1041，1093，1284，1344}且t=3754，则子集S'={1，16，64，256，1040，1093，1284}是一个解。

证明思路：

首先，对于子集和问题的一个实例<S，t>，给定一个“证书”S'，要验证$t=\sum_{i \in S'} i$ 是否成立，显然可在多项式时间内完成。因此，SUBSET-SUM∈NP；

其次，证明VERTEX-COVER$\propto_p$SUBSET-SUM。

哈密顿回路问题（HAM-CYCLE）

问题描述：给定无向图G=(V，E)，判定其是否含有一哈密顿回路。

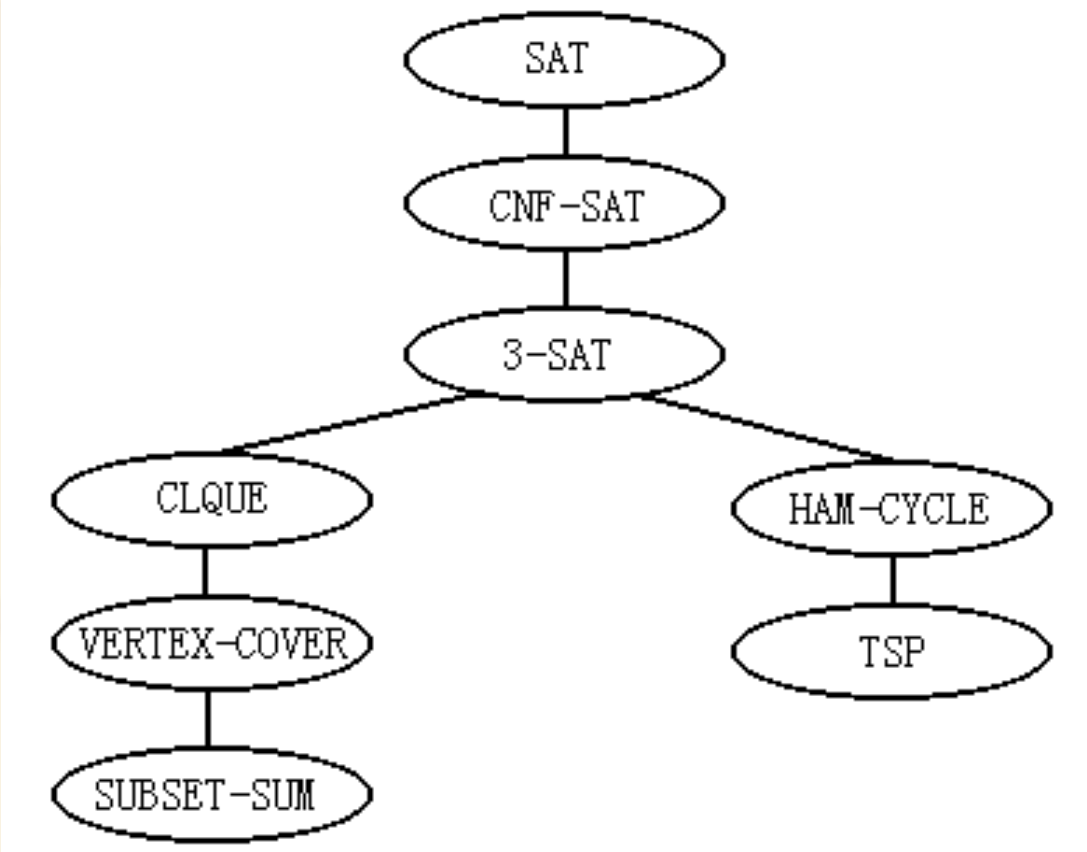
证明思路：

首先，已知哈密顿回路问题是一个NP类问题。

其次，通过证明3-SAT$\propto_p$HAM-CYCLE，

得出：HAM-CYCLE$\in$NPC。

典型的NP完全问题关系



部分NP完全问题树

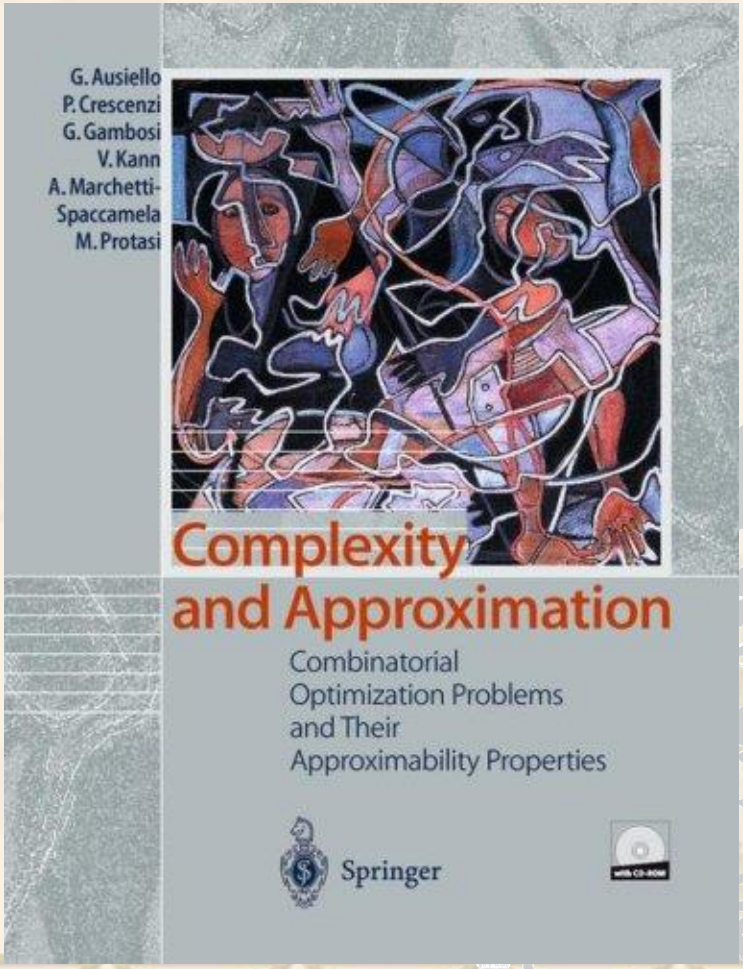
TSP<->Hamilton Circuit

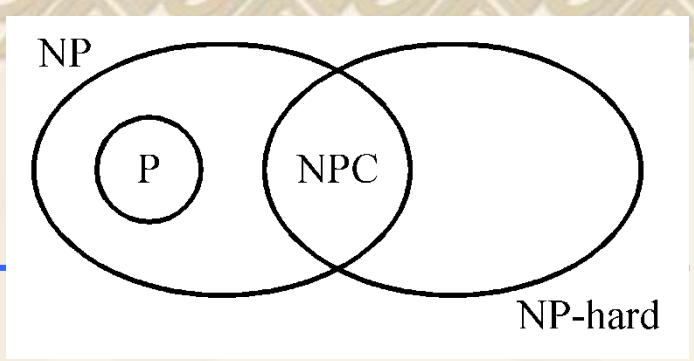
- ✓ 多项式归约方法

A list of NP optimization problems

200个

Graph theory，Network design，Sets and partitions，…





- **四类问题的关系**

NP-C问题具有重要的实际意义和工程背景:

第一个NP完全问题（Cook定理　1971）

六个NP完全问题（Karp 1972)

更多的NP完全问题

- ✦1979年：300多个
- ✦1998年：2000多个

TSP问题、背包问题、装箱问题、Job scheduling问题、集合覆盖问题等均是NP-C问题。

Co-NP

- ✓ Complement of NP

- ✓ Co-P
- ✓ Co-NPC

更难的问题

- ✓ 非NP问题，无法在多项式时间内验证解。例子：问一个图是否不存在Hamilton回路。
- ✓ 以平常的围棋规则在一个 n x n 的棋盘上下，给定一个残局（下了二个子就可以算残局），首先，是否可以确定黑子在最好的下法之下，一定会赢？
- ✓ 不可判定问题是更加困难的，例如停机问题。它们无法在任何给定时间内解决。

邻域

在组合优化中，在一点附件搜索另一个下降的点是组合最优化数值求解 的基本思想.

- 设 D 是一个组合优化问题中决策变量的定义域， 则

$$N : x \in D \to 2^D, \quad x \in N(x)$$

称为一个邻域映射. $N(x)$ 称为 x 的邻域, $y \in N(x)$ 称为 x 的一个邻居.

简单邻域算法

Step 1. 任选一个初始解 $x_0 \in D$.

Step 2. 在 $N(s_0)$ 中按一定规则选择一个 s; 若 $f(s) < f(s_0)$, 则 $s_0 := s$. 否则, $N(s_0) := N(s_0) - \{s\}$; 若 $N(s_0) - \{s_0\} = \emptyset$, 停止; 否则, 重复 step 2.

对优化算法的要求

- ✓ 优化方法要对问题的非线性和多极小性具有克服搜索过程陷入局部极小的能力
- ✓ 要对问题的大规模性和NP-hard性具有一定优化质量意义下的高效搜索能力
- ✓ 要对问题的多目标性和强约束性具有对各目标的合理平衡能力
- ✓ 要对问题的不确定性因素和算法本身参数具有良好的鲁棒性
- ✓ 要对连续与离散变量共存的特点具有搜索操作的灵活性和有效性
- ✓ 即，保证取得全局的优化质量、快速的优化效率、鲁棒的优化性能。