广州大学
GuangZhou University

8. 近似算法和随机算法（Approximation Algorithm && Randomized Algorithms）

广州大学 网络空间先进技术研究院

副教授 李默涵

limohan@gzhu.edu.cn

前情回顾

7.1 图灵机

7.2 P与NP类

7.3 cook定理

7.4 其他NP完全问题

7.5 其他问题类

7.6 应用分析

本章内容

8.1 近似算法

近似算法的基本概念

- 近似算法的基本思想
  - 很多实际应用中问题都是NP-完全问题
  - NP-完全问题是否存在多项式算法还不知道
- 求解NP-完全问题的方法：
  - 如果问题的输入很小,可以使用指数级（或者更高时间复杂度）的算法圆满地解决该问题.
  - 否则使用多项式算法求解问题的近似优化解.

近似算法

- 能够给出一个优化问题的近似解的算法

近似算法的性能分析

- 近似算法的时间复杂性分析
  - 分析的目标和方法与传统算法相同
- 近似算法解的近似度
  - 本节讨论的问题是优化问题
    - 问题的每一个可能的解都具有一个代价，问题的优化解可能具有最大或最小代价
    - 我们希望寻找问题的一个误差最小的近似优化解
    - 我们需要分析近似解代价与优化解代价的差距
      - Ratio Bound
      - 相对误差
      - (1+ ε)-近似

Ratio Bound

定义1(Ratio Bound) 设A是一个优化问题的近似算法, A具有ratio bound $p(n)$, 如果

$$\max\left\{\frac{C}{C*}, \frac{C*}{C}\right\} \leq p(n)$$

其中n是输入大小, C是A产生的解的代价， C^*是优化解的代价.

- 如果问题是最大化问题, max{C/C*, C*/C}=C*/C
- 如果问题是最小化问题, max{C/C*, C*/C}=C/C*
- Ratio Bound不会小于1
- Ratio Bound越大, 近似解越坏

相对误差

定义2(相对误差) 对于任意输入,近似算法的相对误差定义为$|C-C^*|/C^*$,其中C是近似解的代价,C^*是优化解的代价.

定义3(相对误差界) 一个近似算法的相对误差界为$\varepsilon(n)$,如果$|C-C^*|/C^* \le \varepsilon(n)$.

结论1. 设$p(n)$是算法A的ratio bound，则A相对误差$\le p(n)-1$.

Ratio bound:

$$\max\left\{\frac{C}{C^*},\frac{C^*}{C}\right\} \le p(n)$$

证明：

对于最小化问题：相对误差$=|C-C^*|/C^*=(C-C^*)/C^*=C/C^*-1\le p(n)-1$.

对于最大化问题：相对误差$=|C-C^*|/C^*=(C^*-C)/C^*$， $p(n)-1=C^*/C-1=(C^*-C)/C$。因为$C^*\ge C$，所以$(C^*-C)/C^*\le(C^*-C)/C$。

结论1表示，只要求出了Ratio Bound就求出了ε(n)。

对于某些问题，ε(n)和p(n)独立于n，用p和ε表示之。

近似模式

定义4（近似模式） 一个优化问题的近似模式是一个以问题实例I和$\varepsilon>0$为输入的算法$A(I, \varepsilon)$。对于任意固定ε，近似模式是一个算法，称为 $(1+\varepsilon)$-近似算法.

定义5（多项式时间近似模式） 一个近似模式$A(I, \varepsilon)$称为一个多项式时间近似模式，如果对于任意$\varepsilon>0$，$A(I, \varepsilon)$的运行时间是$|I|$的多项式.

例如，$O(n^{2/\varepsilon})$

定义 6（完全多项式时间近似模式） 一个近似模式称为完全多项式时间近似模式，如果它的运行时间是关于$1/\varepsilon$和输入实例大小n的多项式.

例如，$O((1/\varepsilon)^2 n^3)$

顶点覆盖问题

输入：无向连通图$G=(V, E)$

输出：$C \subseteq V$, 满足

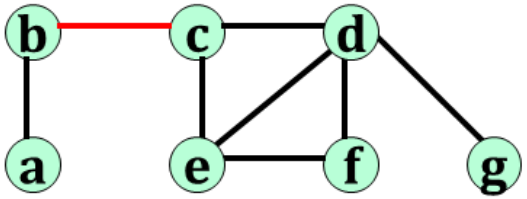
(1). $\forall (u, v) \in E$有$u \in C$或$v \in C$

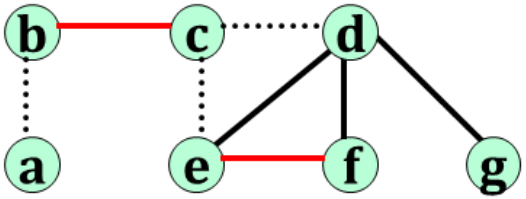
(2). C是满足条件(1)的最小集合。

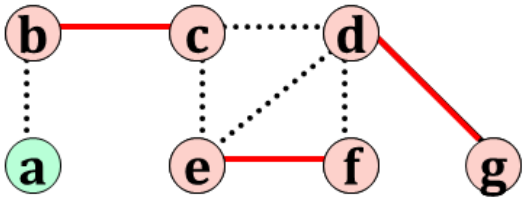
优化顶点覆盖问题是NP-hard问题.

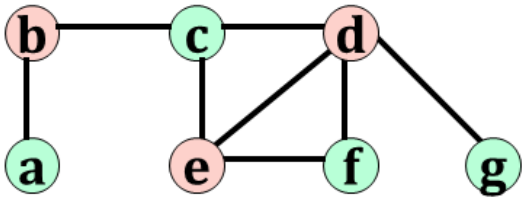
顶点覆盖问题的近似算法

每次任取一条边，节点加入C，删除相关边，重复该过程









算法解: {b, c, e, f, d, g}

优化解: **{b, e, d}**

顶点覆盖问题的近似算法（续）

APPROX-Vertex-Cover (G)

1. $C=0$
2. $E'=E[G]$;
3. While $E'\neq\emptyset$ DO
4. 任取$(u, v)\in E'$;
5. $C=C\cup\{u, v\}$;
6. 从E'中删除所有与u或v相连的边;
7. Return C

- 时间复杂性：T(G)=O(|E|)
- Ratio Bound：2

近似比分析

APPROX-Vertex-Cover (G)
1. $C=0$
2. $E'=E[G]$;
3. While $E'\neq\emptyset$ DO
4. 任取$(u, v)\in E'$;
5. $C=C\cup\{u, v\}$;
6. 从E'中删除所有与u或v相连的边;
7. Return C

定理. Approx-Vertex-Cover的Ratio Bound为2.

证明：

令$A=\{(u, v) \mid (u, v)$是算法第4步选中的边$\}$.

若$(u,v)\in A$，则与(u, v)邻接的边皆从E'中删除.

于是, A中无相邻接边.

第5步的每次运行增加两个结点到C, $|C|=2|A|$.

设C^*是优化解, C^*必须覆盖A.

由于A中无邻接边, C^*至少包含A中每条边的一个结点. 于是,

$|A|\leq|C^*|$, $|C|=2|A|\leq 2|C^*|$, 即$|C|/|C^*|\leq 2$.

集合覆盖问题

- 输入：

  有限集X，X的子集合族F，$X=\cup_{S\in F}S$

- 输出：

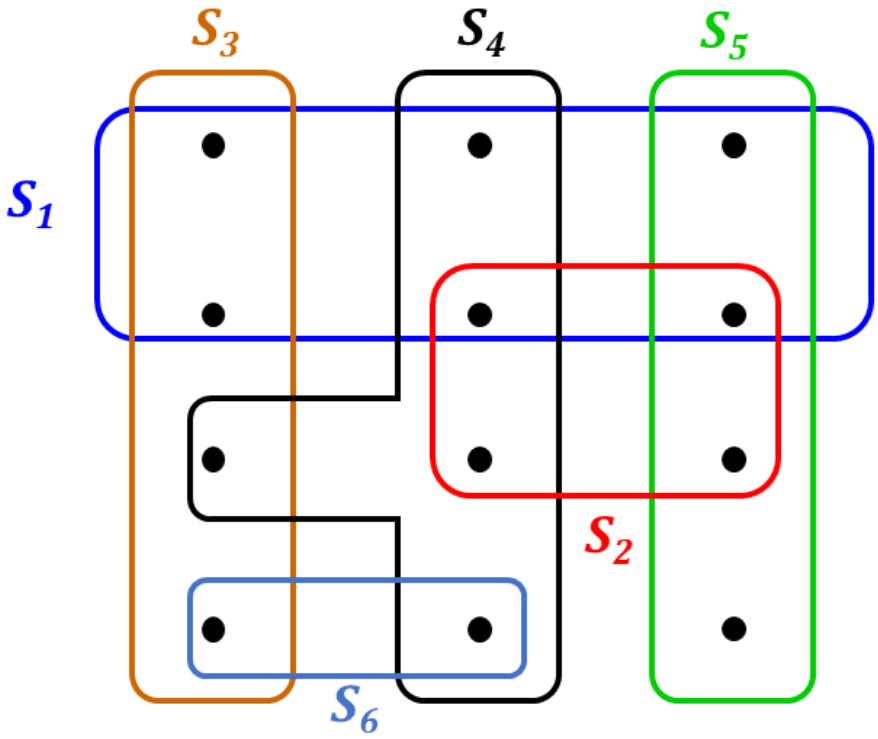
  $C\subseteq F$，满足

  (1) $X=\cup_{S\in C}S$，

  (2) C是满足条件(1)的最小集族，即$|C|$最小.

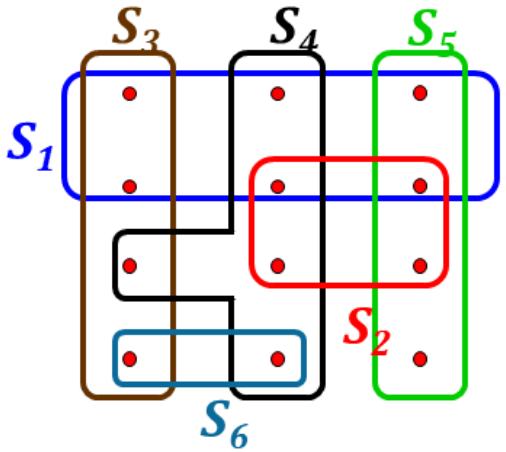
集合覆盖问题实例

- X=12个黑点，F={S_1, S_2, S_3, S_4, S_5, S_6}
- 优化解C={S_3, S_4, S_5}



近似算法的设计

- 基本思想
  - 贪心选择：选择能覆盖最多未被覆盖元素的子集
- $C=\{S_1, S_4, S_5, S_3\}$



Greedy-Set-Cover(X, F)
1. $U \leftarrow X$; /* U是X中尚未被覆盖的元素集 */
2. $C \leftarrow \emptyset$;
3. While $U \neq \emptyset$ and $F \neq \emptyset$ Do
4. Select $S \in F$ 使得$|S \cap U|$最大;
   /* 贪心选择—选择能覆盖最多U元素的子集S */
5. $U \leftarrow U-S$; $F=F-\{S\}$
6. $C \leftarrow C \cup \{S\}$; /* 构造$X$的覆盖 */
7. Return C.

定理1. *Greedy-Set-Covers*的Ratio Bound是$H(d)=\sum_{1 \le i \le d} 1/i$，其中，$d=max\{|S| \mid S \in F\}$.

8.2 随机算法

随机算法的基本概念

- 什么是随机算法
  - 随机算法是一种使用概率和统计方法在其执行过程中对于下一计算步骤作出随机选择的算法
- 随机算法的优越性
  - 对于有些问题:算法简单
  - 对于有些问题:时间复杂性低
  - 对于有些问题:同时兼有简单和时间复杂性低
- 随机算法的随机性
  - 对于同一实例的多次执行，效果可能完全不同
  - 时间复杂性是一个随机变量
  - 解的正确性和准确性（可能）也是随机的

随机算法的分类

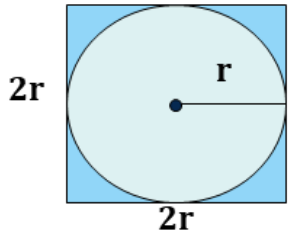
随机数值算法	Monte Carlo算法	Las Vegas算法	Sherwood算法
主要用于数值问题求解	算法可能给出错误解	一旦找到一个解，该解一定是正确的	一定能够求得一个正确解
算法的输出往往是近似解	获得精确解概率与算法执行次数有关	找到解的概率与算法执行时间成正比	确定算法的最坏与平均复杂性差别大时，加入随机性，消除最坏行为与特定实例的联系
近似解的精确度与算法执行时间成正比			

随机算法的性能分析

- 随机算法分析的目标
  - 平均时间复杂性: 时间复杂性随机变量的均值
  - 获得正确解的概率/获得优化解的概率
  - 解的精确度分析

计算π值

- 数学基础
  - 设有一个半径为r的圆及其外切四边形



  - 向正方形随机地投掷n个点，设k个点落入圆内
  - 投掷点落入园内的概率为$(\pi r^2)/(4r^2)=\pi/4$.
  - 用k/n逼近$\pi/4$, 即$k/n \approx \pi/4$, 于是$\pi \approx 4k/n$.
  - 我们可以令r=1用投掷n个点的方法计算π.

计算π值（续）

```
1.  k=0;
2.  For  i=1  To  n
3.      随机地产生四边形中的一点(x, y);
4.      If  x²+y²≤1  Then  k=k+1;
5.  EndFor
6.  Return  (4k)/n
```

- 时间复杂性=O(n)
  - 不是输入的大小, 而是随机样本的大小
- 解的精确度
  - 随着随机样本大小n增加而增加

Max-3-CNF问题定义

- 输入：
  - 合取范式3CNF，
  - 每个析取式具有三个变量，
  - 没有任何变量和它的非出现在同一析取式中
- 输出：
  - 一个变量赋值，最大化值为1的析取式个数

$$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_3 \vee x_4)$$

Clause 1　　Clause 2　　Clause 3

Max-3-CNF——一个随机算法

```
Random-Max-3-CNF(CNF)
1.  For 对于CNF中的每个变量x Do
2.      随机地为x赋值:
            x=0的概率为1/2,
            x=1的概率为1/2;
3. Return.
```

你认为这个算法的效果怎么样？

算法性能分析

定理. Random-Max-3-CNF是一个随机8/7-近似算法.

证明：

假定输入CNF中具有n个变量, m个析取式, 第i个析取式的形式为 $x_{i1} \vee x_{i2} \vee x_{i3}$.

对i=1, 2,..., m, 定义随机变量：

Y_i=1 如果第i个析取式为1, 否则Y_i=0.

$Pr(\text{第i个析取式为0})=Pr(x_{i1}=0)Pr(x_{i2}=0)Pr(x_{i3}=0)=(1/2)^3=1/8$.

$Pr(\text{第i个析取式为1})=1-1/8=7/8$.

$E[Y_i]=7/8$.

令$Y=Y_1+Y_2+\ldots+Y_m$， Y是CNF中值为1的析取式的个数.

$E[Y]=\sum_{1\le i\le m}E[Y_i]=\sum_{1\le i\le m}7/8=m\times 7/8$.

显然, 优化解的代价≤m. 于是近似比≤$m/(m7/8)=8/7$.

8.3 应用分析

应用分析

- 解决难解问题
- 应对不确定的场景
- 提高覆盖率
- 加速求解速度
- ……

小结

小结

8.1 近似算法

- 8.1.1 近似算法的基本概念
- 8.1.2 近似算法的性能分析
- 8.1.3 近似算法举例

8.2 随机算法

- 8.2.1 随机算法的基本概念
- 8.2.2 随机算法的性能分析
- 8.2.3 随机算法举例

8.3 应用分析