HIT CS&E

第九章

Approximation Algorithm

骆吉洲
计算机科学与工程系

HIT CS&E

提要

HIT CS&E

参考资料

《Introduction to Algorithms》
- 第35章

《网站资料》
- 第9章

HIT CS&E

9.1 Introdution

- 近似算法的基本概念
- 近似算法的性能分析

HIT CS&E

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-完全问题的多项式算法是难以得到的
  - 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
  - 什么是近似算法
    - 能够给出一个优化问题的近似优化解的算法
    - 近似算法主要解决优化问题

HIT CS&E

近似算法的性能分析

- 近似算法的时间复杂性
  - 分析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正的代价
    - 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
    - $(1+\varepsilon)$-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

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

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, $\max\{C/C^*, C^*/C\}=C/C^*$
- 由于$C/C^*<1$当且仅当$C^*/C>1$, 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. $\varepsilon(n)\le p(n)-1$.

证. 对于最小化问题

$\varepsilon(n)=|C-C^*|/C^*=(C-C^*)/C^*=C/C^*-1=p(n)-1$.

对于最大化问题

$\varepsilon(n)=|C-C^*|/C^*=(C^*-C)/C^*=(C^*/C-1)/(C^*/C)$

$=(p(n)-1)/p(n)\le p(n)-1$.

- 对于某些问题, $\varepsilon(n)$和$p(n)$独立于n, 用p和ε表示之.
- 某些NP-完全问题的近似算法满足: 当运行时间增加时, Ratio Bound和相对误差将减少.
- 结论1表示, 只要求出了Ratio Bound就求出了$\varepsilon(n)$

- **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. 对于任意固定ε, 近似模式是一个$(1+\varepsilon)$-近似算法.

定义5 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.

定义 6 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.

HIT
CS&E

9.2 基于组合优化的近似算法

- 9.2.1 顶点覆盖问题
- 9.2.2 装箱问题
- 9.2.3 最短并行调度问题
- 9.2.4 TSP问题
- 9.2.5 子集和问题

HIT
CS&E

9.2.1 *The Vetex-cover Problem*

- 问题的定义
- 近似算法的设计
- 算法的性能分析

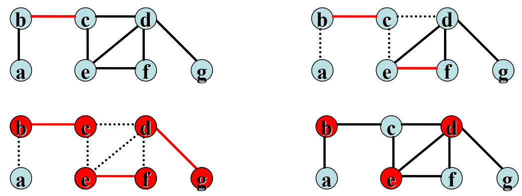
问题的定义

输入： 无向图$G=(V, E)$
输出： $C\subseteq V$，满足
(1). $\forall(u, v)\in E$, $u\in C$或者$v\in C$
(2). C是满足条件(1)的最小集合。

理论上已经证明优化结点覆盖问题是NP-完全问题.

近似算法的设计

- 算法的基本思想



算法解：{b, c, e, f, d, g}

优化解：{b, e, d}

- **算法**

APPROX-Vertex-Cover (G)
1. $C=0$
2. $E'=E[G]$;
3. While $E'\neq 0$ 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. Roturn C

算法的性能分析

- **时间复杂性**
  $T(G)=O(|E|)$
- **Ratio Bound**
  定理. Approx-Vertex-Cover的Ratio Bound为2.
  证. 令A={(u, v) | (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,
  $|A|\leq|C^*|$, $|C|=2|A|\leq2|C^*|$, 即$|C|/|C^*|\leq2$.

9.2.2 Bin-Packing Problem

- 问题的定义
- 近似算法的设计
- 算法的性能分析

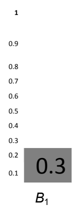
问题的定义

- **输入**
  体积依次为$a_1,\ldots,a_n\in(0,1]$的n个物品
  无穷个体积为1的箱子
- **输出**
  物品的一个装箱方案，使得使用的箱子数量最少

- Bin-Packing是一个著名的NP完全问题.
- 实例：将n种奖券印刷在一些张具有标准尺寸的纸张上，每张奖券是一个物品，纸张是箱子，归一化处理之后变为Bin-Packing问题。

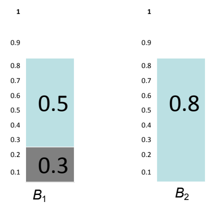
- **算法的基本思想**

  0.3, 0.5, 0.8. 0.2, 0.4



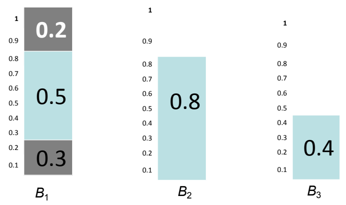
- **算法的基本思想**

  0.3, 0.5, 0.8. 0.2, 0.4



- **算法的基本思想**

  0.3, 0.5, 0.8. 0.2, 0.4



优化解也需要3个箱子

- **算法**

First-Fit (G)

1. $k \leftarrow 0, B_1 \leftarrow \varnothing$
2. For i=1 to n Do
3. 　　从$B_1,\ldots,B_k$中选择能容纳a_i的第一个箱子B_j
4. 　　如果B_j存在，则$B_j \leftarrow B_j \cup \{a_i\}$
5. 　　否则，$k \leftarrow k+1$，$B_k \leftarrow \{a_i\}$
6. 输出$B_1,\ldots,B_k$

时间复杂性$O(n^2)$

- **记号**
  - k^*—最优解所用箱子的个数
  - k—近似解所用箱子的个数
  - $|B_i|$—箱子B_i中物品总体积
  - $|B_i|+|B_j|>1$对任意$i \neq j$成立
- $k^* \geq \sum_{1\leq i\leq k} a_i$
- $|B_1|+\ldots+|B_k|=\sum_{1\leq i\leq k} a_i$
- $k/2<(|B_1|+|B_2|)/2+\ldots+(|B_{k-1}|+|B_k|)/2+(|B_k|+|B_1|)/2$

  $=|B_1|+|B_2|+\ldots+|B_k|$

  $\leq k^*$

定理：First-Fit算法的近似比为2

9.2.3 最短并行调度问题

- 问题的定义
- 近似算法的设计
- 算法的性能分析

HIT CS&E

问题的定义

- 输入
  - 计算时间分别为$t_1,\ldots,t_n$的n个任务
  - m台完全一样的机器
- 输出
  - 计算任务在m台机器上的一个调度策略
  - 使并行时间最短

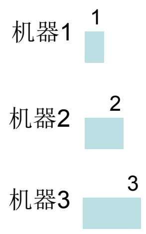
• Bin-Packing是一个著名的NP完全问题.

HIT CS&E

近似算法的设计

- 算法的基本思想

m=3，t_1~t_6=1,2,3,4,5,6

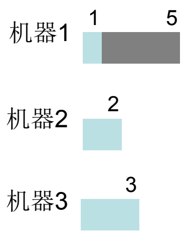


HIT CS&E

近似算法的设计

- 算法的基本思想

m=3，t_1~t_6=1,2,3,4,5,6

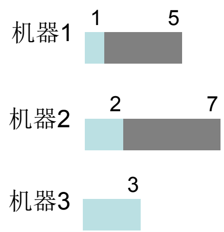


HIT CS&E

近似算法的设计

- 算法的基本思想

m=3，t_1~t_6=1,2,3,4,5,6

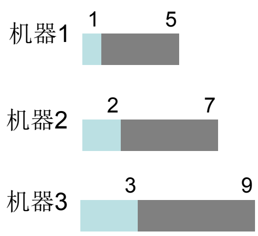


HIT CS&E

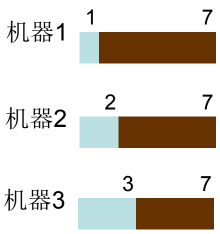
近似算法的设计

- 算法的基本思想

m=3，t_1~t_6=1,2,3,4,5,6



优化解



- 算法

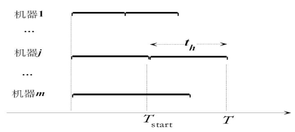
MakeSpanScheduling ()

1. 任意排定所有任务的一个顺序$t_1,\ldots,t_n$
2. For k←1 To m Do
3. T_k←0, M_k←∅
4. For i=1 to n Do
5. 找出j使得$T_j=min_{1\le k\le m}T_k$
6. T_j←T_j+t_i； M_j←M_j∪{i}
7. 输出$M_1,\ldots,M_m$

时间复杂性$O(nm)$

近似比的分析

- 记号
  - T^*—最优解的并行时间　　$T^*\geq(\sum_{1\leq i\leq n} t_i)/m$　　$T^*\geq t_i$
  - T—近似解的并行时间
- 近似解中第j台机器的处理时间最长,最后处理任务t_h
  - 记第h个任务开始执行的时间为T_{start}
  - 则$T=T_{start}+t_h$
  - $T_{start}\leq(\sum_{1\leq i\leq n} t_i)/m$
  - $T_{start}\leq T^*$



- $T=T_{start}+t_h\leq 2T^*$

定理:*MakeSpanScheduling*算法的近似比为2

HIT CS&E

9.2.4 The Traveling-salesman Problem

- 问题的定义
- 近似算法设计
- 算法的性能分析

HIT CS&E

问题的定义

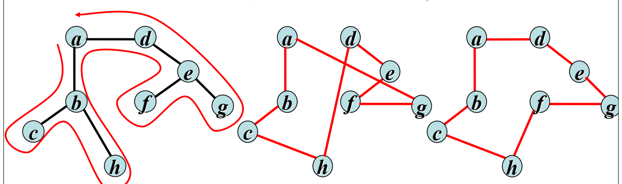
- 输入
  - 完全无向图$G=(V,E)$;
  - 代价函数C: $E\to$非负整数集合
  - C满足三角不等式:
    $C(u,w)\leq C(u,v)+C(v,w)$.
- 输出
  - 具有最小代价的Hamilton环

- Hamilton环是一个包含V中每个结点一次的简单环.
- 代价函数的扩展: 设$A\subseteq E$, $C(A)=\sum_{(u,v)\in E} C(u,v)$.
- 不满足三角不等式的TSP问题无具有常数Ration Bound的近似算法,除非NP=P.

HIT CS&E

近似算法的设计

- 基本思想
  - 首先构造最小生成树(可以使用第五章的算法)
  - 先序遍历最小生成树,构造TSP的解



先序遍历: abchdefga　　先序遍历解　　优化解

HIT CS&E

- 近似算法

  APPROX-TSP-TOUR(G,C)
  1. 选择一个$r\in V[G]$作为生成树的根;
  2. 调用MST-Prim(G, C, r)生成一个最小生成树T;
  3. 先序遍历T,形成有序结点表L;
  4. 按照L中的顺序访问各结点,形成哈密顿环.

HIT CS&E

算法的性能分析

- 时间复杂性
  - 第2步: $O(|E|+|V|\log|V|)=O(|V|^2+|V|\log|V|)=O(|V|^2)$
  - 第3步: $O(|E|)=O(|V|^2)$, 因为G是完全图,
  - 第4步: $O(|V|)$
  - $T(G)=O(|V|^2)$

- **解的精确度**

定理1. APPROX-TSP-TOUR具有Ratio Bound 2.

证.

设H^*是TSP问题的优化解, H是算法产生的近似解.我们需要证明$C(H) \le 2C(H^*)$.

从H^*中删除任意一条边, 可以得到G的一个生成树T'.

设T是算法第2步产生的导致H的最小生成树, 则

$$C(T) \le C(T') \le C(H^*).$$

T的一个full walk W列出了所有结点(第一次访问的和以后从一个子树返回时再访问的). 前面例子的full walk给出顺序: a,b,c,b,h,b,a,d,e,f,e,g,e,d,a

由于W通过每条边两次, $C(W)=2C(T)$, 进而$C(W) \le 2C(H^*)$.

W不是哈密顿环, 因为它通过某些结点多于一次.

根据三角不等式, 我们可以从W中删除对一个结点的任何访问, 而不增加代价. (例如: 从$u \to v \to w$ 删除v得$u \to w$)

反复地应用上述操作, 我们可以从W中删除所有对任何结点的非第一次访问, 得到一个算法中的*preoder walk*.

在我们的例子中, 操作结果是: a, b, c, h, d, e, f, g.

由于T的*preoder walk*导致H, 我们有$C(H) \le C(W)$, 即

$$C(H) \le 2C(H^*),$$

明所欲证.

9.2.5 *The Subset-sum Problem*

- 问题的定义
- 指数时间算法
- 完全多项式时间近似模式

问题定义

- **输入:**

  (S, t), $S=\{x_1, x_2, ..., x_n\}$,

  x_i和t均是正整数

- **输出:**

  $\sum_{x \in A} x$, 满足:

  $A \subseteq S$, $\sum_{x \in A} x \le t$

  $\sum_{x \in A} x = max\{ \sum_{x \in B} x \mid B \subseteq S \}$

指数时间算法

- **算法**

(设S是集合, x是正整数, 定义$S+x=\{s+x \mid s \in S\}$)

Exact-Subset-Sum($S=\{x_1, x_2, ..., x_n\}$, t)

1. $n \leftarrow |S|$;
2. $L_0 \leftarrow <0>$;
3. For $i \leftarrow 1$ To n Do
4. $\quad L_i \leftarrow$Merge-List(L_{i-1}, $L_{i-1}+x_i$);
5. $\quad$删除L_i中所有大于t的元素;
6. Return L_n中最大元素.

- **计算过程:**
  - $L_0=<0>$
  - $L_1=<0, x_1>$ /* 前一个元素所有子集的和(不大于t) */
  - $L_2=<0, x_1, x_2, x_1+x_2>$
    /* 前二个元素所有子集的和(不大于t) */
  - $L_3=<0, x_1, x_2, x_1+x_2, x_3, x_1+x_3, x_2+x_3, x_1+x_2+x_3>$
    /* 前三个元素所有子集的和(不大于t) */
  - L_i=前i个元素所有子集的和(不大于t)

对n作数学归纳法可以证明:
L_n=前n个元素所有子集的和(不大于t)

- 时间复杂性

  第4步：$|L_i|=2|L_{i-1}|=2^2|L_{i-2}|=\ldots=2^i|L_0|=2^i$

  $T(n)=O(2^n)$ 如果t比较大

1. $n\leftarrow|S|$;
2. $L_0\leftarrow<0>$;
3. For $i\leftarrow 1$ To n Do
4. $L_i\leftarrow$Merge-List(L_{i-1}, $L_{i-1}+x_i$);
5. 删除L_i中所有大于t的元素;
6. Return L_n中最大元素.

完全多项式时间近似模式

- 基本思想：

  修剪L，对于多个相近元素，只留一个代表，尽量缩小每个L的长度

  – 设 $\delta(0<\delta<1)$是修剪参数，根据 δ修剪L:

    (1). 从L中删除尽可能多的元素，

    (2). 如果L'是L修剪后的结果，则对每个从L中删除的元素y，L'中存在一个元素$z\leq y$，使得

    $$(1-\delta)y\leq z\leq y$$

  – 如果y被修剪掉，则存在一个代表y的z在L中，而且z相对于y的相对误差小于 δ.

- 修剪算法

```
Trim(L={y1, y2, ..., ym}, δ)  /* yi≤yi+1, 0< δ<1, 输出缩小的表L'. */
m←|L|;
L'←<y1>;
last←y1;
For i←2  To  m  Do
    If  last<(1- δ)yi
    /* 即yi-1<(1- δ)yi, 由L和L'有序, 对∀y∈L', 不满足(1- δ)yi≤y≤yi */
    Then  yi加入到L'末尾;  /* 因L'中目前没有能够表示yi的元素 */
    last←yi;
Return L'.
```

- 复杂性：$O(|L|)=O(m)$

- 完全多项式近似模式

  输入：$S=\{x_1, x_2, \ldots, x_n\}$, $t\geq 0$, $0<\varepsilon<1$

  输出：近似解z

  Approx-Subset-Sum(S, t, ε)

  1. $n\leftarrow|S|$;
  2. $L_0\leftarrow<0>$
  3. For $i\leftarrow 1$ To n Do
  4. $L_i\leftarrow$Merge-List(L_{i-1}, $L_{i-1}+x_i$);
  5. $L_i\leftarrow$Trim(L_i, ε/n) /* 修剪参数 $\delta=\varepsilon/n$ */
  6. 从L_i中删除大于t的元素;
  7. 令z是L_n中最大值;
  8. Return z.

- 性能分析

  定理1. Approx-Subset-Sum是子集求和问题的一个完全多项式时间近似模式.

  证. 令$P_0=\{0\}$, $P_i=\{x \mid x=\sum_{y\in A}y, A\subseteq\{x_1, x_2, \ldots, x_i\}\}$. 例如，

  令$S=\{1, 4, 5\}$, 则

  $P_1=\{0, 1\}$,

  $P_2=\{0, 1, 4, 5\}$,

  $P_3=\{0, 1, 4, 5, 6, 9, 10\}$.

  使用数学归纳法可以证明: $P_i=P_{i-1}\cup(P_{i-1}+x_i)$.

  使用数学归纳法可以证明L_i是P_i中所有不大于t的元素的有序表.

L_i经第5步修剪以及第6步的大于t元素的删除，仍然有$L_i\subseteq P_i$. 于是，第8步返回的z是S的某个子集的和. 我们需证明

(1). $C^*(1-\varepsilon)\leq z$, 即$(C^*-z)/C^*\leq\varepsilon$, C^*是优化解, z是近似解.

注意，由于子集合求和问题是最大化问题，$(C^*-z)/C^*$是算法的相对误差.

(2). 算法是关于$|S|$和$1/\varepsilon$的多项式时间算法.

(1). 往证$C^*(1-\varepsilon)\le z$

对i作归纳法证明: $\forall y\in P_i, y\le t$, 存在一个$z'\in L_i$使$(1-\varepsilon/n)^i y\le z'\le y$.

当$i=0$时$P_i=\{0\}$, $L_i=\{0\}$, 命题成立.

设当$i\le k$时命题成立. $P_{k+1}= P_k\cup\{P_k+x_{k+1}\}$.

由归纳假设, $\forall y\in P_{k+1}\cap P_k, y\le t$, 存在$z'\in L_k\subseteq L_{k+1}$使

$$(1-\varepsilon/n)^k y\le z'\le y.$$

于是, $(1-\varepsilon/n)^{k+1}y\le z'\le y$.

对于$\forall y'\in P_{k+1}-P_k$, $y'=y+x_{k+1}\le t$, $y\in P_k$. 由归纳假设, 存在$z'\in L_k\subseteq L_{k+1}$使$(1-\varepsilon/n)^k y\le z'\le y$. 于是,

$$(1-\varepsilon/n)^k y+x_{k+1}\le z'+x_{k+1}\le y+x_{k+1}.$$

由于$z'\in L_k$, $z'+x_{k+1}\in L_{k+1}$, 而且

$$((1-\varepsilon/n)^k y+x_{k+1})-((1-\varepsilon/n)^{k+1}(y+x_{k+1}))$$
$$=(1-\varepsilon/n)^k(y-(1-\varepsilon/n)y)+(x_{k+1}-(1-\varepsilon/n)^{k+1}x_{k+1})>0,$$

即$(1-\varepsilon/n)^{k+1}(y+x_{k+1})\le(1-\varepsilon/n)^k y+x_{k+1}\le z'+x_{k+1}\le y+x_{k+1}$.

最后, 若$C^*\in P_n$是子集合求和问题的优化解, 则存在一个$z'\in L_n$, 使$(1-\varepsilon/n)^n C^*\le z'\le C^*$. 因算法解$z=\max(L_n)$, $(1-\varepsilon/n)^n C^*\le z'\le z\le C^*$.

由于$(1-\varepsilon/n)^n$的一阶导数大于0, $(1-\varepsilon/n)^n$是关于n递增的函数. 因为$n>1$, $(1-\varepsilon)<(1-\varepsilon/n)^n$.

于是, $(1-\varepsilon)C^*\le z$, 即近似解z与优化解的相对误差不大于ε.

(2). 往证算法的时间复杂性是n与$1/\varepsilon$的多项式

先计算$|L_i|$的上界. 修剪后, L_i中的相邻元素z和z'满足:

$$z'<(1-\varepsilon/n)z, \text{ 即 } z/z'>1/(1-\varepsilon/n).$$

如果L_i中具有$k+2$个元素, 则必有$y_0=0$, $y_1=z_0$, $y_2>z_0\cdot 1/(1-\varepsilon/n)$, $y_3>z_0\cdot 1/(1-\varepsilon/n)^2,\ldots, y_{k+1}>z_0\cdot 1/(1-\varepsilon/n)^k$, 而且$z_0\cdot 1/(1-\varepsilon/n)^k\le t$.

由$z_0\cdot 1/(1-\varepsilon/n)^k\le t$, $k\le\log_{1/(1-\varepsilon/n)}t$, 对$\log_{1/(1-\varepsilon/n)}t$, 台劳展开$\ln(1-\varepsilon/n)$,

$$|L_i|=k+2\le 2+\log_{1/(1-\varepsilon/n)}t=2+(\ln t/-\ln(1-\varepsilon/n))\le 2+n\ln t/\varepsilon.$$

算法的运行时间是$|L_i|$的多项式, 即n和$1/\varepsilon$的多项式.

HIT CS&E

9.3 基于贪心策略的近似算法

- 9.3.1 集合覆盖问题
- 9.3.2 不相交路径问题

HIT CS&E

9.3.1 The Set-covering Problem

- 问题的定义
- 近似算法的设计
- 算法的性能分析

HIT CS&E

问题的定义

- 输入:

  有限集X, X的所有子集族F, $X=\cup_{S\in F}S$

- 输出:

  $C\subseteq F$, 满足

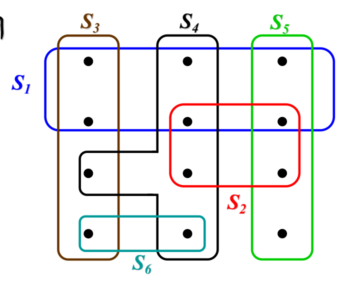
  (1). $X=\cup_{S\in C}S$,

  (2). C是满足条件(1)的最小集族, 即$|C|$最小.

*最小集合覆盖问题是很多实际问题的抽象.

*最小集合覆盖问题是NP-完全问题.

•问题的实例

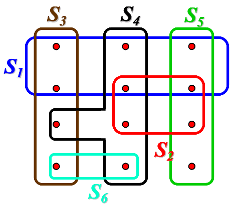


X=12个黑点, $F=\{S_1, S_2, S_3, S_4, S_5, S_6\}$

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



$C=\{S_1, S_4, S_5, S_3\}$

- **算法**

Greedy-Set-Cover(X, F)

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

算法性能的分析

- **时间复杂性**
  - 3-6的循环次数至多为$min(|X|, |F|)$
  - 计算$|S \cap U|$需要时间$O(|X|)$
  - 第4步需要时间$O(|F||X|)$
  - $T(X,F)=O(|F||X|min(|x|,|F|))$

- **Ration Bound**

定理1. 令$H(d)=\sum_{1\leq i\leq d} 1/I$. Greedy-Set-Covers是多项式$p(n)$-近似算法, $p(n)=H(max\{|S| \mid S\in F\})$.

证. 我们已经算法是多项式算法,仅需计算Ratio Bound.
设C^*是优化集合覆盖, C^*的代价是$|C^*|$.
令S_i是由Greedy-Set-Cover选中的第i个子集.
当把S_i加入C时, C的代价加1. 我们把选择S_i增加的代价均匀分配到由S_i首次覆盖的所有结点.
$\forall x\in X$, 令c_x是分配到x的代价. 若x被S_i首次覆盖, 则

$$c_x = \frac{1}{|S_i - (S_1 \cup S_2 \cup \ldots \cup S_{i-1})|}$$

显然, 算法给出的解C的代价为$|C|$, $|C|$平均地分布到X的所有点. 由于C^*也覆盖X, 我们有

$$|C| = \sum_{x\in X} c_x \leq \sum_{S\in C^*}\sum_{x\in S} c_x$$

注意: 上式的小于成立是因为C^*中各子集可能相交, 某些c_x被加了多次, 而左式每个c_x只加一次.

如果$\forall S\in F$, $\sum_{x\in S} c_x \leq H(|S|)$成立, 则

$$|C| \leq \sum_{S\in C^*} H(|S|) \leq |C^*| \cdot H(max\{|S| \mid S\in F\}),$$

即$|C|/|C^*| \leq H(max\{|S| \mid S\in F\})$, 定理成立.

下边我们来证明: 对于$\forall S\in F$, $\sum_{x\in S} c_x \leq H(|S|)$.

对于$\forall S\in F$和$i=1,2,\ldots|C|$, 令$u_i=|S-(S_1 \cup S_2 \cup \ldots \cup S_i)|$是$S_1$、$S_2$、…、$S_i$被选中后, S中未被覆盖的点数. S_i先于S被选中.

令$u_0=|S|$, k是满足下列条件的最小数: $u_k=0$, 即S中每个元素被S_1、S_2、…、S_k中至少一个覆盖.

显然, $u_{i-1}\geq u_i$, $u_{i-1}-u_i$是S中由S_i第一次覆盖的元素数. 于是,

$$\sum_{x\in S} c_x = \sum_{i=1}^{k}(u_{i-1}-u_i)\bullet\frac{1}{|S_i-(S_1\cup S_2\cup\ldots\cup S_{i-1}|}$$

注意: $|S_i-(S_1 \cup S_2 \cup \ldots \cup S_{i-1})| \geq |S-(S_1 \cup S_2 \cup \ldots \cup S_{i-1}|=u_{i-1}$, 因为Greed算法保证: S不能覆盖多于S_i覆盖的新结点数, 否则S将在S_i之前被选中. 于是,

$$\sum_{x\in S} c_x \leq \sum_{i=1}^{k}(u_{i-1}-u_i)\bullet\frac{1}{u_{i-1}}$$

HIT
CS&E

$$\sum_{x\in S} c_x \le \sum_{i=1}^{k} (u_{i-1}-u_i)\bullet \frac{1}{u_{i-1}}$$
$$= \sum_{i=1}^{k} \sum_{j=u_i+1}^{u_{i-1}} \frac{1}{u_{i-1}}$$
$$\le \sum_{i=1}^{k} \sum_{j=u_i+1}^{u_{i-1}} \frac{1}{j} \qquad (\because j \le u_{i-1})$$
$$= \sum_{i=1}^{k} \left(\sum_{j=1}^{u_{i-1}} \frac{1}{j} - \sum_{j=1}^{u_i} \frac{1}{j} \right)$$
$$= \sum_{i=1}^{k} H(u_{i-1}) - H(u_i)$$
$$= H(u_0) - H(u_k)$$
$$= H(u_0) - H(0) \qquad (\because u_k = 0)$$
$$= H(u_0) = H(|S|) \qquad (\because H(0)=0,\ u_0 = S)$$

HIT
CS&E

复杂性分析

推论1. Greedy-Set-Cover是一个多项式ln(|x|+1)-近似算法.

证. 由不等式$H(n)\le \ln(n+1)$可知
$H(max\{|S| \mid S\in F\})\le H(|X|)\le \ln|X|+1$.

HIT
CS&E

9.3.2 Path-disjoint Problem

- 问题的定义
- 近似算法的设计
- 算法的性能分析

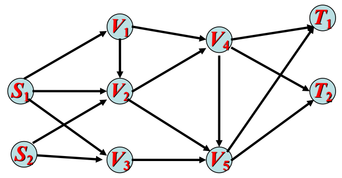
HIT
CS&E

问题的定义

- 输入:
  图$G=(V,E)$,源顶点集S,汇顶点集T
- 输出:
  $A\subseteq S\times T$
  (1). A中的所有顶点对在G中存在无公共边的路径
  (2). $|A|$最大.

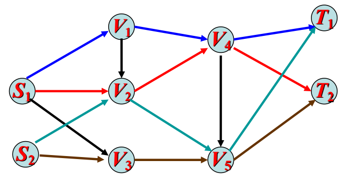
*不相交问题是很多实际问题的抽象.
*不相交问题是NP-完全问题.

•问题的实例



图G
$S=\{S_1, S_2\}$ $T=\{T_1, T_2\}$

•问题的实例

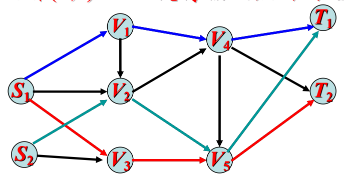


图G
$S=\{S_1, S_2\}$ $T=\{T_1, T_2\}$
优化解$A=\{(S_1,T_1),(S_1,T_2),(S_2,T_1),(S_2,T_2)\}$

HIT CS&E

近似算法的设计

- 基本思想
  - 贪心选择:选择$(u,v)\in S\times T$使得该顶点对间路径最短



$A=\{(S_1,T_1), (S_1,T_2), (S_2,T_1)\}$

精确解$A^*=\{(S_1,T_1),(S_1,T_2),(S_2,T_1),(S_2,T_2)\}$

- 算法

EdgeDisjointPath(G,S,T)

1. $A\leftarrow\varnothing$; $B\leftarrow S\times T$
2. While true Do
3. 计算B中所有顶点对在G中的最短路径构成P;
4. IF $P=\varnothing$ Then break;
5. 选择P中长度最短的路径$P_{u,v}$ /*贪心选择*/
6. $A\leftarrow A\cup\{(u,v)\}$; $G\leftarrow G-P_{u,v}$; $B\leftarrow B-\{(u,v)\}$;
   /*根据贪心选择更新A、图G和B*/
7. 输出A.

时间复杂度 $O(|S||T||V|^4)$

第3步的开销$O(|V|^4)$,第5-6步开销为$O(|E|)$

- 解的精确度

定理. 算法EdgeDisjointPath的近似比为$O(m^{1/2})$,其中$m=|E|$

证.

A^*—问题精确

A—近似算法输出的近似解

k—参数,任意固定的值

证明思想:

用参数k将A^*划分为两个部分S^*,L^*使得$A^*=S^*\cup L^*$

S^*—A^*中长度小于等于k的路径(短路径), $|S^*|\le k|A|$(引理2)

L^*—A^*中长度大于k的路径(长路径),$|L^*|\le(m/k)|A|$(引理1)

$|A^*|=|S^*|+|L^*|\le(k+m/k)|A|$ (对任意k成立)

取$k=m^{1/2}$时得到 $|A^*|\le 2m^{1/2}|A|$

HIT CS&E

引理1. $|L^*|\le(m/k)|A|$对任意k成立。

证. A^*—精确解

L^*—A^*中长度大于k的路径

A—近似解, $1\le|A|$

- L^*中任意两条路径的无公共边
- L^*中的所有路径至少用到$k|L^*|$条边
- $k|L^*|\le|E|=m$
- $|L^*|\le(m/k)|A|$

引理2. $|S^*|\le k|A|$对任意k成立。

证. A^*—精确解

S^*—A^*中长度$\le k$的路径

A—近似解

- 任意$p^*\in A^*$必然与A中某条路径相交(有公共边)
  - 否则,近似算法终止时,p^*仍然存在于图G中,与终止条件矛盾
- 任意$p^*\in S^*\subseteq A^*$
  - p^*中至多有k条边
  - p^*至少与A中某一条路径相交(A是算法操作逐渐添加路径得到的)
  - 记近似算法得到A时,第一条与p^*相交的路径为p
  - 算法选择p加入A而未选择p^*,说明p比p^*更短,$|p|<k$
  - A^*(继而S^*)中与p有公共边的路径至多有k条
- $|S^*|\le k\times|\{p\in A$: p与某条S^*中某条短路径相交$\}|\le k|A|$

HIT CS&E

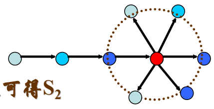
9.4 基于局部搜索的近似算法

- 9.4.1 局部搜索原理
- 9.4.2 最大割问题
- 9.4.3 设施定位问题

HIT CS&E

9.4.1局部搜索原理

- 计算问题有很多可行解
- 从任意可行解出发,进行局部修改，产生其他可行解
- 达到一个局部优化解(比相邻可行解代价更优)
- 输出局部优化解作为近似解
- 可行解有向图
  - 每个顶点表示一个可行解
  - S_1到S_2之间存在边$\Leftrightarrow S_1$由局部修改可得S_2



- 局部搜索适用条件
  - 顶点的度较小(多项式)，确保多项式时间达到“相邻”解
  - 从任意可行解开始，多项式时间内必然能找到局部优化解

HIT CS&E

9.4.2 最大割问题

- 问题定义
- 局部搜索算法
- 时间复杂度分析
- 近似比分析

HIT CS&E

问题的定义

- 输入:

  加权图$G=(V,E)$，权值函数$W:E\rightarrow N$

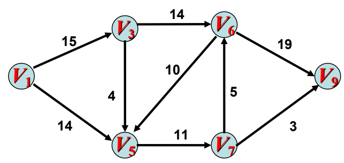
- 输出:

  $S\subseteq V$ 使得$\sum_{u\in S, v\in V-S}W(uv)$最大

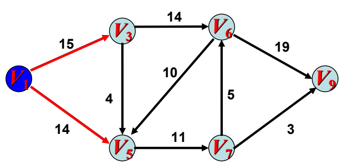
*最小割问题存在多项式时间算法.

*最大割问题是NP-完全问题.

•问题的实例

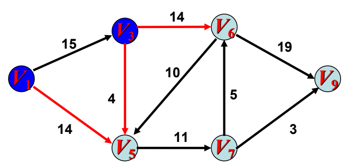


HIT CS&E



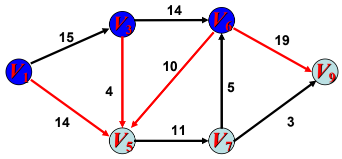
$S_0=\{V_1\}$ 代价29 交换V_3在S和V-S中的位置

HIT CS&E



$S_0=\{V_1\}$ 代价29 交换V_3在S和V-S中的位置

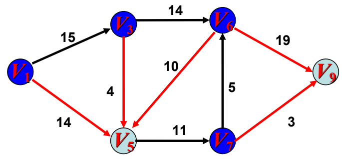
$S_1=\{V_1,V_3\}$ 代价32 交换V_6在S和V-S中的位置



$S_0=\{V_1\}$ 代价29 交换V_3在S和V-S中的位置

$S_1=\{V_1,V_3\}$ 代价32 交换V_6在S和V-S中的位置

$S_1=\{V_1,V_3,V_6\}$ 代价57 交换V_7在S和V-S中的位置



$S_0=\{V_1\}$ 代价29 交换V_3在S和V-S中的位置

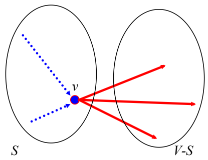
$S_1=\{V_1,V_3\}$ 代价32 交换V_6在S和V-S中的位置

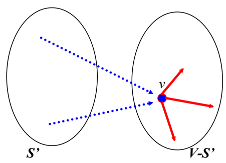
$S_1=\{V_1,V_3,V_6,V_7\}$代价60 局部优化解

全局优化解

算法思想:

- 局部修改操作—交换v在S和V-S中的位置
- 只有与v关联的部分边会影响割的代价





- $cost(v,S)=\sum_{u\in S:uv\in E}W(uv)-\sum_{u\in V\text{-}S:vu\in E}W(vu)$ $\quad v\in S$
- $cost(v,S)=\sum_{u\in V\text{-}S:vu\in E}W(vu)-\sum_{u\in S:uv\in E}W(uv)$ $\quad v\in V\text{-}S$

- **算法**

ApproxMaxCut($G(V,E)$)

输入: 加权图$G=(V,E)$，权值函数$W:E\to N$

输出: $S\subseteq V$ 使得$\sum_{u\in S,v\in V\text{-}S}W(uv)$最大

1. $S\leftarrow\{u\}$; /*u是V中任意顶点*/
2. repeat
3. 任取使得cost(v,S)>0的顶点v,交换v在S,V-S中的位置;
4. until 不存在$v\in V$使得cost(v,S)>0
5. 输出S.

- **算法复杂度**
  - 第3步每次运行的时间开销为$O(|V|^2)$
    - ✓ 计算cost(v,S)的开销为$|V|$,至多为$|V|$个顶点计算
  - 第2-4步至多运行$\sum_{uv\in E}W(uv)\le|V|^2\times max_{uv\in E}W(uv)=O(|V|^2W)$
    - ✓ 循环每做一次，割的代价至少增大1
  - 总的时间开销为$O(|V|^4W)$

近似比分析

- **定理**：ApproxMaxCut的近似比为2。

证明：由于算法输出局部最优解，

- $\forall v\in S,\ cost(v,S)\le 0$
  - ✓ $\sum_{u\in S}W(uv)\le\sum_{u\in V\text{-}S}W(uv)$
  - ✓ $\sum_{u\in S}W(uv)+\sum_{u\in V\text{-}S}W(uv)\le 2\times\sum_{u\in V\text{-}S}W(uv)$
  - ✓ $(1/2)\times\sum_{u\in V}W(uv)\le\sum_{u\in V\text{-}S}W(uv)$
- $\forall v\in V\text{-}S,\ cost(v,S)\le 0$
  - ✓ $(1/2)\times\sum_{u\in V}W(uv)\le\sum_{u\in S}W(uv)$
- $\sum_{uv\in E}W(uv)\le 2W(S)$
- $W(S^*)\le\sum_{uv\in E}W(uv)\le 2W(S)$

9.4.3 设施定位问题（简介）

HIT CS&E

问题的定义

- 输入:

  设施集合F,用户集合U, 距离函数d:U×F→R^+
  - 距离函数满足三角不等式;
  - $\forall i\in F$,开启设施i的代价为f_i, $\forall S\subseteq F$的开启代价$C_f(S)=\sum_{i\in S}f_i$
  - $\forall j\in U$, $\forall S\subseteq F$,
    S向j提供服务的代价为$r(j,S)=\min_{i\in S}d(j,i)$
    S向U提供服务的总代价为$C_r(S)=\sum_{j\in U}r(j,S)$
  - $\forall S\subseteq F$, S的代价定义为C(S)=$C_f(S)+C_r(S)$
- 输出:
  $S\subseteq F$ 使得$C(S)$最小

*设施定位问题是很多实际问题的抽像.
*设施定位问题是NP-完全问题.

- 算法思想:局部修改操作
- 对任意可行解S可以施行如下三类局部操作
  - 添加——向S添加一个设施
  - 删除——从S中删除一个设施
  - 替换——将S中的一个设施i替换为另一个设施i'
- 算法

  LocalSearchFacility(F,U,ε)
  1. $S\leftarrow F$的任意子集;
  2. IF 存在添加、删除或替换操作使S的代价下降因子(1- ε/n^2) Then
     执行该操作
  3. 重复第2步,直到不存在满足条件的操作
  4. 输出S.

•该算法在多项式时间内终止,且近似比为3+$o(\varepsilon)$(参见讲义)

HIT CS&E

9.5 基于动态规划的近似算法

- 9.5.1 用动态规划与近似算法
- 9.5.2 0-1背包问题的完全多项式近似模式
- 9.5.3 Bin-Packing 问题的近似模式

HIT CS&E

9.4.1动态规划与近似算法

- 动态规划算法
  - 问题具有优化子结构
  - 重叠子问题
  - 用系统化的方法搜索优化子结构涉及的所有子问题
- 近似策略1
  - 原问题的实例I转换为一个特殊实例I'
  - 用动态规划方法求解实例I'
  - 将I'的解转化为I的近似解
  - 近似比取决于变形过程的性质
- 近似策略2
  - 将动态规划方法视为解空间的枚举过程
  - 仅枚举整个解空间的一个子空间,则得到一个近似解
  - 近似比取决于所枚举的子空间与整个空间之间的"间隙"大小

HIT CS&E

9.5.2 0-1背包问题的完全多项式近似模式

- 问题定义及其动态规划算法
- 问题的变形
- 完全多项式近似模式
- 近似比分析

HIT CS&E

问题定义

给定n种物品和一个背包,物品i的重量是w_i,价值v_i,背包承重为C,问如何选择装入背包的物品,使装入背包中的物品的总价值最大?

对于每种物品只能选择完全装入或不装入,一个物品至多装入一次。

- 输入:$C>0, w_i>0, v_i>0, 1\le i\le n$
- 输出:$(x_1, x_2, \ldots, x_n)$, $x_i\in\{0, 1\}$, 满足

  $\sum_{1\le i\le n}w_ix_i\le C$, $\sum_{1\le i\le n}v_ix_i$ 最大

0-1背包问题是NP完全问题

HIT CS&E

第4章曾介绍了0-1背包问题的动态规划算法

- 将第i, i+1,...,n个物品装入背包中
- $b_{i,j}$表示欲获得总价值量j所需的最小背包容量
- 其中$1 \le j \le \sum_{1\le i\le n} v_i$
- 问题的优化子结构可以重述为：

$$b_{i,j}=b_{i+1,j} \quad j<v_i$$
$$b_{i,j}=\min(b_{i+1,j}, b_{i+1,j}+w_i) \quad j\ge v_i$$
$$b_{n,j}=w_n \quad j\ge v_n$$
$$b_{n,j}=0 \quad j<v_n$$

- 用相同计算过程可得到$O(n\sum_{1\le i\le n} v_i)$时间的DP算法
- 将该算法称为BoolPacking算法

HIT CS&E

问题定义的实例的变形

•给定参数ε,令$K=\varepsilon/n$, $v_{max}=max_{1\le i\le n}v_i$

•实例$I=\langle w_1,\ldots,w_n,v_1,\ldots,v_n\rangle$变形为$I'=\langle w_1,\ldots,w_n,v'_1,\ldots,v'_n\rangle$

• $v'_i=\lfloor v_i\times(K/v_{max})\rfloor$

•$\sum_{1\le i\le n}v'_i= \sum_{1\le i\le n}\lfloor v_i\times(K/v_{max})\rfloor \le \sum_{1\le i\le n}K\times(v_i/v_{max}) \;\; =n^2/\varepsilon$

•BoolPacking算法在I'上仅需运行多项式时间

HIT CS&E

完全多项式近似模式

算法ApproxPacking(W[1:n],V[1:n],C,ε)

输入：容量C,重量数组W[1:n],价值数组V[1:n],误差参数ε

输出：0-1背包问题的1+ε-近似解$\langle x_1,\ldots,x_n\rangle$

1. $K=n/\varepsilon$;
2. $v_{max}\leftarrow max_{1\le i\le n}v_i$;
3. $V[i]\leftarrow\lfloor K\times(V[i]/v_{max})\rfloor$; /*$i$=1,2,…,$n$*/
4. $\langle x_1,\ldots,x_n\rangle$←BoolPacking($W$[1:$n$],$V$[1:$n$],$C$) ; /*DP*/
5. 输出$\langle x_1,\ldots,x_n\rangle$

时间复杂度 $O(n^2/\varepsilon^2)$

主要取决于定4步的开销

定理：算法ApproxPacking是一个1+ε-近似算法

证：$\langle z_1,\ldots,z_n\rangle$—优化解，$Z=\sum_{1\le i\le n}v_iz_i$—优化解代价,$Z'=\sum_{1\le i\le n}v'_iz_i$

$\langle x_1,\ldots,x_n\rangle$—近似解 ,$X=\sum_{1\le i\le n}v_ix_i$—近似解代价

$\langle x_1,\ldots,x_n\rangle$是$I'$的优化解, $X'=\sum_{1\le i\le n}v'_ix_i$是$I'$最优代价

- $\langle z_1,\ldots,z_n\rangle$是$I'$的近似解 $\Rightarrow Z'\le X'$
- $v'_i=\lfloor v_i\times(K/v_{max})\rfloor \le (K/v_{max})v_i \Rightarrow X'\le (K/v_{max})X$

$X\ge(v_{max}/K)X' \ge(v_{max}/K)Z'$ (联立两个不等式)

$=(v_{max}/K)\sum_{1\le i\le n}v'_iz_i$ (Z'的定义)

$=(v_{max}/K)\sum_{1\le i\le n}\lfloor v_i\times(K/v_{max})\rfloor z_i$

$\ge(v_{max}/K)\sum_{1\le i\le n}[v_i\times(K/v_{max})-1]z_i$

$=\sum_{1\le i\le n}v_iz_i-(v_{max}/K)\sum_{1\le i\le n}z_i$

$=Z-\varepsilon\cdot v_{max}$

$\ge Z(1-\varepsilon)$

HIT CS&E

9.5.3 Bin-Packing问题的近似模式

- 问题定义
- 问题的变形
- 多项式近似模式
- 近似比分析

HIT CS&E

问题的定义

- 输入
  体积依次为$a_1,\ldots,a_n\in(0,1]$的n个物品
  无穷个体积为1的箱子
- 输出
  物品的一个装箱方案，使得使用的箱子数量最少

HIT CS&E

基本想法—给定实例I,ε

- 小体积物品对优化解的影响较小
  - 多个小体积物品的可以容纳在少数箱子中
- 可以先忽略小体积物品得到实例I^{up}
  - 缩小问题的解空间
  - 实例I^{up}的优化解具有某种优良的性质
- 在I^{up}上用动态规划算法得到精确解s'
- 将s'调整为I的近似解s

HIT CS&E

算法框架

算法ApproxBinPacking(I,ε)

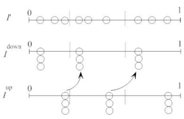
输入:装箱问题的实例I和相对误差参数ε<1

输出:I的一个近似最优的装箱方案;

- $I', I^{down}, I^{up} \leftarrow$ Transfrom(I,ε); /*变换*/
- $S' \leftarrow$ DynamicSearch($I^{up}, 1/\varepsilon^2$); /*DP*/
- $S \leftarrow$ SolutionTrans(S', I, ε); /*得到近似解*/
- 输出S;

下面依次介绍第1,2,3步,实现复杂度为$O(n^{2/\varepsilon^2})$的算法

实例变化算法Transform(I,ε)



输入:装箱问题的实例I和变换参数ε

输出:变形后的三个实例I', I^{down}和I^{up}

1. 删除I中所有体积小于ε的物品,得到I',记$n=|I'|$;
2. 将I'中所有物品按体积大小递增排序,划分为$K=1/\varepsilon^2$组,每组$n\varepsilon^2$个物品;
3. 将各组内物品的体积修改为组内最大体积,得I^{up};
4. 将各组内物品的体积修改为组内最小体积,得I^{down};
5. 输出I', I^{down}和I^{up};

Transform(I,ε)的时间复杂度为$O(n\log n)$

Transform(I,ε)算法的性质

引理1:I', I^{down}和I^{up}满足下列性质

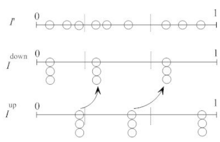
(1)每个箱子至多容纳I', I^{down}和I^{up}的$L=\lceil 1/\varepsilon \rceil$个物品;

(2)I^{down}和I^{up}中物品体积至多有$K=1/\varepsilon^2$个不同的取值;

(3)$Opt(I^{down}) \leq Opt(I')$,且$n\varepsilon \leq Opt(I')$,其中$n$是$I'$中物品个数;

证明:

- $Opt(I^{down}) \leq Opt(I')$
  - I'的每个可行解也是I^{down}的可行解
- $n\varepsilon \leq Opt(I')$
  - 所有物品的总体积至少为$n\varepsilon$



Transform(I,ε)算法的性质

引理1:I', I^{down}和I^{up}满足下列性质

(1)每个箱子至多容纳I', I^{down}和I^{up}的$L=\lceil 1/\varepsilon \rceil$个物品;

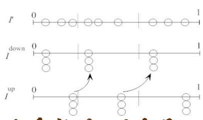
(2)I^{down}和I^{up}中物品体积至多有$K=1/\varepsilon^2$个不同的取值;

(3)$Opt(I^{down}) \leq Opt(I')$,且$n\varepsilon \leq Opt(I')$,其中$n$是$I'$中物品个数

(4)$Opt(I^{up}) \leq (1+\varepsilon)Opt(I')$;

证明:I^{down}的优化解S'可修改得到I^{up}的可行解S

- 所有箱子去除I^{down}第1组的物品
- 所有箱子属于I^{down}第i组的物品替换为I^{up}第i-1组的物品



- I^{up}最后一组每个物品用一个新箱子,至多新增$n\varepsilon^2$箱子

$Opt(I^{up}) \leq Opt(I') + n\varepsilon^2 \leq Opt(I') + \varepsilon Opt(I') = (1+\varepsilon)Opt(I')$

HIT CS&E

- 用动态规划算法求解问题实例I^{up}
  - n个物品,
  - 体积至多有$K=1/\varepsilon^2$中不同取值的取值$s_1,\ldots,s_K$
  - 实例可以描述为K元组$\langle n_1,\ldots,n_K \rangle$
  - 请同学们自己描述问题的优化子结构并实现算法DynamicSearch($I^{up}, 1/\varepsilon^2$),要求时间复杂度为$O(Kn^K)$

解转换算法SolutionTrans(S,I,ε)

输入:实例I，I中体积大于ε的物品的近似装箱方案S
输出:I的一个近似解

1. For I中体积小于ε的每个物品i Do
2. If S中存在箱子能容纳物品i Then 将i装入该箱子
3. Else 开启新箱子将i装入，将更新后的方案仍记为S;
4. 输出更新后的装箱方案S;

SolutionTrans(I,ε)的时间复杂度为$O(n^2)$

近似比分析 $I \xrightarrow{\varepsilon} I' \xrightarrow{\text{缩放}} I^{up}$

- I^{up}的优化解作为I'的近似解S $Opt(I^{up}) \le (1+\varepsilon)\, Opt(I')$
  - S中使用的箱子个数即为Opt(I^{up})
- SolutionTrans新开的箱子个数记为new;
  - Approx(I) = $Opt(I^{up})+new$
- 若new=0, 则
  - Approx(I) = $Opt(I^{up})+new \le (1+\varepsilon)\, Opt(I') \le (1+2\varepsilon) Opt(I)$
- 若$new \ne 0$,则
  - 近似解中，除最后一个箱子之外，每个箱子的空闲空间都小于ε
  - 这些箱子内物品总体积>(1-ε)[Approx(I) -1],但<所有物品总体积
  - Opt(I)≥所有物品总体积
  - (1-ε)[Approx(I) -1] ≤ Opt(I)
  - Approx(I) ≤ [1/(1-ε)]Opt +1 ≤ (1+2ε)$Opt(I)$

定理：ApproxBinPacking是一个1+2ε-近似算法。

9.6 基于线性规划的近似算法

9.6.1线性规划

线性规划问题：在约束条件为线性表达式的前提下对一个线性目标函数进行优化

例1 minimize $7x_1+x_2+5x_3$

subject to $x_1-x_2+3x_3 \ge 10$

$5x_1+2x_2-x_3 \ge 6$

$x_1,x_2,x_3 \ge 0$

maximize $10y_1+6y_2$

subject to $y_1+5y_2 \le 7$

$-y_1+2y_2 \le 1$

$3y_1-y_2 \le 5$

$-y_1,-y_2 \le 0$

线性规划的一般形式

最小化问题

min cx

st. $Ax \ge b$

最大化问题

max by

st. $By \le c$

满足所有约束条件的一组变量称为线性规划问题的**可行解**

使得目标函数达到最优取值的可行解称为线性规划问题的**最优解**

minimize $7x_1+x_2+5x_3$

subject to $x_1-x_2+3x_3 \ge 10$

$5x_1+2x_2-x_3 \ge 6$

$x_1,x_2,x_3 \ge 0$

x=(2,1,3)是上述线性规划问题的一个可行解;x=(7/4,0,11/4)是上述线性规划问题的最优解，目标函数的最优值为26.

线性规划问题可以在多项式时间内求解：Karmarkar算法

线性规划问题的对偶问题

minimize $7x_1+x_2+5x_3$

subject to $x_1-x_2+3x_3 \ge 10$

$5x_1+2x_2-x_3 \ge 6$

$x_1,x_2,x_3 \ge 0$

maximize $10y_1+6y_2$

subject to $y_1+5y_2 \le 7$

$-y_1+2y_2 \le 1$

$3y_1-y_2 \le 5$

$-y_1,-y_2 \le 0$

对偶问题

$7x_1+x_2+5x_3 \ge x_1-x_2+3x_3 \ge 10$

$7x_1+x_2+5x_3 \ge 5x_1+2x_2-x_3 \ge 6$

$7x_1+x_2+5x_3 \ge y_1(x_1-x_2+3x_3) + y_2(5x_1+2x_2-x_3) \ge 10y_1+6y_2$

$(y_1+5y_2)x_1 + (-y_1+2y_2)x_2 + (3y_1-y_2)x_3$

对偶问题 26 原问题

对于一般的线性规划问题

$$\text{Min}\ \ cx$$

$$\text{St}\quad Ax \ge b \qquad \text{原问题}$$

$$x \ge 0$$

其对偶问题为

$$\text{Max}\ \ b^T y$$

$$\text{St}\quad A^T y \le c^T \qquad \text{对偶问题}$$

$$y \ge 0$$

对偶定理

定理1. 在线性规划问题中，原问题的最优值有限当且仅当对偶问题的最优值有限。并且，如果 $x^*=(x_1^*,\dots,x_n^*)$ 和 $y^*=(y_1^*,\dots,y_m^*)$ 分别是原问题和对偶问题的最优解，则 $cx^*=b^T y^*$。

定理2. 在线性规划问题中，如果 $x=(x_1,\dots,x_n)$ 和 $y=(y_1,\dots,y_m)$ 分别是原问题和对偶问题的可行解，则 $cx \ge by$。

证明：

由于 y 是对偶问题的可行解，且 x_j 非负

$$\sum_{j=1}^{n} c_j x_j \ge \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i\right) x_j$$

由于 x 是对偶问题的可行解，且 y_i 非负

$$\sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j\right) y_i \ge \sum_{i=1}^{m} b_i y_i$$

注意到

$$\sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i\right) x_j = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j\right) y_i$$

证毕

定理3. 如果 $x=(x_1,\dots,x_n)$ 和 $y=(y_1,\dots,y_m)$ 分别是原问题和对偶问题的可行解，则 x 和 y 分别是原问题和对偶问题的最优解当且仅当下面的条件同时成立：

原问题的补松弛条件：

$$\text{对于} 1 \le j \le n\text{：}\ x_j = 0 \text{或者} \sum_{i=1}^{m} a_{ij} y_i = c_j$$

对偶问题的补松弛条件：

$$\text{对于} 1 \le i \le m\text{：}\ y_i = 0 \text{或者} \sum_{j=1}^{n} a_{ij} x_j = b_i$$

定理4. 如果 $x=(x_1,\dots,x_n)$ 和 $y=(y_1,\dots,y_m)$ 分别是原问题和对偶问题的可行解，且满足

原问题的补松弛条件：$\alpha \ge 1$

$$\text{对于}\ 1 \le j \le n\text{：}\ x_j = 0 \text{或者} \frac{c_j}{\alpha} \le \sum_{i=1}^{m} a_{ij} y_i \le c_j$$

对偶问题的补松弛条件：$\beta \ge 1$

$$\text{对于}\ 1 \le i \le m\text{：}\ y_i = 0 \text{或者}\ b_i \le \sum_{j=1}^{n} a_{ij} x_j \le \beta \cdot b_i$$

则

$$\sum_{j=1}^{n} c_j x_j \le \alpha \cdot \beta \cdot \sum_{i=1}^{m} b_i y_i$$

证明：$\sum_{j=1}^{n} c_j x_j \le \alpha \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i\right) x_j = \alpha \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j\right) y_i \le \alpha \cdot \beta \cdot \sum_{i=1}^{m} b_i y_i$

基于Primal-dual schema的近似算法以定理4为理论基础

9.6.2 Min-max关系

最大流问题

输入：有向图 $G=(V,E)$，源顶点 $s \in V$，接收顶点 $t \in V$，每条边 e 的流量限制 $c(e)>0$.

输出：从 s 到 t 的最大流。

即，对每条边 e 赋值 $f(e)$ 使得 $\sum_{ut \in E} f(ut)$ 最大且满足

流量约束：$f(e)<c(e)$

守恒约束：$\sum_{uv \in E} f(uv) = \sum_{uv' \in E} f(uv')$ $\quad u \in V\ \ u \ne s,\ u \ne t$

HIT CS&E

最小割问题

输入：有向图$G=(V,E)$，源顶点$s\in V$，接收顶点$t\in V$，每条边e的流量限制$c(e)>0$.

输出：s和t之间的最小割。

即，$s\in X\subseteq V$，$t\in V-X$使得$\sum_{u\in X,v\in V-X}c(uv)$最小

最小割问题与最大流问题间的min-max关系

任意s,t-割的容量给出了任意s,t-可行流的流量的上界

因此：如果一个s,t-割的容量等于一个s,t-可行流的流量，则该割是一个最小割且该流是一个最大流

HIT CS&E

最大流问题的线性规划表示

$$\begin{aligned} \max\quad & f_{ts} \\ s.t.\quad & f_{ij}\le c_{ij} & ij\in E \\ & \sum_{j:ji\in E} f_{ji}-\sum_{j:ij\in E} f_{ij}\le 0 & i\in V \\ & f_{ij}\ge 0 & ij\in E \end{aligned}$$

$$\begin{aligned} \min\quad & \sum_{ij\in E} c_{ij}d_{ij} \\ s.t.\quad & d_{ij}-p_i+p_j\ge 0 & ij\in E \\ & p_s-p_t\ge 1 \\ & d_{ij}\ge 0 & ij\in E \\ & p_i\ge 0 & i\in V \end{aligned}$$

$$\begin{aligned} \min\quad & \sum_{ij\in E} c_{ij}d_{ij} \\ s.t.\quad & d_{ij}-p_i+p_j\ge 0 & ij\in E \\ & p_s-p_t\ge 1 \\ & d_{ij}\in\{0,1\} & ij\in E \\ & p_i\in\{0,1\} & i\in V \end{aligned}$$

考虑整数规划的解

$p_s^*=1,\ p_t^*=0$

$X=\{i\mid p_i^*=1\}$　$V-X=\{i\mid p_i^*=0\}$

X,V-X是一个最小s,t-割

HIT CS&E

9.6.3　两类基于LP方法的近似算法

很多组合优化问题可以表达成整数线性规划问题

将整数约束条件放宽，即得到一个线性规划问题　LP-松弛问题

线性规划问题可以用Karmarkar算法在多项式时间内求解

如何将线性规划问题的解，变成整数得到原问题的一个近似解？

方法1：舍入法　保证舍入得到的近似解代价不会大幅度增加

方法2：primal-dual schema

构造LP-松弛问题的一个整数可行解x作为输出

构造LP-松弛问题的对偶问题的可行解z

比较上述两个解的代价可以得到近似比的界限

两种方法的主要区别在于运行时间，第一种方法需要精确求解线性规划，而第二种不需要。此外，由第二种方法得到的算法可能能够转换成组合优化算法。

HIT CS&E

9.6.4　应用实例之一

- 求解Max-3-CNF问题随机近似算法
- 求解最小节点覆盖问题的线性规划算法

HIT CS&E

求解Max-3-CNF问题随机近似算法

- 基本概念

定义1. 设C是随机近似算法RAS产生的问题P的近似解的代价，C^*是问题P的准确解的代价，n是P的大小. 若$max(C/C^*, C^*/C)\le p(n)$，则称$RAS$具有近似比$p(n)$. 我们也称$RAS$是一个随机$p(n)$-近似算法.

HIT CS&E

- Max-3-CNF问题的定义

输入：合取范式CNF，

每个析取式具有三个变量，

没有任何变量和它的非在同一个析取式中

输出：一个变量赋值，最大化值为1的析取式个数

- 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.

- 性能分析

定理. Random-Max-3-CNF是一个随机8/7-近似算法.

证.

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$i=1, 2,\ldots, m$, 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(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(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.

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

显然, 优化解的代价为m. 于是近似比$=m/(m\,7/8)=8/7$.

求解最小点覆盖问题的线性规划算法

- 问题的定义
  - 输入: 无向图$G=(V, E)$, 每个节点具有权$w(v)$.
  - 输出: $C\subseteq V$, 满足
    (1). $\forall(u, v)\in E$, $u\in C$或者$v\in C$
    (2). $w(C)$最小, $w(C)=\sum_{c\in C}w(c)$.

以前的节点覆盖算法不再适用!

- 问题转化为0-1线性规划问题P_{0-1}
  - 对于 $\forall v\in V$, 定义 $x(v)\in\{0, 1\}$如下:
    - 若v在节点覆盖中, 则$x(v)=1$, 否则$x(v)=0$.
    - $\forall(u, v)\in E$, 若u、v或两者在覆盖中, 则$x(u)+x(v)\ge 1$.
  - 对应的0-1整数规划问题P_{0-1}
    - 优化目标: 最小化 $\sum_{v\in V}w(v)x(v)$
    - 约束条件: $x(u)+x(v)\ge 1$ for $\forall v\in V$
      $x(v)\in\{0, 1\}$ for $\forall v\in V$
  - 0-1整数规划问题是NP-完全问题
  - 我们需要设计近似算法

- 用线性规划问题的解近似0-1规划问题的解
  - 对于 $\forall v\in V$, 定义 $x(v)\in[0, 1]$
  - P_{0-1}对应的线性规划问题LP
    - 优化目标: 最小化 $\sum_{v\in V}w(v)x(v)$
    - 约束条件: $x(u)+x(v)\ge 1$ for $\forall v\in V$
      $x(v)\in[0, 1]$ for $\forall v\in V$
  - 线性规划问题具有多项式时间算法
  - P_{0-1}的可能解是LP问题的可能解
  - P_{0-1}解的代价$\ge LP$的解的代价

- 近似算法

Approx-Min-VC(G, w)

1. $C=0$;
2. 计算LP问题的优化解y;
3. For each $v\in V$ Do
4. If $x(v)\ge 1/2$ Then $C=C\cup\{v\}$;
   /* 用四舍五入法把LP的解近似为P_{0-1}的解 */
5. Return C.

- 算法的性能

定理. Approx-Min-VC是一个多项式时间2-近似算法

证.

由于求解LP需多项式时间, Approx-Min-VC的For循环需要多项式时间, 所以算法需要多项式时间.

下边证明Approx-Min-VC的近似比是2.

往证算法产生的C是一个节点覆盖.

$\forall(u, v)\in E$, 由约束条件可知 $x(u)+x(v)\ge 1$. 于是, $x(u)$和$x(v)$至少一个大于等于1/2, 即u、v或两者在C中. C是一个覆盖.

往证$w(C)/w(C^*)\le 2$.

令C^*是P_{0-1}的优化解, z^*是LP优化解的代价. 因为C^*是LP的可能解, $w(C^*)\ge z^*$.

$$
\begin{aligned}
z^* = \sum_{v\in V} w(v)x(v) &\ge \sum_{v\in V:\, x(v)\ge 1/2} w(v)x(v)\\
&\ge \sum_{v\in V:\, x(v)\ge 1/2} w(v)1/2\\
&= \sum_{v\in C} w(v)1/2\\
&= (1/2)\sum_{v\in C} w(v)\\
&= (1/2)w(C).
\end{aligned}
$$

由$w(C^*)\ge z^*$, $w(C^*)\ge(1/2)w(C)$, 即 $w(C)/w(c^*)\le 2$.

9.6.5 应用实例之二

- 集合覆盖问题
- 集合覆盖问题的线性规划表示
- 对偶拟合方法
- 舍入法
- 随机舍入方法
- Primal-dual schema

问题的定义

- **输入:**

  有限集U, U的一个子集族$\mathcal{S}$, $X=\bigcup_{S\in\mathcal{S}} S$, 每个集合$S$的代价$c(S)$

- **输出:**

  $C\subseteq\mathcal{S}$, 满足

  (1). $U=\bigcup_{S\in C} S$,

  (2). C是满足条件(1)的代价最小的集族, 即$\sum_{S\in C} c(S)$最小.

集合覆盖问题的线性规划表示

对F中的每个集合S, 引入一个变量x_S

$x_S=0$ 表示$S\notin C$　　　　$x_S=1$ 表示$S\in C$

集合覆盖问题的线性规划表示

$$
\begin{aligned}
\text{minimize} \quad & \sum_{S\in\mathcal{S}} c(S)x_S \\
\text{subject to} \quad & \sum_{S:e\in S} x_S \ge 1, \qquad e\in U \\
& x_S\in\{0,1\}, \qquad S\in\mathcal{S}
\end{aligned}
$$

LP-松弛问题—原问题

$$
\begin{aligned}
\text{minimize} \quad & \sum_{S\in\mathcal{S}} c(S)x_S \\
\text{subject to} \quad & \sum_{S:e\in S} x_S \ge 1, \qquad e\in U \\
& x_S \ge 0, \qquad S\in\mathcal{S}
\end{aligned}
$$

对偶问题

$$
\begin{aligned}
\text{maximize} \quad & \sum_{e\in U} y_e \\
\text{subject to} \quad & \sum_{e\in S} y_e \le c(S), \qquad S\in\mathcal{S} \\
& y_e \ge 0, \qquad e\in U
\end{aligned}
$$

贪心算法（对偶拟合方法）

贪心算法:

1. $F=\phi$;　$C=\phi$;
2. while $F\ne U$ do
   - 从$\mathcal{S}$中挑选一个集合S使得$c(S)/|S-F|$最小;
   - $\alpha= c(S)/|S-F|$;
   - 对任意$e\in S-F$, 令$price(e)=\alpha$;
   - $C=C\cup\{S\}$, $F=F\cup S$, $\mathcal{S}=\mathcal{S}-\{S\}$;
3. 输出C

C的代价为$\sum_{e\in U} price(e)$

引理1. 对任意$e \in U$令$y_e = price(e)/H_n$，则由所有y_e构成的向量y是对偶问题的一个可行解。

证明：显然$y_e \geq 0$对任意$e \in U$成立，只需对任意$S \in \mathcal{S}$验证$\sum_{e \in S} y_e \leq c(S)$成立.

假设将S中所有元素依它们在算法中被覆盖的先后次序列出为$e_1, \ldots, e_k$

考察e_i第一次被覆盖的时刻，由于S中此时还有$k-i+1$个元素未被覆盖，将$c(S)$平摊到这些元素上，每个元素分得的代价为$c(S)/(k-i+1)$。

由于在覆盖e_i时，S本身也是优选集合，且选择集合S'时要求$c(S')/|S'-F|$达到最小，故$price(e_i) \leq c(S)/(k-i+1)$。进而

$$y_{e_i} \leq \frac{1}{H_n} \frac{c(S)}{k-i+1}$$

故

$$\sum_{e \in S} y_e = \sum_{i=1}^{k} y_{e_i} = \frac{c(S)}{H_n}\left(\frac{1}{k} + \frac{1}{k-1} + \ldots + \frac{1}{1}\right) = \frac{H_k}{H_n} c(S) \leq c(S)$$

证毕

定理5. 贪心算法的近似比为H_n。

证明：

OPT_f　　OPT

对偶问题　　LP-松弛问题(原问题)

对偶问题的可行解的值不超过集合覆盖问题最优解的值OPT

故

$$\sum_{e \in U} y_e = \sum_{e \in U} \frac{price(e)}{H_n} \leq OPT$$

即　$\sum_{e \in U} price(e) \leq H_n OPT$

注意：不等式的左端恰好是近似解的代价。

证毕

舍入法

频率：对于$e \in U$，e的频率指的是$\mathcal{S}$中包含e的集合的个数

f: U中元素的最大频率

集合覆盖问题的LP-舍入算法

1. 用Karmarkar算法求解LP-松弛问题
2. For $S \in \mathcal{S}$ Do

   IF $x_S \geq 1/f$ THEN $C = C \cup \{S\}$
3. 输出C

定理6. 对于集合覆盖问题，LP-舍入算法的近似比为f

证明：

对于任意$e \in U$，由于e至多属于f个集合中，为了确保

$$\sum_{S \in \mathcal{S},\, e \in S} x_S \geq 1$$

必有某个x_S使得$x_S \geq 1/f$。因此，算法输出的集族中必有一个集合包含了e；进而，算法的输出覆盖了U。

在舍入过程中，对任意$S \in C$，x_S被舍入为1，至多被放大f倍。因此

$$OPT_f = \sum_{S \in \mathcal{S}} c(S) x_S = \sum_{S \in C} c(S) x_S + \sum_{\text{其他}S} c(S) x_S \geq \sum_{S \in C} c(S) \frac{1}{f} + \sum_{\text{其他}S} c(S) x_S \geq \frac{1}{f} \sum_{S \in C} c(S)$$

从而，$\sum_{S \in C} c(S) \leq f\, OPT_f \leq f\, OPT$.

证毕

集合覆盖问题的LP-随机舍入算法

LP-随机舍入算法

1. 用Karmarkar算法求解LP-松弛问题得到最优解$x = \langle x_S : S \in \mathcal{S} \rangle$
2. $C = \phi$
3. For $\forall S \in \mathcal{S}$ Do

   独立地产生一个随机数 *rand*

   IF *rand* > $1 - x_S$ THEN $C = C \cup \{S\}$;

   /* S被选入C的概率为x_S */
4. 输出C

定理7. 对于集合覆盖问题的LP-随机舍入算法，C的代价的数学期望为OPT_f，其中OPT_f是LP-松弛问题的最优解的值。

证明：$E(cost(C)) = \sum_{S \in \mathcal{S}} p_r[S\text{被选入}C].c(S)$

$= \sum_{S \in \mathcal{S}} x_S \,.\, c(S)$

$= OPT_f$　　证毕

定理8. 对于集合覆盖问题的LP-随机舍入算法，$\forall a \in U$被C覆盖的概率大于$1 - 1/e$。

证明：

设a属于$\mathcal{S}$的k个集合中，将LP-松弛问题中这些集合对应的变量记为$x_1, \ldots, x_k$.

在LP-松弛问题的优化解中，$x_1 + \ldots + x_k \geq 1$。

P_r[a未被C覆盖] $= (1-x_1)(1-x_2)\ldots(1-x_k)$

$\leq (1-(x_1+\ldots+x_k)/k)^k$

$\leq (1-1/k)^k$

Pr[a被C覆盖] $= 1-$ Pr[a未被C覆盖]

$\geq 1-(1-1/k)^k$

$\geq 1-1/e$

证毕

改进策略：为了得到完整的集合覆盖，独立运行LP-随机舍入算法$c\log n$次，其中c满足$(\frac{1}{e})^{c\log n} \leq \frac{1}{4n}$，将所有输出集合求并得到$C'$，然后输出$C'$。

P_r[C'未覆盖U] $\leq \sum_{a\in U} P_r$[C'未覆盖a] $\leq n\cdot[(1/e)^{c\log n}] = 1/4$

$E(cost(C'))=OPT_f\cdot c\cdot \log n$

$P_r[cost(C') \geq OPT_f 4c\log n] \leq 1/4$

Markov 不等式：$P_r(X>t) \leq \frac{E(X)}{t}$

P_r[C'覆盖U 且 $cost(C') \leq OPT_f.4c\log n$] $= 1-$ P_r[C'未覆盖U 或 $cost(C') \geq OPT_f.4c\log n$]

$\leq 1-(1/4+1/4)=1/2$

HIT CS&E

Primal-dual schema

基于Primal-dual Schema的集合覆盖近似算法

1. $x \leftarrow 0$; /*向量，$\mathcal{S}$中的每个集合S对应一个分量x_s*/
2. $y \leftarrow 0$; /*向量，U中的每个元素e对应一个分量y_e*/
3. $F \leftarrow \phi$; /*记录被覆盖的子集*/
4. while $F \neq U$ Do
5. 取$e_0 \in U-F$;
6. 增大y_{e_0}直到 $\sum_{e:\, e\in S} y_e = c(S)$ 对某个$S\in\mathcal{S}$成立；
7. 对第6步中满足$\sum_{e:\, e\in S} y_e = c(S)$ 的任意$S\in\mathcal{S}$，令$x_s=1$, $F=F\cup S$;
8. 输出x中$x_s=1$的所有集合构成的子集族；

引理2. 在上述算法中，while循环结束后，x和y分别是原问题和对偶问题的可行解。

证明：

1. While循环结束后，U中的所有元素均被覆盖。
2. 算法初始时，$0 = \sum_{e:\, e\in S} y_e \leq c(S)$对任意$S\in\mathcal{S}$成立。

   算法运行过程中，当$\sum_{e:\, e\in S} y_e = c(S)$对某个$S\in\mathcal{S}$成立后，$S$中的所有元素均被加入到$F$中，因此在算法以后运行的各个阶段内第5步不会再选中S中的任何元素，即$\sum_{e:\, e\in S} y_e$不会再增加。

   基于以上两条原因，算法结束后，$\sum_{e:\, e\in S} y_e \leq c(S)$对任意$S\in\mathcal{S}$成立。

引理3. 在上述算法中，while循环结束时x和y满足以下两个性质：

(1) 对于$\forall S\in\mathcal{S}$，$x_s \neq 0 \Rightarrow \sum_{e:\, e\in S} y_e = c(S)$；

(2) 对于$\forall e\in U$，$y_e \neq 0 \Rightarrow \sum_{S:\, e\in S} x_s \leq f$（$U$中元素的最大频率）

证明：

1. 根据算法的第7步即可证得1。
2. 根据f的定义，对于$\forall e\in U$, e至多属于f个集合；且，对于$\forall S\in\mathcal{S}$，$x_s=1$或0；从而结论(2)成立。

定理9. 基于primal-dual schema的集合覆盖近似算法的近似比为f

证明：由引理2和引理3，我们知道，算法结束时x和y分别是原问题和对偶问题的可行解，且

(1) 对于$\forall S\in\mathcal{S}$，$x_s=0$ 或 $c(S)/1 \leq \sum_{e:\, e\in S} y_e \leq c(S)$；

(2) 对于$\forall e\in U$，$y_e=0$ 或 $1 \leq \sum_{S:\, e\in S} x_s \leq f\cdot 1$；

这恰好是定理4中的条件（$\alpha=1$，$\beta=f$）。

由定理4，$cost(C) = \sum_{S:\, S\in C} c(S) = \sum_{S:\, S\in\mathcal{S}} x_s c(S) \leq 1.f.\sum_{e\in U} y_e$

由于y是对偶问题的可行解，故$\sum_{e\in U} y_e \leq cost(C^*)$.

在算法设计过程中，我们实际上要先寻找恰当的α和β使得定理4中的条件得到满足，然后用算法确保该条件成立。通常，我们往往固定α或β为1，仅让另一个参数变化。算法近似比的好坏，往往取决于所确定的参数的优劣。