PKU-04833060
算法设计与分析『实验班』

第12讲 近似算法 (下)

罗国杰

gluo@pku.edu.cn

2025年春季学期

复习：0-1 背包问题的近似算法

- 贪心算法 G-KK
- 多项式时间近似方案
  - Polynomial-Time Approximation Scheme (PTAS)
- 完全多项式时间近似方案
  - Fully Polynomial-Time Approximation Scheme (FPTAS)

复习：0-1 背包问题的贪心算法 G-KK

贪心算法G-KK

1. 按单位重量的价值从大到小排列物品. 设
   $v_1/w_1 \geq v_2/w_2 \geq \ldots \geq v_n/w_n$.
2. 顺序检查每一件物品, 只要能装得下就将它装入背包,
   设装入背包的总价值为 V.
3. 求 $v_k = \max\{ v_i \mid i = 1, 2, \ldots, n \}$.
   若$v_k > V$, 则将背包内的物品换成物品 k.

实例1 (w_i,v_i): (3,7), (4,9), (5, 9), (2,2); B=6.

实例2 (w_i,v_i): (3,7), (4,9), (5,10), (2,2); B=6.

G-KK给出实例1的解是装入(3,7)和(2,2), 总价值为9.

G-KK给出实例2的解是装入(5,10), 总价值为10.

这两个实例的最优解都是装入(4,9)和(2,2), 总价值为11.

复习：G-KK 的性能

定理 对0-1背包问题的任何实例 I, 有

$$\mathrm{OPT}(I) < 2\mathrm{G\text{-}KK}(I).$$

证 设物品 l 是第一件未装入背包的物品, 由于物品按单位重量的价值从大到小排列, 故有

$$\begin{aligned}\mathrm{OPT}(I) &< \mathrm{G\text{-}KK}(I) + v_l \\ &\le \mathrm{G\text{-}KK}(I) + v_{\max} \\ &\le 2\ \mathrm{G\text{-}KK}(I).\end{aligned}$$

G-KK是2-近似算法.

复习：多项式时间近似方案

算法 Polynomial-Time Approximation Scheme（PTAS）

输入 $\varepsilon > 0$和实例 I.

1. 令 $m = \lceil 1/\varepsilon \rceil$.

2. 按单位重量的价值从大到小排列物品. 设

$v_1/w_1 \geq v_2/w_2 \geq \ldots \geq v_n/w_n$.

3. 对每一个 $t = 1,2,\ldots, m$ 和 t 件物品，检查这 t 件物品的重量之和. 若它们的重量之和不超过B，则接着用 G-KK把剩余的物品装入背包.

4. 比较得到的所有装法，取其中价值最大的作为近似解.

PTAS是一簇算法.对每一个固定的 $\varepsilon > 0$, PTAS是一个算法, 记作 PTAS_ε .

复习：例子

取ε=0.1，m=10.

t=1: 尝试 n次，装入背包物品集分别为{1},{2},…,{n}，再使用 G-KK算法

t=2: 尝试 $n(n-1)/2$次，装入物品集分别是是{1, 2},{1, 3}, … ,{n-1, n}，再使用 G-KK算法.

…

t=10: 尝试 C_n^{10} 次，装入物品集为 {1,…9,10}, {1,…9,11}, …, {n-9,n-8,…,n-1,n}，再用G-KK算法.

总计运行G-KK算法次数：

$$C_n^1 + C_n^2 + ... + C_n^{10}$$

复习： PTAS 的性能

定理　对每一个$\varepsilon > 0$和 0-1 背包问题的实例 I,

$\mathrm{OPT}(I) < (1+\varepsilon)\ \mathrm{PTAS}_{\varepsilon}(I)$,

且 $\mathrm{PTAS}_{\varepsilon}$的时间复杂度为$O(n^{1/\varepsilon+2})$.

证　设最优解为S^*. 若 $|S^*| \le m$, 则算法必得到 S^*. 设$|S^*| > m$. 考虑计算中以 S^*中 m 件价值最大的物品为基础，用G-KK得到的结果 S. 设物品 l 是 S^* 中第一件不在 S 中的物品, 在此之前 G-KK装入不属于 S^* 的物品 (肯定有这样的物品, 否则应该装入物品 l) 单位重量的价值都不小于 v_l/w_l, 当然也不小于 S^* 中所有没有装入的物品的单位重量的价值, 故有

$$\mathrm{OPT}(I) < \sum_{i\in S} v_i + v_l$$

S 包括 S^* 中 m 件价值最大的物品, 它们的价值都不小于 v_l,

$$v_l \le \sum_{i\in S} v_i/m$$

复习：多项式时间近似方案

$$\begin{aligned}\mathrm{OPT}(I) &< \sum_{i\in S} v_i + v_l \\ &\le \sum_{i\in S} v_i + \sum_{i\in S} v_i/m \\ &\le (1+1/m)\ \mathrm{PTAS}_\varepsilon(I) \\ &\le\ (1+\varepsilon)\ \mathrm{PTAS}_\varepsilon(I)\end{aligned}$$

时间复杂度. 从 n 件物品中取 t 件(t =1,2,…,m), 所有可能取法的个数为

$$c_n^1 + c_n^2 + \cdots + c_n^m \le m \cdot \frac{n^m}{m!} \le n^m .$$

对每一种取法, G-KK的运行时间 为$O(n)$, 故算法的时间复杂度为 $O(n^{m+1})$ $= O(n^{1/\varepsilon+2})$.

多项式时间近似方案：以ε>0和问题的实例作为输入 I，对每一个固定的 ε > 0，算法是 1+ε - 近似的。

完全多项式时间近似方案

完全多项式时间近似方案

(Fully Polynomial-Time Approximation Scheme)

以 $\varepsilon > 0$和问题的实例 I 作为输入，时间复杂度为二元多项式 $p(|I|,1/\varepsilon)$，且对每一个固定的$\varepsilon > 0$, 算法的近似比为 $1+\varepsilon$.

构造FPTAS准备：伪多项式时间算法

动态规划算法A 记$G_k(d)$: 当只考虑前 k 件物品时, 为了得到不小于 d 的价值, 至少要装入的物品重量

$$G_k(d) = \min\left\{ \sum_{i=1}^{k} w_i x \mid \sum_{i=1}^{k} v_i x_i \geq d,\ x_i = 0\text{或}1, 1 \leq i \leq k \right\},$$

$0 \leq k \leq n,\ 0 \leq d \leq D,\ D = v_1 + v_2 + \ldots + v_n,$

约定: $\min\varnothing = +\infty.$

$\mathrm{OPT}(I) = \max\{\, d \mid G_n(d) \leq B \,\}$

构造FPTAS准备：动态规划算法

递推公式

$$G_0(d)=\begin{cases}0, & \text{若}d=0,\\ +\infty, & \text{若}d>0,\end{cases}$$

$$G_{k+1}(d)=\begin{cases}\min\{G_k(d), w_{k+1}\}, & \text{若}d\le v_{k+1},\\ \min\{G_k(d), G_k(d-v_{k+1})+w_{k+1}\}, & \text{若}d>v_{k+1},\end{cases}$$

$0\le k\le n\text{-}1,\ 0\le d\le D.$

A的时间复杂度为$O(nD)=O(n^2v_{\max})$，是伪多项式时间算法.

完全多项式时间近似方案

算法FPTAS (Fully Polynomial-Time Approximation Scheme)

输入 $\varepsilon > 0$和实例 I.

1. 令 $b = \max\left\{\left\lfloor \frac{v_{\max}}{(1+1/\varepsilon)n} \right\rfloor, 1\right\}$.
2. 令 $v_i' = \lceil v_i/b \rceil$, $1 \le i \le n$. 把所有 v_i 换成 v_i', 记新得实例为I'.
3. 对 I'应用算法 A 得到解 S, 把 S 取作实例 I 的解.

定理　对每一个$\varepsilon > 0$和 0-1背包问题的实例 I,

$\mathrm{OPT}(I) < (1+\varepsilon)\ \mathrm{FPTAS}\ (I)$,

并且 FPTAS 的时间复杂度为 $O(n^3(1+1/\varepsilon))$.

证明

$$b=\max\left\{\left\lfloor\frac{v_{\max}}{(1+1/\varepsilon)n}\right\rfloor,1\right\}. \quad v_i'=\lceil v_i/b\rceil$$

证　由于

$$(v_i'-1)b < v_i \le v_i' b \qquad (1)$$

对任意的 $T\subseteq\{1,2,\ldots,n\}$

$$0 \le b\sum_{i\in T} v_i' - \sum_{i\in T} v_i < b|T| \le bn \qquad (2)$$

设 I 的最优解为 S^*, 注意到 S 是 I' 的最优解, 故有

$$\mathrm{OPT}(I)-\mathrm{FPTAS}(I) = \sum_{i\in S^*} v_i - \sum_{i\in S} v_i$$

$$= \left(\sum_{i\in S^*} v_i - b\sum_{i\in S^*} v_i'\right) \quad (\le 0,\ \text{由式(1)}\ v_i \le v_i' b)$$

$$+ \left(b\sum_{i\in S^*} v_i' - b\sum_{i\in S} v_i'\right) \quad (\le 0,\ S\text{是关于输入}v_i'\text{的最优解})$$

$$+ \left(b\sum_{i\in S} v_i' - \sum_{i\in S} v_i\right)$$

$$\le \left(b\sum_{i\in S} v_i' - \sum_{i\in S} v_i\right) < bn \quad (\text{由式(2)})$$

证明

$$b=\max\left\{\left\lfloor\frac{v_{\max}}{(1+1/\varepsilon)n}\right\rfloor,1\right\} \quad v_i'=\lceil v_i/b\rceil$$

$\mathrm{OPT}(I)-\mathrm{FPTAS}(I) < bn$

对每一个$\varepsilon > 0$, 若 $b=1$, 则 I'就是 I, S是 I 的最优解.

设 $b>1$, 则

$$\begin{aligned}&\mathrm{OPT}(I)-\mathrm{FPTAS}(I)\\ &< v_{\max}/(1+1/\varepsilon)\\ &\le \mathrm{OPT}(I)/(1+1/\varepsilon) \qquad (\text{因为} v_{\max}\le \mathrm{OPT}(I))\end{aligned}$$

得 $\mathrm{OPT}(I) < (1+\varepsilon)\mathrm{FPTAS}(I)$.

时间：主要是 A对I'的运算，时间复杂度为

$$O(n^2 v_{\max}/b) = O(n^3(1+1/\varepsilon))$$

顶点覆盖与集合覆盖

- 顶点覆盖：任给图 G = <V, E>，求 G 的顶点数最少的顶点覆盖。
- **Set Cover**：给定集合 U 包含 n 个元素，和 U 的一些子集的集合 $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$，以及一个代价函数 $c(S_i)$，要找到 $\mathcal{S}$ 的一个代价最小的子集使 U 的每一个元素都被覆盖。
- 顶点覆盖 $\leq_p$ Set Cover
  - 任给顶点覆盖的实例 G = <V,E>，构造 Set Cover 的实例 (U, $\mathcal{S}$, c)，其中 U = E，$\mathcal{S}$ = { S_u : u ∈ V}，S_u = { (u,v) ∈ E }，且 c(S_u) = 1。
    - 边 → 元素；点 → 集合
  - 顶点覆盖是 Set Cover 的特例。

集合覆盖问题的贪心算法

- **Set Cover**：给定集合 U 包含 n 个元素，和 U 的一些子集的集合 $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$，以及一个代价函数 $c(S_i)$，要找到 $\mathcal{S}$ 的一个代价最小的子集使 U 的每一个元素都被覆盖。
- 每次选取“性价比”最高的子集 S_i，去掉所覆盖的元素；直到所有元素被覆盖。
- 怎样衡量子集的“性价比”
  - 子集 S 覆盖新元素的单位成本： $\alpha(S) = c(S) / |S-V|$
  - 其中 V 表示已经被覆盖的元素
  - 即 S 覆盖新元素的平均代价

贪心集合覆盖算法

1. 初始化，$V \leftarrow \phi$
2. while $V \neq A$ do
   ① 找到覆盖新元素的单位成本 $\alpha(S) = c(S) / |S - V|$ 最低的子集 S
   ② 选取 S 并对每个 $x \in S - V$，赋值 $price(x) = \alpha(S)$
   ③ $V \leftarrow V \cup S$
3. 输出所有被选取的子集 $\{ S \}$

$price(x)$: x 被某个子集覆盖时的平摊代价

贪心集合覆盖算法的近似比分析（1/2）

假设上述算法覆盖元素的顺序为 $x_1, x_2, \ldots, x_n$ （如果多个元素被同时覆盖，则他们之间的顺序任意选定）。

引理：对于任意的 k ($1 \le k \le n$)，$price(x_k) \le OPT / (n - (k - 1))$。

证明：假设 x_k 在上页算法的某次循环被覆盖，循环前被覆盖的元素集为 V，

循环前未被覆盖的元素集为 $V' = U - V$，则有 $|V'| \ge n - (k-1)$。

最优解 OPT 所选中的集合 $\{S^*\}$ 里，至少有一个集合 S^* 满足 $|V'| \cdot c(S^*)/|S^*-V| \le OPT$。

否则，可得到 $\Sigma_{S\in\{S^*\}} c(S) > \Sigma_{S\in\{S^*\}} |S-V| \cdot OPT/|V'| > OPT$，矛盾。

贪心策略的选择 $price(x_k) \le c(S^*)/|S^*-V| \le OPT/|V'| \le OPT / (n - (k-1))$

贪心集合覆盖算法的近似比分析（2/2）

定理：该贪心算法是H_n-近似算法（H_n =1+1/2+…+1/n）。

证明：

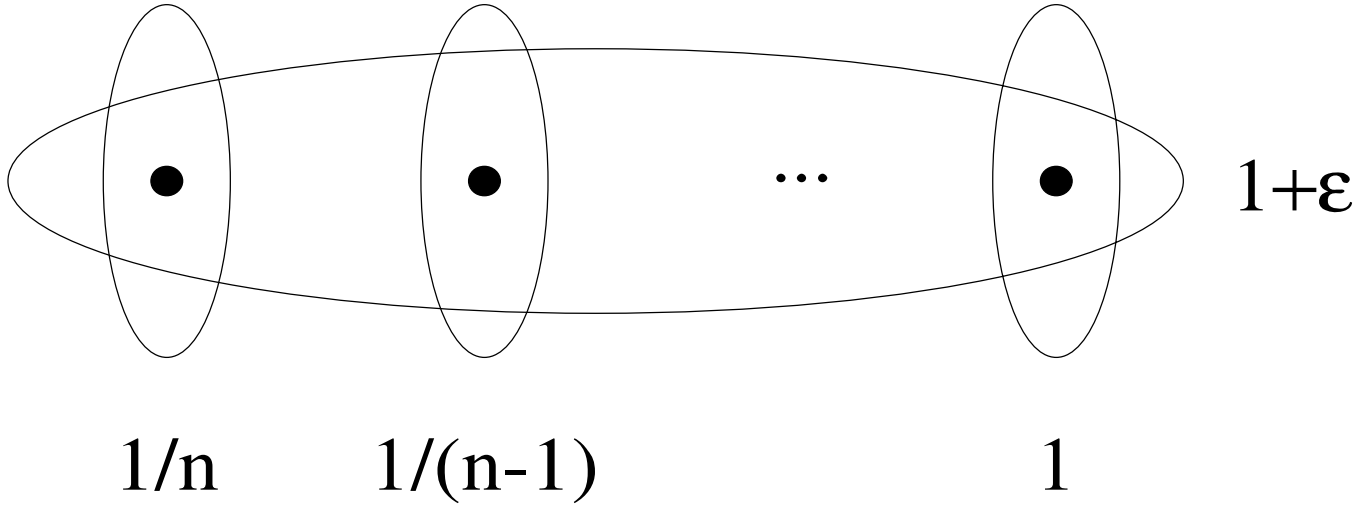
由于贪心算法中每次选取的子集的代价是平均分摊在每个元素上，所以算法中所选取的所有子集的总代价为

$$\sum_{k=1}^{n} price(x_k)$$

根据刚才证明的引理，有

$$\sum_{k=1}^{n} price(x_k) \le (1+\frac{1}{2}+\cdots+\frac{1}{n})\cdot OPT = H_n \cdot OPT$$

贪心集合覆盖算法的紧实例



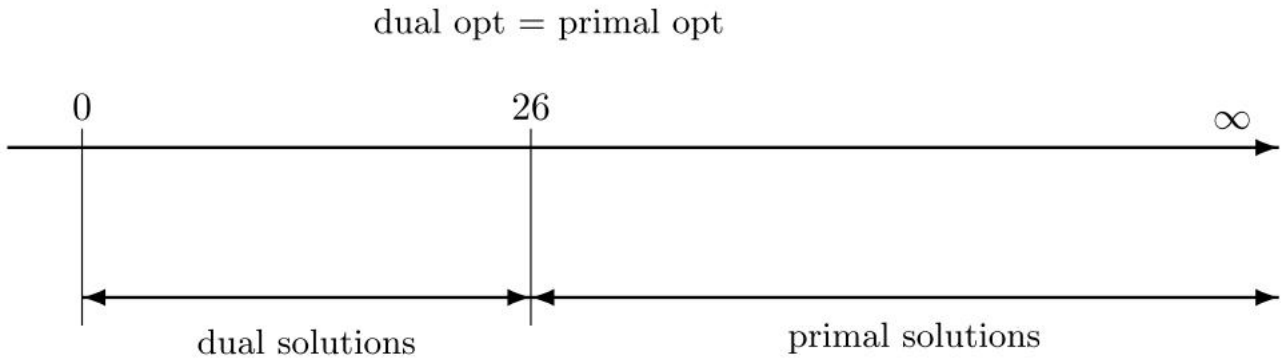
基于线性规划的近似算法 [Chap. 12-15, Vazirani 2001]

- 线性规划的对偶性
- 集合覆盖的对偶拟合解释
- 集合覆盖1/f舍入算法
- 集合覆盖随机舍入算法
- 集合覆盖的原-对偶模式算法
  - 原-对偶模式的近似比

复习：线性规划对偶性

$$\begin{aligned} \text{minimize} \quad & 7x_1 + x_2 + 5x_3 \\ \text{subject to} \quad & x_1 - x_2 + 3x_3 \geq 10 \\ & 5x_1 + 2x_2 - 5x_3 \geq 6 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

$$\begin{aligned} \text{maximize} \quad & 10y_1 + 6y_2 \\ \text{subject to} \quad & y_1 + 5y_2 \leq 7 \\ & -y_1 + 2y_2 \leq 1 \\ & 3y_1 - y_2 \leq 5 \\ & y_1, y_2 \geq 0 \end{aligned}$$



$x = (7/4, 0, 11/4)$ and $y = (2,1)$

复习：线性规划对偶定理

$$\begin{aligned}
&\text{minimize} && \sum_{j=1}^{n} c_j x_j && && \text{maximize} && \sum_{i=1}^{m} b_i y_i && \\
&\text{subject to} && \sum_{j=1}^{n} a_{ij} x_j \geq b_i, && i = 1, \ldots, m && \text{subject to} && \sum_{i=1}^{m} a_{ij} y_i \leq c_j, && j = 1, \ldots, n \\
& && x_j \geq 0, && j = 1, \ldots, n && && y_i \geq 0, && i = 1, \ldots, m
\end{aligned}$$

对偶定理：线性规划原始问题具有有限最优解当且仅当其对偶问题具有有限最优解。并且如果 $x^*=(x^*_1,\ldots,x^*_n)$ 和 $y^*=(y^*_1,\ldots,y^*_m)$ 分别是原始问题和对偶问题的最优解，则

$$\sum_{j=1}^{n} c_j x_j^* = \sum_{i=1}^{m} b_i y_i^*$$

复习：弱对偶定理 与 互补松弛条件

弱对偶定理：如果 $x=(x_1,\ldots,x_n)$ 和 $y=(y_1,\ldots,y_m)$ 分别是原问题和对偶问题的可行解，则

$$\sum_{j=1}^{n} c_j x_j \geq \sum_{i=1}^{m} b_i y_i$$

互补松弛条件：如果 x 和 y 分别是原及对偶问题的可行解，则 x 和 y 是最优解的充要条件为：

原始互补松弛条件：

$$\textbf{For each } 1 \leq j \leq n\textbf{: either } x_j = 0 \textbf{ or } \sum_{i=1}^{m} a_{ij} y_i = c_j$$

对偶互补松弛条件：

$$\textbf{For each } 1 \leq i \leq m\textbf{: either } y_i = 0 \textbf{ or } \sum_{j=1}^{n} a_{ij} x_j = b_i$$

两种 LP 近似算法设计方法

- 方法一：线性规划舍入（LP rounding）
  - 先计算线性松弛问题的分式最优解，再将其转化为整数解
  - 分析整数解与最优解间的间的差距。
- 方法二：原始对偶方案（primal-dual schema）
  - 求解线性松弛问题及其对偶问题。
  - 求线性松弛原始问题的整数解，及对偶问题的可行解。
  - 对偶问题的可行解是最优解OPT的下界，评估解的质量。
- 整性间隙（integrality gap）

$$\sup_{f} \frac{\mathrm{OPT}(I)}{\mathrm{OPT}_f(I)}$$

集合覆盖的对偶拟合

0-1 规划 / 最原始问题

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

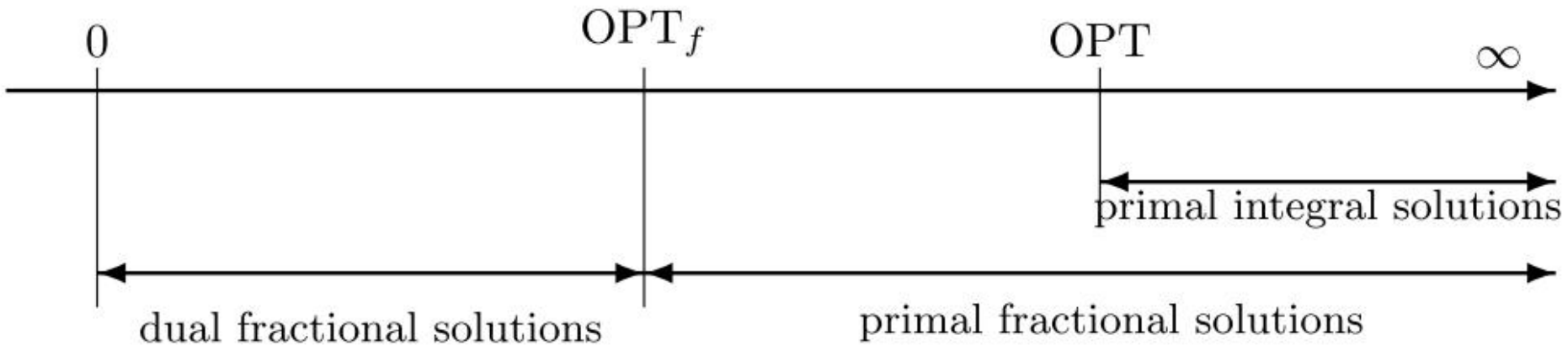
线性松弛 / 原始问题：Covering LP

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

对偶问题：Packing LP

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

小数最优解与整数最优解间的关系



贪心算法的对偶拟合解释（1/3）

Algorithm 2.2 defines dual variables price(e), for each element, e. Observe that the cover picked by the algorithm is fully payed for by this dual solution. However, in general, this dual solution is not feasible (see Exercise 13.2). We will show below that if this dual is shrunk by a factor of H_n, it fits into the given set cover instance, i.e., no set is overpacked. For each element e define,

$$y_e = \frac{\text{price}(e)}{H_n}.$$

Algorithm 2.2 uses the dual feasible solution, $\boldsymbol{y}$, as the lower bound on OPT.

Lemma 13.2 *The vector $\boldsymbol{y}$ defined above is a feasible solution for the dual program (13.3).*

贪心算法的对偶拟合解释（2/3）

Lemma 13.2 *The vector* $\boldsymbol{y}$ *defined above is a feasible solution for the dual program (13.3).*

Proof: We need to show that no set is overpacked by the solution $\boldsymbol{y}$. Consider a set $S \in \mathcal{S}$ consisting of k elements. Number the elements in the order in which they are covered by the algorithm, breaking ties arbitrarily, say $e_1, \ldots, e_k$.

Consider the iteration in which the algorithm covers element e_i. At this point, S contains at least $k-i+1$ uncovered elements. Thus, in this iteration, S itself can cover e_i at an average cost of at most $c(S)/(k-i+1)$. Since the algorithm chose the most cost-effective set in this iteration, $\text{price}(e_i) \leq c(S)/(k-i+1)$. Thus,

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

Summing over all elements in S,

贪心算法的对偶拟合解释（3/3）

$$\sum_{i=1}^{k} y_{e_i} \leq \frac{c(S)}{H_n} \cdot \left(\frac{1}{k} + \frac{1}{k-1} + \cdots + \frac{1}{1} \right) = \frac{H_k}{H_n} \cdot c(S) \leq c(S).$$

Therefore, S is not overpacked. □

Theorem 13.3 *The approximation guarantee of the greedy set cover algorithm is* H_n.

Proof: The cost of the set cover picked is

$$\sum_{e \in U} \text{price}(e) = H_n \left(\sum_{e \in U} y_e \right) \leq H_n \cdot \text{OPT},$$

where OPT denotes the cost of the optimal fractional set cover. The last inequality follows from the fact that $\boldsymbol{y}$ is dual feasible. □

集合覆盖的$\frac{1}{f}$-舍入算法

- 近似比为 f 的集合覆盖近似算法（f 为元素最大出现次数）
- Algorithm 14.1 (Set cover via LP-rounding)
  1. Find an optimal solution to the LP-relaxation
  2. Pick all sets S for which $x_S \geq 1/f$ in this solution

集合覆盖的$\frac{1}{f}$-舍入算法的近似比证明

Theorem 14.2 *Algorithm 14.1 achieves an approximation factor of f for the set cover problem.*

Proof: Let $\mathcal{C}$ be the collection of picked sets. Consider an arbitrary element e. Since e is in at most f sets, one of these sets must be picked to the extent of at least $1/f$ in the fractional cover. Thus, e is covered by $\mathcal{C}$, and hence $\mathcal{C}$ is a valid set cover. The rounding process increases x_S, for each set $S \in \mathcal{C}$, by a factor of at most f. Therefore, the cost of $\mathcal{C}$ is at most f times the cost of the fractional cover, thereby proving the desired approximation guarantee. □

集合覆盖的$\frac{1}{f}$-舍入算法的紧实例

Example 14.3 Let us give a tight example for Algorithm 14.1. For simplicity, we will view a set cover instance as a hypergraph: sets correspond to vertices and elements correspond to hyperedges (this is a generalization of the transformation that helped us view a set cover instance with each element having frequency 2 as a vertex cover instance).

Let $V_1, \ldots, V_k$ be k disjoint sets of cardinality n each. The hypergraph has vertex set $V = V_1 \cup \ldots \cup V_k$, and n^k hyperedges; each hyperedge picks one vertex from each V_i. In the set cover instance, elements correspond to hyperedges and sets correspond to vertices. Once again, inclusion corresponds to incidence. Each set has cost 1. Picking each set to the extent of $1/k$ gives an optimal fractional cover of cost n. Given this fractional solution, the rounding algorithm will pick all nk sets. On the other hand, picking all sets corresponding to vertices in V_1 gives a set cover of cost n. □

集合覆盖的随机舍入算法：预备知识

- 马尔可夫不等式（Markov's inequality）
  - If X is a nonnegative random variable and $a > 0$, then $\mathbf{Pr}[X \geq a] \leq \mathbf{E}[x]/a$
- 并集上界（Union bound）或布尔不等式（Boole's inequality）
  - For a countable set of events $A_1, A_2, A_3, \ldots$, we have $\mathbf{Pr}[\bigcup_{i=1}^{\infty} A_i] \leq \sum_{i=1}^{\infty} \mathbf{Pr}[A_i]$

第12讲"随机算法"将证明/复习上述不等式

集合覆盖的随机舍入算法（1/3）

Let $\boldsymbol{x} = \boldsymbol{p}$ be an optimal solution to the linear program. For each set $S \in \mathcal{S}$, pick S with probability p_S, the entry corresponding to S in $\boldsymbol{p}$. Let $\mathcal{C}$ be the collection of sets picked. The expected cost of $\mathcal{C}$,

$$\mathbf{E}[\text{cost}(\mathcal{C})] = \sum_{S \in \mathcal{S}} \mathbf{Pr}[S \text{ is picked}] \cdot c_S = \sum_{S \in \mathcal{S}} p_S \cdot c_S = \text{OPT}_f.$$

Next, let us compute the probability that an element $a \in U$ is covered by $\mathcal{C}$. Suppose that a occurs in k sets of $\mathcal{S}$. Let the probabilities associated with these sets be $p_1, \ldots, p_k$. Since a is fractionally covered in the optimal solution, $p_1 + p_2 + \cdots + p_k \geq 1$. Using elementary calculus, it is easy to show that under this condition, the probability that a is covered by $\mathcal{C}$ is minimized when each of the p_i's is $1/k$. Thus,

$$\mathbf{Pr}[a \text{ is covered by } \mathcal{C}] \geq 1 - \left(1 - \frac{1}{k}\right)^k \geq 1 - \frac{1}{e},$$

where e is the base of natural logarithms. Hence each element is covered with constant probability by $\mathcal{C}$.

集合覆盖的随机舍入算法（2/3）

To get a complete set cover, independently pick $c \log n$ such subcollections, and compute their union, say $\mathcal{C}'$, where c is a constant such that

$$\left(\frac{1}{e}\right)^{c \log n} \leq \frac{1}{4n}.$$

Now,

$$\mathbf{Pr}[a \text{ is not covered by } \mathcal{C}'] \leq \left(\frac{1}{e}\right)^{c \log n} \leq \frac{1}{4n}.$$

Summing over all elements $a \in U$, we get

$$\mathbf{Pr}[\mathcal{C}' \text{ is not a valid set cover}] \leq n \cdot \frac{1}{4n} \leq \frac{1}{4}.$$

集合覆盖的随机舍入算法（3/3）

Clearly, $\mathbf{E}[\mathcal{C}'] \leq \mathrm{OPT}_f \cdot c \log n$. Applying Markov's Inequality (see Section B.2) with $t = \mathrm{OPT}_f \cdot 4c \log n$, we get

$$\mathbf{Pr}[\mathrm{cost}(\mathcal{C}') \geq \mathrm{OPT}_f \cdot 4c \log n] \leq \frac{1}{4}.$$

The probability of the union of the two undesirable events is $\leq 1/2$. Hence,

$$\mathbf{Pr}[\mathcal{C}' \text{ is a valid set cover and has cost} \leq \mathrm{OPT}_f \cdot 4c \log n] \geq \frac{1}{2}.$$

若随机舍入不满足“C’ is a valid set cover and has cost ≤ OPT_f 4c log n”，
则重复上述随机舍入操作。期望2次随机舍入即能采样到满足要求的 C’。

原始-对偶模式的近似算法

原始问题

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{n} c_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad i = 1, \cdots, m \\ & x_j \geq 0 \quad\quad\quad j = 1, \cdots, n \end{aligned}$$

对偶问题

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{m} b_i y_i \\ \text{subject to} \quad & \sum_{i=1}^{m} a_{ij} y_i \leq c_j \quad j = 1, \cdots, n \\ & y_i \geq 0 \quad\quad\quad i = 1, \cdots, m \end{aligned}$$

Primal complementary slackness conditions

Let $\alpha \geq 1$.

For each $1 \leq j \leq n$: either $x_j = 0$ or $c_j/\alpha \leq \sum_{i=1}^{m} a_{ij} y_i \leq c_j$.

Dual complementary slackness conditions

Let $\beta \geq 1$.

For each $1 \leq i \leq m$: either $y_i = 0$ or $b_i \leq \sum_{j=1}^{n} a_{ij} x_j \leq \beta \cdot b_i$,

原始-对偶模式的近似比

Proposition 15.1 *If $\boldsymbol{x}$ and $\boldsymbol{y}$ are primal and dual feasible solutions satisfying the conditions stated above then*

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

Proof:

$$\begin{aligned}\sum_{j=1}^{n} c_j x_j &\leq \alpha \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j = \alpha \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i \\ &\leq \alpha\beta \sum_{i=1}^{m} b_i y_i \ . \end{aligned} \tag{15.1}$$

The first and second inequalities follow from the primal and dual conditions, respectively. The equality follows by simply changing the order of summation. □

集合覆盖的对偶拟合

原始问题：Covering LP

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

对偶问题：Packing LP

$$
\begin{aligned}
maximize \quad & \sum_{e \in U} y_e & \\
subject\ to \quad & \sum_{e\,:\,e \in S} y_e \leq c(S) & S \in \mathcal{S} \\
& y_e \geq 0 & e \in U
\end{aligned}
$$

集合覆盖的原始-对偶模式算法

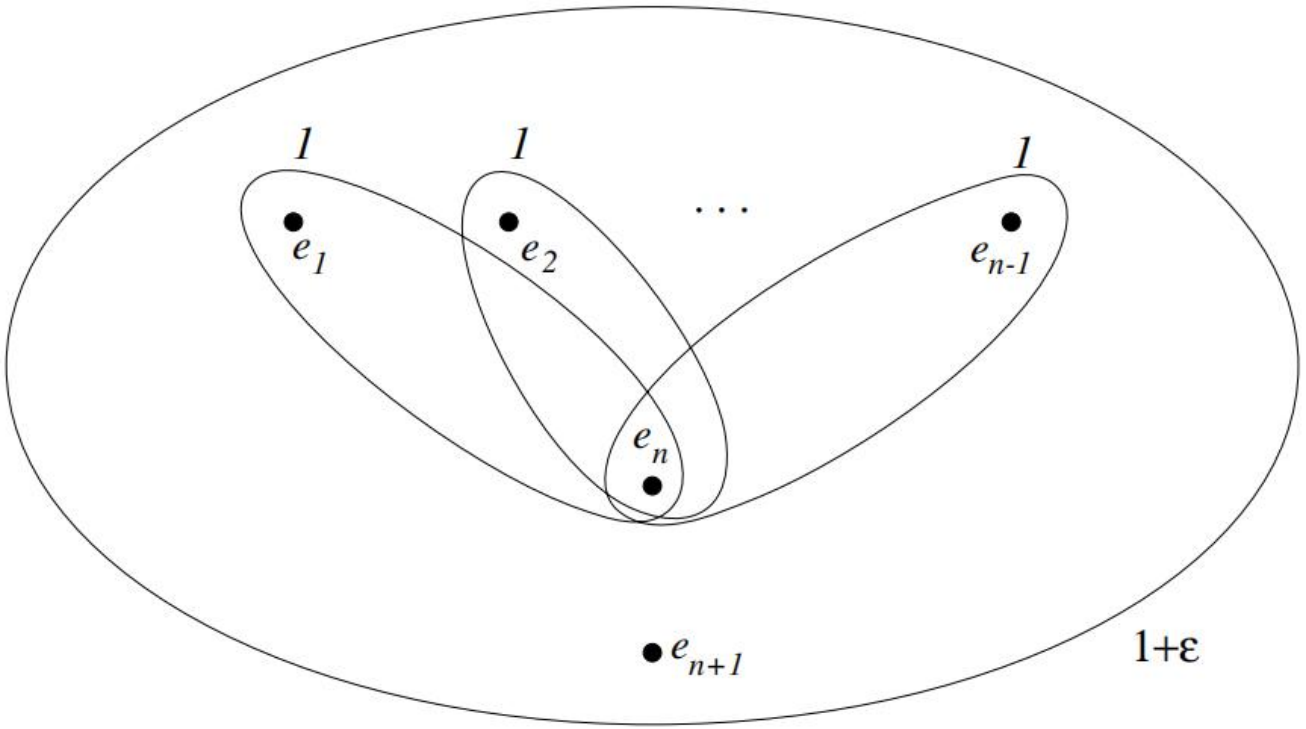
- Algorithm 15.2 (Set cover - factor f)
  1. Initialization: $x \leftarrow 0$; $y \leftarrow 0$
  2. Until all elements are covered, do:
     - Pick an uncovered element, say e, and raise y_e, until some set goest tight.
     - Pick all tight sets in the cover and update x.
     - Declare all the elements occurring in these sets as “covered”.
  3. Output the set cover x.

- 令 $\beta = f$: $y_e \neq 0 \Rightarrow \Sigma_{S:\, e \in S} x_S \leq f$ （“pick an element”：自动满足约束）
- 令 $\alpha = 1$: $\Sigma_{e \in S} y_e = c(S) \Rightarrow x_S \neq 0$ （“pick all tight sets”：$x_S \leftarrow 1$）

Theorem 15.3 *Algorithm 15.2 achieves an approximation factor of f.*

Proof: Clearly there will be no uncovered elements and no overpacked sets at the end of the algorithm. Thus, the primal and dual solutions will both be feasible. Since they satisfy the relaxed complementary slackness conditions with $\alpha = 1$, $\beta = f$ by Proposition 15.1 the approximation factor is f. □

Example 15.4 A tight example for this algorithm is provided by the following set system:



本讲小结

- PTAS 和 FPTAS 近似算法
  (以背包问题为例)
  - 多项式时间近似算法 PTAS
  - 完全多项式时间近似算法 FPTAS
- 基于线性规划的近似算法
  (以集合覆盖为例)
  - 集合覆盖贪心近似的对偶拟合解释
  - 集合覆盖 1/f 舍入算法
  - 集合覆盖随机舍入算法
  - 集合覆盖的原始-对偶模式算法
    - 原-对偶模式的近似比

- 参考资料
  - Vazirani, “Approximation Algorithms”, 2001
  - Willamson and Shmoys, “The Design of Approximation Algorithms”, 2010