Lecture 4 (Part 2)

How good is a greedy algorithm?

(•) In general, it is OK to use, if no better algorithm exists.

We can solve the Set Cover Problem by using Linear Programming.

Definition (Linear Programming)

A "Linear Programming" is the problem of optimizing (minimizing or maximizing) a linear function subject to linear inequality constraints.

Example

Minimize $\sum_{j=1}^{m} c_j x_j$

Subject to $\sum_{j=1}^{n} a_{ij} x_j \geq b_i$, $i = 1, 2, \cdots, m$

$x_j \geq 0$

(Note) 由於答案不一定要 Integer, 有 polynomial time algorithm

"Integer programming is harder."

Set Cover Problem

variable x_j for set S_j = $\begin{cases} 1 & \text{if } S_j \text{ is selected;} \\ 0 & \text{if not.} \end{cases}$

上面例子

$S = [1,6]$, ...

Min. $\sum_{j=1}^{k} c_j x_j = OPT$

$x_j = \begin{cases} 1 & \text{if } S_j \text{ is selected} \\ 0 & \text{if not.} \end{cases}$

subject to

One of S_1 and S_2 must be chosen! →

One of S_5, S_6 and S_7 must be chosen! →

$$\left\{\begin{array}{l} x_1 + x_2 \geq 1 \\ x_1 + x_3 + x_4 \geq 1 \\ x_2 + x_3 + x_5 \geq 1 \\ x_4 + x_6 + x_7 \geq 1 \\ x_5 + x_6 + x_7 \geq 1 \\ x_4 + x_5 + x_6 \geq 1 \end{array}\right\} ?$$

Consider 1, 2, 3, 4, 5, 6

Each element has to be "covered".

1 : S_1 and S_2

2 : S_1, S_3, S_4,

3 : S_2, S_3, S_5

4 : S_4, S_6, S_7

5 : S_5, S_6, S_7

6 : S_4, S_5, S_6

How to apply ?

Here, $c_j = |S_j|$.

x_i's are in $\{0,1\}$.

$S_1 = \{1,2\}$

$S_2 = \{1,3\}$

$S_3 = \{2,3\}$

$S_4 = \{2,4,6\}$

$S_5 = \{3,5,6\}$

$S_6 = \{4,5,6\}$

$S_7 = \{4,5\}$

Review p. 8

More on "Set Cover Problem"

Problem Each set of weight (cost) 1.

Target: Find as few sets as possible.

Example cover [1, 12].

$T_1 = \{1, 2, 3, 4, 5, 6\}$

$T_2 = \{5, 6, 8, 9\}$

$T_3 = \{1, 4, 7, 10\}$

$T_4 = \{2, 5, 7, 8, 11\}$

$T_5 = \{3, 6, 9, 12\}$

$T_6 = \{10, 11\}$

Better

先選一個，再選尚未 cover 的集合它增加最多的新元素！

$T_1 \rightarrow T_4 \rightarrow T_5 \rightarrow T_3$

$C = \{T_1, T_3, T_4, T_5\}$

(A solution but not an optimal one!)

G

Idea of greedy algorithm (Greedy-Set-Cover (X, F)).

mother set, collect

(*) At each stage, the greedy algorithm picks the set that covers the greatest number of elements not yet covered.

1. $U \leftarrow X$ (U = {uncovered elements})
2. $C \leftarrow \phi$ (C = {covered elements})
3. While $U \neq \phi$
4. do select an $S \in F$ that maximizes $|S \cap U|$
5. $U \Leftarrow U \setminus S$.

   Stop at $U = \phi$

Theorem Let $d = \max\{|S'| : S' \in \mathcal{F} \text{ and } \mathcal{F} \text{ is a set cover of } S\}$.

Then Greedy-Set-Cover is a polynomial time $H(d)$-approximation algorithm where $H(d) = \sum_{i=1}^{d} \frac{1}{i} = \ln d + O(1)$.

(Note: If $C(T)$ and $C(OPT)$ are the solution of Greedy-Set-Cover and optimal solution respectively, then $C(T)/C(OPT) \leq H(d)$.)
($C(T) = |C|$, $C(OPT) = |C^*|$, $\leq \approx$)

"Algorithm G"

Proof:

1 分給新加入的元素(平均)

Assign a price "1" to each set $S' \in \mathcal{F}$ selected by the algorithm and distribute this price over the elements (evenly) covered for the first time.

S_i : i^{th} subset selected

$c_x^{(i)}$: the price allocated to $x \in X$ that is covered for the first time at i^{th} iteration:

$$c_x^{(i)} = \frac{1}{|S_i \setminus (\bigcup_{j=1}^{i-1} S_j)|}.$$

$c_x = c_x^{(i)}$ for some i!

This implies that $|C| = \sum_{x \in X} c_x$. (C is a set cover of X)

Now, we use c_x where $x \in X$ to estimate $|C^*|$.

(*) The price assigned to the optimal cover is

$$\sum_{S \in C^*} \sum_{x \in S} c_x$$

(The sets in C^* may overlap.)

$$\Rightarrow \sum_{S \in C^*} \sum_{x \in S} c_x \geq \sum_{x \in X} c_x = |C| .$$

Claim: $\sum_{x \in S} c_x \leq H(|S|)$.

Note that if this claim is true, then

$$|C| \leq \sum_{S \in C^*} H(|S|) \qquad \left(\because |S| \leq \max\{|S| \mid S \in C^*\} \right)$$

$$\leq |C^*| \cdot H(d) \quad \text{where} \quad d = \max\{|S| \mid S \in F\} \qquad (\leq d)$$

$$\Rightarrow \frac{|C|}{|C^*|} \leq H(d).$$

original collection

(At this stage, we don't know $\max\{|S| \mid S \in C^*\}$)

Let $u_i(S)$ be the number of elements in S remaining uncovered after $S_1, S_2, \cdots, S_i$ are selected to C.

$$u_0(S) = |S|, \quad u_i(S) = \left| S \setminus \bigcup_{j=1}^{i} S_j \right|.$$

Clearly $u_0(S) \geq u_1(S) \geq \cdots \geq u_{|S|}(S)$.

Let k be the least index s.t. $u_k(S) = 0$.

Then $u_0(S) \geq u_1(S) \geq \cdots \geq u_{k-1}(S) \geq u_k(S) = 0$

$\Rightarrow$ $k = |S|$

$$\sum_{x \in S} c_x = \sum_{i=1}^{k} (u_{i-1}(S) - u_i(S)) \cdot \frac{1}{|S_i \setminus \bigcup_{j=1}^{i-1} S_j|}$$

Since $|S_i \setminus \bigcup_{j=1}^{i-1} S_j| \geq |S \setminus \bigcup_{j=1}^{i-1} S_j| = u_{i-1}(S)$, (?)

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

($u_i(S) \to u_i$; $u_{i-1}(S) \to u_{i-1}$)

$$= \sum_{i=1}^{k} (u_{i-1} - u_i) \frac{1}{u_{i-1}}$$

$$= \sum_{i=1}^{k} \left(1 - \frac{u_i}{u_{i-1}}\right)$$

$$\leq \left(H(u_0) - H(u_1)\right) + \left(H(u_1) - H(u_2)\right) + \cdots + \left(H(u_{k-1}) - H(u_k)\right)$$

$$= H(u_0) - H(0)$$

($H(0) = 0$)

$$= H(|S|).$$

∎

$$H(u_{i-1}) - H(u_i) = \frac{1}{u_i + 1} + \frac{1}{u_i + 2} + \cdots + \frac{1}{u_{i-1}} \geq \frac{1}{u_{i-1}} \cdot (u_{i-1} -$$

Approximating Shortest Superstring via Set Cover

First, we give an example.

Let $S = \{\underset{①}{CATGC}, \underset{②}{CTAAGT}, \underset{③}{GCTA}, \underset{④}{TTCA}, \underset{⑤}{ATGCATC}\}$.

$\Downarrow$

$S =$

$S_1 = \{CATGC\}$ ①

$S_2 = \{CTAAGT\}$ ②

$S_3 = \{GCTA\}$ ③

$S_4 = \{TTCA\}$ ④

$S_5 = \{ATGCATC\}$ ⑤

$k=1$ $S_6 = \{①, ②\}$

$k=2$ $S_7 = \{①, ③\}$ bad!

$k=2$ $S_8 = \{①, ④\}$ bad!

$k=4$ $S_9 = \{①, ⑤\}$

$S_{10} = \{②, ③\}$

$S_{11} = \{②, ④\}$

$S_{12} = \{②, ⑤\}$

$k=0$ $S_{13} = \{③, ④\}$ bad!

$S_{14} = \{③, ⑤\}$

$S_{15} = \{④, ⑤\}$

Clearly, S can be covered by a sub-collection of S.

But, we need a collection with minimum cost.

$c(S_1) = 5$, $c(S_2) = 6$, $c(S_3) = 4$, $c(S_4) = 4$, $c(S_5) = 7$,

$c(S_6) = 10$ . . . $c(S_{15}) = 10$

CATGC
CTAAGT
$\Downarrow$
CATGCTAAGT

TTCA
ATGCATC
$\Downarrow$
TTCATGCATC.

Definition

Let $S = \{\vec{s}_1, \vec{s}_2, \cdots, \vec{s}_n\}$. For strings $\vec{s}_i$ and $\vec{s}_j$, if the last $k > 0$ symbols (characters) of $\vec{s}_i$ ($\vec{s}_j$) are the same as the first k symbols of $\vec{s}_j$ ($\vec{s}_i$), let $\sigma_{i,j,k}$ denote the string obtained by overlapping these k symbols of $\vec{s}_i$ and $\vec{s}_j$.

Let I be the set of $\sigma_{i,j,k}$'s for all valid choices of i, j, k, i.e., the set of all "good" superstrings of pairs of strings in S, i.e., $k > 0$.

(如果 $k = 0$, 则不选!)

We use $\text{set}(\sigma_{i,j,k})$ to denote $\{\vec{s}_i, \vec{s}_j\}$.

Now, $F = \{\text{set}(\sigma) : \sigma \in S \cup I\}$.

(*) 如果能找到一个 set cover, 对应的 σ 就可以把它们串起来成为 S 的一个 superstring (with $\vec{s}_i$'s as substrings).

Algorithm G' Greedy-Set-Cover with Cost (X, F)

1. $C \leftarrow \phi$
2. $U \leftarrow X$
3. While $U \neq \phi$ do
4. Find $S \in F \setminus C$ that minimizes $\alpha =_{def} \frac{cost(S)}{S \cap U}$.
5. for each $x \in S \cap U$ do
6. price(x) $\leftarrow \alpha$
7. $C \leftarrow C \cup \{S\}$
8. $U \leftarrow U \setminus S$
9. Return C

Algorithm $\tilde{S}$ (Superstring Algorithm)

1. Compute S.C. in Algorithm G'
2. Let $\{set(\sigma_1), \ldots, set(\sigma_k)\}$ be the collection of sets returned by Algorithm G'.
3. return $\vec{s} =_{def} \sigma_1 \cdot \sigma_2 \cdot \cdots \cdot \sigma_k$.

Lemma Let OPT_{SC} denote the cost of an optimal solution to the S.C. instance in (X, F), and OPT_{SS} denote the length of the shortest superstring of S. Then $OPT_{SC} \leq 2 \cdot OPT_{SS}$.

Proof. Let $set(a_1), \cdots, set(a_t)$ be a solution S.C. (不一定 optimal)

Let $a_1, a_2, \cdots, a_t$ be the corresponding strings. Since input string is covered by at most two a strings,

$\sum_i |a_i| \leq 2\, OPT_{SS}$. (?)

Theorem Greedy SCP $\leq 2 H_d\, OPT_{SSP}$