Advanced Algorithms*
Lecture 11: Approximation algorithms for Set Cover

Notes taken by Cheng Chen, Lingxiao Huang

March 2011

Summary: This lecture provides two approximation algorithm for Set Cover problem and apply LP-rounding on Vertex Cover problem to get a 2-approx algorithm.

1 Set Cover Problem

1.1 Basic Concepts

We have a set U of n elements $e_1, e_2, \cdots, e_n$, and a set S of k subsets $S_1, S_2, \cdots, S_k$ from U. A set cover is a collection of subsets from S satisfied that every elements in U belongs to one of the subsets.
In addition, a cost function $c : \mathrm{S} \to \mathbb{Z}^+$ and the cost of a set cover is denoted by the sum of costs of each subsets in the collection of selected subsets. Our objective is to find the collection of subsets that minimizes the cost.

Example 1. $\mathrm{U} = \{1, 2, \cdots, 6\}$ and $\mathrm{S}' = \{S_1, S_2, S_3\}$ where $S_1 = \{1, 2\}$, $S_2 = \{2, 3, 4\}$ and $S_3 = \{5, 6\}$. In addition, we have the cost of the three sets are 1, 2 and 3 respectively. The example is described as below:

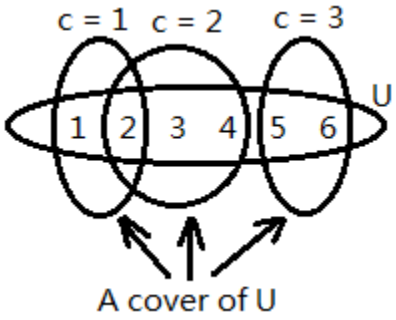

Figure 1: Set Cover Example

By definition, $\mathrm{S}'_1 = \{S_2, S_3\}$ is not a set cover of $(\mathrm{U}, \mathrm{S}, c)$ for that S'_1 does not cover 1. Oppositely, $\mathrm{S}'_2 = \{S_1, S_2, S_3\}$ is the only set cover of this example and $c(\mathrm{S}'_2) = c(S_1) + c(S_2) + c(S_3) = 6$.

1.2 Definitions

A Set Cover problem $(\mathrm{U}, \mathrm{S}, c)$ is denoted by
Input:

* Lecture notes for the course Advanced Algorithms, Spring 2011 - full of typos! Use at your own risk.

$$\begin{aligned} \mathrm{U} &= \{e_1, e_2, \cdots, e_n\} \\ \mathrm{S} &= \{S_1, S_2, \cdots, S_k\} \qquad S_i \subseteq \mathrm{U} \\ c &: \mathrm{S} \to \mathbb{Z}^+ \end{aligned}$$

Output:

$$\min_{\mathrm{S}'} c(\mathrm{S}') = \min_{\mathrm{S}'} \sum_{S \in \mathrm{S}'} c(S) \qquad \text{subject to} \quad \forall i, \exists j \text{ s.t. } S_j \in \mathrm{S}' \text{ and } e_i \in S_j$$

1.3 Complexity

Theorem 2. *The Set Cover Problem is NP-hard.*

Proof. We prove by reduction from Vertex Cover. For any given instance of Vertex Cover in graph $G = (\mathrm{V}, \mathrm{E})$ and an integer j, the Vertex Cover problem determines whether there are j vertexes covering all edges. We can construct a corresponding Set Cover Problem $(\mathrm{U}, \mathrm{S}, c)$ with $\mathrm{U} = E$ and for each vertex in V, there is a set $S \in \mathrm{S}$ containing all the edges in E directly linked to the vertex. Further, for every $S \in \mathrm{S}$, we have $c(S) = 1$. This construction can be done in time that is polynomial in the size of Vertex Cover instance. Suppose that G has a vertex of size at most j, then we have a set cover in the constructed problem by simply choosing the subsets corresponding to the selected vertexes. On the other hand, suppose that we have a set cover in the constructed problem, we can therefore choose the vertexes corresponding to the selected subsets. In this way, we have Vertex Cover $\leq_P$ Set Cover. So that Set Cover Problem is NP-hard for that Vertex Cover is NP-hard. □

2 Greedy Set Cover

2.1 Algorithm Description

Input: element set $\mathrm{U} = \{e_1, e_2, \cdots, e_n\}$, subset set $\mathrm{S} = \{S_1, S_2, \cdots, S_k\}$ and cost function $c : \mathrm{S} \to \mathbb{Z}^+$
Output: set cover C with minimum cost

```
C ← φ;
while C ≠ U do
    Calc cost effectiveness, which will be defined later, α_1, α_2, ··· , α_l of the unpicked sets S_1, S_2, ··· , S_l
    respectively;
    pick a set S with the minimum cost effectiveness α;
    C ← C ∪ S;
end
output the picked sets S;
```

Algorithm 1: Greedy Set Cover Algorithm

The cost effectiveness α of subset S above is denoted by

$$\alpha = \frac{c(S)}{|S - \mathrm{C}|}$$

C is the picked sets in the corresponding iteration, and $|S - \mathrm{C}|$ means the size of the substraction set of S and all union of all subsets in C. It is obvious that for any particular subset S, its cost effectiveness α varies from one iteration to another.

Example 3. $\mathrm{U} = \{1, 2, \cdots, 6\}$ and $\mathrm{S}' = \{S_1, S_2, \cdots, S_{10}\}$. The elements and cost of each set is as below

$$\begin{array}{rclrcl} S_1 & = & \{1,2\} & c(S_1) & = & 3 \\ S_2 & = & \{3,4\} & c(S_2) & = & 3 \\ S_3 & = & \{5,6\} & c(S_3) & = & 3 \\ S_4 & = & \{1,2,3,4,5,6\} & c(S_4) & = & 14 \\ S_5 & = & \{1\} & c(S_5) & = & 1 \\ S_6 & = & \{2\} & c(S_5) & = & 1 \\ S_7 & = & \{3\} & c(S_5) & = & 1 \\ S_8 & = & \{4\} & c(S_5) & = & 1 \\ S_9 & = & \{5\} & c(S_5) & = & 1 \\ S_{10} & = & \{6\} & c(S_5) & = & 1 \end{array}$$

the example is described as in the following graph

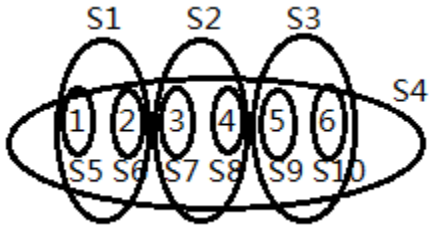

Figure 2: Greedy Set Cover Example

There are several feasible set covers in this example, such as $\{S_1, S_2, S_3\}$ and $\{S_2, S_5, S_6, S_9, S_{10}\}$. However, the set cover with the minimum cost is $\{S_5, S_6, S_7, S_8, S_9, S_{10}\}$, and the corresponding cost is 6.
For example, if $\mathrm{C} = \{S_1, S_2, S_9\}$ in this example, for subset S_3 we have

$$\alpha = \frac{c(S_3)}{|S_3 - \mathrm{C}|} = \frac{3}{1} = 3$$

2.2 Approximation Rate of the Greedy Set Cover

We sort the elements $e_1, e_2, \cdots, e_n$ by the iteration when they were added. Let this be $e'_1, e'_2, \cdots, e'_n$.

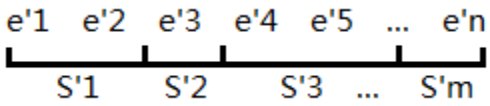

Figure 3: sort elements by iteration time

Notice that the above subsets S'_i is different from the original S'_i, it only includes the elements added by itself to C.
For each e' which is added to S', we define

$$price(e') = \alpha_{S'}$$

namely the cost effectiveness of the set where e was covered.

Observation 4. $A(I) = \sum_{k=1}^{n} price(e'_k)$

We denote the cost of the set cover of Greedy Set Cover by $A(I)$, where I is the input. Therefore, we have

$$\begin{aligned} A(I) &= c(\mathrm{C}) \\ &= \sum_{S'_i \in \mathrm{C}} c(S'_i) \\ &= \sum_{S'_i \in \mathrm{C}} \sum_{e' \in S'_i} \frac{c(S'_i)}{|S'_i|} \\ &= \sum_{k=1}^{n} price(e'_k) \end{aligned}$$

The third equation above is breaking the cost of a subset into the cost of elements in the subset.

Lemma 5. *Let $k = 1, 2, \cdots, n$, we have $price(e'_k) \leq OPT/(n-k+1)$.*

Proof. Note that the remaining elements $e'_k, e'_{k+1}, \cdots, e'_n$ can be covered at cost at most OPT, namely in the OPT solution of the problem. Therefore there is a subset in the OPT solution which has not been selected and with a cost effectiveness at most $OPT/(n-k+1)$. Further, our algorithm is choosing to cover a set which minimum cost effectiveness, so that we should have $price(e'_k) \leq OPT/(n-k+1)$ to guarantee our algorithm could pick the selected subset. □

Theorem 6. *Greedy Set Cover is $\ln(n)$-approx*

Proof.

$$\begin{aligned} A(I) &= \sum_{k=1}^{n} price(e'_k) \\ &\leq (\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1)OPT \\ &\leq \lceil \ln(n) \rceil OPT \end{aligned}$$

The first equation is due to the observation and the second is derived from the above lemma. Thus, we have prove the Greedy Set Cover is $\ln(n)$-approx. □

2.3 Tightness of Greedy Set Cover

$\mathrm{U} = \{e_1, e_2, \cdots, e_n\}$ and $\mathrm{S} = \{S_1, S_2, \cdots, S_n\}$. The elements and cost of each set is as below

$$\begin{array}{rclrcl} S_1 &=& \{1\} & c(S_1) &=& 1/n \\ S_2 &=& \{2\} & c(S_2) &=& 1/(n-1) \\ S_3 &=& \{3\} & c(S_3) &=& 1/(n-2) \\ & & \cdot & & & \cdot \\ & & \cdot & & & \cdot \\ & & \cdot & & & \cdot \\ S_{n-1} &=& \{n-1\} & c(S_{n-1}) &=& 1/2 \\ S_n &=& \{1, 2, \cdots, n\} & c(S_n) &=& 1+\epsilon \end{array}$$

The ϵ above is a negligible value. It is obvious that the OPT Set Cover is $\{S_n\}$ with cost n. However, in our algorithm, for iteration i, we will pick S_i for that S_i has the minimum cost effectiveness, and ϵ is used here to make sure the cost effectiveness of S_i is smaller that of S_n. In this way, we will choose all the n subsets and get a solution with cost

$$\frac{1}{n} + \frac{1}{n-1} + \cdots + 1 = \lceil \ln(n) \rceil$$

So we construct a worst case for any n with approximation rate $\ln(n)$. In conclusion, the $\ln(n)$ approximation rate is tight bound for Greedy Set Cover.

3 LP-rounding Set Cover

3.1 Integer Program

We express the Set Cover problem into an integer programs in the form of
Minimize

$$\sum_{i=1}^{k} c(S_i)u_i$$

Subject to

$$\begin{array}{rcl} \sum_{i:e\in S_i} u_i \geq 1 & , & \forall e \in \mathrm{U} \\ u_i \in \{0,1\} & , & i = 1, 2, \cdots, k \end{array}$$

where u_i is the boolean variable shows whether S_i is chosen. S_i is not chosen when $u_i = 0$, while S_i is chosen when $u_i = 1$.

Example 7. $\mathrm{U} = \{1, 2, 3, 4\}$ and the elements and cost of each set is as below

$$\begin{array}{rclrcl} S_1 & = & \{1, 2\} & c(S_1) & = & 5 \\ S_2 & = & \{2, 3\} & c(S_2) & = & 100 \\ S_3 & = & \{2, 3, 4\} & c(S_3) & = & 1 \end{array}$$

We can change the set cover problem above into the integer programs as below
Minimize

$$5u_1 + 100u_2 + u_3$$

Subject to

$$\begin{array}{rcrcl} 1 & : & u_1 & \geq & 1 \\ 2 & : & u_1 + u_2 + u_3 & \geq & 1 \\ 3 & : & u_2 + u_3 & \geq & 1 \\ 4 & : & u_3 & \geq & 1 \end{array}$$

$$u_i \in \{0, 1\} \qquad i \in \{1, 2, 3\}$$

3.2 Relaxing to Linear Program

The above change construct an integer program for the original set cover problem, however, the integer program is still NP-hard.
To get an approximation algorithm, we relax the integral constraints and get a linear program

Integer Program	**Linear Program**
Minimize	**Minimize**
$\sum_{i=1}^{k} c(S_i)u_i$	$\sum_{i=1}^{k} c(S_i)u_i$
Subject to	**Subject to**
$\sum_{i:e\in S_i} u_i \geq 1 \quad , \quad \forall e \in \mathrm{U}$	$\sum_{i:e\in S_i} u_i \geq 1 \quad , \quad \forall e \in \mathrm{U}$
$u_i \in \{0,1\} \quad , \quad i = 1, 2, \cdots, k$	$0 \leq u_i \leq 1 \quad , \quad i = 1, 2, \cdots, k$

We relax the only non-linear constraint for u_i. In this way, we can easily solve this linear program in polynomial time. However, the relax of the constraints bring about new problems: How can we interpret the result vector $u = (u_1, u_2, \cdots, u_k)$ when some u_i are fractions.

3.3 LP-rounding Set Cover

The algorithm first changes the original set cover problem into an integer program, then it relaxes the constraints and thus we obtain a linear program. By solving this linear program, we get a solution $\overline{u} \in \mathbb{R}^k$. Furthermore, we round each element of the result vector using f as below.

Input: element set $\mathrm{U} = \{e_1, e_2, \cdots, e_n\}$, subset set $\mathrm{S} = \{S_1, S_2, \cdots, S_k\}$ and cost function $c : \mathrm{S} \to \mathbb{Z}^+$
Output: set cover C with minimum cost

1 Write the original Set Cover problem into Integer Program;
2 Relax the Integer Program into Linear Program;
3 Solve the Linear Program using the Ellipsoid Algorithm and we obtain $\overline{u} = (u_1, u_2, \cdots, u_k) \in \mathbb{R}^k$;
4 Let f be the maximum frequency (the number of times that element appears in distinct subsets);
5 Output deterministic rounding $\overline{u'} = (u'_1, u'_2, \cdots, u'_k) \in \{0, 1\}^k$, where u'_i satisfies $u'_i = \begin{cases} 1 & , \quad u_i \geq 1/f \\ 0 & , \quad \text{otherwise} \end{cases}$;

Algorithm 2: LP-rounding Set Cover Algorithm

3.4 Effectiveness of LP-rounding

Lemma 8. *LP-rounding Set Cover is an f-approximation.*

Proof. First, we prove that $\overline{u'}$ is a feasible solution. In order to show this, we need to examine each constraints and satisfy it. Suppose

$$u_1 + u_2 + \cdots + u_l \geq 1$$

be one of the constraints. By definition, we have $l \leq f$. So that, there must be some i, $u_i \geq 1/l \geq 1/f$. In this way, u_i will be rounded to 1, and therefore makes this constraint satisfied. Thus, we have proved that the result vector $\overline{u'}$ is a feasible solution.
On the other hand, we look into the rounding process from u_i to u'_i:

$$u'_i = \begin{cases} 1 & , \quad u_i \geq 1/f \\ 0 & , \quad \text{otherwise} \end{cases}$$

We can easily check that $u'_i \leq f \cdot u_i$ for both cases. So that

$$\begin{aligned} \sum_{i=1}^{k} c(S_i) \cdot u'_i &\leq \sum_{i=1}^{k} c(S_i) \cdot f \cdot u_i \\ &= f \sum_{i=1}^{k} c(S_i) \cdot u_i \\ &\leq f \cdot OPT \end{aligned}$$

The first inequality is due to the lemma above, and the second inequality is because when relaxing the constraints enlarge the feasible solution area, and OPT in the Integer Program will be include in the Linear Program.

$$IP \xrightarrow{relax} LP \qquad \longrightarrow \qquad OPT_{IP} \geq OPT_{LP}$$

Above all, LP-rounding Set Cover produces feasible solution for the problem and the approximation rate is f. □

3.5 Vertex Cover

As we proved in section 1.3, Any vertex cover problem can be changed into Set Cover problem.

Example 9. $G = (V, E)$, where $V = \{u_1, u_2, u_3, u_4, u_5\}$ and $E = \{e_1, e_2, \cdots, e_8\}$. We can change the above Vertex Cover problem into a Set Cover problem, where

$$\begin{aligned} \mathrm{U} &= \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\} \\ \mathrm{S} &= \{(e_1, e_2, e_3), (e_1, e_4, e_5), (e_3, e_4, e_6, e_7), (e_2, e_6, e_8), (e_5, e_7, e_8)\} \\ c(S_i) &= 1 \quad , \quad \forall S_i \in \mathrm{S} \end{aligned}$$

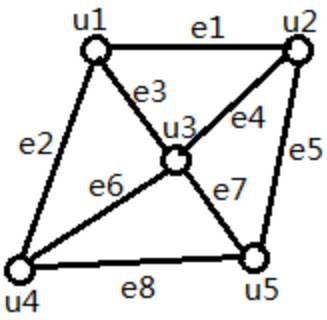

Figure 4: Vertex Cover Example

From the construction, when using LP-rounding Set Cover to solve the vertex cover problem, the maximum frequency is the number of vertices belong to a particular edge, which is 2. Therefore, we have $f = 2$ and using LP-rounding Set Cover, we can get a approximation algorithm with approximation rate 2 for Vertex Cover Problem.