SET COVER APPROXIMATION ALGORITHM

Narasimhan Kovalai
Roll Number-20CS01075
School of Electrical Sciences

March 16, 2022

ABSTRACT : We explore the mathematical aspect of various set cover versions in this report to support the claims made in the presentation. Software like MS Excel is used in this article to solve the Linear Programming Problem.

1 Deriving the Upper Bound Expression

Lemma 1. *For all* $c \geq 0$,

$$(1 - \frac{1}{c})^{c} \leq \frac{1}{e}$$

Proof. From the Taylor's expansion of e^z :

$$e^z = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!}...$$

which results to

$$e^z \geq 1 + z$$

Substituting $z = -\frac{1}{c}$, we get

$$e^{-\frac{1}{c}} = 1 - \frac{1}{c}$$

Raising both sides by power of c, the result is obtained.

$$(1 - \frac{1}{c})^{c} \leq \frac{1}{e}$$

□

We now prove the approximation bound using the above result.

Theorem 1. *Given any set system* $\Sigma = (X, S)$*, let* G *be the output of the greedy cover and let* O *be an optimum cover. Then* $|G| \leq |O| \cdot \ln m$*, where* $m = |X|$.

Proof. Let c be the optimum cover and let g=$c^* - 1$, where c^* is the greedy set cover. We proceed to show that $g \leq c \cdot \ln |m|$. Though this is not the same as expected ($c^* \leq c \cdot \ln |m|$), it provides a reasonably exact upper bound and simplifies calculation.

We now shift our focus to see how many elements are covered in each iteration of the **GREEDY SET COVER APPROXIMATION ALGORITHM**. Initially, there are $m_0 = m$ elements to be covered. Let m_i denote the number of elements remaining to be covered after i iterations of the greedy algorithm. After $i - 1$ iterations, there are m_{i-1} elements remain uncovered.

We know that if there is a cover of size c for these elements (namely, the optimal cover),by Pigeonhole Principle, there exists some set that covers at least

$\frac{m_{i-1}}{c}$ elements. Since the greedy algorithm selects the set covering the largest number of uncovered elements, it is always expected to select a set that covers at least these many elements. The number of elements that remain to be covered is upper bounded by;

$$m_i \leq m_{i-1} - \frac{m_{i-1}}{c} = m_{i-1}(1 - \frac{1}{c})$$

From the above, the reduction factor in the number of remaining elements is given by:

$$r_f = (1 - \frac{1}{c})$$

Thus after i iterations,

$$m_i \leq m(1 - \frac{1}{c})^i$$

Since the **GREEDY SET COVER APPROXIMATION ALGORITHM** ran for $c^* = g + 1$ iterations, then after g iterations, there must be at least one uncovered element. so:

$$1 \leq m_g \leq m(1 - \frac{1}{c})^g = m(1 - \frac{1}{c})^{c \cdot \frac{g}{c}}.$$

Applying the lemma earlier proven, we get :

$$1 \leq m(\frac{1}{e})^{\frac{g}{c}}$$

Multiplying by $e^{\frac{g}{c}}$ on both sides and taking natural logarithm, we get the desired bound:

$$g \leq c \cdot \ln |m|$$

A more or less similar treatment is done for weighted set cover problem. □

2 Proving that Set Cover Problem is NP-Complete

Proving that the set cover is NP-Complete involves two steps:

- Set cover Problem is NP.
- Set cover Problem is NP-hard.(every other problem in the NP class is reducible to a Set Cover instance.)

The first criteria is trivial to prove. If we are given a certificate solution Ψ with $|\Psi| = k$ to the set cover problem, we perform the following algorithm:

Algorithm 1 VERIFICATION OF SET COVER SOLUTION(Ψ, X)

```
1: M=Ψ
2: while M ≠ Φ do
3:     Mark the elements of S_i ∈ M as covered in the universal set X
4:     M = M − S_i
5: end while
6: if x.cover==True for each x ∈ X then
7:     return True
8: end if
9: return False
```

Clearly, for a certificate solution, the algorithm terminates at line 7 and hence the set cover would be NP in nature.

It remains to be shown that the problem is NP-hard. For this, we will reduce the instance of a vertex cover problem to an instance of the set cover problem, since we know that vertex cover problem is a proven NP-Complete problem(thus both NP and NP-hard).

We can reduce any instance of the decision vertex cover for $G = (V, E)$ with an integer parameter k by letting $X = E$ and letting the family of sets $F = (S_v | v \in V \cap \deg(v) > 0)$, such that S_v contains all edges in E adjacent to v

We will now prove that for any vertex cover V' for G and a set cover C for$X$ and E respectively, the following holds:

$$\exists V' \subseteq V : |V'| \leq k \iff \exists C \subseteq F : |C| \leq K$$

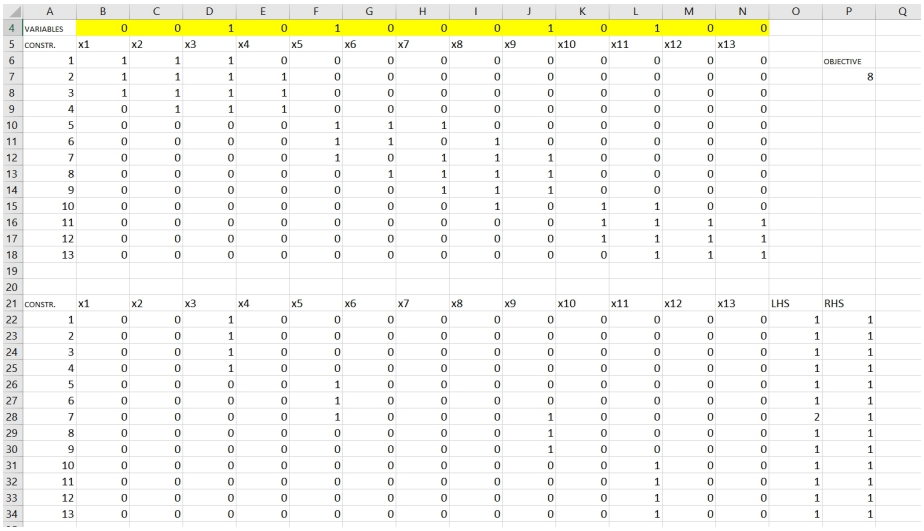
Since $X = E$, and we know V' is a vertex cover for $G = (V, E)$ we can find C for (X, F) by selecting from F all sets corresponding to $v' \in V'$

Similarly, if C exists, by the way we defined the reduction, we know it covers $X = E$. This means we can find V' by taking the vertices corresponding to $S_v \in C$. Due to the 1:1 correspondence of vertices in V' and sets in C: $|V'| = |C|$.

So set cover problem is NP-hard as well. Hence, it is proved that it is **NP-Complete** in nature.

3 Weighted Set cover Problem using Linear Programming

We can use the simplex algorithm to solve the linear programming problem of the weighted set cover problem. An excel solver has been used for this purpose and the results are attached below.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
4	VARIABLES		0	0	1	0	1	0	0	0	1	0	1	0	0		
5	CONSTR.	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13			
6	1	1	1	1	0	0	0	0	0	0	0	0	0	0		OBJECTIVE	
7	2	1	1	1	1	0	0	0	0	0	0	0	0	0		8	
8	3	1	1	1	1	0	0	0	0	0	0	0	0	0			
9	4	0	1	1	1	0	0	0	0	0	0	0	0	0			
10	5	0	0	0	0	1	1	1	0	0	0	0	0	0			
11	6	0	0	0	0	1	1	0	1	0	0	0	0	0			
12	7	0	0	0	0	1	0	1	1	1	0	0	0	0			
13	8	0	0	0	0	0	1	1	1	1	0	0	0	0			
14	9	0	0	0	0	0	0	1	1	1	0	0	0	0			
15	10	0	0	0	0	0	0	0	1	0	1	1	0	0			
16	11	0	0	0	0	0	0	0	0	0	1	1	1	1			
17	12	0	0	0	0	0	0	0	0	0	1	1	1	1			
18	13	0	0	0	0	0	0	0	0	0	0	1	1	1			
19																	
20																	
21	CONSTR.	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	LHS	RHS	
22	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	
23	2	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	
24	3	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	
25	4	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	
26	5	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	
27	6	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	
28	7	0	0	0	0	1	0	0	0	1	0	0	0	0	2	1	
29	8	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	
30	9	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	
31	10	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	
32	11	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	
33	12	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	
34	13	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	

Figure 1: Figure showing the results of the excel solver

Important Observations :

- The optimum set cover chooses S_3, S_5, S_9, S_11. Thus the cost is $2 + 4 + 0 + 2 = 8$.
- The approximation ratio is $10/8 = 1.25$ which is upper bounded by $\ln 13 = 2.565$.
- If a tie occurs between 2 ratios, we select the subset with the lower individual cost.

4 Linear Greedy Set Cover Algorithm

We provide a linear time algorithm without its proof to avoid complex mathematics involved.

Algorithm 2 LINEAR GREEDY SET COVER

```
compute the sizes of every S ∈ F, storing them in S.size. let A be an array of length max_S|S|, consisting of empty lists
for S ∈ F do
    add S to A[S.size]
end for
let A.max be the index of the largest nonempty list in A.
let L be an array of length |∩_S ∈ FS| consisting of empty lists
for S ∈ F do
    for l ∈ S do
        add S to L[l]
    end for
end for
let C be the set cover that we will be selecting, initially empty.
let T be the set of letters that have been covered, initially empty.
while A.max > 0 do
    Let S_0 be any element of A[A.max].
    add S_0 to C
    remove S_0 from A[A.max]
    for l ∈ S_0 do
        for S ∈ L[l] do
            Remove S from A[A.size]
            S.size = S.size − 1
            Add S to A[S.size]
            if A[A.max]isEmpty then
                A.max=A.max-1
            end if
        end for
        add l to T
    end for
end while
```

5 CONCLUSION

We thus proved the approximation ratio of the Set-Cover Problem to be $ln|X|$ and also proved that the decision vertex of the problem is NP-Complete in nature. We used Excel Solver to solve the Linear Programming Problem to find the optimal solution. Then, we cited a linear greedy set cover algorithm which further improvises the algorithm. Hence, an exhaustive analysis of a practical NP-Complete problem has been presented.