Greedy Approximation Algorithm for Set Cover[1]

- In the set cover problem, we are given a universe U of n elements, and a collection of subsets $\{S_1, \ldots, S_m\}$ of the universe, the goal is to pick the smallest number of sets from among this collection so that their union is the whole universe. More precisely, we need to find[2] $I \subseteq [m]$ such that $\bigcup_{j \in I} S_i = U$. Note that we may assume $\bigcup_{i=1}^{m} S_i = U$, otherwise we can simply answer no solution.

  Remark: *There is a weighted generalization where each subset S_i also has a cost $c(S_i)$ and the objective is to minimize the total cost of the sets picked. We will also consider this version.*

- The Set Cover problem is one of the ***canonical*** problems in approximation algorithms. It generalizes many different problems and arises in many applications. One example is task assignment : you have n jobs and m workers, and worker i can perform the subset S_i of the jobs. What is the minimum number of workers you need to complete all jobs? Another is the vertex cover problem : the universe now is the collection of edges in a graph $G = (V, E)$, and the $|V|$ subsets S_v, corresponding to vertex v, is the subset of edges incident on v. The set cover problems boils down to : what is the minimum number of vertices C such that every edge is incident on at least one vertex in C? Since vertex cover is NP-hard, so is set cover.

  Exercise: ☕ *Given an undirected graph $G = (V, E)$, a **dominating set** U is a subset of V such that every vertex of V is either in U or has a neighbor in U. Explain why the minimum cardinality dominating set is a set cover problem.*

  Set Cover is also canonical in that many algorithmic ideas from approximation algorithms can be illustrated using this problem. It is also one of the oldest problems for which approximation algorithms were studied. And indeed the most natural algorithm for the problem turns out to be pretty good.

- **The Greedy Algorithm.**

```
1: procedure GREEDY SET COVER(U, (S_1, ..., S_m)):
2:     Initialize X ← U. ▷ X will denote the collection of uncovered elements.
3:     Initialize I ← ∅. ▷ I denotes the indices of the sets picked in our solution.
4:     while X ≠ ∅ do:
5:         Pick j ∈ [m] which maximizes |S_j ∩ X|.
6:         ▷ That is, the set which covers the maximum number of uncovered elements.
7:         I ← I ∪ j.
8:         X ← X \ S_j.
9:     return I.
```

[1] *Lecture notes by Deeparnab Chakrabarty. Last modified : 10th Jan, 2022*
These have not gone through scrutiny and may contain errors. If you find any, or have any other comments, please email me at deeparnab@dartmouth.edu. Highly appreciated!

[2] $[m]$ denotes $\{1, 2, \ldots, m\}$.

Theorem 1. GREEDY SET COVER is a $(1 + \ln n)$-approximation algorithm for the set cover problem.

Proof. Fix an instance $(U, (S_1, \ldots, S_m))$ with $|U| = n$. Let $O \subseteq [m]$ be the optimum solution, and let $k = |O|$. Let I denote the set cover returned by the algorithm. We need to prove $|I| \leq (1 + \ln n)k$. In fact, we prove something stronger. We show that if $n > 1$ then $|I| \leq \lceil k \ln n \rceil$ which is $\leq (1 + \ln n)k$. When $n = 1$, then clearly both $|I| = |O| = 1$ and the theorem vacuously holds.

To this end, let x_j denote the number of set of uncovered elements *before* the jth while loop, and let n_j denote the number of elements we cover in the jth while loop. In other words, $x_j = |X|$ before the jth while loop, and $n_j = |S_j \cap X|$ where S_j is the set the greedy algorithm picks in the jth while loop. Note that $|I|$ is the number of while loops. Now, the x_j and n_j's satisfy the following.

$$x_1 = n; \quad x_{j+1} = x_j - n_j; \quad n_j \geq \frac{x_j}{k} \tag{1}$$

The first two follow from definition. The third is where we use the "greediness" of the algorithm and is key to the analysis. Why is it true? Well, x_j is the number of uncovered elements at the beginning of loop j. We know that the sets of the optimal set cover indexed by O covers all these elements. Therefore, some S_t for $t \in O$ must cover at least $\frac{1}{k}$th of these elements. That is, there exists $t \in O$ with $|S_t \cap X| \geq \frac{x_j}{k}$. The algorithm picks S_j with $|S_j \cap X| \geq |S_t \cap X|$, implying $n_j \geq x_j/k$.

The rest of the analysis is "playing around" with (1). We claim the following, and the interested reader may want to prove this on their own.

Claim 1. For any $1 \leq j \leq |I|$, $x_j \leq n \cdot \left(1 - \frac{1}{k}\right)^{j-1}$.

We leave the proof of the claim as an exercise. We now explain why the claim implies $|I| \leq \lceil k \ln n \rceil$. Suppose not, and say $|I| > \lceil k \ln n \rceil$. Since $|I|$ is an integer, this implies $|I| \geq \lceil k \ln n \rceil + 1 \geq k \ln n + 1$. At the beginning of the $\ell := |I|$th loop, by Claim 1, we have

$$x_\ell \leq n \cdot \left(1 - \frac{1}{k}\right)^{k \ln n} < n \cdot e^{-\ln n} = 1$$

where we have used the oft used and very important inequality "for every real $z \neq 0$, $(1 + z) < e^z$" with $z = -1/k$. Since $n > 1$, we have $k \ln n > 0$. However, $x_\ell < 1$, which implies $x_\ell = 0$ since x_ℓ is an integer, is a contradiction : that would mean $X = \emptyset$ at the beginning of the while loop, and so the loop would terminate. □

Exercise: ☕ *Show that one can analyze a bit better : argue that* $|I| \leq k \cdot (1 + \ln(n/k))$. *Furthermore, use this to argue that if every set* $|S_i| \leq d$, *then* GREEDY SET COVER *is in fact an* $(1 + \ln d)$*-approximation algorithm.*

Exercise: ☕ *Prove Claim 1 using induction and* (1).

Exercise: ☕☕ *The* MAX-COVERAGE *problem has the same input as set cover, but with an extra parameter k which is a positive integer in $[m]$. The objective is to pick k sets, indexed by I with $|I| = k$ such that $|\bigcup_{j \in I} S_j|$ is maximized. Describe a natural greedy algorithm for this problem and prove that is an $(1 - \frac{1}{e})$-approximation.*

- Tightness of Analysis. Whenever one analyses an algorithm, one needs to also wonder if one can do better. In approximation algorithms, where inequalities abound, one could wonder perhaps some inequality is "loose", and perhaps with some more cleverness one can say something better. For instance, the $\ln n$-factor is loose in the sense that the above exercise could improve to $1 + \ln d$-factor which is way better when the subsets are of small size. Can one do even better? Unfortunately, the answer is in the negative. The following picture illustrates this.

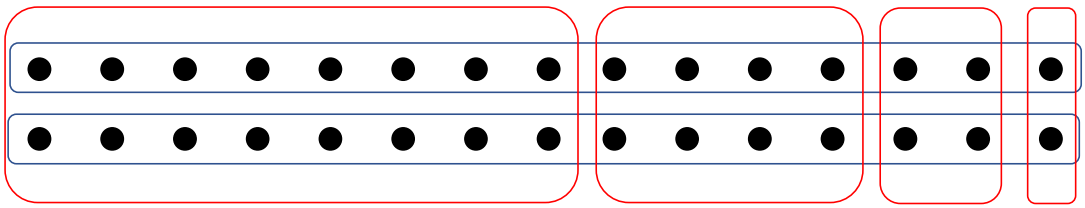

Figure 1: *The optimum solution is to pick the 2 blue horizontal sets. The greedy algorithm picks the 4 red "fatter" sets. Do you see why? This shows that the approximation factor of the algorithm is at least 2. Now consider the situation where we have $2^\ell - 1$ points in each row. The optimum solution still picks the two horizontal sets, but the algorithm would end up picking ℓ sets. This shows that the factor is at least $\frac{\log_2(1+d)}{2}$. Now consider what happens when there are k-horizontal lines. How would the construction change?*

The Weighted Version

- Suppose now that every set S_j also had a cost $c(S_j)$, and the objective was to pick the minimum *cost* set cover. One then modifies the above algorithm in the obvious way : in the while loop, instead of picking the set which maximizes the number of new elements covered, one picks the set which has the best "bang-for-the-buck", that is, maximizes the number of new elements covered divided by the cost of the set. For simplicity of the analysis, we describe the algorithm using the reciprocal.

```
1: procedure WEIGHTED GREEDY SET COVER(U, (S_1, ..., S_m) + costs):
2:     Initialize X ← U. ▷ X will denote the collection of uncovered elements.
3:     Initialize I ← ∅. ▷ I denotes the indices of the sets picked in our solution.
4:     while X ≠ ∅ do:
5:         Pick j ∈ [m] which minimizes c(S_j) / |S_j ∩ X|.
6:         I ← I ∪ j.
7:         X ← X \ S_j.
8:     return I.
```

- **Analysis.** Suppose the number of loops is r, and let's rename the sets such that S_i is the set the algorithm picks in loop i. We use alg to denote $\sum_{i=1}^{r} c(S_i)$ the cost of the algorithm. Let's call the sets picked in the optimum set cover $O_1, \ldots, O_\ell$, and so $\mathtt{opt} = \sum_{j=1}^{\ell} c(O_j)$. We use X_i to denote the set of uncovered elements just before loop i. Thus, $X_1 = U$ and $X_{r+1} = \emptyset$. So, $S_i \cap X_i$ is the set of elements covered at iteration i, and this is precisely $X_i \setminus X_{i+1}$.

Theorem 2. WEIGHTED GREEDY SET COVER is an H_n-approximation algorithm, where $H_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n}$ is the nth Harmonic number.

Proof. Greedy choice tells us for all loops $i \in [r]$, we have

$$\forall j \in [\ell], \quad \frac{c(S_i)}{|S_i \cap X_i|} \leq \frac{c(O_j)}{|O_j \cap X_i|} \tag{2}$$

This is because each set O_j was a possible choice in the ith loop, but the algorithm picked S_i instead.

And now we do an "averaging trick" which seems pretty slick, but one gets used to it. We use the fact that for any two rational numbers $\frac{a}{b}, \frac{c}{d}$, the rational number $\frac{a+c}{b+d}$ is *at least* $\min\left(\frac{a}{b}, \frac{c}{d}\right)$. Applying this on the RHS of (2), we get that

$$\forall i \in [r], \quad \frac{c(S_i)}{|S_i \cap X_i|} \leq \frac{\sum_{j=1}^{\ell} c(O_j)}{\sum_{j=1}^{\ell} |O_j \cap X_i|} \leq \frac{\mathtt{opt}}{|X_i|} \tag{3}$$

The last inequality follows since O_j's form a cover and thus, $\bigcup_{j=1}^{\ell} (O_j \cap X_i) = X_i$. Taking the $|S_i \cap X_i|$ to the right hand side and adding over all i we get

$$\begin{aligned}
\mathsf{alg} = \sum_{i=1}^{r} c(S_i) &\underbrace{\leq}_{(3)} \mathtt{opt} \cdot \sum_{i=1}^{r} \frac{|S_i \cap X_i|}{|X_i|} \\
&\underbrace{=}_{\text{since } S_i \cap X_i = X_i \setminus X_{i+1}} \mathtt{opt} \cdot \sum_{i=1}^{r} \frac{|X_i| - |X_{i+1}|}{|X_i|} \\
&\leq \mathtt{opt} \cdot \left(\frac{1}{|U|} + \frac{1}{|U|-1} + \cdots + 1 \right) \qquad (4) \\
&= \mathtt{opt} \cdot H_n
\end{aligned}$$

where (4) is another simple inequality which follows by noticing that the worse case occurs when each new set covers exactly one element. □

Exercise: ☕ *Prove* (4).

- **A Different and Better Analysis.** We now show a slightly different way of analyzing the above algorithm which illustrates an analysis tool called the *charging trick* in the parlance. This analysis also gives a better factor. Let $d := \max |S_i|$ be the size of the largest cardinality set in the collection.

Theorem 3. WEIGHTED GREEDY SET COVER is an H_d-approximation algorithm

Proof. Once again let $\{S_1, \ldots, S_r\}$ be the sets picked by the algorithm in that order. Recall that we pick set S_i in iteration i because $\frac{c(S_i)}{|S_i \cap X_i|} \leq \frac{c(S)}{|S \cap X_i|}$ for any set S_t. However, when we pick the set S_i in our solution, the cost increases by $c(S_i)$ and not the ratio. Somehow, we need to argue that although we are locally optimizing for the ratio, we can still upper bound the numerator.

To this end, we do the following "charging trick." For every element $j \in S_i \cap X_i$, that is, each new element covered by S_i, we assign this element a charge $\alpha_j = \frac{c(S_i)}{|S_i \cap X_i|}$. Note that $c(S_i)$, by design, is $\sum_{j \in S_i \cap X_i} \alpha_j$. Now, at the end of the algorithm, every element will be charged once and only once, and furthermore

$$\mathsf{alg} = \sum_{j \in U} \alpha_j$$

To see why this is helpful, pick any set O in the input. Suppose O has $d_O \leq d$ elements, and rename them to be $\{1, 2, \ldots, d_O\}$ *in the order* in which they were covered by the greedy algorithm. This is important.

Now take any element $j \in O$. What do we know about α_j? Well, when this element j was being covered by our algorithm by some set S_i in iteration i, we had the choice of picking O. Furthermore, by design, at this iteration none of the elements in $\{j, j+1, \ldots, d_O\}$ were covered. That is, $\{j, j+1, \ldots, d_O\} \subseteq X_i$, where X_i is the set of uncovered elements at the beginning of this loop. Since we picked S_i instead of O, we have that the cost-to-new-elements-covered ratio of S_i is at most that of O. And since the former is the charge on the element j, we get

$$\alpha_j \leq \frac{c(O)}{|O \cap X_i|} \leq \frac{c(O)}{d_O - j + 1} \underbrace{\Rightarrow}_{\text{summing over } j = 1 \text{ to } d_O} \sum_{j \in O} \alpha_j \leq c(O) \cdot H_{d_O} \leq c(O) \cdot H_d \qquad (5)$$

In other words, the charges that the greedy algorithm induces on the elements has the following property : for any set O of the input, the total charge on the elemnts in O is at most H_d times more than the cost of the set.

Now, let $\{O_1, \ldots, O_\ell\}$ be the sets picked by the optimal set cover. Using the fact that it is a cover, we can finish the proof of our theorem as follows.

$$\mathsf{alg} = \sum_{j \in U} \alpha_j \underbrace{\leq}_{O_i\text{'s form a cover}} \sum_{i=1}^{\ell} \sum_{j \in O_i} \alpha_j \underbrace{\leq}_{(5)} \sum_{i=1}^{\ell} c(O_i) \cdot H_d = \mathtt{opt} \cdot H_d$$

□

Notes

The set cover problem is *the* classic approximation algorithms problem. The analysis of the greedy algorithm is present in the papers [4] by Johnson and [5], while the weighted set cover result is from the paper [1] by Chvatal. On the other hand, obtaining an $(1 - \varepsilon) \ln n$-approximation for any constant $\varepsilon > 0$ would imply $P = NP$. These result is present in the papers [6], [3], and [2] by Lund and Yannakakis who showed an $\Omega(\ln n)$ hardness, Feige who showed this under a stronger complexity theoretic assumption, and Dinur and Steurer, who obtained the final result. See [7] for a short survey on the set cover problem.

References

[1] V. Chvátal. A greedy heuristic for the set covering problem. *Math. Oper. Res.*, 4:233–235, 1979.

[2] I. Dinur and D. Steurer. Analytical approach to parallel repetition. In *Proc., ACM Symposium on Theory of Computing (STOC)*, pages 624–633, 2014.

[3] U. Feige. A Threshold of ln n for Approximating Set Cover. *Journal of the ACM*, 45, 1998.

[4] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.*, 9:256–278, 1974.

[5] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete mathematics*, 13(4):383–390, 1975.

[6] C. Lund and M. Yannakakis. On the Hardness of Approximating Minimization Problems. *Journal of the ACM*, 41(5):960–981, Sept. 1994.

[7] N. E. Young. Greedy set-cover algorithms. *Encyclopedia of algorithms*, pages 379–381, 2008.