Advanced Algorithms	18 September 2018

Lecture 1: Approximation Algorithms I

Lecturer: Mohsen Ghaffari *Scribe: Davin Choo*

1 Approximation algorithms

Unless $\mathbb{P} = \mathbb{NP}$, we do not expect efficient algorithms for $\mathbb{NP}$-hard problems. However, we are often able to design efficient algorithms that give solutions that are provably close/approximate to the optimum . We next formalize this.

Definition 1 (α-approximation)**.** *An algorithm $\mathcal{A}$ is an α-approximation algorithm for a* minimization *problem with respect to a cost metric c if for any problem instance I and for some optimum algorithm OPT, $c(\mathcal{A}(I)) \leq \alpha \cdot c(OPT(I))$.*

Remark Maximization problems are defined similarly with $c(\mathcal{A}(I)) \geq \alpha \cdot c(OPT(I))$.

2 Minimum set cover

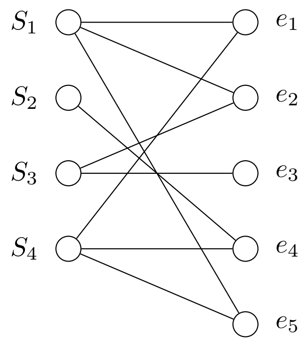
Consider a universe $\mathcal{U} = \{e_1, \ldots, e_n\}$ of n elements, a collection of subsets $\mathcal{S} = \{S_1, \ldots, S_m\}$ of m subsets of $\mathcal{U}$ such that $\mathcal{U} = \bigcup \mathcal{S}$, and a non-negative cost function $c : \mathcal{S} \to \mathbb{R}^+$. Suppose $S_i = \{e_1, e_2, e_5\}$, then we say that S_i *covers* elements e_1, e_2, and e_5. For any subset $T \subseteq \mathcal{S}$, we define $c(\bigcup_{S_i \in T} S_i) = \sum_{S_i \in T} c(S_i)$.

Definition 2 (Minimum set cover problem)**.** *Given $\mathcal{U}$, $\mathcal{S}$, and $c : \mathcal{S} \to \mathbb{R}^+$, find a subset $S^* \subseteq \mathcal{S}$ such that:*

(i) (Set cover): $\bigcup_{S_i \in S^*} S_i = \mathcal{U}$

(ii) (Minimum cost): $c(S^*)$ is minimized.

Example



In this example, there are $n = 5$ vertices and $m = 4$ subsets $\mathcal{S} = \{S_1, S_2, S_3, S_4\}$. Suppose the cost function is defined as $c(S_i) = 2^i$. Even though $S_3 \cup S_4$ covers all vertices, it costs $c(S_3 \cup S_4) = c(S_3) + c(S_4) = 9 + 16 = 25$. One can verify that the minimum set cover is $S^* = \{S_1, S_2, S_3\}$ with a cost of $c(S^*) = 14$. Notice that we want a minimum cover with respect to c and not the number of subsets chosen from $\mathcal{S}$ (unless c is uniform cost).

2.1 A greedy minimum set cover algorithm

Minimum set cover is known to be **NP**-complete, hence we are interested in algorithms that give us a good approximation for the optimum. In this section, we describe a greedy algorithm and prove that it is a H_n-approximate algorithm.

Algorithm 1 (cite?) is a greedy set cover algorithm. The intuition is as follows: Spread the cost $c(S_i)$ amongst the vertices that are newly covered by S_i. The algorithm then greedily selects the set that has the lowest price-per-item.

Algorithm 1 GreedySetCover($\mathcal{U}, \mathcal{S}, c$)

$T \leftarrow \emptyset$ ▷ Selected subset of $\mathcal{S}$
$C \leftarrow \emptyset$ ▷ Covered vertices
while $C \neq \mathcal{U}$ **do**
  $S_i \leftarrow \arg\min_{S_i \in S \setminus T} \frac{c(S_i)}{|S_i \setminus C|}$ ▷ Pick the set with the lowest price-per-item
  $T \leftarrow T \cup \{S_i\}$ ▷ Add S_i to selection
  $C \leftarrow C \cup S_i$ ▷ Update covered vertices
end while
return T

Consider a run of Algorithm 1 on the earlier example. On the first iteration, price-per-item(S_1) = 2/3, price-per-item(S_2) = 4, price-per-item(S_3) = 9/2, and price-per-item(S_4) = 16/3; So, S_1 is chosen. On the second iteration, price-per-item(S_2) = 4, price-per-item(S_3) = 9, and price-per-item(S_4) = 16; So, S_2 was chosen. In the third iteration, price-per-item(S_3) = 9, and price-per-item(S_4) = ∞; so S_3 was chosen. Since all vertices are now covered, the algorithm terminates (coincidentally to the minimum set cover). Notice that the price-per-item for the remaining sets change according to which vertices remain uncovered. Furthermore, one can simply ignore S_4 when it was no longer covers any uncovered vertices.

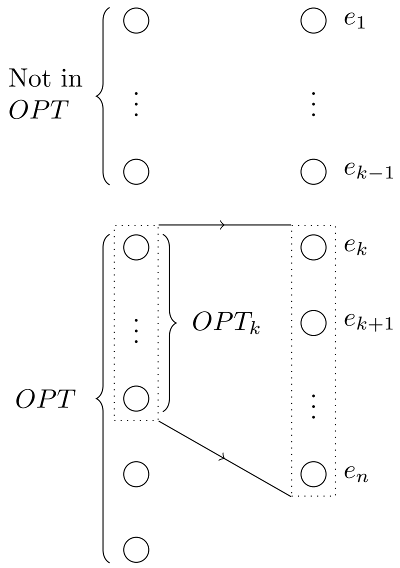
Theorem 3. *Algorithm 1 gives us a H_n-approximation for minimum set cover.*

Proof. Since $\mathcal{U} = \bigcup \mathcal{S}$, by the termination condition of algorithm 1, the output T is a valid set cover.

Consider any *fixed* minimum set cover OPT. It remains to show that $c(T) \leq H_n \cdot c(OPT)$. Let $e_1, \ldots, e_n$ be the elements in the order they are covered by algorithm 1. Define price(e_i) as the price-per-item of the set that covered e_i during the run of the algorithm.

Consider the moment in the algorithm where elements $e_1, \ldots, e_{k-1}$ are already covered. Since there is a cover of cost at most $c(OPT)$ for the remaining $n-k+1$ elements, then there must be an element whose price is at most $\frac{c(OPT)}{n-k+1}$. We formalize this intuition with the argument below.

Since OPT is a set cover, there exists a subset of $OPT_k \subset OPT$ that covers $e_k \ldots e_n$.



Suppose $OPT_k = \{O_1, \ldots, O_p\}$. We know the following:

1. $O_1, \ldots, O_p \in \mathcal{S} \setminus T$. Otherwise, some element in $e_k, \ldots, e_n$ would have been covered.
2. $n - k + 1 = |\mathcal{U} \setminus C| \leq |O_1 \cap (\mathcal{U} \setminus C)| + \cdots + |O_p \cap (\mathcal{U} \setminus C)|$, because some elements may be covered more than once.
3. By definition, for each $j \in \{1, \ldots, p\}$, price-per-item(O_j) = $\frac{c(O_j)}{|O_j \cap (\mathcal{U} \setminus C)|}$.

Since the greedy algorithm will pick a set in $\mathcal{S} \setminus T$ with the lowest price-per-item, price(e_k) $\leq$ price-per-item(O_j) for all $j \in \{1, \ldots, p\}$. Hence,

$$c(O_j) \geq \text{price}(e_k) \cdot |O_j \cap (\mathcal{U} \setminus C)|, \forall j \in \{1, \ldots, p\} \quad (1)$$

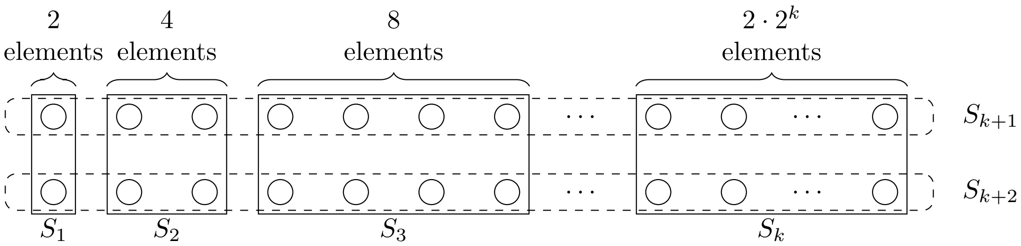
Summing over all p sets, we have: $c(OPT) \geq c(OPT_k) = \sum_{i=1}^{p} c(O_i) \geq \text{price}(e_k) \cdot \sum_{j=1}^{p} |O_j \cap (\mathcal{U} \setminus C)| \geq \text{price}(e_k) \cdot |\mathcal{U} \setminus C| = \text{price}(e_k) \cdot (n-k+1)$, where the second inequality is due to Equation (1). Rearranging, we get: $\text{price}(e_k) \leq \frac{c(OPT)}{n-k+1}$. Summing over all elements, we have:

$$c(T) = \sum_{S \in T} c(S) = \sum_{k=1}^{n} \text{price}(e_k) \leq \sum_{k=1}^{n} \frac{c(OPT)}{n-k+1} = c(OPT) \sum_{k=1}^{n} \frac{1}{k} = c(OPT) \cdot H_n$$

The second equality is because the cost of sets is partitioned into the price$(\cdot)$ of all n vertices. □

Tight bound example for algorithm 1 During lecture, it was mentioned, without an explicit example, that the bound is tight. We construct the example here.

Note that $H_n = \ln(n) + \gamma \leq \ln(n) + 0.6 \in \mathcal{O}(\log(n))$, where γ is the Euler-Mascheroni constant[1]. Consider the following setup with $n = 2 \cdot (2^k - 1)$ elements, for some $k \in \mathbb{N} \setminus \{0\}$. Partition the elements into groups of size $2 \cdot 2^0, 2 \cdot 2^1, 2 \cdot 2^2, \ldots, 2 \cdot 2^{k-1}$. Let $\mathcal{S} = \{S_1, \ldots, S_k, S_{k+1}, S_{k+2}\}$. For $1 \leq i \leq k$, let S_i cover the group of size $2 \cdot 2^{i-1} = 2^i$. Let S_{k+1} and S_{k+2} cover half of each group (i.e. $2^k - 1$ elements each).



Suppose $c(S_i) = 1, \forall i \in \{1, \ldots, k+2\}$. The greedy algorithm will pick S_k, then S_{k-1}, ..., and finally S_1. This is because $2 \cdot 2^k > n/2$ and $2 \cdot 2^i > (n - \sum_{j=i+1}^{k} 2 \cdot 2^j)/2$, for $1 \leq i < k$. This greedy set cover costs $k = \mathcal{O}(\log(n))$. On the other hand, the minimum set cover is $S^* = \{S_{k+1}, S_{k+2}\}$ with a cost of 2.

A series of works by Lund and Yannakakis [LY93], Feige [Fei98], and Moshkovitz [Mos15] showed that it is **NP**-hard to always approximate set cover to within $(1 - \epsilon) \ln |\mathcal{U}|$, for any constant $\epsilon > 0$.

Theorem 4 ([Mos15]). *It is **NP**-hard to always approximate set cover to within $(1 - \epsilon) \ln |\mathcal{U}|$, for any constant $\epsilon > 0$.*

Proof. See [Mos15] □

2.2 Special cases

In this section, we show that one may improve the approximation factor from H_n if we have further assumptions on the set cover instance. Define $\Delta = max_{i \in \{1,\ldots,m\}} \text{degree}(S_i)$ and $f = max_{i \in \{1,\ldots,m\}} \text{degree}(e_i)$. Consider the following two special cases of set cover instances:

1. Δ is small. i.e. All sets are small.
2. f is small. i.e. There is a small number of sets that cover any fixed element.

2.2.1 Small Δ

Theorem 5. *Algorithm 1 gives us a H_Δ-approximation for minimum set cover.*

Proof. Suppose $OPT_k = \{O_1, \ldots, O_p\}$. Consider a set $O_i = \{e_{i,1}, \ldots, e_{i,d}\}$ with degree$(O_i) = d \leq \Delta$. Without loss of generality, suppose that the greedy algorithm covers $e_{i,1}$, then $e_{i,2}$, and so on. For $1 \leq k \leq d$, when $e_{i,k}$ is covered, $\text{price}(e_{i,k}) \leq \frac{c(O_i)}{d-k+1}$ (The inequality could possibly be equal and O_i could be chosen by the greedy algorithm, covering $e_{i,k}, \ldots, e_{i,d}$). Hence, the greedy cost of covering elements in O_i (i.e. $e_{i,1}, \ldots, e_{i,d}$) is at most $\sum_{k=1}^{d} \frac{c(O_i)}{d-k+1} = c(O_i) \cdot \sum_{k=1}^{d} \frac{1}{k} = c(O_i) \cdot H_d \leq c(O_i) \cdot H_\Delta$. Summing over all p sets to cover all n elements, we have $c(T) \leq H_\Delta \cdot c(OPT)$. □

[1] https://en.wikipedia.org/wiki/Euler-Mascheroni_constant

Remarks We apply the same greedy algorithm for small Δ but analyzed in a more localized manner. Crucially, in our analysis, we always work with the exact degree d and only use the fact $d \leq \Delta$ *after* summation. Observe that $\Delta \leq n$ and the approximation factor equals that of Theorem 3 when $\Delta = n$.

2.2.2 Small f

We first look at the case when $f = 2$, show that it is related to another graph problem, then generalize the approach for general f.

Vertex cover as a special case of set cover

Definition 6 (Minimum vertex cover problem)**.** *Given a graph $G = (V, E)$, find a subset $S \subseteq V$ such that:*

(i) (Vertex cover): $\forall e = (u, v) \in E$, $u \in S$ or $v \in S$

(ii) (Minimum cost): $|S|$ is minimized

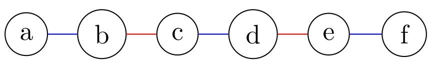
When $f = 2$ and $c(S_i) = 1, \forall S_i \in \mathcal{S}$, the minimum set cover problem is essentially a minimum vertex cover problem — Each element is an edge with endpoints being the two sets that cover it. One way to obtain a 2-approximation to minimum vertex cover (and hence 2-approximation for this special case of set cover) is to use a maximal matching.

Definition 7 (Maximal matching problem)**.** *Given a graph $G = (V, E)$, find a subset $M \subseteq E$ such that:*

(i) (Matching): $\forall e_i, e_j \in M$, edges e_i and e_j do not share an endpoint.

(ii) (Maximal): $\forall e_k \notin M$, adding $M \cup \{e_k\}$ is not a matching (violates first property).

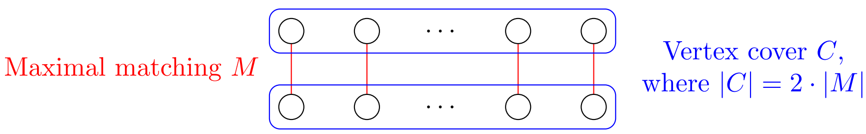
A related concept to maximal matching is maxi*mum* matching, where one tries to maximize the set of M. By definition, any maximum matching is also maximal matching, but the converse is not necessarily true. Consider the line graph of 6 vertices and 5 edges below. Both the set of blue edges $\{(a, b), (c, d), (e, f)\}$ and the set of red edges $\{(b, c), (d, e)\}$ are valid maximal matchings, where the maximum matching is the former.



Algorithm 2 GreedyMaximalMatching(V, E)

$M \leftarrow \emptyset$ ▷ Selected edges
$C \leftarrow \emptyset$ ▷ Set of incident vertices
while $E \neq \emptyset$ **do**
  $e_i = (u, v) \leftarrow$ Pick any edge from E
  $M \leftarrow M \cup \{e_i\}$ ▷ Add e_i to the matching
  $C \leftarrow C \cup \{u, v\}$ ▷ Add endpoints to incident vertices
  Remove all edges in E that are incident to u or v
end while
return M

Algorithm 2 is a greedy maximal matching algorithm. The algorithm greedily adds any available edge e_i that is not yet incident to M, then exclude all edges that are adjacent to e_i.



Theorem 8. *The set of incident vertices C in Algorithm 2 is a 2-approximation for minimum vertex cover.*

Proof. Suppose, for a contradiction, that C is not a vertex cover. Then, there exists an edge $e = (u, v)$ such that $u \notin C$ and $v \notin C$. If such an edge exists, it would not be removed from E during in the greedy algorithm. This is a contradiction, hence C is a vertex cover.

Consider the matching M. Any vertex cover has to include either endpoints, hence the minimum vertex cover OPT has at least $|M|$ vertices. By picking C as our vertex cover, $|C| = 2 \cdot |M| \leq 2 \cdot |OPT|$. Therefore, C is a 2-approximation. □

We now generalize beyond $f = 2$ by considering hypergraphs. Hypergraphs are a generalization of graphs in which an edge can join any number of vertices. Formally, a hypergraph $H = (X, E)$ consists of a set of vertices/elements X and a set of hyperedges E where each hyperedge is a non-empty subset of $\mathcal{P}(X)$, the powerset of X. The minimum vertex cover problem and maximal matching problems are defined similarly on a hypergraph.

Remark A hypergraph $H = (X, E)$ can be viewed as a bipartite graph where the partitions X and E respectively and the edges are between element $x \in X$ and hyperedge $e \in E$ if $x \in e$.

Example Suppose $H = (X, E)$ where $X = \{a, b, c, d, e\}$ and $E = \{\{a, b, c\}, \{b, c\}, \{a, d, e\}\}$. A minimum vertex cover of size 2 would be $\{a, e\}$ (there are multiple size 2 vertex covers). Maximal matchings would be $\{\{a, b, c\}\}$ and $\{\{b, c\}, \{a, d, e\}\}$, where the latter is the maximum matching.

Claim 9. *For general f, we can find a f-approximation for minimum vertex cover.*

Sketch of Proof

- Greedily compute a maximal matching in the hypergraph, removing any edge involving vertices that appear in the hyperedge of the greedy selection.
- Let C be the set of all vertices involved in the greedily selected edges.
- C can be showed to be an f-approximation in a similar manner as the proof in Theorem 8.

□

References

[Fei98] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[LY93] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. In *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 286–293, 1993.

[Mos15] Dana Moshkovitz. The projection games conjecture and the np-hardness of ln n-approximating set-cover. *Theory of Computing*, 11(1):221–235, 2015.