COMPSCI 330: Design and Analysis of Algorithms	April 21, 2016
Approximation Algorithms	
Lecturer: Debmalya Panigrahi	*Scribe: Tianqi Song*

1 Overview

In this lecture, we introduce approximation algorithms and their analysis in the form of approximation ratio. We also review a few examples.[1]

2 Approximation Algorithms

It is uncertain whether polynomial time algorithms exist for NP-hard problems, but in many cases, polynomial time algorithms exist which approximate the solution.

Definition 1. *Let P be an optimization problem for minimization, with an approximation algorithm $\mathcal{A}$. The **approximation ratio** α of $\mathcal{A}$ is:*

$$\alpha = \max_{I \in P} \frac{\text{ALGO}(I)}{\text{OPT}(I)}$$

Each I is an input/instance to P. $\text{ALGO}(I)$ is the value $\mathcal{A}$ achieves on I, and $\text{OPT}(I)$ is the value of the optimal solution for I. An equivalent form exists for maximization problems:

$$\alpha = \min_{I \in P} \frac{\text{ALGO}(I)}{\text{OPT}(I)}$$

*In both cases, we say that $\mathcal{A}$ is an α-**approximation** algorithm for P.*

A natural way to think of this (as we maximize over all possible inputs) is the worst-case performance of $\mathcal{A}$ against optimal. We will often use the abbreviations ALGO and OPT to denote the worst-case values which form α.

3 2-Approximation for Vertex Cover

A *vertex cover* of a graph $G = (V, E)$ is a set of vertices $S \subseteq V$ such that every edge has at least one endpoint in S. The VERTEX-COVER decision problem asks, given a graph G and parameter k, whether G admits a vertex cover of size at most k. The optimization problem is to find a vertex cover of the minimum size. We will provide an approximation algorithm for VERTEX-COVER with an approximation ratio of 2. Consider a very naive algorithm: while an uncovered edge exists, add one of its endpoints to the cover. It turns out this algorithm is rather difficult to analyze in terms of approximation ratio. A small variation gives a very straightforward analysis: instead of adding one vertex of the uncovered edge, add both.

[1]Some materials are from a previous note by Allen Xiao.

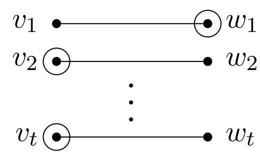

Figure 1: The set of v_i, w_i are the vertices chosen by the approximation algorithm. The optimal vertex cover must cover all these edges; at least one vertex from each edge must have been used in OPT as well.

Algorithm 1 Vertex Cover 2-Approximation

$U \leftarrow E$
$S \leftarrow \emptyset$
while U is not empty **do**
    Choose any $(v, w) \in U$.
    Add both v and w to S.
    Remove all edges adjoining v or w from U.
end while
return S

Consider the vertices added by this procedure. The vertex pairs added by the algorithm are a set of disjoint edges, since the algorithm removes adjoining vertices for every vertex it adds. OPT must cover each of these edges (v_i, w_i), and must therefore pick at least one endpoint from each edge. It follows that $\text{OPT}(G)$ is at least half the size of $|S|$, so the approximation ratio for this algorithm is at most 2.

4 Greedy Approximation for Set Cover

Given a universe of n objects X and a family of subsets $S = s_1, \ldots, s_m$ ($s_i \subseteq X$) a *set cover* is a subfamily $T \subseteq S$ such that every object in X is a member of at least one set in T (i.e. $\bigcup_{s \in T} s = X$). Let $c(\cdot)$ be a cost function on the covers, and let the cost of the set cover $c(T) = \sum_{s \in T} c(s)$. The weighted set cover optimization problem asks for the minimum cost set cover of X using covers S.

As with vertex cover, we will use a simplistic algorithm and prove its approximation ratio. Let $F \subseteq X$ be the set of (remaining) uncovered elements. Each step, we add the set which *pays the least per uncovered element it covers.*

$$\min_{s \in S} \frac{c(s)}{|s \cap F|}$$

Intuitively, this choice lowers the average cost of covering an element in the final set cover.

Algorithm 2 Greedy Set Cover

$F \leftarrow X$
$T \leftarrow \emptyset$
while F is not empty **do**
    $s \leftarrow \text{argmin}_{s' \in S} \frac{c(s')}{|s' \cap F|}$
    $T \leftarrow T \cup \{s\}$
    $F \leftarrow F \setminus s$
end while
return T

Correctness follows from the same argument as the vertex cover analysis: Elements are only removed from F (initially X) when they are covered by the set we add to T, and we finish with F empty. Therefore all elements of X are covered by some set in T.

To prove the approximation ratio, consider the state of the algorithm before adding the ith set. For clarity, let F_i be F on this iteration (elements not yet covered), but let T denote the final output set cover, and T^* the optimal set cover. By optimality of T^*:

$$\sum_{s \in T^*} c(s) = c(T^*) = \text{OPT}$$

T^* covers X, and therefore covers F_i:

$$\sum_{s \in T^*} |s \cap F_i| \geq |F_i|$$

We can consider how the sets in T^* perform on the cost-per-uncovered ratio that is minimized in the algorithm.

$$\min_{s \in T^*} \frac{c(s)}{|s \cap F_i|} \leq \frac{\sum_{s \in T^*} c(s)}{\sum_{s \in T^*} |s \cap F_i|} \leq \frac{\text{OPT}}{|F_i|}$$

The second inequality used "the minimum is at most the average". Now notice that the algorithm takes a minimum over all subsets S. Since $S \supseteq T^*$, the chosen set must have had at least as low a ratio as the minimum from T^*.

$$\min_{s \in S} \frac{c(s)}{|s \cap F_i|} \leq \min_{s \in T^*} \frac{c(s)}{|s \cap F_i|} \leq \frac{\text{OPT}}{|F_i|}$$

Finally, the cost of T is the sum of costs of its sets. Using the notation above, we can write this expression as a weighted sum of the minimized ratios, and then apply the above inequality to find an upper bound linear in OPT. Let $s^{(i)}$ be the ith set selected.

$$\begin{aligned}
\text{ALGO} = c(T) &= \sum_{s \in T} c(s) = \sum_{i=1}^{|T|} c(s^{(i)}) \\
&= \sum_{i=1}^{|T|} \frac{c(s^{(i)})}{|s^{(i)} \cap F_i|} \cdot |s^{(i)} \cap F_i| \\
&= \sum_{i=1}^{|T|} \frac{c(s^{(i)})}{|s^{(i)} \cap F_i|} \cdot (|F_i| - |F_{i+1}|) \\
&\leq \sum_{i=1}^{|T|} \frac{\text{OPT}}{|F_i|} \cdot (|F_i| - |F_{i+1}|)
\end{aligned}$$

Analyzing the sum will give us an expression for the approximation ratio. Since each sum term is $\text{OPT}/|F_i|$ duplicated $(|F_i| - |F_{i-1}|)$ times, we can replace the denominator terms to get an upper bound.

$$\begin{aligned}
\frac{\text{OPT}}{|F_i|} \cdot (|F_i| - |F_{i+1}|) &= \Bigg(\underbrace{\frac{\text{OPT}}{|F_i|} + \cdots + \frac{\text{OPT}}{|F_i|}}_{(|F_i| - |F_{i+1}|) \text{ times}} \Bigg) \\
&\leq \left(\frac{\text{OPT}}{|F_i|} + \frac{\text{OPT}}{|F_i| - 1} + \frac{\text{OPT}}{|F_i| - 2} + \cdots + \frac{\text{OPT}}{|F_{i-1}| + 1} \right) \\
&= \sum_{j=0}^{|F_i| - |F_{i+1}| - 1} \frac{\text{OPT}}{|F_i| - j}
\end{aligned}$$

Returning to the original sum, we realize this is actually a big descending sum of $\text{OPT}/(n-j)$ terms.

$$\begin{aligned}
\sum_{i=1}^{|T|} \left(\sum_{j=0}^{|F_i| - |F_{i+1}| - 1} \frac{\text{OPT}}{|F_i| - j} \right) &= \left(\frac{\text{OPT}}{|F_0|} + \cdots + \frac{\text{OPT}}{|F_1| + 1} \right) + \left(\frac{\text{OPT}}{|F_1|} + \cdots + \frac{\text{OPT}}{|F_2| + 1} \right) + \cdots \\
&= \frac{\text{OPT}}{n} + \frac{\text{OPT}}{n-1} + \cdots + \frac{\text{OPT}}{1} \\
&= \sum_{j=0}^{n-1} \frac{\text{OPT}}{n-j} \\
&= \sum_{k=n}^{1} \frac{\text{OPT}}{k}
\end{aligned}$$

In the last step, we applied a change of variables with $k = n - j$. This familiar sum is the nth harmonic number (times OPT).

$$\begin{aligned}\text{ALGO} &\leq \sum_{k=n}^{1} \frac{\text{OPT}}{k} \\ &= \text{OPT} \cdot H_n \\ &= \text{OPT} \cdot \Theta(\log n)\end{aligned}$$

Rearranging, we see that the approximation factor for the greedy algorithm is no more than some constant multiple of $\log n$.

$$\frac{\text{ALGO}}{\text{OPT}} = O(\log n)$$