COMPSCI 532: Design and Analysis of Algorithms	October 14, 2015
Lecture 12	
Lecturer: Debmalya Panigrahi	*Scribe: Allen Xiao*

1 Overview

In this lecture, we introduce approximation algorithms and their analysis in the form of approximation ratio. We review a few examples, then introduce an analysis technique for linear programs known as dual fitting.

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 $\mathscr{A}$. The* ***approximation ratio*** α *of $\mathscr{A}$ 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 $\mathscr{A}$ 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 $\mathscr{A}$ 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 $\mathscr{A}$ 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.

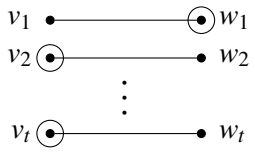

Figure 1: The set of v_i, w_i are the vertices chosen by the approximation algorithm. The optimal vertex cover must cover all these edges; at least one vertex from each edge must have been used in OPT as well.

Algorithm 1 Vertex Cover 2-Approximation

```
U ← E
S ← ∅
while U is not empty do
    Choose any (v, w) ∈ 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.

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

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 $\log n$.

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

5 Dual Fitting

Dual fitting is an analysis technique for approximation algorithms on linear programming problems. In short, we maintain a *feasible dual* corresponding to the infeasible (primal) solution we have built so far, and show that the ratio between their values is bound by a constant. This gives us a bound on the approximation ratio, thanks to strong duality. To see this, suppose we have a minimization primal, and let:

1. P be the value of the integral primal solution output by the algorithm. This is equivalent to ALGO.

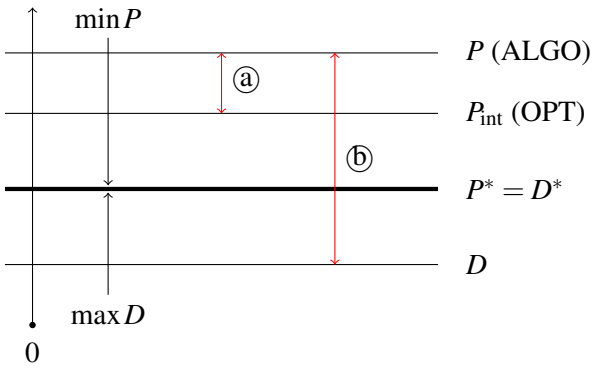

Figure 2: We have an optimization problem expressed as a minimization LP. The approximation algorithm produces some primal solution P which we wish to compare against the integral optimal P_{int} (ⓐ above). Dual fitting bounds this gap by constructing a feasible dual D whose gap with P is known (ⓑ above).

2. P_{int} be the optimal value for an integral solution to the primal. This is equivalent to OPT.
3. D be the value of the feasible dual we constructed to associate with P.
4. P^*, D^* be the primal and dual fractional optimal values, respectively.

Suppose we manage to construct feasible D such that $P \leq \alpha D$ for constant $\alpha \geq 1$. First, notice that the approximation factor is defined to be:

$$\frac{\text{ALGO}}{\text{OPT}} = \frac{P}{P_{\text{int}}}$$

Since the integral optimal in minimization is always at least the fractional optimal:

$$\frac{P}{P_{\text{int}}} \leq \frac{P}{P^*}$$

Invoke strong duality:

$$\frac{P}{P^*} = \frac{P}{D^*}$$

D is feasible, therefore $D \leq D^*$:

$$\frac{P}{D^*} \leq \frac{P}{D}$$

Combined, we have that:

$$\frac{\text{ALGO}}{\text{OPT}} \leq \frac{P}{D} \leq \alpha$$

See Figure 2 for a pictorial representation of the inequalities we used. We will go over a few examples of problems and construct feasible duals for them.

5.1 Dual fitting for vertex cover

Recall the 2-approximation algorithm for vertex cover from before (see Algorithm 1). To perform a dual fitting analysis, we will first write the relaxed (fractional) linear programs for vertex cover.

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ \text{s.t.} \quad & x_v + x_w \geq 1 \quad \forall (v,w) \in E \\ & x_v \geq 0 \qquad\qquad \forall v \in V \end{aligned}$$

The dual is:

$$\begin{aligned} \max \quad & \sum_{(v,w) \in E} y_{vw} \\ \text{s.t.} \quad & \sum_{v} y_{vw} \leq 1 \quad \forall v \in V \\ & y_{vw} \geq 0 \qquad \forall v \in V \end{aligned}$$

We will interpret each vertex selection as $x_v = 1$. Initially, our solutions are $x_v = 0$ for all v in the primal, and $y_{vw} = 0$ for all (v,w) in the dual. Notice that the primal solution is infeasible, and the dual solution is feasible. When (v,w) is added to S:

1. In the primal, we set $x_v = 1$ and $x_w = 1$. Since the edges selected are disjoint, $x_v, x_w = 0$ previously. Therefore, $\Delta P = 2$ for each iteration.

2. For the dual, we will do the natural thing and set $y_{vw} = 1$ as well. Since the edges selected are disjoint, the edges with $y_{vw} = 1$ form a matching on the graph. No vertex constraint in the dual has value more than 1, and $y_{vw} = 0$ before this addition. Thus, D remains feasible and $\Delta D = 1$ for each iteration.

At the end of the algorithm, P becomes primal feasible (no edge of E left uncovered). At each step, $\Delta P = 2\Delta D$. Let P_t (resp. D_t) be the primal (resp. dual) value after the tth iteration.

$$\frac{P_t}{D_t} = \frac{\sum_{i=1}^{t} (\Delta P)_i}{\sum_{i=1}^{t} (\Delta D)_i} = \frac{2t}{t} = 2$$

P is exactly the size of the output cover (ALGO). Since D is a feasible dual at the end of the algorithm, dual fitting gives us that:

$$\frac{\text{ALGO}}{\text{OPT}} \leq \frac{P}{D} = 2$$

5.2 Dual fitting in general

It is not exactly necessary that dual we maintain is feasible. Suppose instead that we maintain an infeasible dual D and $P/D = \alpha$, where D/β is feasible instead. Applying the dual fitting argument with $D' = D/\beta$ tells us that:

$$\frac{\text{ALGO}}{\text{OPT}} \leq \frac{P}{D'} = \frac{P}{D/\beta} = \alpha\beta$$

In general, there are two ratios we can control in dual fitting:

1. The primal-dual ratio: $P/D = \alpha$ by construction, though D may not be feasible.

2. The infeasibility ratio: D/β is feasible.

One might ask, "why not just maintain D/β instead?" For certain problems, maintaining infeasible D is more semantically useful than its feasible scaled counterpart. The next example will demonstrate this.

5.3 Dual fitting for greedy set cover

Recall the algorithm for greedy set cover, which we proved using other techniques to be $\log n$-approximate.

Algorithm 3 Greedy Set Cover

```
F ← X
T ← ∅
while F is not empty do
    s ← argmin_{s'∈S} c(s')/|s'∩F|
    T ← T ∪ {s}
    F ← F \ s
end while
return T
```

The relaxed linear programming form of weighted set cover is:

$$\begin{aligned} \min \quad & \sum_{s\in S} c(s)x_s \\ \text{s.t.} \quad & \sum_{s:e\in s} x_s \geq 1 \quad \forall e \in X \\ & x_s \geq 0 \qquad\quad \forall s \in S \end{aligned}$$

The dual is:

$$\begin{aligned} \max \quad & \sum_{e\in X} y_e \\ \text{s.t.} \quad & \sum_{e\in s} y_e \leq c(s) \quad \forall s \in S \\ & y_e \geq 0 \qquad\quad \forall e \in X \end{aligned}$$

Initially, all $y_e, x_s = 0$. This is infeasible for the primal, but feasible in the dual. Each iteration of the algorithm, we add a set s to T. F is the set of uncovered elements each iteration.

1. In the primal, we set $x_s = 1$. Then, $\Delta P = c(s) \cdot 1 = c(s)$.
2. We will set the dual to maintain $\Delta D = \Delta P = c(s)$. Set $y_e = c(s)/|s \cap F|$ for the new elements $e \in s$ covered. Intuitively, this is charging the "purchase" of s to the elements it newly covers ($s \cap F$). Since each element is first covered exactly once, y_e is unchanged after the first time e is covered and until the end of the algorithm. The total charge is $\Delta D = |s \cap F| \cdot c(s)/|s \cap F| = c(s)$.

Lemma 1. *$D/\log n$ is dual feasible.*

Proof. We will do the analysis on D (without scaling) first. On each iteration, the dual constraint for a fixed set s^* changes if any of its elements were covered. Without loss of generality, we will only consider iterations which cover some $e \in s^*$. Recall that each iteration picks some set s:

$$\min_{s'\in S} \left(\frac{c(s')}{\#\text{ uncovered elements left in } s'} \right) = \frac{c(s)}{|s \cap F|}$$

As the minimizer, this lower bounds the same ratio for s^*:

$$\frac{c(s)}{|s \cap F|} \leq \frac{c(s^*)}{|s^* \cap F|}$$

We will examine the dual constraint for s^* and use an argument similar to that of the original greedy quicksort analysis to extract a $\log n$ factor. Let s_i be the ith selected set, and F_i be the set of uncovered elements before s_i is added. When s_i is selected, the dual change in the s^* constraint is at most:

$$|s^* \cap Y| \cdot \frac{c(s^*)}{|s^* \cap F|}$$

At the end of the algorithm, the dual constraint for s^* becomes:

$$\begin{aligned}
\sum_{e \in s^*} y_e &= \sum_i \frac{c(s_i)}{|s^* \cap F_i|} \cdot |\{\text{elements of } s^* \text{ covered by } s_i\}| \\
&= \sum_i \frac{c(s_i)}{|s^* \cap F_i|} \cdot (|s^* \cap F_i| - |s^* \cap F_{i+1}|) \\
&\leq \sum_i \frac{c(s^*)}{|s^* \cap F_i|} \cdot (|s^* \cap F_i| - |s^* \cap F_{i+1}|) \\
&\leq \left(\frac{c(s^*)}{|s^* \cap F_1|} + \cdots + \frac{c(s^*)}{1 + |s^* \cap F_2|} \right) + \left(\frac{c(s^*)}{|s^* \cap F_2|} + \cdots + \frac{c(s^*)}{1 + |s^* \cap F_3|} \right) + \cdots \\
&\leq c(s^*) \left(\frac{1}{|s^*|} + \frac{1}{|s^*| - 1} + \cdots + \frac{1}{1} \right) \\
&= c(s^*) H_{|s^*|} \\
&\leq c(s^*) \log |s^*| \leq c(s^*) \log n
\end{aligned}$$

This is better than $\log n$, especially when sets are small. Dividing by $\log n$ gives feasibility.

$$\frac{1}{\log n} \sum_{e \in s^*} y_e \leq c(s^*)$$

It follows that $D / \log n$ is dual feasible. □

We now have that $P/D = 1$, and $D/\log n$ is dual feasible. Applying the dual fitting argument with P and D, we see that the greedy set cover algorithm is $(\log n)$-approximate.