15-854: Approximations Algorithms **Lecturer:** Anupam Gupta
Topic: Randomized Approximation Algorithms **Date:** October 17, 2005
Scribe: Don Sheehy

11.1 Randomized Approximation Algorithms

We have seen several randomized approximation algorithms before. Recall that for the MAX-CUT and MAX-3SAT problems we proved that choosing any solution uniformly at random gives a constant factor approximation in the expectation. For MAX-CUT the approximation factor was 2. For MAX-3SAT, a random assignment of truth values to a variable satisfies a given clause with probability $\frac{7}{8}$.

In the case of MAX-3SAT, it is possible to derandomize the algorithm and get a deterministic performance guarantee of $\frac{8}{7}$. Furthermore, this approximation is tight as it has been shown that finding a better approximation is NP-Hard.

Today, we will look at two other problems where flipping coins allows us to get a good approximation in the expectation. Specifically, we will be looking at VERTEX-COVER and RENT-OR-BUY. In each of these we will have to be more clever with our randomness than we were for MAX-CUT and MAX-3SAT.

11.2 Two flavors of Performance Guarantee for Randomized Approximations

In many cases, we may find that we desire a randomized algorithm that performs well with high probability rather than simply in the expectation. In such cases we want an algorithm A with a guarantee of the form

$$A(I) \leq \rho OPT \tag{11.2.1}$$

with high probability (on the order of $1 - \frac{1}{poly(n)}$) for all instance I. In general, approximation algorithms that do well in the expectation can be converted to algorithms that do well with high probability. That is, given $\epsilon > 0$, it possible to convert algorithm A with guarantee

$$\mathbf{E}[A(I)] \leq \rho OPT \tag{11.2.2}$$

into an algorithm A' with performance guarantee

$$A'(I) \leq \rho(1+\epsilon)OPT \tag{11.2.3}$$

with high probability. The method is simply to run the algorithm that does well in the expectation several times and choose the best answer. Let t be the number of times we run the algorithm to get a good answer. We need to show that t is not too large. First we observe the probability that the algorithm performs worse than expected by a factor of more than $(1+\epsilon)$ is bounded by $\frac{1}{1+\epsilon}$.

$$\mathbf{Pr}[A(I) \geq (1+\epsilon)\mathbf{E}[A(I)]] \leq \frac{1}{1+\epsilon}. \tag{11.2.4}$$

It follows that the probability that after t independent trials, the algorithm has performed worse than the expectation is similarly bounded.

$$\mathbf{Pr}\left[\left(\min_{t\ trials} A(I)\right) \geq (1+\epsilon)\mathbf{E}[A(I)]\right] \leq \frac{1}{(1+\epsilon)^t}. \tag{11.2.5}$$

By choosing $t = c\log_{(1+\epsilon)} n$ for any $c > 0$ we can rewrite the above inequality as follows.

$$\mathbf{Pr}\left[\left(\min_{t\ trials} A(I)\right) \geq (1+\epsilon)\mathbf{E}[A(I)]\right] \leq \frac{1}{n^c}. \tag{11.2.6}$$

Thus, if A' is the algorithm that runs A $c\log_{(1+\epsilon)} n$ times and takes the best answer, then A' achieves the desired performance guarantee of (11.2.3). Also, if A ran in time $O(t(n))$ then A' runs in time $O(t(n)\log n)$.

11.3 VERTEX-COVER

Instance: A graph $G = (V, E)$, and vertex weights $w : V \to \mathbb{R}_{\geq 0}$.
Output: Subset $C \subseteq V$ such that each edge has an end in C and $\sum_{v\in C} w(v)$ is minimized.

As a first exercise, let's consider the following greedy algorithm. For each edge e, if e has not already been covered then add its least expensive endpoint to the set. Now, because this is a first try, we ask: How bad can this approximation be? Let's consider the star on n vertices with $w(v) = 1+\epsilon$ for the center and $w(v) = 1$ for all others. Now, the greedy algorithm will select all of the leaves for its vertex cover. Thus achieving a total weight $n-1$, where selecting the center node would have given a weight $1+\epsilon$ vertex cover. So the greedy algorithm could be off by almost a factor of n.

We now give a randomized version of the above algorithm for the unweighted case, i.e. all vertices have unit weight.

RAND-VC: For each $e = (u, v) \in E$, if e is not already covered, add u to the set with probability $\frac{1}{2}$ and add v otherwise.

Theorem 11.3.1 *RAND-VC is a 2-approximation in expectation for unweighted VERTEX-COVER. Equivalently,*

$$\mathbf{E}[|C|] \leq 2|C^*|, \tag{11.3.7}$$

where C is the vertex cover produced by the algorithm and C^* is the optimal vertex cover.

We only sketch the proof as this result will also follow from the stronger version of the theorem below. Partition the edges of E by assigning each to a vertex $v^* \in C^*$ that covers it. Of all the edges assigned to some $v_i^* \in C^*$, the expected number that will be examined before all these edges are covered is at most 2 because each edge examined has v_i^* as an endpoint and v_i^* covers them all. So, in the expectation, the number of vertices added to the set C will be at most $2|C^*|$.

We now alter the algorithm for the weighted case by choosing an endpoint of an edge with a probability relative to weights of the endpoints. the algorithm is due to Pitt [2].

WtRAND-VC: For each $e = (u, v) \in E$, if e is not covered yet then pick u with probability $\frac{w_v}{w_u + w_v}$ and pick v otherwise.

Theorem 11.3.2 *WtRAND-VC is a 2-approximation in expectation for weighted VERTEX-COVER.*

Proof: As before, we partition the edges by assigning them to vertices in C^*, an optimal vertex cover. Let $e_1, \ldots, e_k$ be the edges assigned to some $v^* \in C^*$. For each edge e_i examined by the algorithm, at most one vertex was added to the vertex cover C. Let S denote the set of such vertices. We first present the following claim.

Claim 11.3.3

$$\mathbf{E}[w(S)] \leq 2w(v^*). \tag{11.3.8}$$

Proof of Claim: The proof is by induction on k. If $k = 0$ then no e_i is incident to v^* and trivially, $w(S) \leq w(v^*) \leq 2w(v^*)$. For the inductive step let $e_k = (u, v^*)$ letting $k = n$ and assume the claim holds for all $k \leq n$.

$$\mathbf{E}[w(S)] = \frac{w(u)}{w(u) + w(v^*)} w(v^*) + \frac{w(v^*)}{w(u) + w(v^*)} (w(u) + \mathbf{E}\big[w(S')\big]), \tag{11.3.9}$$

where S' is the vertex cover produced on $e_i, \ldots, e_{k-1}$. So, using the inductive hypothesis we get the following inequality.

$$\begin{aligned}
\mathbf{E}[w(S)] &= \frac{w(u)}{w(u) + w(v^*)} w(v^*) + \frac{w(v^*)}{w(u) + w(v^*)} (w(u) + \mathbf{E}\big[w(S')\big]) && (11.3.10)\\
&\leq 2\frac{w(u)w(v^*)}{w(u) + w(v^*)} + 2\frac{w(v^*)^2}{w(u) + w(v^*)} && (11.3.11)\\
&\leq 2w(v^*). && (11.3.12)
\end{aligned}$$

■

It follows that the total expected weight of the cover C is just the sum of the expected weights for each neighborhood of vertices in C^*. This is exactly what we bounded in the claim.

$$\begin{aligned}
\mathbf{E}[w(C)] &= \sum_{v_i^* \in C^*} \mathbf{E}[w(S_i)] && (11.3.13)\\
&\leq \sum_{v_i^* \in C^*} 2w(v_i^*) && (11.3.14)\\
&\leq 2w(C^*) && (11.3.15)
\end{aligned}$$

■

11.4 The RENT-OR-BUY Problem

Instance: A metric $G = (V, E)$ with distances $d(u, v)$ satisfying the triangle inequality; A set $R \subseteq V$ of terminals; a designated root $r \in V$; and an integer $M \geq 1$.
Output: A set F of "collection points" (also known as "facilities") such that $r \in F$ and a Steiner

tree T connecting F. The cost $\Phi(F,T)$ of a solution is the sum of the distances of all terminals to F plus M times the size of the Steiner tree T. That is,

$$\Phi(F,T) = \sum_{v\in R} d(v,F) + M \times \mathit{cost}(T) \tag{11.4.16}$$

$$= \sum_{v\in R} d(v,F) + M \left(\sum_{(u,v)\in E[T]} d(u,v) \right). \tag{11.4.17}$$

The desired pair (F^*, T^*) is the one that minimizes the cost Φ. One way to think about this problem is to imagine that we want to connect all of the terminals to the root by paths. We can "buy" some of the edges for M times their distance and use them as many times as we want. We can also "rent" edges, paying only for the distance, but we must pay for each use. Intuitively, if we want to use a particular edge of length d more than M times, it makes sense to buy it for Md dollars rather than renting it for d dollars more than M times. If M is very large we will not buy any edges and shortest paths to r will be optimal. If $M = 1$ then we can always buy and thus the optimal is the mincost Steiner tree.

The Algorithm

1. Start with $F = \{r\}$.
2. For each terminal $v \in R$, add v to F with probability $1/M$.
3. Build the minimum spanning tree T on F. "Buy" the edges of the MST. Recall that the MST is a 2-approximation for the minimum Steiner tree.
4. "Rent" a path from each $v \in R$ to the closest vertex in F. The total cost of these paths is $\sum_{v\in R} d(v,F)$.

In order to show that this algorithm works well, we first make a claim about the expected cost incurred from our choice of the tree T.

Claim 11.4.1 *If T is the tree constructed in the algorithm and OPT is the total cost of an optimal solution, then*

$$\mathbf{E}[M \times \mathit{cost}(T)] \le 2OPT, \tag{11.4.18}$$

Proof: Take an optimal solution (T^*, F^*). Consider the random tree $\widehat{T}$

$$\widehat{T} = T^* \cup \{\text{shortest paths from vertices in } F \text{ to } F^*\}. \tag{11.4.19}$$

Since $\widehat{T}$ is a Steiner tree on F, the MST T has the property that $\mathit{cost}(T) \le 2\mathit{cost}(\widehat{T})$. So, it will suffice to show that $\mathbf{E}\left[M \times \mathit{cost}(\widehat{T})\right] \le OPT$.

$$\mathbf{E}\left[M \times \mathit{cost}(\widehat{T})\right] = \mathbf{E}[M \times \mathit{cost}(T^*)] + \sum_{v\in R} \mathbf{Pr}[v \in F] \times M d(v, F^*) \tag{11.4.20}$$

Recalling that the probability in the above equation was picked explicitly by the algorithm to be $1/M$, we get.

$$\mathbf{E}\Big[M \times cost(\widehat{T})\Big] = M \times cost(T^*) + \sum_{v \in R} d(v, F^*) = OPT \tag{11.4.21}$$

This completes the proof. ■

We now bound the cost associated with the rented edges to get a bound on the expected quality of a solution provided by the algorithm.

Claim 11.4.2 *The expected "renting cost" of the algorithm's solution is at most $2OPT$. That is,*

$$\mathbf{E}\left[\sum_{v \in R} d(v, F)\right] \leq 2OPT. \tag{11.4.22}$$

Proof: By the preceding claim, it suffices to show that the expected renting cost is at most the expected buying cost.

The tree T constructed by the algorithm is the minimum spanning tree of the set F. So, there is an execution of Prim's MST algorithm that produces T. Recall that we added each $v \in R$ to F with probability $1/M$. So we may imagine that the execution of Prim's algorithm was in fact looking at the entire set of terminals R but each time a vertex would be added to the tree, it was added with probability $1/M$ and thrown out otherwise. At each step i, Prim's algorithm considers adding a new vertex v_i to the growing tree T_i because $d(v_i, T_i)$ was minimum. Say that the parent of v_i is the vertex in T_i closest to v_i, denoted P_{v_i}.

Trivially because $P_{v_i} \in F$, we have

$$d(v, F) \leq d(v, P_v). \tag{11.4.23}$$

Also, we can infer from the correctness of Prim's algorithm that

$$\sum_{u \in F} d(u, P_u) = cost(MST(F)), \tag{11.4.24}$$

because the algorithm adds the edges (u, P_u) for all $u \in F$ and the result is the MST.

So, at step i we buy edge (u, P_u) with probability $1/M$.

$$\mathbf{E}[\text{buying cost at step } i \mid \text{all choices made before } i] = \frac{1}{M} \underbrace{(Md(u, P_u))}_{\text{cost of } (u, P_u)} \tag{11.4.25}$$

$$= d(u, P_u) \tag{11.4.26}$$

We now note that the expected renting cost at step i given all previous choices is bounded above by $d(u, P_u)$ and thus

$$\mathbf{E}[\text{renting cost} \mid \text{all previous choices}] \leq \mathbf{E}[\text{buying cost} \mid \text{all previous choices}]. \tag{11.4.27}$$

This implies that $\mathbf{E}[\text{renting cost}] \leq \mathbf{E}[\text{buying cost}] \leq 2OPT$ as desired.

■

By the linearity of expectations the expected total cost is the expected renting cost plus the expected buying cost. So, the two preceding claims imply that the expected total cost is at most $4OPT$. So we have proven the following Theorem.

Theorem 11.4.3 *The randomized algorithm is a* 4*-approximation in the expectation for the RENT-OR-BUY problem.*

11.4.1 Some Notes

The RENT-OR-BUY problem is also referred to as the Connected Facility Location problem. The preceding algorithm is due to Gupta, Kumar, and Roughgarden [1]. A quick look at the preceding analysis reveals that part of the error was introduced by using the MST as a 2-approximation for the minimum Steiner tree on F. In [1], the authors use the better Steiner tree approximation of Robins and Zelikovsky to push the approximation ratio down to 3.55.

Exercise 11.4.4 *Imagine that we pick some* α *and run the randomized RENT-OR-BUY algorithm choosing vertices in* R *with probability* $\frac{\alpha}{M}$ *rather than* $\frac{1}{M}$*. What is the resulting approximation guarantee?*

References

[1] Anupam Gupta and Amit Kumar and Tim Roughgarden. Simpler and better approximation algorithms for network design. STOC 2003.

[2] L. Pitt. A Simple Probabilistic Approximation Algorithm for Vertex Cover, Technical Report YaleU/DCS/TR-404, Department of Computer Science, Yale University, 1985.