

CSE203a

Spring 2006

Scribed by Brian McFee

2006-04-11

1. Approximation algorithms for Vertex Cover

- (a) Factor-2 approximation based on matchings. Although we can't find the best vertex cover efficiently, we can efficiently compute a related structure: a maximal matching M .
- M gives us a lower bound on the size of the optimal vertex cover: imagine the optimal vertex cover on a complete bipartite graph.
 - If we construct a vertex cover by adding all of the vertices in the maximal matching M , we get a cover that is at most twice M . That is, we can get within a factor of two of the lower bound on the optimal vertex cover.
- (b) A naive greedy algorithm for vertex-cover:

VC-Greedy(G)

```
1  $S \leftarrow \emptyset$ 
2 repeat
3   Let  $U$  be the vertex of highest degree in  $G$ .
4   Remove  $U$  and its edges from  $G$ .
5    $S \leftarrow S \cup \{U\}$ 
until  $G$  is empty.
```

- **Claim:** VC-Greedy achieves a $\Omega(\log n)$ approximation ratio.
- **Proof:** Consider a bipartite graph with n vertices on the left side of the bipartition, and $\frac{1}{2}n \lg n$ vertices on the right. Each vertex on the left side has degree n . On the right side, we break up the vertices into groups $A_0 \dots A_{\lg n - 1}$ of $n/2^i$ vertices each, where each vertex in A_i has degree $\lfloor n/2^i \rfloor$. The optimal vertex cover on this graph would be to take the entire left side, resulting in a cover of $|OPT| = n$ vertices. However, the greedy solution could first take all of the vertices in A_0 , since they each have degree n . This leaves all of the left-side vertices with degree $n/2$. The greedy algorithm could then select the nodes from A_1 , leaving the left side with degree $n/4$. This process can repeat until the right side of the bipartition is exhausted, resulting in a vertex cover of size $\frac{1}{2}n \lg n$.

- (c) For VC, we can't do better than a factor $7/6$ approximation ratio [Hås01].

2. (Metric) k -Clustering

- (a) High-level problem description: Given a set of points in a metric space, divide them into k coherent groups.

- (b) A distance function $d(\cdot, \cdot)$ is a metric if for all points x, y, z in the space:

- $d(x, y) \geq 0$,
- $d(x, y) = 0 \Leftrightarrow x = y$,
- $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality),
- $d(x, y) = d(y, x)$ (symmetry).

- (c) k -cluster:

- Input: points $\{x_1, \dots, x_n\}$, metric $d(\cdot, \cdot)$, positive integer k .
- Output: Subsets C_1, \dots, C_k that partition X : $\forall_{i,j} C_i \cap C_j = \emptyset$, and $\bigcup_{i=1}^k C_i = X$.
- Objective: minimize the maximum cluster diameter:

$$\min \left\{ \max_{C_j} \max_{x_a, x_b \in C_j} d(x_a, x_b) \right\}$$

- (d) Algorithm: furthest-first traversal

K-Cluster-Furthest-First(X, d, k)

- 1 Pick any point $\mu_1 \in X$ as the first cluster center.
- 2 **for** $j = 2$ to k
 do
- 3 let $\mu_j = \arg \max_{z \in X} \min_{i=1 \dots j-1} d(z, \mu_i)$, ie. the point in X furthest from the already chosen cluster centers.
- 4 Define C_j as the set of points closer to μ_j than any other μ_i .

(e) Why does this work?

Let z be the point furthest from μ_1, \dots, μ_k . Define Δ as the distance from z to the closest cluster center μ_i .

- **Claim 1:** The cost of the solution found by this algorithm is no more than 2Δ .
- **Proof:** Since z is the furthest point from a cluster center, all other points must be within Δ of their nearest cluster center. Therefore, the radius of each cluster is at most Δ . By the triangle inequality, the diameter of each cluster cannot be more than 2Δ . \square
- **Claim 2:** The cost of the optimal solution is at least Δ .
- **Proof:** Assume we have $k+1$ points: μ_1, \dots, μ_k, z , all of which are at least Δ apart from each-other. By the pigeonhole principle, at least two points belong to the same cluster. Therefore, the optimal clustering must be at least Δ .

(f) High-level similarities between K-clustering and Vertex Cover:

Instead of finding the exact optimal solution, we find a closely related structure that is efficiently computable.

For K-cluster, we find k points which cover the data with radius Δ . This gives a lower bound on the optimal solution. We then build an approximate solution that has a cost no more than some multiple of this lower bound, in this case, 2.

(g) Could we find a tighter bound? For arbitrary metric spaces, factor $(2 - \epsilon)$ approximations are not possible. Euclidean spaces may be easier.

3. Set cover

(a) Problem definition

- i. Input: a base set B , subsets $S_1, \dots, S_m \subseteq B$.
- ii. Output: some collection of subsets which cover all of B , ie.

$$I \subseteq [1, m] \qquad \text{such that } \bigcup_{i \in I} S_i = B$$

iii. Objective: minimize $|I|$.

Note that Vertex Cover can be reduced to set cover, so set cover is at least as hard to solve exactly. Is it also harder to approximate?

(b) Greedy algorithm for set cover

Set-Cover-Greedy(B, S_1, \dots, S_m)

- 1 $I \leftarrow \emptyset$
- 2 **repeat**
- 3 Add to I the set S_i containing the most as-yet-uncovered elements of B .
 until all of B is covered by I .

(c) Let $n = |B|$, and OPT be the size of the best set cover for B .

- **Claim:** Our greedy algorithm returns a solution of cost no more than $OPT \ln n$.
- **Proof:** Let n_t be the number of uncovered elements after t sets have been chosen (start with $n_0 = n$). Since there are OPT sets that cover all of B , there must be some set that covers at least n_t/OPT of the remaining n_t elements. Therefore, we can apply the following convexity argument:

$$\begin{aligned} n_{t+1} &\leq n_t - \frac{n_t}{OPT} \\ \Rightarrow n_t &\leq n_0 \left(1 - \frac{1}{OPT}\right)^t \\ &< n e^{-t/OPT} \end{aligned}$$

To reach our end condition of no uncovered elements, we let $t = OPT \ln n$ so that $n_t < 1$. \square

4. Max-Cut

(a) A cut of a graph $G = (V, E)$ is a partition of V into 2 groups: S and $V \setminus S$. The size of the cut is the number of edges between S and $V \setminus S$.

- (b) We can solve this approximately with a randomized algorithm. The simplest way to do this is simply flipping a fair coin to determine if a vertex goes in S or $V \setminus S$, and repeating until all vertices have been assigned.

Randomized-Max-Cut(G)

```
1  $S \leftarrow \emptyset$ 
2 for each  $u \in V$ 
   do
3     add  $u$  to  $S$  with probability  $\frac{1}{2}$ .
```

- (c) **Claim:** $\mathbf{E}[\text{size of cut}] \geq \frac{1}{2}|E| \geq \frac{1}{2}OPT$.

- (d) **Proof:** for any edge e , the probability that one end goes in S and the other goes in $V \setminus S$ is the same as $\Pr[e \text{ ends up in the cut}] = 1/2$.

Clearly, the optimal cut can be no larger than the number of edges in the graph. \square

References

[Hås01] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.