**CS 787: Advanced Algorithms**

# Greedy Approximations

Instructor: Dieter van Melkebeek

Approximation algorithms give a solution to a problem in polynomial time, at most a given factor away from the correct solution. This lecture covers Greedy Approximation Algorithms, in the context of the Vertex Cover, Metric k-Center, and Set Cover problems.

## 1 Vertex Cover

A vertex cover of a graph is a set of vertices that "covers" the edges of the graph. That is, every edge has at least one endpoint in the covering set, or, equivalently, for every vertex, it is either in the set or adjacent to some vertex which is in the set.

**Given:** Graph $G = (V, E)$

**Goal:** Find a vertex cover of minimum size.

### 1.1 Approximation algorithm

We first give the definition of matching.

**Definition 1** (Matching)**.** *Given a graph $G = (V, E)$, a subset of the edges $M \subseteq E$ is a matching if no two edges of $M$ share an endpoint. A matching of maximum number of edges is called a maximum matching. A maximal matching is one that cannot be extended by adding another edge. (A maximum matching is maximal but the reverse is not always true).*

We observe the following two facts:

1. $OPT_{VC} \geq |M|$, for any matching $M$ in $G$.

   This is so because any vertex cover has to pick at least one endpoint of each edge in the matching, otherwise this edge would be uncovered.

2. For any maximal matching $M$, $S = \bigcup_{(u,v)\in M}\{u, v\}$ is a VC.

   Assume it is not a VC. That means there is some edge $e = (u, v) \in E$ left uncovered by the vertices in $S$. Then neither $u$ or $v$ is an endpoint of any edge in $M$ and thus $M$ could legally be extended by adding edge $e$. Thus $M$ is not maximal and we have a contradiction, so $S$ must be a VC. This also directly implies that $OPT_{VC} \leq |S|$.

So we have $OPT_{VC} \leq |S| = 2|M| \leq 2OPT_{VC}$.

**Theorem 1.** *Vertex Cover problem can be approximated to within factor of 2.*

*Proof.* We can use a greedy algorithm to construct a maximal matching $M$, then by the above two facts, we have $|S| \leq 2OPT_{VC}$, where $S$ is the set of vertices matched in $M$. □

### 1.2 Can the approximation guarantee be improved?

We need to answer the following questions.

Q1: Is "$OPT_{VC} \leq 2|M|$" tight?

Yes. Let $K_{2n+1}$ be a complete graph of $2n+1$ vertices. Then every maximal matching $M$ has $n$ edges and the vertex cover problem for $K_{2n+1}$ also has optimal solution with $OPT = 2n$. So this is a factor 2 approximation with tight bound. For other graphs, we might be able to improve the factor, for example, bipartite graphs have a factor 1 approximation.

Q2: Is the analysis of the algorithm tight, i.e., $|S| \leq 2OPT_{VC}$?

Yes. For example, a complete bipartite graph on $n + n$ vertices has $OPT_{VC} = n$, while the maximal matchings have $|M| = n$ so $|S| = 2|M| = 2n$.

Q3: Is the theorem tight?

We do not know! But by assuming reasonable complexity theoretic hypothesis, the answer is yes.

## 2 (Metric) k-Center

Given the locations of customers, the problem is to decide on the locations of at most $k$ warehouses such that the maximum travel time over all the customers is minimized. We can define the problem formally as follows:

**Given:**

- Complete (di)graph $G = (V, E)$
- Weights on the edges, $d : E \to [0, \infty)$
- $k \in \mathbb{N}$

**Goal:** Find set $C \subseteq V$ with $|C| \leq k$ s.t. the following objective is minimized:

$$\max_{v \in V}(\min_{c \in C}(d(v, c)))$$

$C$ is our set of warehouses in this example, and customers are located at each vertex $v \in V$. Note that $\min_{c \in C}(d(v, c))$ is the distance of customer $v$ to the nearest warehouse. Then $\max_{v \in V}(...)$ effectively finds the distance traveled by the customer that must travel the longest to reach any warehouse. This customer is presumably the unhappiest. The overall goal is to then minimize the maximum unhappiness.

Note that the problem in that original version, called k-center problem, is an NP-complete problem. Moreover, k-center cannot even be approximated (i.e. it cannot be solved efficiently to within any constant factor of the optimal objective). So, we consider the problem with the restriction that $d$ is a metric on $V$.

**Definition 2** (Metric)**.** *A function $d : X \times X \to \mathbb{R}$ is a metric on the set $X$ iff $(\forall x, y, z \in X)$,*

$$
\begin{aligned}
d(x,x) &= 0 \\
d(x,y) &= d(y,x) \\
d(x,z) &\leq d(x,y) + d(y,z)
\end{aligned}
$$

## 2.1 Factor 2 approximation algorithm

We begin with a useful claim.

**Claim 1.** *Within any set of $k+1$ vertices, there is some pair, $u, v$, s.t. $d(u,v) \leq 2OPT$.*

*Proof.* Since there are $k+1$ vertices, and only $k$ centers (in the optimal solution to metric k-center), the pigeonhole principle tells us that at least two points must be associated with the same center. Let the vertices be $u$ and $v$ and the shared center be $c$. Because $c$ is a center, and $d$ is a metric, we have the following:

$$
\begin{aligned}
d(u,v) &\leq d(u,c) + d(c,v) \\
&\leq 2OPT
\end{aligned}
$$

This is also depicted in Figure 1. Two vertices which share a center are not farther than $2OPT$ because of the triangle inequality inherent in a metric function.

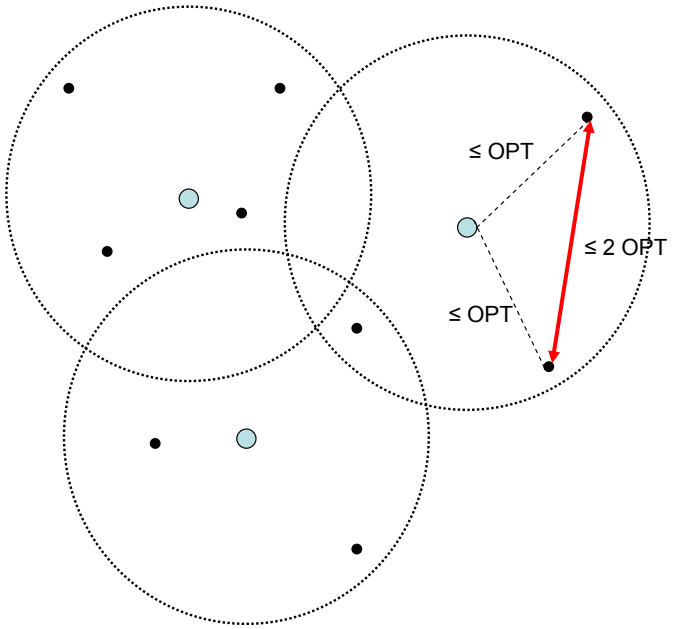


Figure 1: Graphical example for the proof

□

Now we can present the algorithm. The metric version of k-center problem can be approximated by greedily selecting the customer who is furthest from any warehouse at each step, and putting the next warehouse at their location.

**Algorithm 1:** Metric k-center Greedy Algorithm
(1) **if** $k \geq |V|$
(2) **return** $V$
(3) Pick $c_1$ arbitrarily
(4) **for** i=2 **to** k
(5) $c_i \leftarrow argmax_{v \in V}(\min_{j=1,..,i-1}(d(v, c_j)))$
(6) **return** $C$

**Claim 2.** *The algorithm gives a factor 2 approximation.*

*Proof.* Take any vertex $v$ that is not selected by the end of this algorithm's execution. Consider this $v$ along with the $k$ selected vertices in $C$ and also let $c_v \leftarrow argmin_{c \in C}(d(v, c))$.

By Claim 1, we know that some pair of vertices in this group must have distance between them $\leq 2OPT$.

- Case 1: $v$ is one of the pair. Because $d(v, c_v) \leq d(v, c)(\forall c \in C)$, we know that $d(v, c_v) \leq 2OPT$.
- Case 2: $v$ is not one of the pair. Then the pair is $c_i$ and $c_j, (i < j)$ (those vertices selected on iterations $i$ and $j$ in the algorithm). Because on iteration $j$, $c_j$ was selected in favor of $v$, we have the following:

$$\begin{aligned} d(v, c_v) &\leq min_{c \in c_1,..,c_{j-1}}(d(v, c)) \\ &\leq min_{c \in c_1,..,c_{j-1}}(d(c_j, c)) \\ &\leq d(c_j, c_i) \\ &\leq 2OPT \end{aligned}$$

The selection of $v$ was arbitrary, so we know that any vertex not in $C$ will have minimum distance to some $c \in C$ that is at most a factor of 2 from the optimal unhappiest-customer's best distance. □

## 2.2 Tightness example

Now, let's look over the tight example below that shows us the extreme case resulting in the factor of 2 exactly.

*Example:* Consider the metric k-center problem for the graph seen in the Figure 2 for $k = 1$.

Suppose that all the spokes are assigned $d = 1$ whereas all the other edges are assigned $d = 2$. (The figure does not have all these edges but only some of them to prevent a crowding of lines). In this case, observe that $OPT = 1$ (just pick the center vertex). The algorithm is likely to pick one of the other vertices, however, and hence $ALG = 2OPT$. ⊠

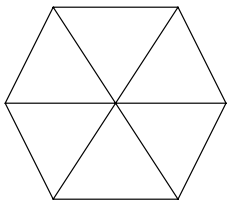

Figure 2: Tightness example for the k-center problem

### 2.3 Better approximation?

*Question:* Can we come up with a better factor than 2? *Answer:* No. (Unless $\mathsf{P} = \mathsf{NP}$)

**Theorem 2.** *For any $\epsilon > 0$, metric k-center problem is not approximable within a factor of $(2-\epsilon)$ unless $\mathsf{P} = \mathsf{NP}$.*

*Proof.* We will translate an instance of vertex cover (VC) into an instance of the k-center problem. Remember the VC problem is to determine for a given graph $G = (V, E)$ whether there is a set of vertices with size at most $k$ such that all vertices are either in the set or adjacent to a vertex in the set by an edge in $G$. Given instance $(G = (V, E), k)$ of VC, we construct an instance $(G', d')$ of metric k-center problem $(G', d')$ as follows:

1. $G' \leftarrow G$
2. Remove isolated vertices (those with no edges) from $G'$
3. Make $G'$ a complete graph by adding all missing edges
4. Define metric $d'$ for each edge $e$

$$d'(e) = \begin{cases} 1 & \text{if } e \in E \\ 2 & \text{if } e \notin E \end{cases}$$

This is a polynomial time construction. Further, this construction would allow us to distinguish yes and no cases for VC in polynomial time by running a factor $(2-\epsilon)$ approximation algorithm for metric k-center on $(G', d')$. This is because if $G$ has a vertex cover of size at most $k$, then $OPT(G') = 1$, otherwise $OPT(G') = 2$.

Suppose algorithm $A$ is a factor $(2-\epsilon)$ approximation algorithm for the metric k-center problem.

$$OPT(G') = 1 \Rightarrow Val(A(G')) \leq (2-\epsilon)$$
$$OPT(G') = 2 \Rightarrow Val(A(G')) \geq 2$$

These are easily distinguishable cases, so if such an algorithm $A$ exists, $\mathsf{P} = \mathsf{NP}$. □

# 3 Set Cover

In the Set Cover problem, we have a bunch of items which we want to cover, and several sets, each covering some number of items, and each with a cost to take up the set. The goal is to cover all items, taking up the the minimum cost selection of sets to do so.

**Given:**

- Universe $U = [m]$
- $S_1, S_2, .., S_n \subseteq U$
- $w_1, w_2, .., w_n \in [0, \infty)$

**Goal:** Find set $I \subseteq [n]$ s.t.

$$\bigcup_{i \in I} S_i = U, \text{ and}$$
$$\sum_{i \in I} w_i \text{ is minimized}$$

Note that the requirement of positive weights is not a limitation. If we do have negative weights, to minimize the total weight we should choose all the subsets with negative weights. Then, we remove the elements in those subsets from the universe. Finally, it becomes a Set Cover problem with non-negative weights.

In fact, Set Cover can be seen as a generalization of Vertex Cover. To reduce Vertex Cover to Set Cover, let $U$ be the set of edges, $S_i$ be the set of edges that connect node $i$ to another node, and $w_i = 1$. This is a restricted version of Set Cover. The subsets are unweighted; i.e., all weigh the same. Also, $f \equiv \max_{u_j \in U} |\{i : u_j \in S_i\}| = 2$, because an edge has exactly 2 endpoints.

For this restricted version of Set Cover, we can directly apply the greedy algorithm for Vertex Cover, which gives a factor of 2 approximation. In general, if we relax $f$ to be any number, the greedy technique of Vertex Cover will give us a factor $f$ approximation. However, if there is any element that is in every subset, then $f = m$ and we gain nothing from the approximation.

## 3.1 Factor $(1 + \ln m)$ approximation algorithm

A greedy algorithm for Set Cover is presented below. The idea is to keep adding subsets that have minimum marginal cost per new element covered until all elements in $U$ are covered.

**Algorithm 2:** Set Cover Greedy Algorithm
(1) $C \leftarrow \emptyset$
(2) $I \leftarrow \emptyset$
(3) **while** $C \neq U$
(4) Pick $i \in [n]$ s.t. $|S_i \cap \overline{C}| > 0$ and $\frac{w_i}{|S_i \cap \overline{C}|}$ is minimized
(5) $I \leftarrow I \cup \{i\}$
(6) $C \leftarrow C \cup S_i$
(7) **return** $I$

**Theorem 3.** *The greedy algorithm for Set Cover is an approximation algorithm with factor $H_m$, where $H_m = \sum_{j=1}^{m} \frac{1}{j}$.*

*Proof.* Let $u_1, u_2, ..., u_m$ be the order in which elements of $U$ are added to $C$ (This ordering is important for the following proof). We define $Price(u_j)$ to be the marginal cost per element at the time $u_j$ is added, i.e.

$$Price(u_j) = \frac{w_i}{|S_i \cap \overline{C}|} \text{ at the time } u_j \text{ is added.}$$

Note that $Cost(I) = \sum_{i \in I} w_i = \sum_{j=1}^{m} Price(u_j)$.

**Claim 3.** *At the time $u_j$ is covered, $Price(u_j) \leq \frac{OPT}{|\overline{C}|}$, where $OPT$ denotes the cost of an optimal solution.*

To see this, note that there is an optimal cover, $I_{OPT}$. If we knew $I_{OPT}$, then we could select all at once those $i$ in $I_{OPT}$ that are not yet in $I$ for extra cost no greater than $OPT$. Selecting these optimal sets would yield an average $Price$ for the elements in $\overline{C}$ of no more than $\frac{OPT}{|\overline{C}|}$. Because the minimum value in a group must be no more than the average value in the group, there must be some set in this optimal selection which alone would cover one or more elements at $Price$ no more than $\frac{OPT}{|\overline{C}|}$. Since the $Price$ for a set cannot decrease over time, and our algorithm selects the set with minimum $Price$, we know that this bound is achieved.

**Claim 4.** *At the time $u_j$ is covered, $|\overline{C}| \geq m - j + 1$.*

Let $u_{j'}$ be the "first" of the elements covered in the iteration in which it is covered. This means that the number covered so far is $|C| = (j' - 1)$, and we also have $j \geq j'$, so

$$\begin{aligned} |\overline{C}| &= |U| - |C| \\ &= m - (j' - 1) \\ &= m - j' + 1 \\ &\geq m - j + 1 \end{aligned}$$

By Claim 3 and Claim 4, we have

$$\begin{aligned} Cost(Solution) &= \sum_{j=1}^{m} Price(u_j) \\ &\leq \sum_{j=1}^{m} \frac{OPT}{m - j + 1} \\ &= \left( \sum_{j=1}^{m} \frac{1}{j} \right) OPT \\ &= (H_m) OPT && (H_m : mth \text{ harmonic number}) \\ &\leq (1 + \ln m) OPT \end{aligned}$$

□

Note that $(\int_1^m \frac{1}{x} dx) \leq (\sum_{j=1}^{m} \frac{1}{j}) \leq (1 + \int_1^m \frac{1}{x} dx)$. Thus, $\ln n \leq H_n \leq 1 + \ln n$.

### 3.2 Tightness example

Is the analysis of the algorithm tight? The answer is yes. The following is an example.

*Example:* Let $S_1 = \{u_1\}, S_2 = \{u_2\}, ..., S_m = \{u_m\}, S_{m+1} = \{u_1, u_2, ..., u_m\}$, $w_i = \frac{1}{m-i+1}$(for $i = 1..m$), and $w_{m+1} = 1 + \epsilon$. The optimal solution is to choose $S_{m+1}$ alone, which gives $OPT = 1 + \epsilon$. On the other hand, the approximation algorithm will choose $S_1, S_2, ..., S_m$ and the cost is $H_m$. $\boxtimes$