Approximation Algorithm

Approximation Ratios and optimizations problems

We are trying to minimize (or maximize) some cost function c(S) for an optimization problem. E.g.

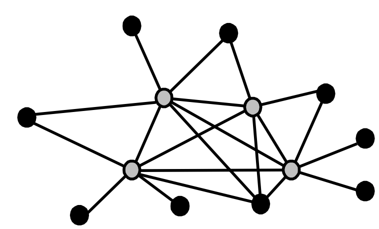
- Finding a minimum spanning tree of a graph.
  - Cost function – sum of weights of edges in the graph
- Finding a cheapest traveling salesperson tour (TSP) in a graph.
- Finding a smallest vertex cover of a graph
  - Given G(V,E), find a **smallest** set of vertices so that each edge touches at least one vertex of the set.

Approximation Ratios

- An approximation produces a solution T
  - T is a **δ-approximation** to a minimization problem if c(T) ≤ δ· OPT
  - We assume δ>1
  - **Examples:**
  - Will show how to find a p path in a graph, that visits all vertices, and w(p) ≤ δ w(p*). Here p* is the cheapest TSP path.

Vertex Cover

- A **vertex cover** of graph G=(V,E) is a subset $C \subseteq V$ of vertices, such that, for every (u,v) ∈ E, either $u \in C$ or $v \in C$ (or both $\in C$)
- Application:
  - Given graph of Facebook friends, find set of influencers - vertices that cover all edges of the graph.
  - Given maps of roads, find junctions to place monitoring cameras, so we could monitor the whole traffic.
- OPT-VERTEX-COVER: Given an graph G, find a vertex cover of G with smallest size.
- OPT-VERTEX-COVER is NP-hard.



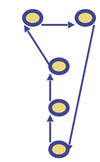
A 2-Approximation for Vertex Cover

Algorithm *VertexCoverApprox*(G)
  Input graph G
  Output a vertex cover C for G
  $C \leftarrow$ empty set ; $H \leftarrow E$
  /* H – what is left to be covered */
  while H has edges (not empty){
    $(u,v) \leftarrow$ An edge of H.
    Add both u and v to C
    for each edge f **of** H incident to v or w
      Remove f **from** H
  }
  return C

- Analysis: How large could C be, comparing to OPT ?
- Let OPT be the opt solution.
- Every chosen edge e has both ends in C.
- But e must be covered by at least one vertex of OPT. So, one end of e must be in OPT.
- $|C| \leq 2\ |OPT|$.
- (there are ≤ 2 vertices of C for each vertex of OPT.)
- That is, C is a 2-approx. of OPT
- Running time: O(|E|)

Approximating the Traveling Salesperson Problem (TSP)

- OPT-TSP: Given a weighted graph $G(V,E)$, find a cycle of minimum cost that visits each vertex at least once.
- OPT-TSP is NP-hard
- However, it is very easy to find a tour that costs $\leq$ twice opt.
- First Step: Compute the Minimum Spanning Tree MST(G) (for example, using Kruskal algorithm)
- Just to remind ourself: MST(G) is a set of edges which are
  1. Contains every vertex of V
  2. Connected (a path from every vertex to every other vertex). That is, it **spans** G.
  3. Among all the graphs satisfying (1) +(2), has the smallest sum of weights of edges.
- Observation: The edges of TSP, they also span G



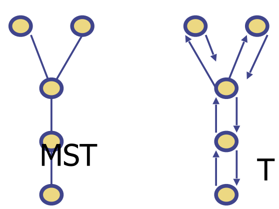
From MST to cycles

Given a MST of G, a traversal T of MST is constructed by picking a source vertex s, and visit the nodes of the graph in a DFS order.

- Let w(MST) and w(OPT-TSP) be the sum of weights of edges of MST and of OPT-TSP. (an edge is counted once, even if appearing multiple times).
- Cost(OPT-TSP) $\geq w(OPT-TSP)$, since possibly the same edge was used more than once.
- Claim: $w(\text{OPT-TSP}) \geq w(MST)$
  - (explanation: Both OPT-TSP and MST spans G, but OPT-TSP optimize other parameter, which MST minimizes sum of weights.
- T is a tour that uses twice every edge of MST. so $w(T) = 2w(MST)$.
- OPT-TSP is a spanning graph (graph that connects all vertices of V.)
  Obviously $Cost(T) \geq cost(\text{OPT-TSP})$. However

$$cost(\text{OPT-TSP}) \geq w(\text{OPT-TSP}) \geq w(MST)$$
$$2cost(\text{OPT-TSP}) \geq 2 \cdot w(\text{OPT-TSP}) \geq 2 \cdot w(MST) = cost(T)$$

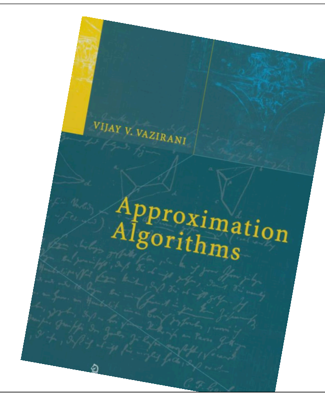
Conclusion: Traversing MST gives a factor 2 approx to TSP.



Approximation Algorithm for Set Cover

Dave's Mount Lecture Notes:

Dorit S. Hochbaum and Anu Pathria. Analysis of the Greedy Approach in Problems of Maximum k-Coverage. Naval Research Logistics, Vol. 45 (1998)

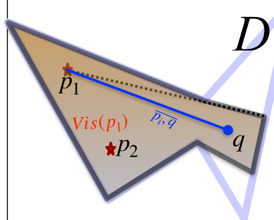


Set-Cover Problems

Facility location problems: Given: A map of Tucson, place min number of charging station, so every house is at distance ≤ 5 miles from a charging station,

Budget Set Cover. With a budget of $\leq k$ stations, cover as much of Tucson as possible.

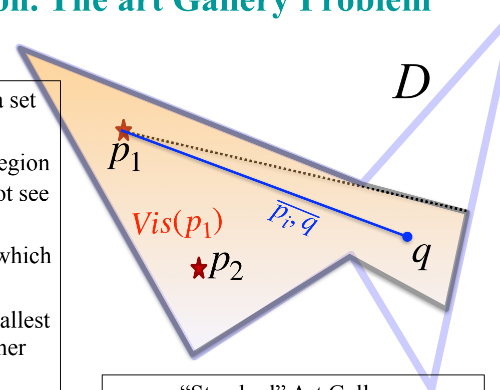
- Given - a polygon domain D, and a set $P = \{p_1 \ldots p_n\}$ of **potential** guard - we **might** place a camera at p_i .
- Each potential guard p_i **sees** some region $Vis(p_i)$ of the polygon, but could not see through walls.
- Formally, p_i sees every point q for which the segment $\overline{p_i\, q}$ is fully in D.
- **Art Gallery Problem** - find the smallest set of guards (all from P) that together see the whole D.
- Budget Art Gallery - with at most k guards, see as much as possible.



- Set cover is NP-hard (and extremely practical)
- $a_i = Area(Vis(p_i))$ the area (in meters^2) that it sees.

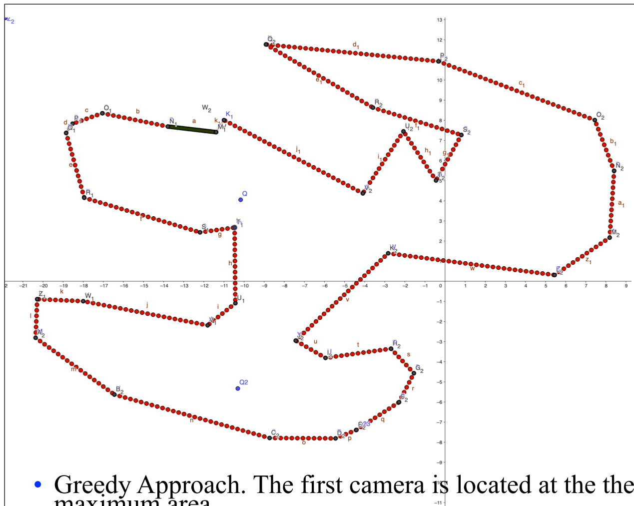
Visibility in a polygon. The art Gallery Problem

- Given - a polygon domain D, and a set $P = \{p_1 \ldots p_n\}$ of potential guards.
- Each potential guard p_i sees some region $Vis(p_i)$ of the polygon, but could not see through walls.
- Formally, p_i sees every point q for which the segment $\overline{p_i\, q}$ is fully in D.
- **Art Gallery Problem** - find the smallest set of guards (all from P) that together see the whole D.
- NP-hard (and extremely practical)
- $\mu_i = Area(Vis(p_i))$ the area (in meters^2) that it sees.
- Budget Art-Gallery Problem: Given a number k (\`budget'), find a set G of $\leq k$ guards from P, that sees together the maximum area.

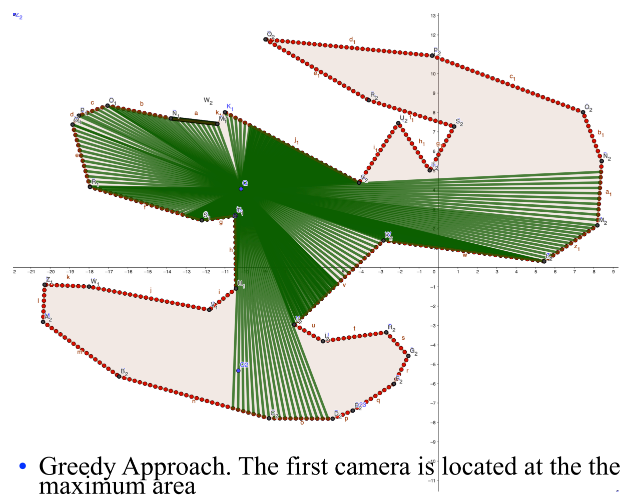


"Standard" Art Gallery:
Find the **smallest** set $\{g_1, g_2 \ldots g_r\} \subseteq P$
s.t
$D = Vis(g_1) \cup Vis(g_i) \cup \ldots Vis(g_r)$

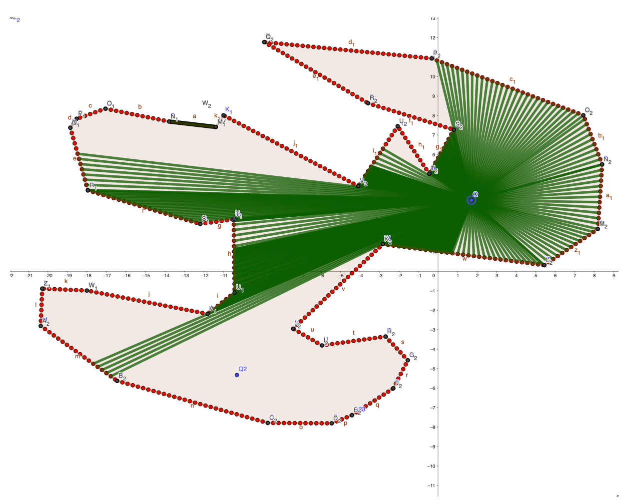
Budget Art Galley:
Given k, find $\{g_1, g_2 \ldots g_k\} \subseteq P$
Maximize
$Area(\ Vis(g_1) \cup Vis(g_2) \cup \ldots Vis(g_k))$

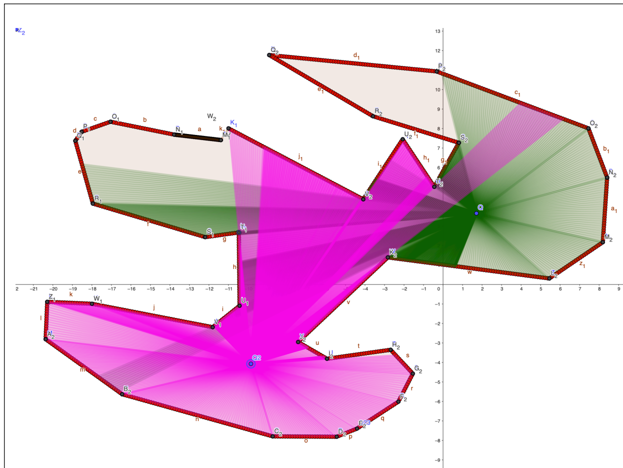


- Greedy Approach. The first camera is located at the the point of P that sees maximum area
- The second camera g_2 is located where it sees the maximum area **that** g_1 **does not see**
- g_3 sees the max area not seen by neither g_1 nor g_2, etc…
- Stop when either P is covered, or (in the budget case) when used k cameras.

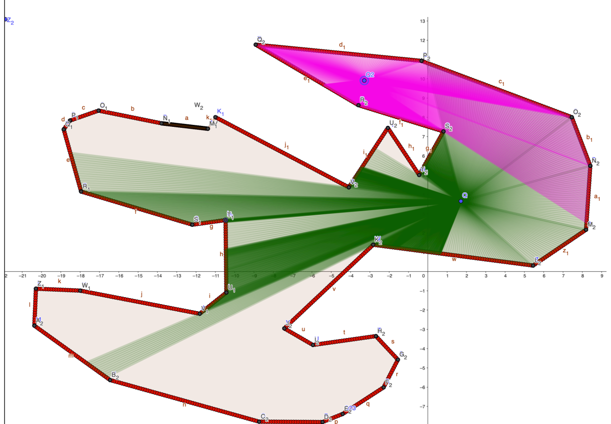


- Greedy Approach. The first camera is located at the the point of P that sees maximum area
- The second camera g_2 is located where it sees the maximum area **that** g_1 **does not see**
- g_3 sees the max area not seen by neither g_1 nor g_2, etc…
- Stop when either P is covered, or (in the budget case) when used k cameras.



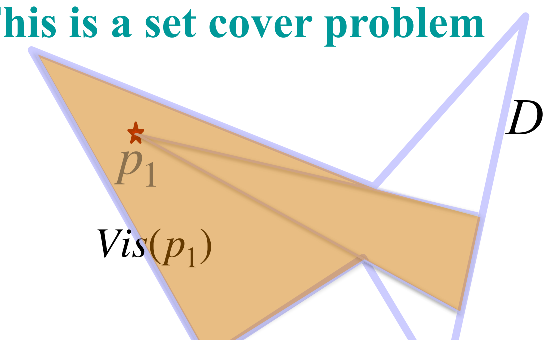


- Greedy Approach. The first camera is located at the the point of P that sees maximum area
- The second camera g_2 is located where it sees the maximum area **that** g_1 **does not see**
- g_3 sees the max area not seen by neither g_1 nor g_2, etc…
- Stop when either P is covered, or (in the budget case) when used k cameras.



- Greedy Approach. The first camera is located at the the point of P that sees maximum area
- The second camera g_2 is located where it sees the maximum area **that** g_1 **does not see**
- g_3 sees the max area not seen by neither g_1 nor g_2, etc…
- Stop when either P is covered, or (in the budget case) when used k cameras.

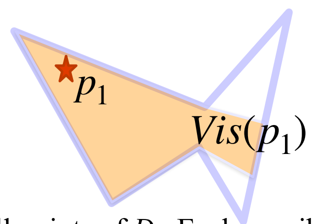
This is a set cover problem



- Given - a polygon domain D, and a set $P = \{p_1 \dots p_n\}$ of potential guards.
- Every potential guard p_i defines a **set**. This set is $Vis(p_i)$. A set cover problem is to find a collection of sets that together covers the whole domain.
- Greedy Approach. The first guard is the point that sees maximum area $g_1 = \arg\max_{p \in P} \mu(p)$
- The second guard g_2 sees the maximum area that g_1 does not see
- g_3 sees the max area not seen by neither g_1 nor g_2, etc…

Set Cover Problems - terminology

General problem: Given a **universe** $X = \{x_1 \ldots x_m\}$, each x_i is an **atoms**.
Also given a range space (also called set system). It is a collection of subsets of X. $\mathbf{R} = \{S_1, S_2 \ldots\}$ a collection of subsets of X. $(S_i \subseteq X)$

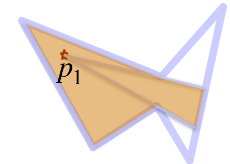


Examples:

1. In a polygon D, the atoms are all points of D. Each possible guard p_i defines $Vis(p_i)$. $\mathbf{R} = \{Vis(p_i) \ | \ p_i \in P\}$
2. Given a graph $G(V, E)$, we could treat V as the universe. Each edge is a set of two atoms. (edge-cover)
3. In a graph $G(V, E)$, the atoms are the **edges**. Each vertex $v_i \in V$ defines the set S_i of all the edges that v_i is adjacent to. (vertex cover)

Set Cover Problems

General problem: Given a **universe** $X = \{x_1 \ldots x_m\}$, each x_i is an atoms.
Also given R=$\{S_1, S_2 \ldots\}$ a collection of subsets of X. $(S_i \subseteq X)$



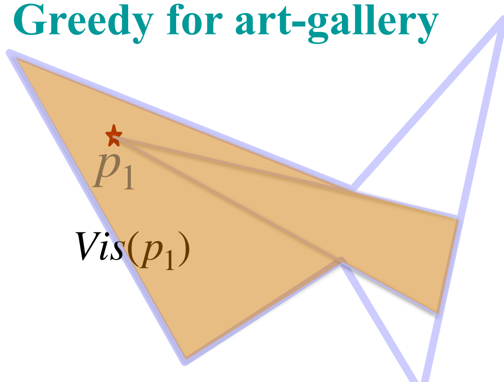
Examples:

1. In a polygon D, the atoms are all points of D. Each possible guard p_i define $Vis(p_i)$ which is the region of D that p_i sees.
2. Given a graph $G(V, E)$, we could treat V as the universe. Each edge is a set of two atoms. (edge-cover)
3. In a graph $G(V, E)$, the atoms are the **edges**. Each vertex $v_i \in V$ defines the set S_i of all the edges that v_i is adjacent to.

- We say that $\mathbf{C} = \{S'_1, S'_2 \ldots S'_k\}$ covers X if $X = S'_1 \cup S'_2 \cup \ldots \cup S'_k$, (that is, each atom is contained in at least one set of **C**.)
- **Set cover problem**: Find the collection $\mathbf{C} = \{S'_1, S'_2 \ldots S'_k\}$ of min number of sets that covers X. That is, minimize $|\mathbf{C}|$.
- In the case of the art-gallery problem, find a min-cardinality set of guards that see all the polygon .

- **Budget Set cover problem:** Given an integer $k > 1$ (budget).
- Find a collection $\mathbf{C} = \{S'_1, S'_2 \ldots S'_k\}$ of no more than k sets from **R** such that the number of atoms in $S'_1 \bigcup S'_2 \bigcup \ldots \bigcup S'_k$ is as large as possible.
- In the case of the art gallery, find a set of k guards that see as much as possible.

Greedy for art-gallery



1. Pick the guard g_1 maximizing the area of $g_1 = \arg_{g \in D} \ area(Vis(g))$
2. Pick the guard g_2 maximizing the area not seen by g_1.
   $g_2 = \arg_{g \in D} \ area\big(Vis(g) \setminus Vis(g_1)\big)$
3. Pick the guard g_3 maximizing the area not seen by g_1.
   $g_3 = \arg_{g \in D} \ area\big(Vis(g) \ \setminus (Vis(g_1) \cup Vis(g_2))\big)$

   Etc until either seeing all D, or until using k guards.

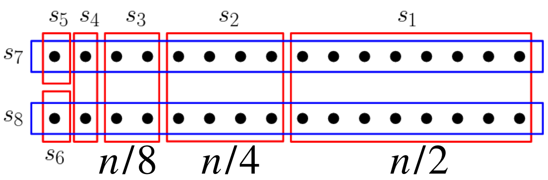
Greedy Algorithm for the budget case:

$X' = X = \{x_1 \ldots x_m\}$
For i=1 to k {
  Let S'_i be the set $S \in \mathbf{R}$ that maximizing $|S \bigcap X'|$.
  $X' \leftarrow X' \ \setminus \ S'_i$ *//Only care for uncovered atoms*
  }
Return $[S'_1, S'_2 \ldots S'_k]$

For standard set cover, the stoping condition is that $X' = \emptyset$ Algorithm for the budget case:

Dorit S. Hochbaum and Anu Pathria. Analysis of the Greedy Approach in Problems of Maximum k-Coverage. Naval Research Logistics, Vol. 45 (1998)

Greedy could be far away from opt, if we insist of covering X



Opt: $\{s_5, s_6\}$

Greedy: $\{s_1, s_2, s_3, s_4, s_5, s_6\}$

- It is known that it could be much worse than opt.
- In the opt problem above, $Opt = \{s_7, s_8\}$ (two sets)
- Greedy might start from s_1, then pick s_2 … could be $\geq \log_2 n$
- Approximation factor:
- Approximation factor $= \dfrac{\text{Numer of sets that greedy finds}}{\text{Numer of sets that OPT finds}} = \dfrac{\log_2 n}{2} = \Omega(\log n)$
- This is actually a tight bound (we will see shortly)

- However, greedy is doing much better for the budget case (number of sets is given ***k*** - maximize the area / the number of atoms

Analysis of greedy algorithm for the budget set-cover problem

Consider a range space (atoms and subsets)

Let k (the budget) be a given fixed positive integer.

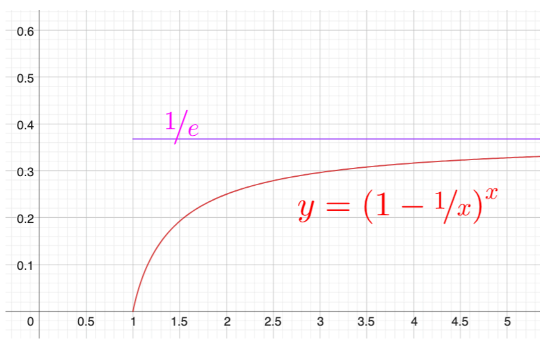
Let OPT be the maximum number of atoms that we can cover with $\leq k$ sets

Theorem: Greedy algorithm will produces a solution that covers $\geq OPT\left(1 - \frac{1}{e}\right) \approx 0.64 \cdot OPT$

Before we start - useful inequalities

If $k \geq 1$ then $\left(1 - \frac{1}{k}\right)^k \leq \frac{1}{e}$

Conclusion $1 - \left(1 - \frac{1}{k}\right)^k \geq 1 - \frac{1}{e}$



Proof: Analysis of the Greedy Algorithm (for the budget case)

- Let OPT be the maximum area that k guards could see.
  …(or the max number of atoms that ***k*** sets could contain)
- Let a_l be the gain from adding the l's set (for $1 \leq l \leq k$). How much area did the l'th add.
- Let $s(l) = a_1 + a_2 + \ldots a_l$ be the total area guard by that the first l guards that greedy picked.

$(1 - 1/k)^k \leq \frac{1}{e}$

Claim: $a_1 \geq OPT / k$.

Lemma 1 : $a_l \geq \dfrac{OPT - s(l-1)}{k}$. (this is the heart of the proof.)

Lemma 2: $s(l) \geq OPT \cdot \left\{1 - \left(1 - \frac{1}{k}\right)^l\right\}$

Lets assume that the lemma is proven.
This will implies that greedy, after picking the k'th set, covers area which is at least
$s(k) \geq OPT\left\{1 - (1 - 1/k)^k\right\} \geq OPT \cdot (1 - \frac{1}{e}) \approx 0.64 \cdot OPT$

That is, greedy reaches 64% of what OPT could reach.

Let a_l be the gain from adding the l's set (for $1 \le l \le k$). Let $s(l) = a_1 + a_2 + \ldots a_l$ be the number of atoms of the first l sets that greedy picks. So a_l is the gain from adding the l's set (the atoms that were not covered by previous sets)

$\boxed{\text{Claim: } a_1 \ge \frac{OPT}{k}}$ $\boxed{\text{Lemma 1: } a_{l+1} \ge \frac{OPT - s(l)}{k}}$ $\boxed{\text{Lemma2: } s(l) \ge OPT\left(1 - (1 - 1/k)^l\right)}$

Proof Lemma 2 by induction on l.

The base case $l = 1$ is just Lemma 1, since $s(0) = 0$. Let $Q = (1 - 1/k)$.

Induction hypothesis claims that $s(l) \ge (1 - Q^l)OPT$.

Need to show $s(l+1) \ge (1 - Q^{l+1})OPT$.

Proof: $s(l+1) \overset{\text{def}}{=} s(l) + a_{l+1} \overset{\text{Lemma 1}}{\ge} s(l) + \frac{OPT - s(l)}{k} =$

$= s(l)\left(1 - \frac{1}{k}\right) + \frac{OPT}{k} = s(l)Q + \frac{OPT}{k} \overset{\text{Induction}}{\ge}$

$Q(1 - Q^l)OPT + \frac{OPT}{k} = (1 - \frac{1}{k})OPT - Q^{l+1}\,OPT + \frac{OPT}{k} = (1 - Q^{l+1})OPT$. QED

Another attempt to prove the theorem, possibly in a simpler way

$\boxed{\text{Claim: } a_1 \ge \frac{OPT}{k}}$ $\boxed{\text{Lemma 1: } a_{l+1} \ge \frac{OPT - s(l)}{k}}$

- Let a_i be the gain from adding the i's set (for $1 \le i \le k$). Let $s(l) = a_1 + a_2 + \ldots a_l$ be the number of atoms of the first l sets that greedy picks. So a_l is the gain from adding the l's set (the atoms that were not covered by previous sets)
- Let $\Delta(i) = OPT - s(i)$ denote the gap between OPT and what greedy gained till the i'th step. Note: $\Delta(0) = OPT$

Note that lemma 1 implies $a_i \ge \frac{\Delta(i-1)}{k}$. So

$$\Delta(i) = \Delta(i-1) - a_i$$
$$\le \Delta(i-1) - \frac{\Delta(i-1)}{k}$$
$$= \Delta(i-1)\left(1 - \frac{1}{k}\right)$$

This implies

$$\Delta(k) \le \Delta(k-1)\left(1 - \frac{1}{k}\right) \le$$
$$\le \left\{\Delta(k-2)\left(1 - \frac{1}{k}\right)\right\}\left(1 - \frac{1}{k}\right) = \Delta(k-2)\left(1 - \frac{1}{k}\right)^2$$
$$\le \left\{\Delta(k-3)\left(1 - \frac{1}{k}\right)\right\}^2\left(1 - \frac{1}{k}\right) = \Delta(k-3)\left(1 - \frac{1}{k}\right)^3$$
$$\le \ldots$$
$$\Delta(0)\left(1 - \frac{1}{k}\right)^k = OPT\left(1 - \frac{1}{k}\right)^k \le OPT\left(\frac{1}{e}\right)$$

Using the fact that $(1 - 1/k)^k \le \frac{1}{e}$

Hence $\Delta(k) = OPT - s(k) \le OPT/e$,

Thus : $s(k) \ge OPT(1 - 1/e)$ QED

This result is even stronger :

In many cases, it is hard or impossible to compute the largest set at each stage.

Assume that greedy picks at every stage a set S_i such that $size(S_i) \ge \beta \cdot size(\max(S))$. That is, we only pick a $\beta-$approximation to the largest set.

Then $Greedy \ge OPT\left(1 - \frac{1}{e^\beta}\right)$

For example, if $\beta = 0.95$, then we get $Greedy \ge OPT \cdot 0.61$

This result is even stronger :

Note that $f(A \cup \{x\}) - f(A)$ measures the benefit of adding an atom x to the set A.

Submodularity: We say that a function $f(S)$, is submodular if

$\forall A \subseteq B, \qquad f(A \cup \{x\}) - f(A) \ge f(B \cup \{x\}) - f(B)$

That is, this is the idea of **diminishing return**, if B is larger than A ($A \subseteq B$) then adding x to B is less significant than adding it to A.

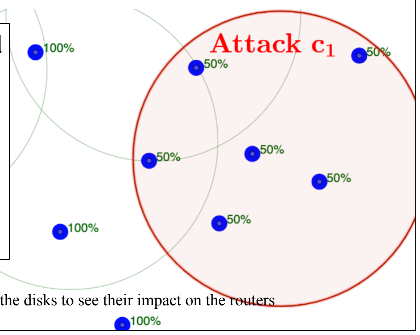
Theorem: Using the budget-greedy algorithm on any sub modular function yields a solution that is $\ge (1 - 1/e)OPT$.

Submodularity and greedy. The EMP case

- We have a items (routers), indicated by blue points $P = \{p_1 \dots p_n\}$
- Electromagnetic pulse (EMP) attack *might* cause some of them to burn.
- Each attack is abstracted by a disk. Each router p_i outside the disk is not effected. A router p_i inside the disk c has 50% chance to survive.
- Given a budget of k attacks, and a possible set of candidate attacks, where should the attacker place the k attacks $S = \{c_1, c_2 \dots c_k\}$ to maximize the **expected** damage $\sum_{p_i \in P} Pr(p_i \text{ did not survice the attackS })$
- NP-hard, so use greedy.

For a subset $\{c_1, c_2 \dots c_k\}$ of attack disks, and a router $p \in \mathbb{R}^2$, lets define $depth(p, S) =$ #disks from S containing p.

Reward function
$f(S) = f(\{c_1, c_2 \dots c_k\}) = \sum_{p_i \in P} 2^{-depth(p,S)}$



In this app (link) add attack disks by changing the slides k, and move the disks to see their impact on the routers

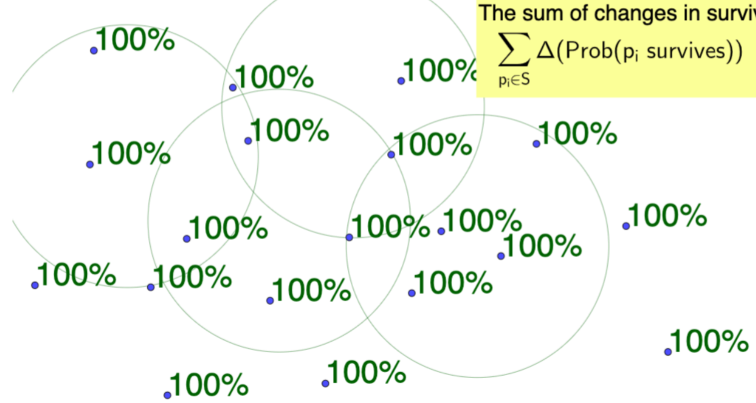
---

Example of submodularity and greedy: An "Attack" is limited to a disk.

Each element in the attack region survives with probability 1/2.

The number next to each item is its survival probability

Revenue from an attack:
The sum of changes in survival probabilitie
$\sum_{p_i \in S} \Delta(\text{Prob}(p_i \text{ survives}))$

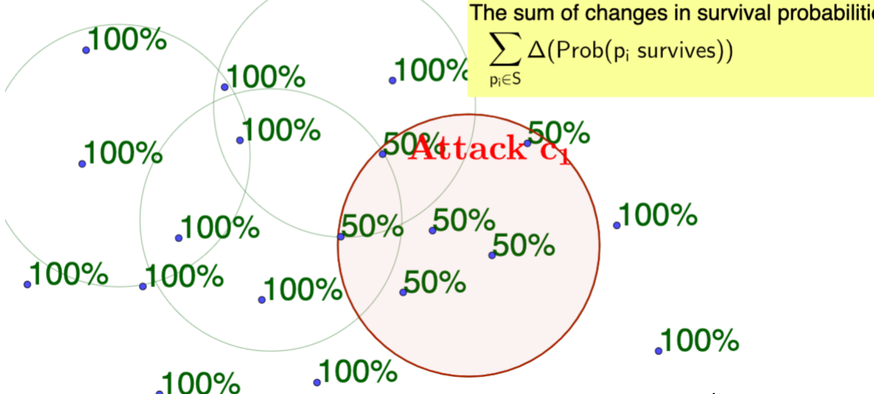


Example of submodularity and greedy: An "Attack" is limited to a disk.

Each element in the attack region survives with probability 1/2.

The number next to each item is its survival probability

Revenue from an attack:
The sum of changes in survival probabilitie
$\sum_{p_i \in S} \Delta(\text{Prob}(p_i \text{ survives}))$



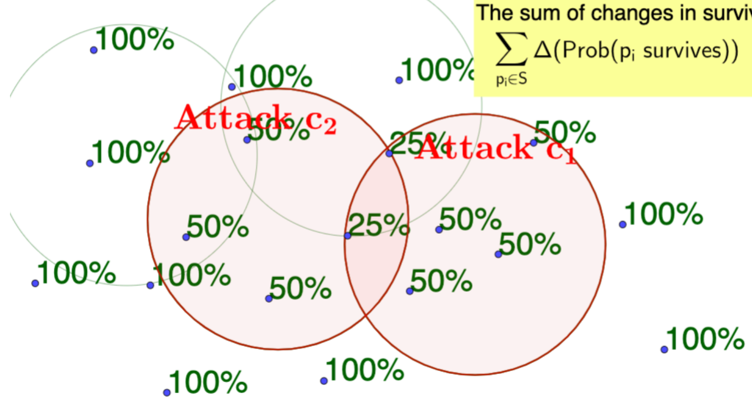
revenue: $6 \cdot 50\,\%$

Example of submodularity and greedy: An "Attack" is limited to a disk.

Each element in the attack region survives with probability 1/2.

The number next to each item is its survival probability

Revenue from an attack:
The sum of changes in survival probabilitie
$\sum_{p_i \in S} \Delta(\text{Prob}(p_i \text{ survives}))$



Example of submodularity and greedy: An "Attack" is limited to a disk.

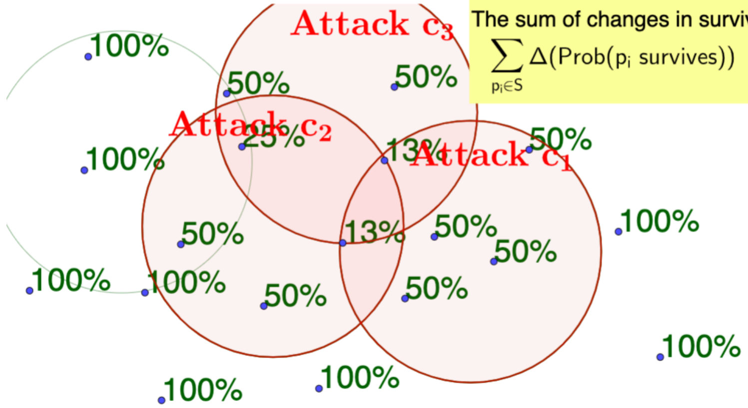
Each element in the attack region survives with probability 1/2.

The number next to each item is its survival probability

Revenue from an attack:

The sum of changes in survival probabiliti

$$\sum_{p_i \in S} \Delta(\text{Prob}(p_i \text{ survives}))$$



Example of submodularity and greedy: An "Attack" is limited to a disk.

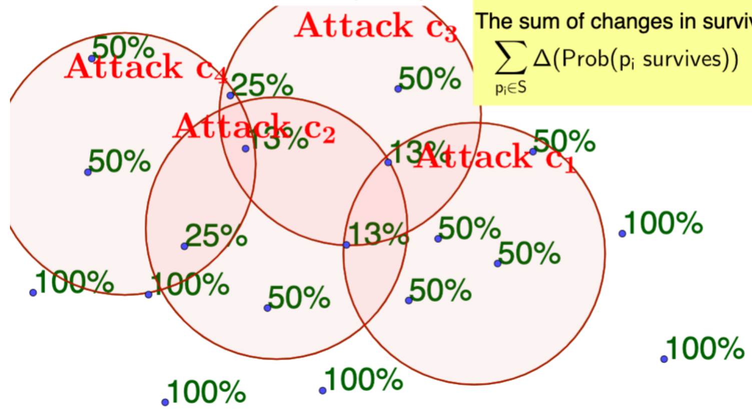
Each element in the attack region survives with probability 1/2.

The number next to each item is its survival probability

Revenue from an attack:

The sum of changes in survival probabilities

$$\sum_{p_i \in S} \Delta(\text{Prob}(p_i \text{ survives}))$$



Back to covering the whole universe

Given a universe $X = \{x_1 \ldots x_m\}$ with $\boldsymbol{m}$ atoms, each x_i is an atoms. R=$\{S_1, S_2 \ldots\}$is a collection of subsets of X. ($S_i \subseteq X$). Find the smallest collection $\mathbf{C} = \{S'_1, S'_2 \ldots S'_k\}$ that covers X.

Again use greedy, but stop when X is covered. How many sets greedy produces ?

- $s(i) = a_1 + a_2 + \ldots a_i$.

Recall $\Delta(\mathbf{i}) = OPT - s(\mathbf{i}) \le OPT(1 - 1/k_{opt})^{\mathbf{i}}$.

k_{opt} =min number of sets that covers X

Now OPT is the universe with m atoms, so $\Delta(i) = m - s(i) \le m(1 - 1/k_{opt})^i$.

Assume greedy needs $r + 1$ iteration until it covers all m atoms of the universe. At the **r'** th iteration (last but one) at least one atom is left uncovered, or $\Delta(r) \ge 1$.

How large could r be ?

$$1 \le \Delta(r) \le m(1 - 1/k_{opt})^r = m\{(1 - 1/k_{opt})^{k_{opt}}\}^{r/k_{opt}}$$

$$\le m\ \{1/e\}^{l/k_{opt}} = m/e^{r/k_{opt}} \text{ or } e^{r/k_{opt}} \le m$$

Take $ln(\cdot)$ from both sides, yield: $r/k_{opt} \le \ln m$ or $r \le k_{opt} \ln m$.

Theorem: Greedy gives $\ln m$ approximation factor to the smallest number of sets needed to cover X.

Thm: Could show: The actual bound is $\ln(\max|S_i|)$

Clustering points in $\mathbb{R}^d$ with bottleneck penalty

(Source: Mount's notes)

Given $S = \{p_1 \ldots p_n\} \in \mathbb{R}^d$ set of n points. Want to divide into k clusters (k is given)

Let $C = \{c_1 \ldots c_k\}$ be the set of centers of clusters. Naturally we would like to assign each p_i to one cluster. How ? We just assign p_i to the nearest cluster.

For every point q let $NN(q, C)$ be the nearest center of q.

Let $\Delta(C) = \max_{p_i \in S} dist(p_i, NN(p_i, C))$ the max distance to the nearest cluster. This is the **bottleneck distance**. And the pair (p_i, c_j) that defined this distance is the bottleneck pair.

$\Delta(C)$ is the measure of the quality of the clustering. The problem is to pick k centers so this distance is as small as possible. link link

Greedy Approx k-center

Given $S = \{p_1 \ldots p_n\} \in \mathbb{R}^d$ set of n points. Want to divide into k clusters (k is given)

The first center $c_1$in G_{greedy} is arbitrary - let's pick p_1 . This is the first center c_1

$G_{greedy} = \{p_1\}$; $\forall p_i \in S$ do $d[p_i] = |p_i - p_1|$

$for(i = 2 \ldots k)$

$c_i = \arg\max_{p_j \in S} d[p_j]$ / / find the bottleneck pair

Add c_i to G_{greedy}

$\forall p_j \in S$ do $d[p_j] = \min\{d[p_j], |p_j - c_i|\}$

return G_{greedy}

For the proof, we will also consider G_{k+1}, obtained by performing another iteration of the algorithm.

Theorem: $\Delta(G_{greedy}) \le 2\Delta(Opt)$

Proof: Let $G_i = G_{greedy}$ after i iteration of the algorithm. (for i=1..k)

$G_i = \{c_1, c_2 \ldots c_i\}$

Claim: $\Delta(G_i) \ge \Delta(G_{i+1})$ (for i=1..k)

Lemma: Let $r = \Delta(G_{i-1})$. The distance between every two centers in G_i is $\ge r$.

Proof: c_i is at distance exactly r from the nearest center of G_{i-1}, so $|c_i - c_l| \ge r$ for all $l < i$.

For smaller values of i, remember that $\Delta(G_l) \ge r$ (from the first claim)

Greedy Approx k-center

The first center $c_1$in G_{greedy} is arbitrary - let's pick p_1 . This is the first center c_1

$G_{greedy} = \{p_1\}$; $\forall p_i \in S$ do $d[p_i] = |p_i - p_1|$

$for(i = 1 \ldots k)$

$c_i = \arg\max_{p_j \in S} d[p_j]$ / / find the bottleneck pair

Add c_i to G_{greedy}

$\forall p_j \in S$ do $d[p_j] = \min\{d[p_j], |p_j - c_i|\}$

return G_{greedy}. Repeat one more iteration to produce G_{k+1}

Claim: $\Delta(G_i) \ge \Delta(G_{i+1})$ (for i=1..k)

Lemma: Let $r = \Delta(G_{i-1})$. The distance between any two centers in G_i is $\ge r$.

In particular, the distance between every two centers in G_{k+1} is $\ge \Delta(G_{greedy})$.

Theorem: $\Delta(G_{greedy}) \le 2\Delta(Opt)$

Pf of Theorem: Let $C' = \{c'_1 \ldots c'_k\}$ be any set of exactly k centers. (for example, opt)

By Definition of C', each $p \in S$ has a center $c \in C$ at distance $\le \Delta(C)$ from p.

Some center $c' \in C$ has two centers $c_i, c_j \in G_{k+1}$ in the cluster of c.

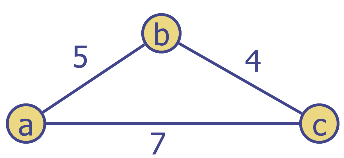
$|c_i - c_j| \ge \Delta(G_{greedy})$ (from the Lemma). So

$$\Delta(G_{greedy}) \le |c_i - c_j| \le |c'_r - c'| + |c'_r - c'|$$
$$\le \Delta(C') + \Delta(C') = 2\Delta(C')$$

QED

Approximating the Traveling Salesperson Problem

- OPT-TSP: Given a **complete**, weighted graph, find a cycle of minimum cost that visits each vertex.
  - OPT-TSP is NP-hard
  - Special case: edge weights satisfy the triangle inequality (which is common in many applications):
    - w(a,b) + w(b,c) ≥ w(a,c)



Complete – there is an edge between every pair of vertices

From MST to cycles

Given a MST of ***G***, a traversal ***T*** of ***MST*** is constructed by picking a source vertex *s*, and visit the nodes of the graph in a DFS order.

- Let w(MST) and w(OPT-TSP) be the sum of weights of edges of MST and of OPT-TSP.
- Since OPT-TSP does not visit a vertex twice, it does not use an edge twice. So its weight w(OPT-TSP) is the sum of weights of its edges.
- T is a tour that uses twice every edge of MST. so $w(T) = 2w(MST)$.
- OPT-TSP is a spanning graph (graph that connects all vertices of V .)

$$w(\text{OPT-TSP}) \ge w(MST)$$
$$2 \cdot w(\text{OPT-TSP}) \ge 2 \cdot w(MST) = w(T)$$