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.

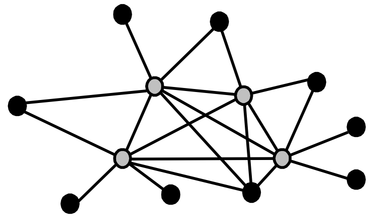
- Finding a minimum spanning tree of a graph.
  - Cost function – sum of weights of edges in the graph
- Finding a cheapest traveling salesperson tour (TSP) in a graph.
- Finding a smallest vertex cover of a graph
  - Given G(V,E), find a **smallest** set of vertices so that each edge touches at least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

Approximation Ratios

- An approximation produces a solution T
  - T is a **δ-approximation** to a minimization problem if $c(T) \le \delta \cdot OPT$
  - We assume $\delta > 1$
  - **Examples:**
  - Will show how to find a p path in a graph, that visits all vertices, and $w(p) \le \delta\, w(p^*)$. 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 $(u,v) \in E$, 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.



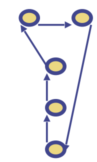
A 2-Approximation for Vertex Cover

Algorithm *VertexCoverApprox(G)*
Input graph ***G***
Output a vertex cover ***C*** for ***G***
C ← empty set ; ***H*** ← ***E***
/ H – what is left to be covered */*
while *H* has edges (not empty)**{**
(u,v) ← 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***

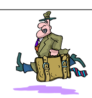
- Analysis: How large could C be, comparing to OPT ?
- Let OPT be the opt solution.
- Every chosen edge e has both ends in C.
- But e must be covered by at least one vertex of OPT. So, one end of e must be in OPT.
- |C| ≤ 2 |OPT|.
- (there are ≤ 2 vertices of C for each vertex of OPT.)
- That is, C is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

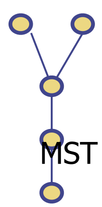


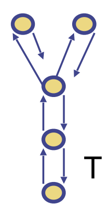
Given a MST of ***G***, a traversal ***T*** of ***MST*** is constructed by picking a source vertex *s*, and visit the nodes of the graph in a DFS order.

- 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) ≥ $w(OPT-TSP)$, since possibly the same edge was used more than once.
- Claim: w(OPT-TSP) ≥ $w(MST)$
  - (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 $Cost(T) \geq cost$(OPT-TSP). However

$cost$(OPT-TSP) ≥ w(OPT-TSP) ≥ $w(MST)$
2$cost$(OPT-TSP) ≥ 2 · w(OPT-TSP) ≥ 2 · $w(MST) = cost(T)$

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



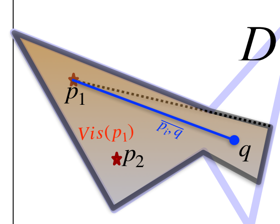


Set-Cover Problems

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,

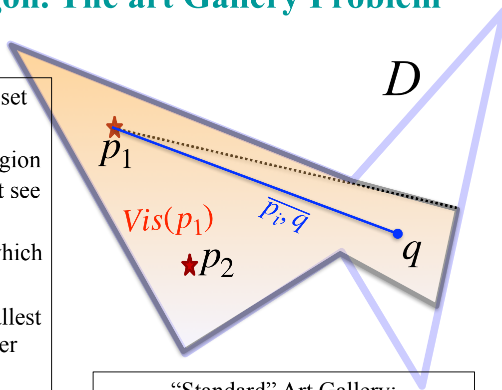
Budget Set Cover. With a budget of ≤ k stations, cover as much of Tucson as possible.

- Given - a polygon domain D, and a set $P = \{p_1 \dots p_n\}$ 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

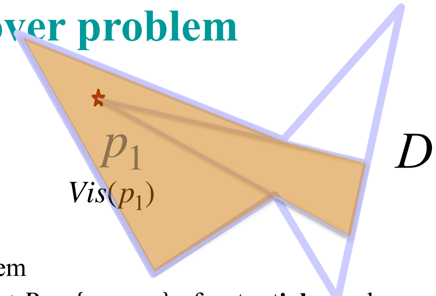


- Given - a polygon domain D, and a set $P = \{p_1 \ldots p_n\}$ of potential guards.
- Each potential guard p_i sees some region $Vis(p_i)$ of the polygon, but could not see through walls.
- Formally, p_i sees every point q for which the segment $\overline{p_i\ q}$ is fully in D.
- **Art Gallery Problem** - find the smallest set of guards (all from P) that together see the whole D.
- NP-hard (and extremely practical)
- $a_i = Area(Vis(p_i))$ 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.

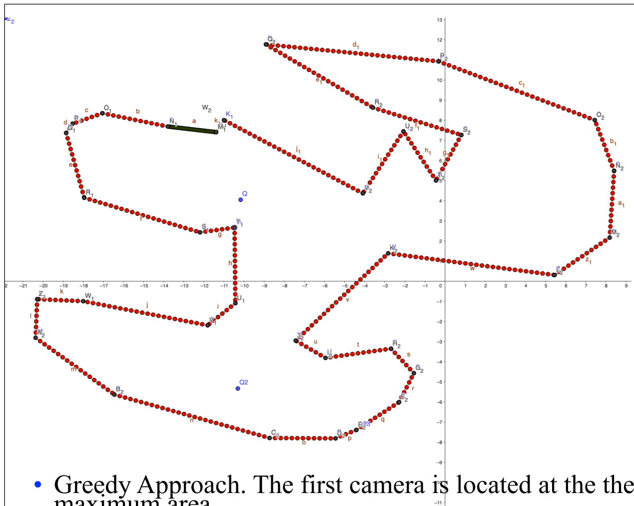
"Standard" Art Gallery:
Find the **smallest** set $\{g_1, g_2 \ldots g_r\} \subseteq P$
s.t
$D = Vis(g_1) \cup Vis(g_i) \cup .. Vis(g_r)$

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

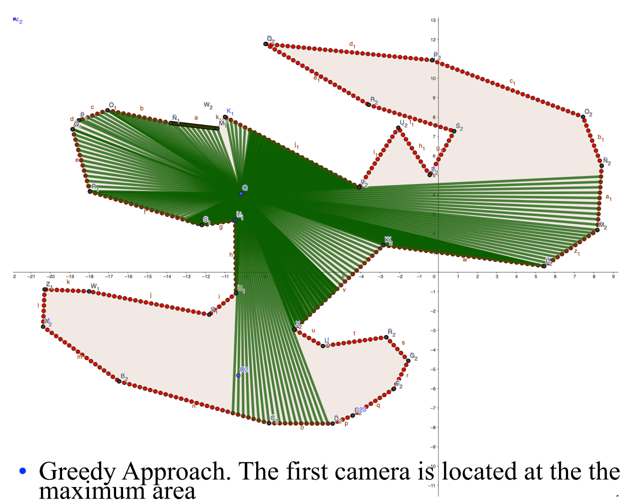
This is a set cover problem



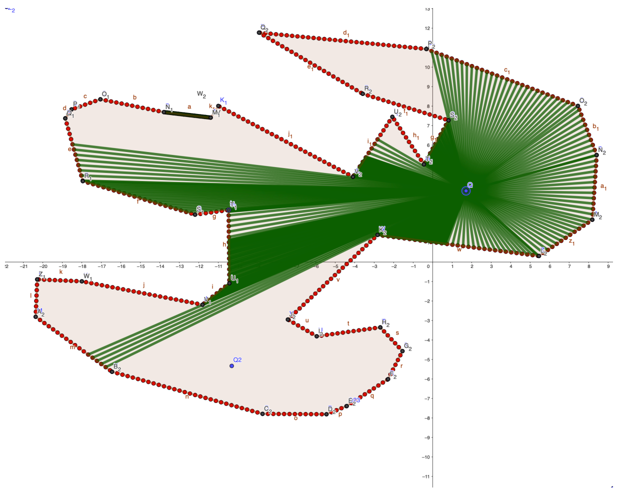
- Greedy Algorithm for set cover problem
- Given - a polygon domain D, and a set $P = \{p_1 \ldots p_n\}$ of **potential** guards.
- Every potential guard p_i defines a **set**. This set is $Vis(p_i)$. A set cover problem is to find a collection of sets that together covers the whole domain.
- Greedy Approach. The first camera is located at the the point of P that sees maximum area (in square feet)
- 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 D is covered, or (in the budget case) when used k cameras.

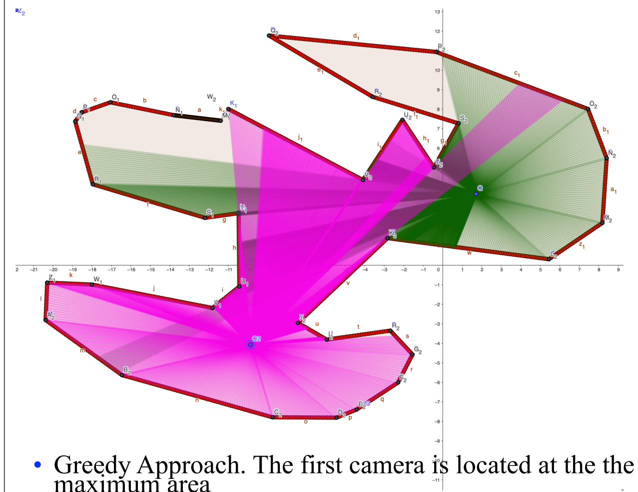


- Greedy Approach. The first camera is located at the the point of P that sees maximum area
- The second camera g_2 is located where it sees the maximum area **that** g_1 **does not see**
- g_3 sees the max area not seen by neither g_1 nor g_2, etc…
- Stop when either P is covered, or (in the budget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

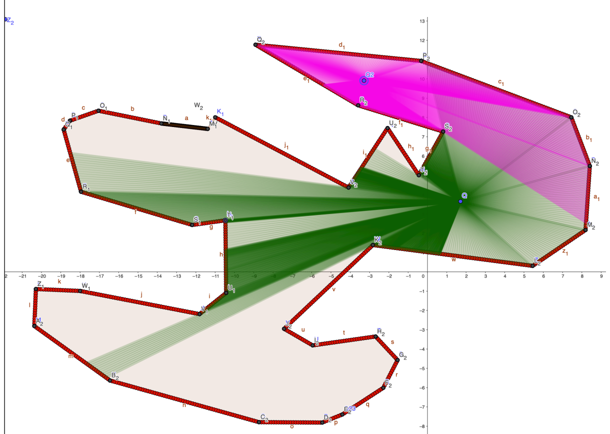


- Greedy Approach. The first camera is located at the the point of P that sees maximum area
- The second camera g_2 is located where it sees the maximum area **that** g_1 **does not see**
- g_3 sees the max area not seen by neither g_1 nor g_2, etc…
- Stop when either P is covered, or (in the budget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

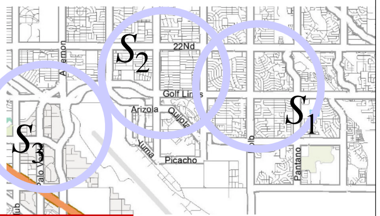


- Greedy Approach. The first camera is located at the the point of P that sees maximum area
- The second camera g_2 is located where it sees the maximum area **that** g_1 **does not see**
- g_3 sees the max area not seen by neither g_1 nor g_2, etc…
- Stop when either P is covered, or (in the budget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

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.

Greedy Algorithm for the set of items $\{x_1 \ldots x_m\}$ (house)
Also given: Collections of sets R=$\{S_1 \ldots S_m\}$ (for each potential charging station at p_i, S_i is a set of houses at distance ≤ 5 from p_i. That is, houses that could be covered by p_i.



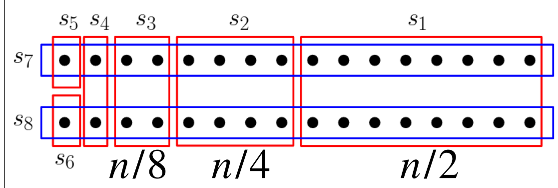
$X' = X = \{x_1 \ldots x_m\}$ //houses yet not covered
For i=1 to k (or when nothing left to be covered) {
Let S_i' be the set $S \in \mathbf{R}$ that maximizing $|S \bigcap X'|$.
//Find a charging station maximizes the number of houses
//not covered yet
$X' \leftarrow X' \setminus S_i'$ //*Only care for uncovered atoms*
}
Return $[S_1', S_2' \ldots S_k']$

\# houses of X' in S

For non-budget version, the only stopping condition is nothing left to be covered.

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 $s_2 \ldots$ could be $\geq \log_2 n$
- Approximation factor:
- $\text{Approximation factor} = \dfrac{\text{Numer of sets that greedy finds}}{\text{Numer of sets that OPT finds}} = \dfrac{\log_2 n}{2} = \Omega(\log n)$
- 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 case of covering the whole region (we stop only when nothing left to cover)

Let Opt be the solution with the smallest number of sets that covers Tucson.
Let C be the result of the greedy algorithm

- |Opt|- number of set in optimal cover (e.g. number of charging stations)
- |C| number of sets produced by the greedy solutions

Theorem: $|Opt| \leq |C| \leq |Opt| \cdot \ln n$

(actually a better bound could be shown: Let m_0 be the max number of houses covered by a single set. Then $|C| \leq |Opt| \cdot \ln m_0$

In practice, this is an excellent and very popular algorithm.

Analysis of greedy algorithm for the budget set-cover problem

- Let k (the budget) be a given fixed positive integer.
- **Opt**- set of *k* stations/cameras that maximizes ***area(Opt)*** - the area in secure feet seen by any set of k cameras. Or number of houses covered by k stations.
- Let ***C*** be the set of k cameras that the greedy algorithm returns.
- Let ***area(C)*** be the area seen by these cameras
- Theorem:

$$\text{Area}(Opt) \geq \text{Area}(C) \geq (1 - \frac{1}{e}) \cdot \text{Area}(Opt) \geq 0.64 \cdot \text{Area}(Opt)$$

That is, greedy covers at least 64 % of the area that Opt covers.