Approximation Algorithms

Is Close Enough
Good Enough?

Motivation

- By now we've seen many NP-Complete problems.
- We conjecture none of them has polynomial time algorithm.

Motivation

- Is this a dead-end? Should we give up altogether?

Motivation

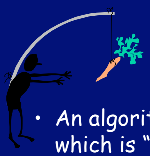
- *Or maybe we can settle for good approximation algorithms?*

Introduction

- Objectives:
  - To formalize the notion of approximation.
  - To demonstrate several such algorithms.
- Overview:
  - Optimization and Approximation
  - VERTEX-COVER, SET-COVER

Optimization

- Many of the problems we've encountered so far are really *optimization problems*.
- I.e - the task can be naturally rephrased as finding a maximal/ minimal solution.
- For example: finding a maximal clique in a graph.



Approximation

- An algorithm that returns an answer C which is "close" to the optimal solution C^* is called an *approximation algorithm*.
- "Closeness" is usually measured by the ratio bound $\rho(n)$ the algorithm produces.
- Which is a function that satisfies, for any input size n, $\max\{C/C^*, C^*/C\} \leq \rho(n)$.

Network Power

Say you have a network, with links between some components

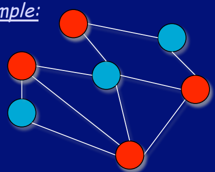
Each link requires power supply, hence, you need to supply power to a set of nodes that cover all links

Obviously, you'd like to connect the smallest number of nodes

VERTEX-COVER

- Instance: an undirected graph $G=(V,E)$.
- Problem: find a set $C \subseteq V$ of minimal size s.t. for any $(u,v) \in E$, either $u \in C$ or $v \in C$.

Example:



Minimum VC NP-hard

Proof: It is enough to show the decision problem below is NP-Complete:

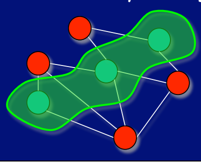
- Instance: an undirected graph $G=(V,E)$ and a number k.
- Problem: to decide if there exists a set $V' \subseteq V$ of size k s.t for any $(u,v) \in E$, $u \in V'$ or $v \in V'$.

This follows immediately from the following observation.

Minimum VC NP-hard

Observation: Let $G=(V,E)$ be an undirected graph. The complement $V \setminus C$ of a vertex-cover C is an independent-set of G.

Proof: Two vertices outside a vertex-cover cannot be connected by an edge. ■

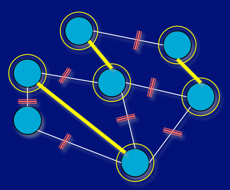


COR(B) 523-524

VC - Approximation Algorithm

- $C \leftarrow \phi$
- $E' \leftarrow E$
- **while** $E' \neq \phi$
  - **do** let (u,v) be an arbitrary edge of E'
  - $C \leftarrow C \cup \{u,v\}$
  - remove from E' every edge incident to either u or v.
- return C.

Demo



Compare this cover to the one from the example

Polynomial Time

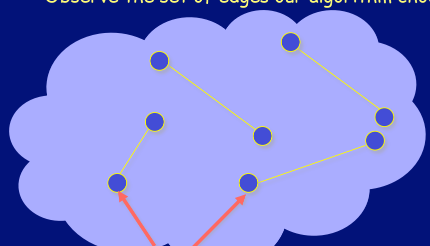
- $C \leftarrow \phi$
- $E' \leftarrow E$ } $O(n^2)$
- **while** $E' \neq \phi$ **do** — $O(n^2)$
  - let (u,v) be an arbitrary edge of E' — $O(1)$
  - $C \leftarrow C \cup \{u,v\}$
  - remove from E' every edge incident to either u or v — $O(n)$
- return C

Correctness

The set of vertices our algorithm returns is clearly a vertex-cover, since we iterate until every edge is covered.

How Good an Approximation is it?

Observe the set of edges our algorithm chooses



no common vertices! ⇒ any VC contains 1 in each

our VC contains both, hence at most twice as large

Mass Mailing

@email

Say you'd like to send some message to a large list of people (e.g. all campus)

There are some available mailing-lists, however, the moderator of each list charges $1 for each message sent

You'd like to find the smallest set of lists that covers all recipients

SET-COVER

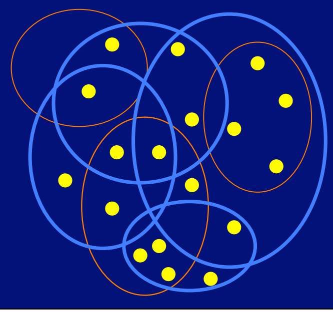
- Instance: a finite set X and a family F of subsets of X, such that

$$X = \bigcup_{S \in F} S$$

- Problem: to find a set $C \subseteq F$ of minimal size which *covers* X, i.e -

$$X = \bigcup_{S \in C} S$$

SET-COVER: Example

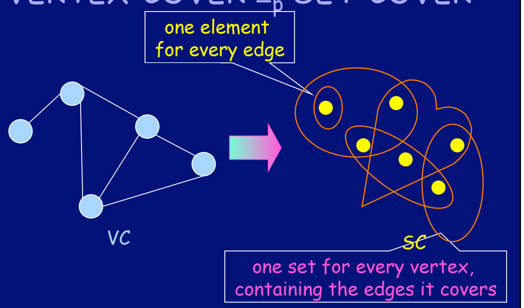


SET-COVER is NP-Hard

Proof: Observe the corresponding decision problem.

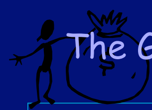
- Clearly, it's in NP (Check!).
- We'll sketch a reduction from (decision) VERTEX-COVER to it:

VERTEX-COVER $\leq_p$ SET-COVER



COR(B) 530-533

The Greedy Algorithm



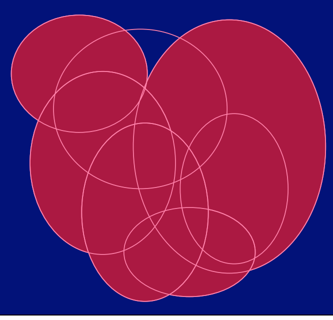
- $C \leftarrow \phi$
- $U \leftarrow X$
- **while** $U \neq \phi$ **do**
  - select $S \in F$ that maximizes $|S \cap U|$ } $O(|F| \cdot |X|)$
  - $C \leftarrow C \cup \{S\}$
  - $U \leftarrow U - S$
- return C

min{|X|,|F|}

Demonstration

compare to the optimal cover

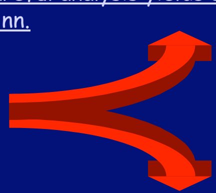
5



Is Being Greedy Worthwhile?

How Do We Proceed From Here?

- We can easily bound the approximation ratio by logn.
- A more careful analysis yields a tight bound of lnn.



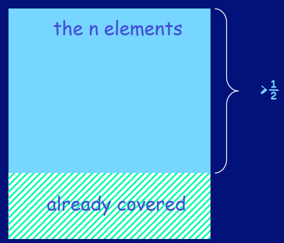
The Trick

- We'd like to compare the number of subsets returned by the greedy algorithm to the optimal
- The optimal is unknown, however, if it consists of k subsets, then any part of the universe can be covered by k subsets!
- Which is exactly what the next 3 distinct arguments take advantage of

Loose Ratio-Bound

<u>Claim</u>: If ∃ cover of size k, then after k iterations the algorithm have covered at least ½ of the elements

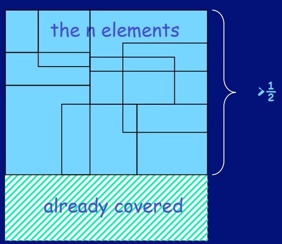
Suppose it doesn't and observe the situation after k iterations:



Loose Ratio-Bound

<u>Claim</u>: If ∃ cover of size k, then after k iterations the algorithm have covered at least ½ of the elements

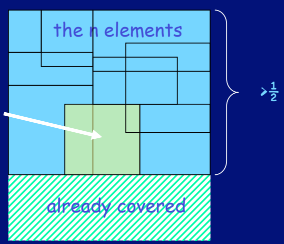
Since this part → can also be covered by k sets...



Loose Ratio-Bound

<u>Claim</u>: If ∃ cover of size k, then after k iterations the algorithm have covered at least ½ of the elements

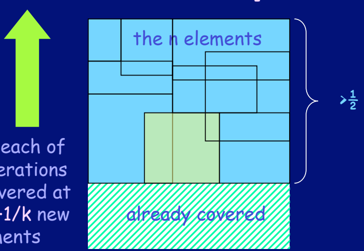
there must be a set not chosen yet, whose size is at least ½n·1/k



Loose Ratio-Bound

<u>Claim</u>: If ∃ cover of size k, then after k iterations the algorithm have covered at least ½ of the elements

and the claim is proven!



Thus in each of the k iterations we've covered at least ½n·1/k new elements

Loose Ratio-Bound

<u>Claim</u>: If ∃ cover of size k, then after k iterations the algorithm covered at least ½ of the elements.

⇓

Therefore after klogn iterations (i.e - after choosing klogn sets) all the n elements must be covered, and the bound is proved. ■

Better Ratio Bound

Let $S_1, \ldots, S_t$ be the sequence of sets outputted by the greedy algorithm. Let, for $0 \le i \le t$

$$U_i \equiv X - \bigcup_{j=1}^{i} S_j$$

Since, for every i, U_i can be covered by k sets, it follows

$$|U_{i+1}| = |U_i - S_{i+1}| \le |U_i| \frac{k-1}{k}$$

Better Ratio Bound

$$|U_{i+1}| = |U_i - S_{i+1}| \le |U_i| \frac{k-1}{k}$$

Hence, for any $0 \le i < j \le t$

$$|U_j| \le |U_i| \cdot \left(\frac{k-1}{k}\right)^{j-i}$$

Which implies that for every i

$$|U_{i \cdot k}| \le |U_0| \cdot \left(\frac{k-1}{k}\right)^{i \cdot k} \le |X| \cdot \frac{1}{e^i}$$

Therefore, $t \le k \ln(n) + 1$

Tight Ratio-Bound

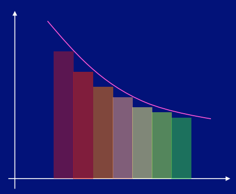
Claim: The greedy algorithm approximates the optimal set-cover to within a factor $H(\max\{ |S| : S \in F \})$

Where $H(d)$ is the d-th harmonic number:

$$H(d) \stackrel{def}{=} \sum_{i=1}^{d} \frac{1}{i}$$

Tight Ratio-Bound

$$\sum_{k=1}^{n} \frac{1}{k} = \sum_{k=2}^{n} \frac{1}{k} + 1 \le \int_{1}^{n} \frac{1}{x} dx + 1 = \ln n + 1$$

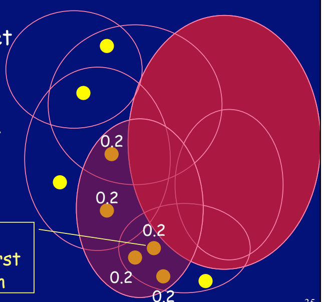


Claim's Proof

Charge $1 for each set

Split cost between covered elements

Bound from above the total fees paid



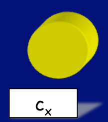
each recipient pays the fractional cost for the first mailing-list it appears in

Analysis

- Thus, every element $x \in X$ is charged

$$c_x \stackrel{def}{=} \frac{1}{|S_i - (S_1 \cup \ldots \cup S_{i-1})|}$$

- Where S_i is the first set that covers x.



Lemma

<u>Lemma</u>: For every $S \in F$

$$\sum_{x \in S} c_x \le H(|S|)$$

number of members of S still uncovered after i iterations

<u>Proof</u>: Fix an $S \in F$. For any i, let

$$u_i \overset{def}{=} |S - (S_1 \cup \ldots \cup S_i)|$$

$\forall 1 \le i \le k$: S_i covers $u_{i-1} - u_i$ elements of S

Let k be the smallest index, s.t. $u_k = 0$

Lemma

$$\sum_{x \in S} c_x = \sum_{i=1}^{k} \frac{u_{i-1} - u_i}{|S_i - (S_1 \cup \ldots \cup S_{i-1})|} \le \sum_{i=1}^{k} \frac{u_{i-1} - u_i}{|S - (S_1 \cup \ldots \cup S_{i-1})|} =$$

sum charges

else greedy strategy would have taken S instead of S_i

definition of u_{i-1}

$$\sum_{i=1}^{k} \frac{u_{i-1} - u_i}{u_{i-1}} \le \sum_{i=1}^{k} H(u_{i-1}) - H(u_i) = H(u_0) - H(u_k) = H(|S|)$$

$\forall a < b$ $H(b) - H(a) = \frac{1}{a+1} + \ldots + \frac{1}{b} \ge \frac{b-a}{b}$

Telescopic sum

$H(u_k) = H(0) = 0$

$H(u_0) = H(|S|)$

Analysis

Now we can finally complete our analysis:

$$|C| = \sum_{x \in X} c_x \le \sum_{S \in C^*} \sum_{x \in S} c_x \le |C^*| \cdot H(\max\{|S| : S \in F\})$$

■

Summary

- As it turns out, we can sometimes find efficient approximation algorithms for NP-hard problems.
- We've seen two such algorithms:
  - for VERTEX-COVER (factor 2)
  - for SET-COVER (logarithmic factor).

What's Next?

- But where can we draw the line?
- Does every NP-hard problem have an approximation?
- And to within which factor?
- Can approximation be NP-hard as well?!