Focus here today; often this is sufficient and problem must be solved quickly - military. Most effective if have guarantee on solution.

example: primality testing for RSA cryptosystem

valid solution that is not optimal - Traveling Salesperson problem, non-prime (decease)

be called prime (decease)

very small probability that non-prime would

How to solve problems that are NP-hard

For sufficiently small input apply exponential algorithm

may require harnessing large computing resources

Accept potentially incorrect solution in polynomial running time

gives correct solution

Approximation algorithm

will violate this today

last week

Violated this last week

(unsolvable problems)

General Flow of Algorithm Presentation

Problem Analysis

Running Time - polynomial or better

Correctness
Formalism

Given optimization problem on input of size $n$

Let $C^* =$ "cost" of optimal soln (not running time, but)

Let $C =$ cost of approx alg soln

Then

Ratio Bound $\rho(n)$

$$\max \left( \frac{C}{C^*}, \frac{C^*}{C} \right) \leq \rho(n) \quad \text{for any } n$$

Relative Error Bound $\varepsilon(n)$

$$\frac{|C - C^*|}{C^*} \leq \varepsilon(n) \quad \text{for any } n$$

Examples

1. TSP: 2-approximation algorithm (can be improved to 1.5)
2. Set cover: $O(\log n)$-approximation algorithm
3. Vertex cover: 2-approximation algorithm
**TSP (optimization) NP-hard**

**Input:** Undirected complete graph with edge lengths \( c(u,v) \)

**Output:** A tour of minimum length that visits each vertex exactly once.

(Note: NP-hard)

**Approx-TSP-Tour**

1. Build minimum spanning tree (MST) \( T \) for \( G \).
2. Let \( L \) be the list of vertices visited in preorder tree walk of \( T \).
3. Return a tour \( T \) that visits vertices in order \( L \).

**Example**

\[ \text{Problem} \]
\[ \text{MST} \]
\[ \text{Tour} \]

Total distance \( = 24.00 \)

\[ \text{Optimal} \text{ (visit } d \text{ later)} \]

Total distance \( = 20.44 \)
Theorem: \text{Approx-TSP-Tour} (with triangle inequality) has \text{ratio bound of 2}.

Proof:
- Let $H^*$ be an optimal tour for $G$.
- Since $T$ is a MST for $G$, $C(T) \leq C(H^*)$.
- Let $W$ be a "full walk" of $T$ (each MST visited twice).
  \[ c(W) = 2c(T) \]
- A tour obtained by shortcutting $T$.
  \[ \Delta\text{-inequality } \Rightarrow C(H) \leq C(W) = 2c(T) \leq 2C(H^*) \]

Note: Even with simple MST-PRI, running time is $\Theta(V^2)$,
\[ \Rightarrow \text{polynomial} \]
\[ \Rightarrow \text{a ratio bound was proven without access to the optimal solution} \]
Set Cover (another NP-complete problem)

Input: Collection of sets \( S_1, \ldots, S_m \subseteq U \), \( U \cap S_i = U \), \( l \leq k \leq m \)

Goal: Find \( I = \{ i_1, \ldots, i_k \} \) of smallest size with \( U \cap \bigcup_{i \in I} S_i = U \)

Why?
\( U \) = set of tasks
\( S_i \) = set of tasks that person \( i \) can perform
Finds "skeleton crew"

Greedy solution
Repeat until all elements covered
- Choose a new set \( S_i \) containing max # uncovered elements
- Add \( i \) to \( I \)
- Mark all elements from \( S_i \) as covered

Example:
\[
\begin{align*}
E & = \{1, 2, 3, 4, 5, 6, 7\} \\
S_1 & = \{1, 2, 3\} \\
S_2 & = \{3, 4\} \\
S_3 & = \{5, 6\} \\
S_4 & = \{1, 3, 5, 7\}
\end{align*}
\]

Optimal: \( I = \{1, 2, 3\} \)
Greedy: \( I = \{1, 2, 3, 4\} \)

However, this is \( (\log m) \)-approximation algorithm
Proof: Let $k$ be size of minimal cover.

Fact: At any point, there is at least one set $S$ that covers $\geq \frac{1}{k}$ fraction of uncovered elements.

Because: if each set covered $< \frac{1}{k}$ fraction of uncovered elements, there would be no way to cover $U$ in only $k$ sets.

\[ \ln m \text{-approximation} \]

- Let $U_i$ be # uncovered elements after $i$th iter.
- $U_{i+1} \leq (1 - \frac{1}{k}) U_i$
- Initially $U_0 = M$, so $U_i \leq (1 - \frac{1}{k})^i M$
- At $i = k \ln M \Rightarrow U_i \leq (1 - \frac{1}{k})^{k \ln M} M \leq e^{-\ln M} M = 1$
  so at $k \ln m + 1$ steps algorithm covers all elements and stops.
- Thus, algorithm is $(\ln m + 1)$-approximate.

\[ \text{Again} \]

- Prove bound without access to optimal solution.
- Note that greedy algorithms can be exact or approximate.
Vertex Cover Problem (NP-complete)

Input: Undirected graph $G(V,E)$

Output: Minimum size set $C \subseteq V$ such that

each edge in $E$ has at least one endpoint in $C$.

(Special case of set cover, in which every element

repeats edge is contained in only 2 sets [Endpoints].)

Approx-Vertex-Cover ($G$)

$C \leftarrow \emptyset$

$E' \leftarrow E[G]$

While $E' \neq \emptyset$

Do let $(u,v)$ be an arbitrary edge of $E'$

$C \leftarrow C \cup \{(u,v)\}$

Remove from $E'$ every edge incident on either $u$ or $v$

Return $C$
Example

\[
\begin{align*}
\text{Done: \text{Cover } = 4} \\
(\text{Optimal } = 3 \text{ - remove top vertex})
\end{align*}
\]

Analysis

Correct
- only remove covered edges from \( E' \)
- iterate until \( E' \) is empty

Running Time: \( \Theta(V + E) \) - Each edge is added to \( E' \) on initialization and removed once. Each vertex is added to \( C \) at most once.

Theorem: \textbf{APPROX-VERTEX-COVER} has ratio bound of \( 2 \)

Proof:
- \( A = \{ \text{edges picked at do } \} \)
- no 2 edges in \( A \) share an endpoint
- \( |C| = 2|A| \)
- optimal cover must include at least one endpoint for each edge in \( A \)
- \( |A| \leq |C^*| \Rightarrow |C| \leq 2|C^*| \)
Why aren't the following improvements?

1. Only pick one endpoint:

   ![Diagram]

   could have n-1 covers, rather than optimal, of 1
   (greedy gives 2)

2. Be more greedy
   - Pick vertex with largest # of incident edges at each iteration
   - Turns out worst case is \( \Theta(n^2) \) approximate

3. Could be greedy in edge selection (always pick edge whose endpoints have highest degree)
   - does not improve worst case bound
   - complicates analysis

Bill\[#] Post-processing can help practically
   - remove extra vertices at end
   - can't weaken approximation bound
   - can be quite useful in practical situations