What to do in the face of complexity: 

**Answer 2**

[Approximation, Randomized] Algorithms
\[ \text{VERTEX-COVER} = \{ \langle G, k \rangle \mid G \text{ is an undirected graph with a } k\text{-node vertex cover} \} \]

\text{VERTEX-COVER is NP-complete}
MIN-VERTEX-COVER: Given an undirected graph $G$ find the minimum vertex cover
If you can solve \textsc{Min-Vertex-Cover} efficiently, then you can solve \textsc{Vertex-Cover} efficiently.
MIN-VERTEX-COVER: Given an undirected graph $G$ find the minimum vertex cover

MIN-VERTEX-COVER is NP-hard
Figure 11.10 The feasible region of a simple linear program.
Linear Programming

\[ \text{LP} \in \textbf{NP} \]

In fact, \( \text{LP} \in \text{co-NP} \)

Further, \( \text{LP} \in \textbf{P} \)
Vertex Cover using **Integer Programming**

\[
\min \sum_i x_i \\
x_i + x_j \geq 1 \quad (i, j) \in E \\
x_i \in \{0, 1\} \quad i \in V
\]
Integer Programming is **NP-complete**!
Vertex Cover using Linear Programming

\[
\begin{align*}
\min \ & \sum_i x_i \\
\text{s.t.} \ & x_i + x_j \geq 1 \quad \forall (i, j) \in E \\
x_i \ & \in [0, 1] \quad \forall i \in V
\end{align*}
\]
Let \( n_{LP} = \sum_i x_i \) be the solution of the linear program

Let \( n^* = \sum_i x_i \) be the solution of the integer program

\[ n_{LP} \leq n^* \]

[0,1] choice on vertices

binary choice on vertices
Vertex Cover using Linear Programming

\[
\begin{align*}
\min & \sum_i x_i \\
x_i + x_j & \geq 1 \quad (i, j) \in E \\
x_i & \in [0, 1] \quad i \in V
\end{align*}
\]

To produce a VC *round* the values

Why is this a VC?
Vertex Cover using Linear Programming

\[ \min \sum_i x_i \]
\[ x_i + x_j \geq 1 \quad (i, j) \in E \]
\[ x_i \in [0, 1] \quad i \in V \]

To produce a VC *round* the values
\[ n_{VC} \leq 2 \, n_{LP} \]
Vertex Cover using Linear Programming

\[ n_{\text{LP}} \leq n^* \]
\[ n_{\text{VC}} \leq 2 \ n_{\text{LP}} \]

This algorithm produces a VC of \textit{at most} twice the number of vertices as a min-VC.
Vertex Cover

another approximation

Pick an edge. Add *both* vertices to vertex cover. Delete any edges incident to either vertex. Repeat with (smaller) subgraph.
Vertex Cover

another 2-approximation

This algorithm produces a VC of at most twice the number of vertices as a min-VC
3SAT = \{ \langle \Phi \rangle \mid \Phi \text{ is a satisfiable 3CNF formula} \}

3SAT is NP-complete
Constraints as Boolean formula

planning and scheduling, automotive design

What if they are unsatisfiable?

**MAX-SAT**: Given a set of input clauses, find a truth assignment that satisfies *as many as possible*. 
MAX-3SAT using Randomization

Algorithm:
Set each variable to True with probability 1/2
Algorithm:
Set each variable to True with probability 1/2

We can expect \(~88\%\) of the optimal number of clauses to be satisfied by this algorithm
MAX-3SAT using Randomization

$Z : \text{number of satisfied clauses}$

$$Z_i = \begin{cases} 1 & \text{if clause } i \text{ is satisfied} \\ 0 & \text{o.w.} \end{cases}$$

$$Z = \sum_i Z_i$$
\[ Z_i = \begin{cases} 1 & \text{if clause } i \text{ is satisfied} \\ 0 & \text{o.w.} \end{cases} \]

\[ E[Z_i] = 1 \cdot (\text{prob that } C_i \text{ is satisfied}) + 0 \cdot (\text{prob that } C_i \text{ is satisfied}) \]

\[ = \text{prob that } C_i \text{ is satisfied} \]
MAX-3SAT using Randomization

$E[Z_i] = \text{prob that } C_i \text{ is satisfied}$

$\text{prob that } C_i \text{ is satisfied}$
$= 1 - \text{prob that } C_i \text{ is not satisfied}$

$\text{prob that } C_i \text{ is not satisfied}$
$= (\text{literal 1 evaluates to false})$
$\quad \ast (\text{literal 2 evaluates to false})$
$\quad \ast (\text{literal 3 evaluates to false})$

$= \frac{1}{8}$

$\text{prob that } C_i \text{ is satisfied} = \frac{7}{8} = E[Z_i]$
MAX-3SAT using Randomization

Z : number of satisfied clauses

\[ E[Z] = E[\sum_i Z_i] \]
\[ = \sum_i E[Z_i] \]
\[ = (7/8)k \]
Algorithm:
Set each variable to True with probability 1/2

We can expect ~88% of the optimal number of clauses to be satisfied by this algorithm
The END

...and the beginning...