Reductions & Approximation

\[ \rightarrow \text{Informal Refresher: } \text{NP-completeness} \]
\[ \rightarrow \text{Reductions: } \text{cSAT} \rightarrow \text{SAT} \rightarrow \text{3SAT} \rightarrow \text{Vertex Cover} \]
\[ \rightarrow \text{Approximation Algorithms} \]

NP-Completeness

- NP: contains all decision problems ("yes"/"no" answers) whose "yes" inputs have poly-short & poly-time verifiable certificates

  e.g.
  - Is there a 2-d matching in a graph?
  - Is there a 3-d matching in a graph?
  - Is there a path from s to t of length \( \leq K \)?
  etc.

- Reductions: We say that \( \Pi_1 \leq_p \Pi_2 \) if there is a polynomial-time algorithm \( R \) s.t.
  - if \( x \) is an input to \( \Pi_1 \), \( R(x) \) is an input to \( \Pi_2 \)
  - \( \Pi_1(x) = \text{"yes"} \iff \Pi_2(R(x)) = \text{"yes"} \)

In words, "\( \Pi_2 \) is at least as hard as \( \Pi_1 \)."
- **NP-complete**: "the hardest problems in NP"
  
  Formally, \( \mathcal{T} \) is NP-complete iff
  
  i. \( \mathcal{T} \in \text{NP} \)
  
  ii. \( \forall \mathcal{T}' \in \text{NP}: \mathcal{T}' \leq_p \mathcal{T} \)
  
  if only ii, then NP-hard

- **Cook's Theorem [1971]**: circuit-SAT is NP-complete.

  given Boolean circuit with gates AND, OR, NOT and no feedback, is there a way to set input wires to 0/1 values so that output wire is 1.

- **Karp [1972]**: showed that many other problems are NP-complete by reducing circuit-SAT to them.

- **SAT**: given formula \( \varphi \) in conjunctive-normal-form (AND of ORs) is it satisfiable?

  \[ \varphi = (x_1 \lor \overline{x}_2) \land x_3 \land (\overline{x}_3 \lor x_1 \lor x_2) \]

  is satisfiable by setting

  \( x_1 = 1, x_2 = 0/1, x_3 = 1 \)

  \[ \varphi = (x_1 \lor \overline{x}_2) \land (x_2 \lor \overline{x}_3) \land x_3 \land \overline{x}_1 \]

  is not satisfiable

- clearly in NP & Karp showed circuit-SAT \( \leq_p \) SAT
- hence SAT is NP-complete
A bit of terminology (by example):

In CNF formula \( \phi = (x_1 \lor \overline{x_2}) \land x_3 \land (\overline{x_3} \lor x_1 \lor x_2) \)

- conjunctive normal form
- variables: \( x_1, x_2, x_3 \)
- literals: \( x_1, x_2, x_3, \overline{x_2}, \overline{x_3} \)
- clauses: \( x_1 \lor \overline{x_2}, x_3, \overline{x_3} \lor x_1 \lor x_2 \)

3-SAT: given formula \( \phi \) in CNF form w/ 3 literals per clause, is \( \phi \) satisfiable?

E.g., \( \phi = (x_1 \lor \overline{x_2}) \land x_3 \land (\overline{x_3} \lor x_1 \lor x_2) \) is not a valid input to 3-SAT,

but \( \phi' = (x_1 \lor \overline{x_2} \lor x_3) \land (x_2 \lor x_3 \lor x_4) \) is

- clearly 3-SAT \( \in \) NP
- easy to see: 3-SAT \( \leq_p \) SAT (but we already knew it, since SAT is NP-complete)
- Karp: SAT \( \leq_p \) 3-SAT \( \Rightarrow \) 3-SAT is NP-complete as well

So far: circuit-SAT \( \leq_p \) SAT \( \leq_p \) 3-SAT

- all these are circuit/formula satisfiability problems
- how about graph problems?
3-SAT ≤p VERTEX COVER:

- **Vertex Cover (VC):** Given graph \( G = (V, E) \) and an integer \( k \), does there exist a subset \( S \subseteq V \) s.t. \( |S| \leq k \) and every \( e \in E \) is incident to at least one vertex in \( S \)?

- Clearly, \( VC \in NP \): indeed if \( (G, k) \) is a "yes" instance then a poly-short certificate for this is a subset \( S \) of \( \leq k \) vertices which in poly-time can be checked to be incident to all edges.

- **Reduction from 3-SAT to VC:**
  - Input to reduction: \( CNF \) formula \( \phi \)
    - \( n = \# \) variables
    - \( m = \# \) clauses
  - Goal of reduction: encode \( \phi \) via a graph \( G = (V, E) \) \& number \( k \) s.t. \((\phi \text{ satisfiable}) \iff (\exists \text{ vertex cover of size } k)\)
  - Technique: gadget construction
    - for each variable \( a \) a gadget (i.e. a subgraph of \( G \)) representing its truth value
ii. for each clause a gadget representing the fact that at least one literal must be true

iii. edges connecting these kinds of gadgets

- Construction:

- each variable \( x_i \) \( \rightarrow \) \( P_{x_i} \rightarrow N_{x_i} \)

  note: choosing one vertex is necessary and sufficient to cover the edge (*)

- every clause \( C \) \( \rightarrow \)

  \( S_c \rightarrow f_c \rightarrow t_c \rightarrow S_c \)

  note: choosing two vertices is necessary and sufficient to cover these three edges (**) 

  e.g. \( \psi = (x_1 \lor x_2 \lor x_3) \land (\overline{x_1} \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_2 \lor \overline{x_3}) \)

\[ \begin{align*}
P_{x_1} & \rightarrow N_{x_1} \\
P_{x_2} & \rightarrow N_{x_2} \\
P_{x_3} & \rightarrow N_{x_3} \\
f_{c_1} & \rightarrow S_{c_1} \rightarrow t_{c_1} \\
f_{c_2} & \rightarrow S_{c_2} \rightarrow t_{c_2} \\
f_{c_3} & \rightarrow S_{c_3} \rightarrow t_{c_3}
\end{align*} \]
without any further edges, we know that the smallest vertex cover in our graph has size $k = n + 2m$ (this follows from observations (C) and (D)).

trick: this is true regardless of whether $\varphi$ is satisfiable.

we need to encode the relations of clauses to variables.

connections between variable and clause gadgets:

- let’s go back to our example:

$\varphi = (x_1 \lor x_2 \lor x_3) \land (\overline{x}_1 \lor x_2 \lor x_3) \land (\overline{x}_1 \lor x_2 \lor \overline{x}_3)

- formally: if first literal of clause $c_i$ is $x_j$ add edge $(f_{c_i}, p_{x_j})$
if first literal of \( c_i \) is \( \overline{x}_j \) add edge \((f_{c_i}, n_{c_j})\)

similarly for nodes \( s_{c_i} \) and \( t_{c_i} \)

- end of construction

- Correctness:

claim: \( \exists \) vertex-cover of size \( k = 2m + n \) \( \Rightarrow \varphi \) is satisfiable

proof:

- Say exists VC of size \( k \). It must use exactly one vertex from each variable gadget and exactly two vertices from each clause gadget.
  - If \( p_{x_i} \) is in VC, set \( x_i = 1 \); o.w. set \( x_i = 0 \)
  - Want to show that resulting assignment \( x_1, \ldots, x_n \) satisfies every clause.
  - Pick any clause \( c \) of \( \varphi \), e.g. \( x_1 \lor x_2 \lor \overline{x}_3 \)

  subclaim: If none of \( n_{x_1}, p_{x_2}, n_{x_3} \) is in VC, then at least one edge adjacent to \( \) clause gadget is uncovered.
pf: Suppose not.
* Exactly two of $f_c, s_c, t_c$ are in $VC$ (by ***).
* Then the blue edge adjacent to the vertex that isn't in $VC$ will be uncovered. $\Box$

claim $\Rightarrow$ at least one of $n_{x_1}, p_{x_2}, n_{x_3}$ is in $VC$ $\Rightarrow$ assignment satisfies clause $\Box$.

claim 2: If $\varphi$ is satisfiable $\Rightarrow \exists$ VC of size $k$.

Proof: easy - say $x$ is satisfying assignment

- If $x_i = 1$, then include $p_{x_i}$ in VC; now $n_{x_i}$
- Pick another 2-m vertices from clause gadgets to cover all edges. $\Box$

- We have completed our proof of $3$-SAT $\leq_p$ VC.

$\Rightarrow VC$ is NP-complete.

* so far: circuit-SAT $\leq_p$ SAT $\leq_p$ 3-SAT $\leq_p$ VC $\leq_p$ Ham-Cycle

$\rightarrow$ recitation

so all of these problems NP-complete
we know thousands more e.g. TSP, 3d-matching, knapsack, independent-set, longest path, max-clique, 0-1 programming, ...

moral of the story: many interesting problems are hard

**Beyond NP-Completeness**

- average case analysis: input may not be worst-case
- special inputs: e.g. tree/bounded treewidth/planar graphs may be easier
- change the problem
- approximation algorithms

**Approximation Algorithms**

- meaningful for optimization versions of problems
- goal: in polynomial-time find solution whose value is within a constant factor of optimal
- for $\alpha > 1$, an $\alpha$-approximation algorithm outputs a solution whose value is guaranteed to be

\[ \frac{\text{opt}}{\alpha} \]

for maximization problems

\[ \alpha \cdot \text{opt} \]

for minimization problems
- E.g. a 2-approximation algorithm for TSP always outputs a cycle whose weight is at most a factor of 2 larger than optimal.

Today: 2-approximation for Max-cut

Max-cut: Given graph $G = (V, E)$ partition $V$ into two non-empty sets $S$ and $V-S$ so as to maximize the number of edges going from $S$ to $V-S$.

E.g.

Max-cut has size 4

Min-cut: Same but want to minimize cut edges.

While min-cut is in P, max-cut is NP-complete (decision version).
Q: Can we approximate max-cut (optimization version)?
A: Yes, almost trivially.

- Proposed Algorithm:
  For every vertex \( v \), include it in \( S \) with probability \( 1/2 \) independently of other vertices.

Analysis:

\[
\mathbb{E} [\# \text{cut edges}] = \sum_{\text{edges } e=(u,v)} \Pr[\text{exactly one of } u,v \text{ is in } S]
\]

\[
= \sum_e \frac{1}{2} = \frac{|E|}{2}
\]

\( \text{OPT} \leq |E| \)

\( \Rightarrow \mathbb{E} [\# \text{cut edges}] \geq \frac{|E|}{2} \)

hence our algorithm is a 2-approximation

- Derandomization?
  - Start w/ arbitrary cut;
  - While \( \exists \) vertex that can be moved to other side of cut to improve size of cut, move it;
  - Return cut.
Analysis:

- \#iterations \leq |E| as the size of the cut increases by one in each iteration
- Checking if \exists \text{ vertex to be moved} takes \( O(|E|) \) (as constant work/edge)
  \( \Rightarrow O(|E|^2) \) time overall

Approximation Guarantee:

at termination:

\[ \forall u \in S: \#\text{edges } (u, v) \geq \#\text{edges } (u, v) \forall v \in S \]

\[ \forall u \notin S: \#\text{edges } (u, v) \geq \#\text{edges } (u, v) \forall v \in S \]

\( \Rightarrow \) at least half of the edges incident to every vertex are cut

\( \Rightarrow \) size of cut \( \geq \frac{|E|}{2} \)

Best Known Poly-time Algorithm:

[Goemans-Williamson '91]: \( \alpha = 1.139 \)