NP and computational intractability

Kleinberg and Tardos, chapter 8

Major Transition

So far we have studied algorithmic patterns:
- Greedy
- Divide and conquer
- Dynamic programming
to develop efficient algorithms.

Now we want to classify and quantify problems that cannot be solved efficiently.

Our tool for doing this is another algorithmic pattern: reductions

Reduction transforms the input and output of an algorithm so that it can be used to solve a different problem.

Reductions

An engineer is handed an empty kettle and is asked to make tea.
The engineer fills up the kettle, boils the water and makes tea.

A mathematician is handed a full kettle and is asked to make tea.

What does the mathematician do?

empties the kettle and hands it to the engineer!

Algorithm Design Patterns and Anti-Patterns

- Algorithm design patterns
  - Greedy
  - Divide-and-conquer
  - Dynamic programming
  - Reductions
  - Approximation algorithms
  - Randomized algorithms

- Algorithm design anti-patterns
  - NP-completeness
  - Undecidability

Definition of tractable problems

Which problems will we be able to solve in practice?

A working definition (Cobham 1964, Edmonds 1965)
Those with polynomial-time algorithms.
Classifying Problems

We would like to: Classify problems according to those that can be solved in polynomial-time and those that cannot.

Some problems provably require exponential-time.

- Given a board position in an n-by-n generalization of chess, can black guarantee a win?

Frustrating news. Large number of fundamental problems have defied classification for decades.

We don’t know of a polynomial algorithm for them, and we cannot prove that no polynomial algorithm exists.

Polynomial-Time Reductions

Suppose we could solve Y in polynomial-time. What else could we solve in polynomial time?

Reduction. Problem X has a polynomial reduction to problem Y if arbitrary instances of problem X can be solved using:
- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem Y.

Notation. \( X \leq_P Y \).

Remarks.
- We pay for time to write down instances sent to black box \( \Rightarrow \) instances of Y must be of polynomial size.

Independent Set

\textbf{INDEPENDENT SET}:

Given a graph \( G = (V, E) \) and an integer \( k \), is there a subset of vertices \( S \subseteq V \) such that \( |S| \geq k \), and for each edge at most one of its endpoints is in \( S \)?

- Is there an independent set of size \( \leq 6 \)? Yes.
- Is there an independent set of size \( \geq 7 \)? No.

Reduction by Equivalence

Basic reduction strategies.
- Reduction by equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.
Optimization vs Decision Problems

**Decision problem.** Does there exist an independent set of size \( \geq k \)?

**Optimization problem.** Find independent set of maximum cardinality.

Easier to focus on decision problems.

---

**Vertex Cover**

**VERTEX COVER:** Given a graph \( G = (V, E) \) and an integer \( k \), is there a subset of vertices \( S \subseteq V \) such that \( |S| = k \), and for each edge, at least one of its endpoints is in \( S \)?

- Is there a vertex cover of size \( \geq 4 \)? Yes.
- Is there a vertex cover of size \( \leq 3 \)? No.

---

**Vertex Cover and Independent Set**

**Claim.** Let \( G = (V, E) \) be a graph. Then a set of nodes \( S \) is an independent set iff \( V - S \) is a vertex cover.

\[
\begin{align*}
\text{\textbf{\text{Claim.}}} & \quad \text{Let } G = (V, E) \text{ be a graph. Then a set of nodes } S \text{ is an independent set iff } V - S \text{ is a vertex cover.} \\
& \quad \Rightarrow \\
& \quad \text{Let } S \text{ be an independent set.} \\
& \quad \text{Consider an arbitrary edge } (u, v). \\
& \quad S \text{ independent } \implies u \notin S \text{ or } v \notin S \implies u \in V - S \text{ or } v \in V - S. \\
& \quad \text{Thus, } V - S \text{ covers } (u, v). \\
& \quad \Leftarrow \\
& \quad \text{Let } V - S \text{ be a vertex cover.} \\
& \quad \text{Consider two nodes } u \in S \text{ and } v \in S. \\
& \quad \text{Observe that } (u, v) \notin E \text{ since } V - S \text{ is a vertex cover.} \\
& \quad \text{Thus, no two nodes in } S \text{ are joined by an edge } \implies S \text{ is an independent set.} \\
\end{align*}
\]

---

**Vertex Cover and Independent Set**

**Claim.** Let \( G = (V, E) \) be a graph. Then a set of nodes \( S \) is an independent set iff \( V - S \) is a vertex cover.

**Use this to prove:** \( \text{VERTEX-COVER} \leq_p \text{INDEPENDENT-SET} \) and \( \text{INDEPENDENT-SET} \leq_p \text{VERTEX-COVER} \).

**Reduction.** Problem \( X \) has a polynomial reduction to problem \( Y \) if arbitrary instances of problem \( X \) can be solved using:
- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem \( Y \).

**Given an instance of IC \((G, k)\), construct an instance of VC \((G', k')\) where:**
- \( G' = G \)
- \( k' = |V| - k \)

The reduction: call the VC oracle on input \((G', k')\), and return its answer. Therefore \( \text{VERTEX-COVER} \leq_p \text{INDEPENDENT-SET} \).
Vertex Cover and Independent Set

Claim. Let $G=(V, E)$ be a graph. Then a set of nodes $S$ is an independent set iff $V - S$ is a vertex cover.

Given an instance of $\text{IC}: (G, k)$, construct an instance of $\text{VC}: (G', k')$ where

$G' = G$
$|V| - k = k$

The reduction: call the VC oracle on input $(G', k')$, and return its answer. Therefore $\text{VERTEX-COVER} \leq_p \text{INDEPENDENT-SET}$.

The other direction is similar, from which we conclude:

$\text{VERTEX-COVER} \equiv_p \text{INDEPENDENT-SET}$.

In general $X \leq_p Y$ does not imply $Y \leq_p X$.

More about reductions

Suppose that $X \leq_p Y$ and there is a polynomial time algorithm for $Y$. What can we say about $X$?

Reduction. Problem $X$ has a polynomial reduction to problem $Y$ if arbitrary instances of problem $X$ can be solved using:

- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem $Y$.

More about reductions

Suppose that $X \leq_p Y$ and there is a polynomial time algorithm for $Y$. What can we say about $X$?

$X$ can be solved in polynomial time! There are a few subtleties.

More about reductions

Suppose that $X \leq_p Y$ and there is no polynomial time algorithm for $X$. What can we say about $Y$?

$Y$ cannot be solved in polynomial time! In other words $Y$ is at least as difficult as $X$.

Reduction from Special Case to General Case

Basic reduction strategies:
- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.
Set Cover

**SET COVER:** Given a set $U$ of elements, a collection $S_1, S_2, \ldots, S_m$ of subsets of $U$, and an integer $k$, is there a collection of at most $k$ of these sets whose union is equal to $U$?

**Example:**

$U = \{1, 2, 3, 4, 5, 6, 7\}$

$k = 2$

$S_1 = \{3, 7\}$

$S_4 = \{2, 4\}$

$S_2 = \{3, 4, 5, 6\}$

$S_5 = \{5\}$

$S_3 = \{1\}$

$S_6 = \{1, 2, 6, 7\}$

**Vertex Cover Reduces to Set Cover**

**Claim:** $\text{VERTEX-COVER} \leq \text{P SET-COVER}$.

**Proof:** Given a $\text{VERTEX-COVER}$ instance $G = (V, E)$, $k$, we construct a set cover instance whose size equals the size of the vertex cover instance.

- $k = k$, $U = E$, $S_v = \{e \in E : e$ incident to $v\}$

- Set-cover of size $\leq k$ iff vertex cover of size $\leq k$.

**Generalization of Independent set**

Set cover is a generalization of Vertex cover

What would be the corresponding generalization of the independent set problem?

**Reductions via "Gadgets"**

Basic reduction strategies:

- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction via "gadgets."

**Satisfiability**

**Literal:** A Boolean variable or its negation. $x_i \lor \overline{x_i}$

**Clause:** A disjunction of literals. $C_i = x_1 \lor x_2 \lor x_3$

**Conjunctive Normal Form:** A propositional formula $\Phi$ that is a conjunction of clauses.

**SAT:** Given a CNF formula $\Phi$, does it have a satisfying truth assignment?

**3-SAT:** SAT where each clause contains exactly 3 literals.

**Example:** $(\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3) \land (x_2 \lor x_3 \lor \overline{x_1})$

Yes: $x_1$ true, $x_2$ true, $x_3$ false.

11/4/12
Satisfiability

**SAT**: Given a CNF formula \( \Phi \), does it have a satisfying truth assignment?

**3-SAT**: SAT where each clause contains exactly 3 literals.

Example: \((x_1 \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3) \land (x_1 \lor \overline{x_2} \lor \overline{x_3})\)

Yes: \(x_1 = \text{true}, x_2 = \text{true}, x_3 = \text{false}\).

When does a CNF formula evaluate to true?

---

**3-SAT Reduces to Independent Set**

**Claim**: 3-SAT \( \leq \text{P INDEPENDENT-SET} \).

**Proof**: Given an instance \( \Phi \) of 3-SAT, we construct an instance \((G, k)\) of INDEPENDENT-SET that has an independent set of size \(k\) iff \( \Phi \) is satisfiable.

**Construction**.
- \(G\) contains 3 vertices for each clause, one for each literal.
- Connect 3 literals in a clause in a triangle.
- Connect literal to each of its negations.
- We call these connected triangles gadgets, creating a useful instance of independent set.

\[
\Phi = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3) \land (x_1 \lor \overline{x_2} \lor \overline{x_3})
\]

**Review**

**Basic reduction strategies**.
- Equivalence: INDEPENDENT-SET \( \equiv \text{P VERTEX-COVER} \).
- Special case to general case: VERTEX-COVER \( \equiv \text{P SET-COVER} \).
- Encoding with gadgets: 3-SAT \( \leq \text{P INDEPENDENT-SET} \).

**Transitivity**. If \( X \leq \text{P Y} \) and \( Y \leq \text{P Z} \), then \( X \leq \text{P Z} \).

**Proof idea**: Compose the two algorithms.

**Example**: 3-SAT \( \leq \text{P INDEPENDENT-SET} \leq \text{P VERTEX-COVER} \leq \text{P SET-COVER} \).