Chapter 4

Lecture 4: NP and computational intractability

Listen to: Find the longest path, Daniel Barret

What do we do today:

- polynomial time reduction
- NP, co-NP and NP complete problems
- some examples for polynomial reduction to prove NP-completeness
- NP problems with special cases

At the same time we move from problems that had polynomial time algorithm solutions to problems where such solutions do not exist.

We will use the following problems as examples:

Problem. Circuit-satisfiability: (Cook and Levin 1971) A circuit is a labelled, directed graph (tree), with boolen operations at the nodes. Is there an assignment of values to the inputs such that the output takes yes?

Problem. 3-Satisfiability (3-SAT): 3-SAT will be our "root problem" later. Suppose we have a set of X Boolean variables $\{x_1, x_2, ...x_n\}$. A term t is x_i or its negation $\bar{x}_i$. A clause C_i is a disjunction of 3 terms, $t_1 \vee t_2 \vee t_3$, and the conjunctive normal form is the conjunction $C_1 \wedge C_2 \wedge ... C_k$. Is the set of clauses satisfiable?

Problem. Independent set: In graph $G = (V, E)$, two nodes $i, j \in V$ are independent if $e_{ij} \notin E$. Given graph G and number k, does G contain an independent set of nodes of at least size k?

Problem. Vertex cover: Given graph G, a set of nodes $S \subseteq V$ is a vertex cover, if for each $e_{ij} \in E$ at least one of i or $j \in S$. Given graph G and a number k, does G contain a vertex cover of size at most k?

Problem. Set cover: Given a set U of elements, a collection of S_i subsets of U, and a number k. Does there exists a collection of at most k subsets, such as $\cup S_i = U$?

Problem. Hamiltonian cycle: Given graph G, is there a cycle that visits each vertex exactly once?

Problem. Traveling salesman: Given a set of distances on n cities, and a bound D, is there a tour of length at most D?

Problem. Subset sum: Given natural numbers $w_i, w_2, ...w_n \in \mathbb{N}$ and a target number W, is there a subset of w_i-s such that they add up to precisely W? (Note we considered the relaxed version of this problem under dynamic programming.)

Problem. 3D matching: Given disjoint sets X, Y and Z each of size n and ordering triplets $T \subseteq X \times Y \times Z$, does there exist a set of n triplets in T, such that each element in $X \cup Y \cup Z$ is contained in exactly one of these?

4.1 Polynomial time reduction

Objective: we would like to classify problems according their difficulty. Tool: compare their relative difficultly. This is what we call *reduction.*

X is at least as hard as Y.

Moreover, we would like to find a solution that works also for polynomial problems, and therefore we introduce *polynomial time reduction*:

Can arbitrary instance of Y be solved using a polynomial number of computational steps, plus polynomial number of calls to a black box that solves X*?*

If the answer is *yes*, we write:

$$Y \leq_p X,$$

and say *Y is polynomial-time reducible to X* or *X is at least as hard as Y.*

Theorem 31. *Suppose $Y \leq_p X$. If X can be solved in polynomial time, then Y can be solved in polynomial time.*

Theorem 32. *Suppose $Y \leq_p X$. If Y can not be solved in polynomial time, then X can not be solved in polynomial time.*

Example. Independent set and Vertex cover. We can show that *Independent set* $\leq_p$ *Vertex cover* and *Vertex cover* $\leq_p$ *Independent set.* It means that the two problems have the same difficulty.

Example. Vertex cover and Set cover. We can show that *Vertex cover* $\leq_p$ *Set cover.* We can not show the opposite direction, so it means that: if we know how to solve set cover, we can call it polynomial number of times to solve Vertex cover. Set cover is at least as hard as vertex cover.

Here is a possible 3-SAT expression:

$$(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_4)$$

.

Example. 3-SAT and Independent set: 3-SAT $\leq_p$ Independent set.

Theorem 33. Transitivity of reductions: If $Z \leq_p Y$ and $Y \leq_p X$, then $Y \leq_p X$

Example. Considering our previous examples, we have: 3-SAT $\leq_p$ Independent set $\leq_p$ Vertex cover $\leq_p$ Set cover, that is, 3-SAT $\leq_p$ Set cover.

We an translate it to say that the Set cover is at least as difficult as the 3-SAT.

4.2 Definition of complexity classes P, NP, co-NP

Superficial definition of NP class: a decision problem is in NP, if a "yes" answer can be proved in polynomial time. E.g., the satisfiability of a 3-SAT can be proved by giving a string $\bar{x}$ that solves the problem. The existence of a Hamiltonial cycle can be proved by the sequence of edges that form a Hamiltonial cycle. In both cases the correctness of the solution can be checked in polynomial time.

Now let's see the exact definition.

Definition. Optimization/computation problem: the results is some finite string s. **Decision problem X:** defined by the set of strings on which the answer is *yes.* An algorithm A for a decision problem receives an input s, and outputs *yes* or *no.* A solves X, if $A(s) = yes$ iff $s \in X$.

As an example, Problem Primes = {2,3,5,7,...}. AgrawalKayalSaxena (AKS) primality test is an algorithm that solves the problem.

Another example, for $G(V, E)$, the Problem Hamiltonial Cycle is given by all the sequences of edges that form such a cycle. It is easy to construct and algorithm, that checks whether a list of edges form a H-cycle.

Definition. Efficient certifier: (other names exist, like oracle, witness) Algorithm $B(s, t)$ is a certifier for decision problem X if for every string s, $s \in X$ iff there exists a string t such that $C(s, t) =$ yes. B is an efficient certifier, if B as well as t is polynomial in s.

Example, back to the Hamilton cycle: t is the string of edges, $B(t, s)$ is the algorithm that checks whether t is a Hamilton-cycle.

Example: composite numbers: t is a nontrivial factor of number s, $B(t, s)$ checks whether $1 < t < s$ and s is multiple of t.

Example: independent set problem: t is the identity of a set of at least k vertexes, and B checks that no edge connects any pair of them ($O(k^2)$).

Definition. NP: the set of all decision problems for which there is an efficient certifier.

NP comes from *nondeterministic-polynomial*, and stands for algorithms that run in polynomial time, if we allow nondeterministic algorithm, these algorithms evaluate all branches of a decision tree simultaneously.

Theorem 34. *$P \subseteq NP$.*

Proof. Consider a problem $X \in$ P. We will show that it is in NP as well. Since it is in P, there is a polynomial algorithm A that solves X.

We design the certifier B as follows: independently from t, $B(s, t)$ simply returns the value $A(s)$. Note, X is a decision problem, so the answer is yes or no.

□

The fundamental question whether P=NP is open.

Definition. EXP: the set of all problems for which there is an exponential-time algorithm .

Definition. NP-complete problem: A decision problem X, (i) $X \in$ NP, and ii) for all $Y \in$ NP $Y \leq_P X$.

In words: X is NP complete, if it is in NP and every problem in NP can be reduced to X. (X is at least as difficult as any other problem in NP.)

Theorem 35. *Suppose X is NP-complete. Then X is solvable in polynomial time iff P=NP.*

It is not at all straightforward though that NP-complete problems exist.

History: first NP-complete problem is the circuit satisfiability, then it was shown that it can be reduced to 3-SAT. Which means, 3-SAT is NP-complete. Finally, 3-SAT can be reduced to several other problems (see the Karp tree), and so on.

Theorem 36. *The circuit-SAT problem is NP-complete.*

We do not prove this here. However, the proof is based on the idea to convert any algorithm to a circuit.

Example. Transforming the independent set problem to circuit-SAT.

Given a graph G, does it contain a two-node independent set?

This problem is of course NP. Now we construct a circuit that is an efficient certifier for the problem. This certifier needs to be constructed for this specific graph G. The example in the book is a certifier for a three node graph with edges vu and vw. The circuit gives a result "yes", if the input is v=0, u=1, w=1.

Theorem 37. *If Y is NP-complete and X is a problem in NP with the property that $Y \leq_P X$, then X is NP-complete.*

4.2.1 Proving NP-Completeness

Given a new problem X we would like to prove that it is NP-complete.

Cook reduction (also known as polynomial-time Turing reduction) General reduction with polynomial number of calls to black box:

1. Prove that $X \in$ NP (that is, there is a poly-time certifier)
2. Choose a problem Y that is known to be NP-complete
3. Prove that $Y \leq_P X$, that is, Y can be solved by multiple calls to a black box solving X.

Karp reduction, when there is only one call to the black-box: that is, we transform every instance of Y into a single instance of X with the same answer (note, that Y is the problem that is known to be NP-complete).

1. Prove that $X \in$ NP
2. Choose a problem Y that is known to be NP-complete
3. Consider an *arbitrary* instance of s_Y, and show how to construct in polynomial time an instance s_X, such that s_X=yes iff s_Y-yes.

4.2.2 Co-NP

Note that according to the definition of NP, there is no short proof (certificate) of a "no" answer.

Definition. Complementary problem of X is $\bar{X}$: $s \in \bar{X}$ iff $s \notin X$.

Definition. A problem X is in co-NP iff the complementary problem belongs to NP.

We do not know whether NP = co-NP. But we know that such equality would have serious consequences:

Theorem 38. *If NP $\neq$ co-NP, then P $\neq$ NP.*

Proof. Prove the contrapositive statement: $P = NP \Rightarrow NP = co - NP$.

Assume $P = NP$:

$$X \in NP \Rightarrow X \in P \Rightarrow \bar{X} \in P \Rightarrow \bar{X} \in NP \Rightarrow X \in co - NP$$

and in the opposite direction

$$X \in co - NP \Rightarrow \bar{X} \in NP \Rightarrow \bar{X} \in P \Rightarrow X \in P \Rightarrow X \in NP.$$

.

Since we found $NP \subseteq co - NP$ and $co - NP \subseteq NP$, $NP = co - NP$.

□

We know that if $X \in$ P, then $X \in$ NP and $X \in$ co-NP, but we do not know whether $P = NP \cap co - NP$.

4.2.3 NP-hard

Definition. A problem X is **NP hard**, if for all $Y \in$ NP $Y \leq_P X$.

Note, that X does not need to be in NP, that is, it does not need to be a decision problem, and there is no need for polynomial-time certificate.

So, we discussed the concept of NP NP-complete for the class of **decision problems**, but most often we meet **optimization problems**. Clearly, by solving the optimization problem we immediately have answer for the related decision problem:

decision problem $\leq_P$ optimization problem.

It also means that the optimization problem is at least as hard as the decision problem. It also means that if the decision problem is hard, then the optimization problem is hard too. NP-Hard but not NP-Complete problems are typically these optimization problems. Like, *integer linear programming is NP-hard.*

Question: How about the opposite direction???? Often we can solve an optimization problem by calling a decision problem in linear or sublinear times.

4.2.4 Reduction examples

Here we list three somewhat different examples. More can be found in the book.

Theorem 39. *Independent set* $\leq_P$ *Vertex cover.*

Proof. That is, we want to show that the Independent set problem can be solved by calling a Vertex cover solver polynomial time. Consequence: vertex cover is at least as hard as independent set. Consequence: if Independent set is NP (NP-complete) then the Vertex cover is NP (NP-complete).

Note, both of the problems are in NP, given the set of nodes it can be checked in $O(k^2)$ time that none of them is connected by an edge, alt. it can me checked in $O(m)$ time that all edges are covered.

Lemma 17. *Let* $G = (V, E)$*. Then* $S \subseteq V$ *is an independent set iff its complement* $V - S$ *is a vertex cover.*

Proof. $S \subseteq V$ is an independent set $\Rightarrow V - S$ is a vertex cover: Consider an edge e_{ij}. Since S is independent set, one of i, j must be in $V - S$, that is, e_{ij} is covered in $V - S$. It holds for all edges, so $V - S$ is vertex cover.

$V - S$ is a vertex cover $\Rightarrow S \subseteq V$ is an independent set: Proof by contradiction. Consider u, v in S, such that there is an edge e_{uv}. But then there is this edge e_{uv} that is not covered by $S \subseteq V$, which contradicts to the assumption the it is a vertex cover. Consequently, there are no two nodes connected by an edge in S, which means it is an independent set. □

So let us go back to our main theorem. Consider a graph $G = (V, E)$. We would like to know whether there is at least k independent nodes. We can get an answer by asking the Vertex cover solver, whether there is a vertex cover with size at most $|V| - k$. □

Theorem 40. *Vertex cover* $\leq_P$ *Set cover.*

We just proved that Vertex-cover is NP-complete. Clearly, Set cover is in NP, a set of subsets S_i is a good certificate. If we prove the above theorem, we prove that Set cover is NP-complete.

Proof. We are looking for a solution where Vertex cover calls the black box that solves set cover. That is, for any graph G, we need to construct subset S_is.

We need to cover edges, therefore we select the set of items for set cover $U = E$.

A node covers all the edges it is connected to, so we define $S_i \subseteq U$ by all e_{ij}.

Now we claim that U can be covered with at most k subsets iff G has a vertex cover of at most k.

$\Rightarrow$: If $S_{i_1}, ...S_{i_l}$, $l \leq k$ is a set cover, then every edge in G is incident to one of i_j-s, and set $\{i_1, ...i_l\}$ is a vertex cover.

$\Leftarrow$: Quite the same reasoning.

□

Theorem 41. *Hamiltonial Cycle* $\leq_P$ *Traveling salesman (called traveling salesperson nowadays).*

Proof. That is, we want to show that the Hamiltonial Cycle problem can be solved by calling the Traveling salesman solver polynomial time. Consequence: if Hamiltonial cycle is NP (NP-complete) then the Traveling salesman is NP (NP-complete).

Note, both of the problems are NP, the sequence of nodes/cities visited is a good certificate.

Given graph $G = (V, E)$, the question is, whether G has a Hamiltonial cycle. For (directed) graph G we construct an instance of the TSP as follows: for each node $v \in V$ we have a city v'. We define $d(v', w') = 1$ if there is an edge $e_{v,w} \in G$, and we define it 2 otherwise. There is a tour of length n iff there is a Hamiltonian cycle in G. □

Theorem 42. *3-SAT $\leq_P$ Independent set. (Specifically, 3-SAT with k clauses, and independent set with at least k nodes.)*

Actually we should have started with this, since we know that 3-SAT in NP complete. So with this theorem we can prove that the Independent set problem is NP-complete.

Proof. The independent set problem is NP as we have shown before.

Now we construct an instance of the independent set problem that solves a given 3-SAT problem. That is, we try to define a graph where an independent set represents a solution. We will use the reasoning, that we need to select one item from each claus (of three terms), such that the k selections do not conflict.

Graph: $n = 3k$ nodes, triangles represent a clause. Nodes from different triangles are connected if they conflict in the SAT expression $(x_i, \bar{x}_i)$.

We need to show that 3-SAT is solvable iff there are at least k independent nodes in the graph.

3-SAT is solvable $\Rightarrow$ at least k independent nodes: Consider the nodes that represent terms with value 1 in the SAT solution. There needs to be at least one of them in each triangle. Let S contain one such a node from each triangle. They can not be connected by any "cross triangle" edge, since either x or $\bar{(x)}$ is 1. We have exactly k independent nodes in S.

At least k independent nodes $\Rightarrow$ 3-SAT is solvable: we construct a solution for SAT based on the independent set S. We set to 1 the variables that represent nodes in S, and 0 their negate. If neither x_i or $\bar{x}_i$ is in S, then we set x_i=1. □

4.3 Special cases - when NP is not NP

Problem. Maximum independent set in trees (or forests)

We have seen that the existence of independent set of size k is NP-complete in general graphs. However, if the graph has specific structure, specifically tree of forest, then the problem becomes P.

Proof. Greedy algorithm, based on the exchange argument. □

Consequence: you need to be observant to the structure of the problem, and if you discover that you always have a special case of a general problem, you need to prove that even this special case is in the given complexity class.

4.4 What have we learned today?

- Polynomial reduction to establish complexity relationships.
- Decision and optimization problems.
- The contept of efficient certifier.
- Definition of P, NP, co-NP, NP-complete and NP-hard.
- Relationships of these, and things we do not know.
- Reduction examples.
- Special cases that turn out to be in P.