Reductions and NP-Completeness

Problem

A problem is defined by a language. For a problem P, the corresponding language is L_P over some alphabet Σ, usually $\Sigma = \{0,1\}$. $L \subseteq \Sigma^*$, where $\Sigma^* = \{0,1\}^*$ - the language of all sequences over the alphabet Σ.

A problem / language presents a **decision problem**, i.e. for a given $x \in \{0,1\}^*$, is $x \in L_P$?

Polynomial-time languages

A language L is polynomial-time decidable, i.e. $L \in P$ if there exists an algorithm that runs in polynomial-time for and decides for each x whether it is in L.

Formally:

$L \in P \Leftrightarrow \exists A_L . T_{A_L} \in n^c$ where n is the length of the input and c is some constant.

The model of computation most of the time is a Turing machine. A pushdown automata is a weaker computational model than Turing machine, and a Quantum-computational model is stronger.

We will use the random access model of computation, and a Turing machine.

Decision problem:

Check whether an input belongs to the language of the problem, in contrast to optimization problem, e.g. minimum/maximum problems.

Usually when we have a solution to a decision problem, we can solve an optimization problem.

Verification problem:

Given an input and a witness / certificate, the verification algorithm verifies using the witness that the input is an instance of the language. We have an efficient way of verifying any answer to a problem in polynomial time.

Trivial example: for the language of sorted sequences, a witness is actually the input itself – a sorted sequence, which can be verified polynomially.

CNFsat:

For a sequence of Boolean variables $x_1, \dots, x_n, \overline{x_1}, \dots, \overline{x_n}$ $x_i \in \{0,1\}$ $(x_i = 0 \Leftrightarrow \overline{x_i} = 1)$, and a formula of $\vee$-clauses concatenated by $\wedge$, e.g. $\phi(x_1, x_2, x_3) = (\overline{x_1} \vee \backslash x_2) \wedge (x_1 \vee \backslash \overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x_3}) = c_1 \wedge c_2 \wedge c_3$, we want to know whether the formula ϕ has a satisfying assignment such that $\phi(c_1, c_2, c_3) = 1$.

The formula's form is CNF – conjunction normal form.

DNFsat:

Disjunctive normal form, similar to CNFsat, only opposite $\wedge, \vee$, e.g.:

$\phi' = (x_1 \wedge x_2 \wedge \overline{x_3}) \vee (\overline{x_1} \wedge x_2 \wedge \overline{x_3} \wedge x_4) \vee x_4$ – easy to solve, simply satisfy one clause.

Every CNF can be converted to DNF, but not in polynomial time.

Let $\phi = \wedge_{i=1}^{N} C_i$ where clauses C_i are from the form $(b_1 \vee \dots \vee b_k)$ $(b_i \in \{x_1, \dots, x_n, \overline{x_1}, \dots, \overline{x_n}\}$.

Given a witness which is an assignment of $x_1, \dots, x_n$, verifying if $\phi \in CNF$ is solvable in polynomial time.

Clique:

Given a graph $G = (V, E)$ and an integer k, does the graph G have a clique (set of vertices that are completely connected) of size k.

The corresponding optimization problem is: find the largest clique in the graph G.

Clique can be verified: given a graph G and a certificate – we will simply verify that any pair of vertices in the certificate has a corresponding edge in E, thus verifying that the pair $< G, k > \in Clique$.

Oracle: given an instance of a problem it answers "yes" or "no".

A non-deterministic Turing machine has an oracle access (can access it polynomial number of times).

K-colorability:

Given a graph $G = (V, E)$, can we color the vertices with at most k colors such that no two adjacent vertices have the same color.

2-colorability: equivalent to whether the graph is bipartite – which is polynomially solvable.

3-colorability: very hard problem.

Integer Programming:

Given a set of linear constraints and a linear function, find an integer point maximizing the function under the constraints.

Traveling salesman (TSP):

Equivalent to the Hamiltonian-Cycle problem: find the shortest cycle that goes through all vertices of a graph.

This problem is NP-hard, that is we cannot verify a witness in polynomial time.

Cook's theorem:

CNF is NPC (NP-Complete).

How can we solve in polynomial time with an oracle access:

- Set $x_i = 1$ and remove all clauses that are now satisfied.
- Ask the oracle if the formula is still satisfyable.
  - If yes, continue to x_{i+1}
  - Else set $x_i = 0$ and continue

Reducing:

For a polynomial-reduction of P_1 to P_2:

- If P_2 is easy then P_1 is easy.
- In the opposite direction: if P_2 is hard, then P_1 is at least as hard as P_1.

The reduction function $R: P_1 \rightarrow P_2$ has to be polynomial, that is $R = P$ ($T_R = n^c$).

To show a problem P is hard: create a reduction $R: CNF \rightarrow P$

A problem P is NP-hard if there's a polynomial reduction $R: CNF \rightarrow P$ and we can show there's no witness that can verify the problem in polynomial time.

A problem A is reducible to problem B if there exists an efficient algorithm $R: A \rightarrow B$ such that $\forall x \in A: R(x) \in B$.

NP-Complete problems

Problems in NP can be solved efficiently with a non-deterministic guess and a verify algorithm.

A problem is in NPC if it is NP and every other problem in NP can be reduced to it.

To show a problem is NPC:

- Show it is verifiable in polynomial time.
- Show a reduction $R: CNF \to P$ (show that $CNF \leq_p P$), or any other known NPC problem other than CNF.

coNP:

$L \in coNP \Leftrightarrow \bar{L} \in NP$, where $\bar{L} = \{x \in \{0,1\}^* | x \notin L\}$ – the complement languages to all languages in NP.

Example of reduction: proving that Clique is NPC:

First, it is verifiable: the witness would be a clique in the graph, that can be verified in poly time.

Now we show a reduction from CNF:

Let $\phi(x_1, \dots, x_n) = C_1 \wedge C_2 \wedge \dots \wedge C_m$, where $C_i = (l_1 \vee l_2 \vee \dots \vee l_k), l_j \in \{x_i, \bar{x_i}\}_{i=1}^n$.

The CNF problem is: does an assignment $b_1, \dots, b_n$ exists such that $\phi(b_1, \dots, b_n) = 1$.

The Clique problem:

Given $G = (V, E), k \leq n$, does G contain a clique of size k? That is, there exists $\{v_1, \dots, v_k\} \subseteq V$ such that $G|_{\{v_1, \dots, v_k\}} \approx K_k$ (where K_k is the complete graph of size k, and $\approx$ is the isomorphism relation).

The reduction:

We create a graph G such that:

- Vertices: For each clause C_i, we create a node for each of the instances x_j in that clause (so we may get many copies of x_i).
- Edges: we connect any two vertices in the graph except for nodes that are compliments of each other or belong to the same clause

If G has a clique, we claim that we have a satisfying assignment to ϕ. The relation is actually if and only if.

The reduction is polynomial: the number of literals in ϕ is the number of vertices in the graph, and the maximum number of edges still keeps the reduction polynomial.

⇒ Clique is at least as hard as CNF, therefore Clique is at least NPC. Since we showed that Clique is NP, then it's NPC.

Notation in reducability:

- Karp reducability: $A \leq_m^p B$
- If $A \leq_m^p B$ and $B \leq_m^p A$ then$A \equiv_m^p B$

IS (Independent Set):

For a given graph $G = (V, E)$, an independent set $U \subseteq V$ is an independent set $\Leftrightarrow \forall u, v \in U: (u, v) \notin E$.

This problem is the complement of the Clique problem. As an optimization problem, it's a maximization problem – you want the largest IS (the smallest – just one vertex, the same as for Clique).

The decision version: given a graph G and a number $k \leq n$, does G has an IS U such that $|U| = k$.

Proof that I$IS \in NPC$:

Verify: the witness would be a set of vertices of size k, and we verify by making sure none are connected – easy in polynomial time.

Reduce: we will show that $Clique \leq_m^p IS$:

Given an input for clique $< G', l >$, we create a new graph G with the same set of vertices, and the inverted edge set, that is $E = \overline{E'}$: for any edge in E', there won't be an edge in E, and for every edge not in E', we put that edge in E. Then the input we produce for IS is $< G, l >$, as a clique of size l in G' is a clique of size l in G.

Lastly, the reduction is polynomial, trivial.

Therefore, since $Clique \leq_m^p IS$ and $IS \in NP$ then $IS \in NPC$.

Vertex-Cover:

For a graph G, a vertex-cover is a set $U \subseteq V$ is a set of vertices such that $\forall (u, v) \in E$ either $u \in U$ or $v \in U$. The optimization problem is a minimum problem (we want the smallest-sized set).

The decision problem: given a pair $< G, k >$, does G has a VC of size at most k?

Proof that $VC \in NPC$: