Defining Tractability

Question:

When is a problem "tractable"?

Conventional answer:

iff it allows a polynomial algorithm

Why not: "if it allows an $O(n^2)$ algorithm"?

- this would be arbitrary
- composing two such algorithms may give an $O(n^4)$ algorithm

Since polynomials are closed under most operations, the conventional answer enables the development of an elegant theory.

Outline

- We have seen many problems that allow polynomial solutions
- while for many problems we do not know if they have a polynomial solution
- but many of those problems are related in the sense that if one of them has a polynomial solution then all of them have.

Restrictions:

- we focus on decision problems:

  does x belong to X, yes or no?

  We identify a decision problem with the set of its "yes" instances.
- we can in most cases reduce an optimization problem to a decision problem, and vice versa.
- we only consider deterministic algorithms

The Set $\mathcal{P}$

$\mathcal{P}$ consists of those decision problems that can be solved in time $O(p(x))$ where

- x is the bit size of a "natural encoding" of the input
- p is a polynomial

Example element of $\mathcal{P}$:

in a graph with n nodes,
where edges have lengths,
is there a path from a to b of length ≤ 10*?*

The Set $\mathcal{NP}$, Motivation

Intuitively, $\mathcal{NP}$ should consist of those decision problems where a yes answer can be equipped with a certificate. A couple of examples:

Non-Primality:

- appears hard to check (deterministically) if n is non-prime
- but once m, q are given, easy to verify that $n = mq$

Hamiltonian Cycle: (a cycle that includes all nodes)

- appears not easy to see if given graph contains a Hamiltonian cycle
- but once a list of nodes is given, easy to verify if they do form a Hamiltonian cycle.

The Set $\mathcal{NP}$, Definition

A decision problem X (the set of "yes" instances) is in $\mathcal{NP}$ iff there is a set F and polynomial p such that

- $F \subseteq X \times Q$ with Q the set of certificates (no-instances don't have certificates)
- for all $x \in X$, there exists $q \in Q$ such that $<x, q> \in F$ and the size of q is at most $p(|x|)$
- a polynomial time algorithm can check membership of F.

If $x \in X$ it is thus possible to verify that fact in polynomial time, once a certificate has been given.

- Observe that $\mathcal{P} \subseteq \mathcal{NP}$ (choose say 0 as certificate and ignore it)
- is the inequality strict? that is, is $\mathcal{P} = \mathcal{NP}$? that's the \$1M question (literally)

Reductions

We can ofte connect problems by showing that if we can do one we can also do the other.

- if we can multiply then we can surely also square
- but if we can square then we can also multiply:

$$x * y = \frac{(x+y)^2 - (x-y)^2}{4}$$

Polynomial Many-One Reductions

We say that decision problem X is polynomially many-one reducible to decision problem Y, to be written $X \;\leq_m^p\; Y$, if there exists f such that

$$x \in X \textit{ iff } f(x) \in Y$$

and f can be computed in polynomial time.

- in particular, $|f(x)|$ is polynomial in $|x|$.

Theorem: if $X \;\leq_m^p\; Y$ and $Y \in \mathcal{P}$ then also $X \in \mathcal{P}$.

Transitivity: if $X \;\leq_m^p\; Y$ and $Y \;\leq_m^p\; Z$ then $X \;\leq_m^p\; Z$.

Example Reduction

We have $HC \leq_m^p TS$ where

- HC is the problem of detecting if a graph has a Hamiltonian cycle
- TS is the problem of detecting if a table of distances between each pair of cities allows a traveling salesman to visit each city once, and come back home again, while traveling at most given d

For given a graph $G = (V, E)$, construct table D by stipulating that

- if $(u, v) \in E$ then $D(u, v) = 1$
- if $(u, v) \notin E$ then $D(u, v) = 2$

Thus $G \in HC$ iff D in $TS_{|V|}$

Optimization Problems

For an optimization problem, there often exists a decision problem such that a solution to the former translates into a solution to the latter, and vice versa.

Example: assume we want to study certain kinds of paths (like cycles where each node occurs exactly once).

- the decision problem $\textsc{AtMost}(k)$ asks whether the length of the shortest path is k or less.
- the optimization problem $\textsc{Shortest}$ finds the length of the shortest path.

If we can solve one we can solve the other:

- we can decide $\textsc{AtMost}(k)$ as the result of the comparison $\textsc{Shortest} \leq k$.
- we can find $\textsc{Shortest}$ as the smallest k such that $\textsc{AtMost}(k)$ holds.

If $\textsc{AtMost}$ runs in $O(n^a)$, and the shortest path has length in $O(n^b)$, then $\textsc{Shortest}$ runs in time $O(n^{a+b})$.

Construction Problems

We can often reduce construction problems to decision problems:

- if we in polynomial time can decide if a graph contains a Hamiltonian cycle
- we can in polynomial time find a Hamiltonian cycle.

For we consider each edge e in turn, and ask whether the graph still has a Hamiltonian cycle even if e is removed

- if "yes", remove e
- if "no", make e part of the cycle

If decision is in $O(n^q)$ then construction is in $O(n^{q+2})$.

It is trivial to reduce decision problems to construction problems.

Defining $\mathcal{NP}$-Hardness

We say that X is $\mathcal{NP}$-hard if all problems in $\mathcal{NP}$ can be polynomially many-one reduced to X.

- if X is $\mathcal{NP}$-hard
- but a polynomial time algorithm is found for X
- then $\mathcal{P} = \mathcal{NP}$ (which is unlikely)

If we have shown a problem to be $\mathcal{NP}$-hard, you don't feel bad about not being able to find polynomial solution!

- if X is $\mathcal{NP}$-hard
- and $X \;\leq_m^p\; Y$
- then also Y is $\mathcal{NP}$-hard

If $X \in \mathcal{NP}$ is $\mathcal{NP}$-hard we say that X is $\mathcal{NP}$-complete.

Finding *NP*-Hard Problems

- if X is $\mathcal{NP}$-hard and $X \le_m^p Y$ then Y is $\mathcal{NP}$-hard
- but how do we find just one $\mathcal{NP}$-hard problem?

A "first" $\mathcal{NP}$-hard problem [Cook, Levin] is SAT:

given a boolean formula ϕ
decide if one can assign truth values to variables such that ϕ is true (satisfied)

- SAT is in $\mathcal{NP}$ since the satisfying assignment can be used as certificate.
- SAT is $\mathcal{NP}$-hard because (pages of details omitted) any computation can be represented as a boolean formula.

We shall now see other $\mathcal{NP}$-complete problems.

CSAT

CSAT:

given a boolean formula ϕ in CNF
decide if one can assign truth values to variables such that ϕ is true (satisfied)

- A formula is in CNF if it is a conjunction of clauses
- A clause is a disjunction of literals
- A literal is a variable, or the negation of a variable

Trivially, CSAT $\leq_m^p$ SAT.

- CSAT is in $\mathcal{NP}$
- CSAT is $\mathcal{NP}$-hard, as we shall show by establishing SAT $\leq_m^p$ CSAT

3-SAT

3-SAT:

given a boolean formula ϕ in CNF
where each clause has at most 3 literals
decide if one can assign truth values to variables
such that ϕ is true (satisfied)

Trivially, 3-SAT $\leq_m^p$ CSAT.

- 3-SAT is in $\mathcal{NP}$
- 3-SAT is $\mathcal{NP}$-hard, as we shall show by establishing CSAT $\leq_m^p$ 3-SAT.

Clique

Given an undirected graph (V, E), a clique C is a subset of V such that for all $u \neq w \in C$, the edge (u, w) belongs to E.

- all singleton sets are cliques
- a graph with at least one edge has a clique of size 2

The decision problem Clique asks if a given graph contains a clique of size k.

- Clique is in $\mathcal{NP}$
- Clique is $\mathcal{NP}$-hard, as we shall show by establishing 3-Sat $\leq_m^p$ Clique.

VC

Given an undirected graph (V, E), a vertex cover C is a subset of V such that for all edges $(u, w) \in E$, either $u \in C$ or $w \in C$ (or both).

- for all u, the set $V \setminus \{u\}$ is a vertex cover

The decision problem VC asks if a given graph contains a vertex cover of size k.

- VC is in $\mathcal{NP}$
- VC is $\mathcal{NP}$-hard, as we shall show by establishing CLIQUE $\leq_m^p$ VC.

Outline of Reductions

Recall that we shall establish the chain

$$\textsc{Sat} \leq_m^p \textsc{CSat} \leq_m^p 3\text{-}\textsc{Sat} \leq_m^p \textsc{Clique} \leq_m^p \mathrm{VC}$$

Since $\textsc{Sat}$ is $\mathcal{NP}$-hard (seminal result) this will establish that all the other problems are also $\mathcal{NP}$-hard.

Reducing CLIQUE to VC

Observe that (with $\overline{X}$ the complement of X) the following claims are equivalent:

$$C \text{ is a clique in } (V, E)$$
$$\forall u \neq w \in V : (u, w \in C \Rightarrow (u, w) \in E)$$
$$\forall u \neq w \in V : ((u, w) \notin E \Rightarrow u \notin C \vee w \notin C)$$
$$\forall u \neq w \in V : ((u, w) \in \overline{E} \Rightarrow u \in \overline{C} \vee w \in \overline{C})$$
$$\overline{C} \text{ is a vertex cover for } (V, \overline{E})$$

Given (V, E) with $|V| = n$, we see:

(V, E) *has a clique of size* k *iff*
$(V, \overline{E})$ *has a vertex cover of size* $n - k$

Reducing 3-SAT to CLIQUE

Given CNF formula ϕ with k clauses, each having at most 3 literals, construct graph G such that

- we have a node for each literal (one node for each occurrence)
- we have an edge between l_1 and l_2 iff
  - they occur in different clauses
  - they are not contradictory (l_1 not negation of l_2)

Lemma: ϕ can be satisfied iff G has a k-clique.

- Assume that ϕ is satisfied by A. Then each clause has at least one literal that is true wrt. A; let one of those go into C. Then C has k elements, all of which are connected by edges.
- Assume that C is a clique with k elements. The literals in C do not contradict each other; hence, we can construct a truth assignment A that assigns true to all literals in C. Since C must consist of one literal from each clause, ϕ will be satisfied by A.

Reducing CSAT to 3-SAT

Let us just show how to reduce 4-SAT to 3-SAT; the generalization is straight-forward. So let

$$\phi = x \vee y \vee z \vee w$$

be given. With u a fresh variable, now define

$$\phi' = (x \vee y \vee u) \wedge (z \vee w \vee \neg u).$$

Lemma: A satisfies ϕ iff an extension of A satisfies ϕ'.

- first assume that A satisfies ϕ. Wlog, assume $A(y) = \text{true}$. Now extend A to A' by stipulating $A'(u) = \text{false}$. Then A' satisfies ϕ'.
- Next assume that A satisfies ϕ'. Wlog, assume that $A(u) = \text{true}$. But then A satisfies $z \vee w$ and hence ϕ.

Reducing SAT to CSAT

Given arbitrary boolean expression, first convert it to an equivalent expression ϕ in NNF (negation normal form)

- why not just normalize it all the way into CNF?
- this could cause exponential blow-up.

Instead, convert to ϕ' in CNF such that

- all variables in ϕ occur also in ϕ'
- any satisfying assignment for ϕ can be extended into a satisfying assignment for ϕ'
- the restriction of any satisfying assignment for ϕ' is a satisfying assignment for ϕ

Thus ϕ is satisfiable iff ϕ' is.

- if ϕ is literal, then $\phi' = \phi$
- if $\phi = \phi_1 \wedge \phi_2$, apply induction hypothesis to find ϕ'_1 and ϕ'_2, and then let $\phi' = \phi'_1 \wedge \phi'_2$
- if $\phi = \phi_1 \vee \phi_2$, we need a more complex constrution. Details in *Howell*,p.519-520.