coNP AND FUNCTION PROBLEMS

- The class of **coNP**
- The relationship of **coNP** and **NP**
- The class **coNP** $\cap$ **NP**
- Function problems vs. decision problems
- Classes of function problems
- Total functions

(C. Papadimitriou: *Computational Complexity*, Chapter 10)

1. The class of complement problems **coNP**

- **NP** is the class of problems with succinct certificates.
- $\mathbf{coNP} = \{L \mid \overline{L} \in \mathbf{NP}\}$ is the class of problems with succinct disqualifications.

  Example. Consider the problem of VALIDITY:
  INSTANCE: A Boolean expression ϕ in CNF.
  QUESTION: Is ϕ valid?
- VALIDITY is in **coNP**: for an expression ϕ which is not valid, a falsifying truth assignment is a succinct disqualification.
- HAMILTON PATH COMPLEMENT and SAT COMPLEMENT are also in **coNP**.
- $\mathbf{P} \subseteq \mathbf{coNP}$

coNP-completeness

Definition. A language L is **coNP**-*complete* iff $L \in \mathbf{coNP}$ and $L' \leq_{\mathrm{L}} L$ holds for every language $L' \in \mathbf{coNP}$.

Proposition. HAMILTON PATH COMPLEMENT and VALIDITY are **coNP**-complete.

Proof. Every language $L \in \mathbf{coNP}$ is reducible to VALIDITY, because $\overline{L} \in \mathbf{NP}$ and, hence, there is a reduction R from $\overline{L}$ to SAT such that for every string x, $x \in \overline{L}$ iff $R(x) \in$ SAT. But then for a reduction $R'(x) = \neg R(x)$, $x \in L$ iff $R(x) \notin$ SAT iff $R'(x) = \neg R(x) \in$ VALIDITY.

Similarly for HAMILTON PATH COMPLEMENT. □

2. The Relationship of **coNP** and **NP**

Proposition. If $L \subset \Sigma^*$ is **NP**-complete, then its complement $\overline{L} = \Sigma^* - L$ is **coNP**-complete.

Further observations:

- It is open whether $\mathbf{NP} = \mathbf{coNP}$.
- If $\mathbf{P} = \mathbf{NP}$, then $\mathbf{NP} = \mathbf{coNP}$ (and $\mathbf{P} = \mathbf{coNP}$).
- It is possible that $\mathbf{P} \neq \mathbf{NP}$ but $\mathbf{NP} = \mathbf{coNP}$ (however, it is strongly believed that $\mathbf{NP} \neq \mathbf{coNP}$).
- The problems in **coNP** that are **coNP**-complete are the least likely problems to be in **P** and also in **NP** (see below).

Do coNP and NP coincide?

Proposition. If a **coNP**-complete problem is in **NP**, $\mathbf{NP} = \mathbf{coNP}$.

Proof.

Suppose that L is a **coNP**-complete problem that is in **NP**.

($\supseteq$) Consider $L' \in \mathbf{coNP}$. Then there is a reduction R from L' to L. Then $L' \in \mathbf{NP}$, because L' can be decided by a polynomial time NTM which on input x computes first $R(x)$ and then starts the NTM for L.

($\subseteq$) Consider $L' \in \mathbf{NP}$. Then $\overline{L'} \in \mathbf{coNP}$ and there is a reduction R from $\overline{L'}$ to L. Then similarly $\overline{L'} \in \mathbf{NP}$ and hence $L' \in \mathbf{coNP}$. $\square$

The primality problem PRIMES

INSTANCE: An integer N in binary representation.
QUESTION: Is N a prime number?

- PRIMES $\in \mathbf{coNP}$ as any divisor acts as a succinct disqualification.
- Note that a $\mathrm{O}(\sqrt{N})$ algorithm for PRIMES testing all relevant divisor candidates is only pseudopolynomial.
- PRIMES $\in \mathbf{NP}$ (as shown below) and hence PRIMES $\in \mathbf{coNP} \cap \mathbf{NP}$.
- New result in August 2002:
  M. Agrawal, N. Kayal, N. Saxena: *PRIMES is in* $\mathbf{P}$!!

3. The Class coNP ∩ NP

- Problems in $\mathbf{coNP} \cap \mathbf{NP}$ have both succinct certificates and disqualifications.
- $\mathbf{P} \subseteq \mathbf{coNP} \cap \mathbf{NP}$ as $\mathbf{P} \subseteq \mathbf{coNP}$ and $\mathbf{P} \subseteq \mathbf{NP}$.
- If two problems in **NP** are *dual*, i.e. each is *reducible to the complement* of the other, then both are in $\mathbf{coNP} \cap \mathbf{NP}$.

Example.
MAX FLOW(D): Has a network N a flow of at least K from s to t?
MIN CUT(D): Given a network, is there a set of edges of capacity of at most B such that deleting these disconnects s from t?

Now by the max flow–min cut theorem, N *has* a flow of value at least K iff it *does not have* a cut of capacity $K-1$ or less.

PRIMES has succinct certificates

A succinct certificate for primality can be obtained using the following theorem.

Theorem. A number $p > 1$ is prime iff there is a number $1 < r < p$ such that $r^{p-1} = 1 \bmod p$ and, furthermore, $r^{\frac{p-1}{q}} \neq 1 \bmod p$ for all prime divisors q of $p-1$.

Corollary. PRIMES is in $\mathbf{NP} \cap \mathbf{coNP}$.

- The theorem provides a succinct certificate for the primality of p:
  $$C(p) = (r; q_1, C(q_1), \ldots, q_k, C(q_k))$$
  where $C(q_i)$ is a *recursive* primality certificate for each prime divisor q_i of $p-1$.
- The recursion stops for prime divisors $q_i = 2$ for which $C(q_i) = (1)$.

Verifying the certificate $C(p)$

The following observations can be made:

- The certificate $C(p)$ is polynomial in the length of p (in $\log p$) and it can be checked by division and exponentiation.
- Ordinary multiplication and division are doable in polynomial time in the length of the input (in binary representation).
- Exponentiation $r^{p-1} \bmod p$ can be done in polynomial time by repeated squaring $r^1, r^2, r^4, \ldots r^{2^l} \pmod p$ where $l = \lfloor \log_2(p-1) \rfloor$ and then with at most l additional multiplications.

☞ The certificate $C(p)$ can be checked in polynomial time. □

4. Function Problems vs. Decision Problems

- We have studied decision problems but many problems in practice require a more complicated answer than "yes" / "no".

  Example. Find a satisfying truth assignment for a formula.

  Example. Compute an optimal tour for TSP.
- Such problems are called *function problems*.
- Decision problems are useful surrogates of function problems only in the context of *negative complexity results*.

  Example. SAT and TSP(D) are **NP**-complete. Then unless $\mathbf{P} = \mathbf{NP}$, there is no polynomial time algorithm for finding a satisfying truth assignment or an optimal tour.

The relationship of SAT and FSAT

FSAT: given a Boolean expression ϕ, if ϕ is satisfiable then return a satisfying truth assignment of ϕ otherwise return "no".

- If FSAT can solved in polynomial time, then clearly so can SAT.
- If SAT can be solved in polynomial time, then so can FSAT using the following algorithm given input ϕ with variables $x_1, \ldots, x_n$ ($\phi[x = \mathbf{true}]$ denotes ϕ where variable x is replaced by $\mathbf{true}$):

  if $\phi \notin$ SAT then return "no";
  for all $x \in \{x_1, \ldots, x_n\}$ do
   if $\phi[x = \mathbf{true}] \in$ SAT then $T(x) := \mathbf{true}$; $\phi := \phi[x = \mathbf{true}]$
   else $T(x) := \mathbf{false}$; $\phi := \phi[x = \mathbf{false}]$;
  return T;

The relationship of TSP(D) and TSP

- If TSP can solved in polynomial time, then clearly so can TSP(D).
- If TSP(D) can solved in polynomial time, then so can TSP in the following way.
- An optimal tour can be found using the algorithm below which finds
  1. the cost $0 \le C \le 2^n$ of an optimal tour by binary search and
  2. an optimal tour using the cost C computed in step 1.

  (Here n is the length of the encoding of the problem instance.)
- Both steps involve a polynomial number of calls to the polynomial time algorithm for TSP(D) (assuming that such an algorithm exists).

An algorithm for TSP

An algorithm for TSP(D) is used as a subroutine:

/* Find the cost C of an optimal tour by binary search*/
$C := 0$; $C_u := 2^n$;
while $(C_u > C)$ do
  if there is a tour of cost $\lfloor (C_u + C)/2 \rfloor$ or less then
    $C_u := \lfloor (C_u + C)/2 \rfloor$
  else $C := \lfloor (C_u + C)/2 \rfloor + 1$;
/* Find an optimal tour */
For all intercity distances do
  set the distance to $C+1$;
  if there is a tour of cost C or less, freeze the distance to $C+1$
  else restore the original distance and add it to the tour;
endfor

5. Classes of Function Problems

Definition. Let $L \in \mathbf{NP}$. Then there is a polynomial time decidable and polynomially balanced relation R_L such that for all strings x, there is a string y with $R_L(x,y)$ iff $x \in L$.

The *function problem* associated with L (denoted FL) is:

Given x, find a string y such that $R_L(x,y)$ if such a string y exists; otherwise return "no".

- The class of all function problems associated as above with languages in **NP** is called **FNP**.
- **FP** is the subclass of **FNP** solvable in polynomial time.
- FSAT is in **FNP** and FHORNSAT is in **FP** (but it is open whether TSP is in **FNP**).

Reductions and completeness for function problems

A function problem A *reduces* to a function problem B if there are string functions R, S computable in logarithmic space such that for all strings x, z: if x is an instance of A, then $R(x)$ is an instance of B and if z is a correct output of $R(x)$, then $S(z)$ is a correct output of x.

- Reductions compose among function problems.
- A problem A is *complete* for a class F$\mathcal{C}$ of function problems if it is in F$\mathcal{C}$ and every problem in F$\mathcal{C}$ reduces to A.
- **FP** and **FNP** are closed under reductions.
- FSAT is **FNP**-complete.
- $\mathbf{FP} = \mathbf{FNP}$ iff $\mathbf{P} = \mathbf{NP}$.

6. Total Functions

- There are certain important problems in **FNP** that are guaranteed to never return "no".

  Example. FACTORING: Given an integer N, find its prime decomposition $N = p_1^{k_1} \cdots p_m^{k_m}$.
  (No known polynomial time algorithm).
- FACTORING seems to be different from the other hard problems in **FNP**: it is a total function in a sense:

  Definition. A problem L in **FNP** is called *total* if for every string x there is at least one string y such that $R_L(x,y)$.
- The subclass of **FNP** containing all total function problems is denoted by **TFNP**.

Total functions—cont'd

There are also other problems in **TFNP** with no known polynomial time algorithm.

Example. HAPPYNET:
INSTANCE: An undirected graph $G = (V, E)$ with integer weights w on edges.

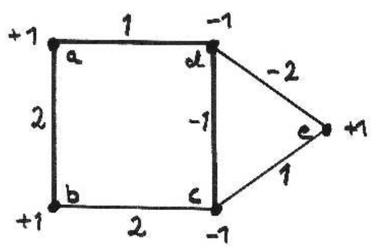
GOAL: Find a state of the graph where all nodes are happy.

- A state is a mapping $S : V \longrightarrow \{-1, +1\}$.
- A node i is happy in a state S of $G = (V, E)$ if

$$S(i) \cdot \sum_{[i,j] \in E} S(j)w[i, j] \geq 0.$$

Example.

- Consider the graph below and a state S such that $S(a) = S(b) = S(e) = 1$ and $S(c) = S(d) = -1$
- Node a is happy as $1 \cdot (1 \cdot 2 + -1 \cdot 1) = 1 \geq 0$
- Node c is unhappy as $-1 \cdot (1 \cdot 2 + -1 \cdot -1 + 1 \cdot 1) = -2 < 0$



Properties of HAPPYNET

- Every instance is guaranteed to have a happy state which can be found using the following algorithm:

  Start with any S and while there is an unhappy node, flip it.
- This algorithm is not polynomial but pseudopolynomial $O(W)$ where W is the sum of all weights.
- No polynomial algorithm known.
- HAPPYNET is equivalent with finding stable states in neural networks in the Hopfield model.

Other total functions

- ANOTHER HAMILTON CYCLE is **FNP**-complete.
- ANOTHER HAMILTON CYCLE for cubic graphs is in **TFNP**.
- EQUAL SUMS:

  Given n positive integers $a_1, \ldots, a_n$ such that $\sum_{i=1}^{n} a_i < 2^n - 1$, find two different subsets that have the same sum.
- EQUAL SUMS in **TFNP**.

  Proof. There are 2^n subsets of $a_1, \ldots, a_n$ and for each of them the sum is an integer between 0 and $2^n - 2$.

  Assume that all subsets have different sums. Then there are 2^n different integers between 0 and $2^n - 2$, a contradiction. Hence, there are two different subsets that have the same sum. □

Learning Objectives

- The definition of **coNP** and examples of languages from this class, e.g., VALIDITY.
- The characterization of **coNP** based on disqualifications.
- Reductions and completeness for function problems
- Relationship of decision problems and function problems