11 Intractable Problems

Theory of computability is the study of what can or cannot be computed by a TM/computer, among all problems. Theory of complexity is the study of what can or cannot be computed efficiently by a TM/computer, among all decidable problems.

11.1 Defining Complexity Classes

- A complexity class is specified by several parameters.
  — Model of computation. (e.g., k-string TM.)
  — Mode of computation: How a machine accepts its input. (e.g., deterministic/nondeterministic modes.)
  — Resource to measure. (e.g., time and space.)
  — Bound on resource. (e.g., polynomial time and exponential space.)
- Some complexity class forms: **TIME**$(f(n))$, **SPACE**$(f(n))$, **NTIME**$(f(n))$, **NSPACE**$(f(n))$.
  (**TIME**$(f(n))$ $\subseteq$**NTIME**$(f(n))$, **SPACE**$(f(n))$ $\subseteq$**NSPACE**$(f(n))$)
  (**NTIME**$(f(n))$ $\subseteq$**SPACE**$(f(n))$, **NSPACE**$(f(n))$ $\subseteq$**TIME**$(k^{\log n+f(n)})$)
- Some complexity classes:
  — **P**$=\cup_{k\geq 0}$**TIME**(n^k).
  — **NP**$=\cup_{k\geq 0}$**NTIME**(n^k). (**P**$\subseteq$**NP**)
  — **PSPACE**$=\cup_{k\geq 0}$**SPACE**(n^k). (**NP**$\subseteq$**PSPACE**)
  — **NPSPACE**$=\cup_{k\geq 0}$**NSPACE**(n^k). (**PSPACE**=**NPSPACE**)
  — **EXP**$=\cup_{k\geq 0}$**TIME**(2^{n^k}). (**P**$\subset$**EXP**, **NP**$\subseteq$**EXP**, **NPSPACE**$\subseteq$**EXP**)
  — **L=SPACE**$(\log n)$.
  — **NL=NSPACE**$(\log n)$. (**L**$\subseteq$**NL**$\subseteq$**P**)
  — **coP**$=\{L : \overline{L} \in$**P**$\}$. (**P=coP**)
  — **coNP**$=\{L : \overline{L} \in$**NP**$\}$. (**NP=coNP**???)
  — **coPSPACE**$=\{L : \overline{L} \in$**PSPACE**$\}$. (**PSPACE=coPSPACE**)
  — **coNPSPACE**$=\{L : \overline{L} \in$**NPSPACE**$\}$. (**NPSPACE=coNPSPACE**)
  Remark: For any complexity class $\mathcal{C}$, its complement **co**$\mathcal{C} = \{L : \overline{L} \in \mathcal{C}\}$. If $\mathcal{C}$ is deterministic, then $\mathcal{C} =$**co**$\mathcal{C}$. However, if $\mathcal{C}$ is nondeterministic, it is sometimes not known whether $\mathcal{C} =$**co**$\mathcal{C}$.

11.2 The class P

Reading: Sipser 7.2

- Definition: **P** is the class of problems solvable in polynomial time (number of steps) by deterministic TMs.
  $O(n^c)$, where c is a constant.
  Tractable (not so hard).
- Why use polynomial as the criterion?
  If a problem is not in **P**, it will be extremely expensive and probably impossible to solve in practice for large sizes.
  Polynomials have nice closure properties: $+, -, *$, and composition.
  P is independent of models of computation: Basic TM, TMs with multi-tracks, multi-tapes, multi-cursors, multi-strings, multi-dimensional tape. (All variations of TM except nondeterministic TM.)
- Examples of problems in **P**: Sorting, Searching, Selecting, Minimum Sapnning Tree, Shortest Path, etc..

11.3 The class NP

Reading: Sipser 7.3

- An NTM is an unrealistic (unreasonable) model of computation which can be simulated by other models with an exponential loss of efficiency. It is a useful concept that has had great impact on the theory of computation.
- NTM $N = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where $\delta : Q \times \Gamma \to 2^P$ for $P = Q \times \Gamma \times \{L, R\}$.
- $\delta(q, X)$ is a set a moves. Which one to choose? This is nondeterminism. The computation can be illustrated by a tree, with each node representing a configuration.
- Nondeterminism can be viewed as a kind of parallel computation wherein multiple independent processes or threads can be running concurrently. When a nondeterministic machine splits into several choices, that corresponds to a process forking into several children, each then proceeding separately. If at least one process accepts, then the entire computation accepts.
- Time complexity of nondeterministic TMs (NTMs): Let N be an NTM that is a decider (where all computation paths halt in the tree for any input). The time complexity of N, $f(n)$, is the maximum number of steps that N uses on any computation path for any input of length n. In other words, $f(n)$ is the maximum height of all computation trees for all input of length n.
- An unreasonable model of computation:

  Theorem; Every $T(n)$-time multi-tape TM has an equivalent $O(T^2(n))$-time single-tape TM.

  Theorem: Every $T(n)$-time single-tape NTM has an equivalent $O(2^{O(T(n))})$-time single-tape DTM.
- Definition: **NP** is the class of problems solvable in polynomial time by nondeterministic TMs.
- Another definition of nondeterministic TMs (algorithms):
  - Guessing phase: Guess a solution (always on target).
  - Verifying phase: Verify the solution.
- Example: TSP (DEC) is in **NP**.

  INSTANCE: An edge-weighted, undirected, and complete graph $G(V, E, w)$ and a bound $B \geq 0$. (The weight on an edge is allowed to be $+\infty$.)

  QUESTION: Is there a tour (a cycle that passes through each node exactly once) in G with total weight no more than B?

  A NTM N that decides TSP in polynomial time can be defined as follows: Assume that a coded instance of TSP is shown on the tape as input. N first generates an arbitrary permutation of the n nodes and then checks whether the permutation represents a tour with total weight no larger than the bound B. If so, N accepts the input. If not. N rejects the input, perhaps hoping that another choice of permutations may end up accepting. Note that the permutation generated can be any of the $n!$ possible permutations. The nondeterminism of the machine guarantees that if the instance is a yes-instance, the right permutation will always be selected.

  Note: The acceptance of an input by a nondeterministic machine is determined by whether there is an accepting computation among all, possibly exponentially many, computations. In the above proof, if there is a solution, it will always be generated by the Turing machine. This is like that a nondeterministic machine has a guessing power. A tour can only be found by a deterministic machine in exponential time, however, it can be found by a nondeterministic machine in just one step. So any nondeterministic proof should always contain two stages: Guessing and verifying.
- Example: Graph Coloring is in **NP**. Consider the decision problem: For a given graph, is there a coloring scheme of the nodes that uses no more than B colors such that no two nodes connected by an edge are given the same color? A nondeterministic TM first guesses a coloring scheme (in polynomial time) and then verifies if the coloring uses no more than B colors and if any two adjacent nodes have different colors (in polynomial time).

- Theorem: **P**$\subseteq$**NP**.

  Any deterministic TM is a special case of a nondeterministic TM.

- Theorem: Any $\Pi \in$**NP** can be solved by a deterministic TM in time $O(c^{p(n)})$ for some $c > 0$ and polynomial $p(n)$.

- Open problem: **NP**$\subseteq$**P**? or **P**=**NP**?

11.4 Polynomial reduction

Reading: Sipser 7.4

- Definition: Let Π_1 and Π_2 be two decision problems, and $\{I_1\}$ and $\{I_2\}$ be sets of instances for Π_1 and Π_2, respectively. We say there is a polynomial reduction from Π_1 to Π_2, or $\Pi_1 \propto_p \Pi_2$ if there is $f : \{I_1\} \rightarrow$ subset of $\{I_2\}$ such that (1) f can be computed in polynomial time and (2) I_1 has a "yes" solution if and only if $f(I_1)$ has a "yes" solution.

- Theorem: If $\Pi_1 \propto_p \Pi_2$, then $\Pi_2 \in$**P** implies $\Pi_1 \in$**P**.

- Theorem: If $\Pi_1 \propto_p \Pi_2$ and $\Pi_2 \propto_p \Pi_3$, then $\Pi_1 \propto_p \Pi_3$.

- *Remark:* $\propto_p$ means "no harder than".

11.5 The class NPC

Reading: Sipser 7.4

- Definition: **NPC** (**NP**-complete) is the class of the hardest problems in **NP**, or equivalently, $\Pi \in$**NPC** if $\Pi \in$**NP** and $\forall \Pi' \in$**NP**, $\Pi' \propto_p \Pi$.

- Theorem: If $\Pi_1 \in$**NPC**, $\Pi_2 \in$**NP**, and $\Pi_1 \propto_p \Pi_2$, then $\Pi_2 \in$**NPC**.

- Theorem: If $\exists \Pi \in$**NPC** such that $\Pi \in$**P**, then **P**=**NP**.

- Theorem: If $\exists \Pi \in$**NPC** such that $\Pi \notin$ **P**, then **P**$\neq$**NP**.

- Satisfiability (SAT):

  INSTANCE: A boolean formula α in *CNF* with variables $x_1, \ldots, x_n$ (e.g., $\alpha = (x_1 \vee x_2) \wedge (\neg x_2 \vee x_3)$).

  QUESTION: Is α satisfiable? (Is there a truth assignment to $x_1, \ldots, x_n$ such that α is true?)

  Cook's Theorem: SAT$\in$**NPC**. (Need to prove (1) SAT$\in$**NP** and (2) $\forall \Pi \in$**NP**, $\Pi \propto_p$SAT.)

- Proof of Cook's Theorem:

  SAT is clearly in **NP** since a NTM exists that guesses a truth assignment and verifies its correctness in polynomial time. Now we wish to prove $\forall \Pi \in$**NP**, $\Pi \propto_p$SAT, or equivalently, for any polynomial-time NTM M, $L(M) \propto_p L_{SAT}$.

  For any NTM M, assume $Q = \{q_0, q_1(\text{accept}), q_2(\text{reject}), \ldots, q_r\}$ and $\Gamma = \{s_0, s_1, s_2, \ldots, s_v\}$. Also assume that the time is bounded by $p(n)$, where n is the length of the input. We wish to prove that there is a function $f_M : \Sigma^* \rightarrow \{\text{instances of SAT}\}$ such that $\forall x \in \Sigma^*$, $x \in L(M)$ iff $f_M(x)$ is satisfiable. In other words, we wish to use a Boolean expression $f_M(x)$ to describe the computation of M on x.

  Variables in $f_M(x)$:

  — State: $Q[i,k]$. M is in q_k after the ith step of computation (at time i).

  — Head: $H[i,j]$. Head points to tape square j at time i.

  — Symbol: $S[i,j,l]$. Tape square j contains s_l at time i. (Assume the tape is one-way infinite and the leftmost square is labeled with 0.)

For example, initially $i = 0$. Assume the configuration is q_0abba. Let $s_0 = B$, $s_1 = a$, and $s_2 = b$. Therefore, we set the following Boolean variables to be true: $Q[0,0]$, $H[0,0]$, $S[0,0,1]$, $S[0,1,2]$, $S[0,2,2]$, $S[0,3,1]$ and $S[0,j,0]$ for $j = 4,5,\ldots$. A configuration defines a truth assignment, but not vice versa.

Clauses in $f_M(x)$:

— At any time i, M is in exactly one state.

$Q[i,0] \vee \cdots \vee Q[i,r]$ for $0 \leq i \leq p(n)$.

$\neg Q[i,k] \vee \neg Q[i,k']$ for $0 \leq i \leq p(n)$ and $0 \leq k < k' \leq r$.

— At any time i, head is scanning exactly one square.

$H[i,0] \vee \cdots \vee H[i,p(n)]$ for $0 \leq i \leq p(n)$.

$\neg H[i,j] \vee \neg H[i,j']$ for $0 \leq i \leq p(n)$ and $0 \leq j < j' \leq p(n)$.

— At any time i, each square contains exactly one symbol.

$S[i,j,0] \vee \cdots \vee S[i,j,v]$ for $0 \leq i \leq p(n)$ and $0 \leq j \leq p(n)$.

$\neg S[i,j,l] \vee \neg S[i,j,l']$ for $0 \leq i \leq p(n)$, $0 \leq j \leq p(n)$ and $0 \leq l < l' \leq v$.

— At time 0, M is in its initial configuration. Assume $x = s_{l_1} \cdots s_{l_n}$.

$Q[0,0]$.

$H[0,0]$.

$S[0,0,l_1],\ldots,S[0,n-1,l_n]$.

$S[0,j,0]$ for $n \leq j \leq p(n)$.

— By time $p(n)$, M has entered q_1 (accept)). (If M halts in less than $p(n)$ steps, additional moves can be included in the transition function.)

$Q[p(n),1]$.

— Configuration at time $i \rightarrow$ configuration at time $i+1$. Assume $\delta(q_k,s_l) = (q_{k'},s_{l'},D)$, where $D = -1,1$.

If the head does not point to square j, symbol on j is not changed from time i to time $i+1$.

$H[i,j] \vee \neg S[i,j,l] \vee S[i+1,j,l]$ for $0 \leq i \leq p(n)$, $0 \leq j \leq p(n)$, and $0 \leq l \leq v$.

If the current state is q_k, the head points to square j which contains symbol s_l, then changes are made accordingly.

$\neg H[i,j] \vee \neg Q[i,k] \vee \neg S[i,j,l] \vee H[i+1,j+D]$,

$\neg H[i,j] \vee \neg Q[i,k] \vee \neg S[i,j,l] \vee Q[i+1,k']$, and

$\neg H[i,j] \vee \neg Q[i,k] \vee \neg S[i,j,l] \vee S[i+1,j,l']$, for $0 \leq i \leq p(n)$, $0 \leq j \leq p(n)$, $0 \leq k \leq r$, and $0 \leq l \leq v$.

Let $f_M(x)$ be the conjunction of all the clauses defined above. Then $x \in L(M)$ iff there is an accepting computation of M on x iff $f_M(x)$ is satisfiable. f_M can be computed in polynomial time since $|f_M(x)| \leq$ (number of clauses) $*$(number of variables) $= O(p(n)^2) * O(p(n)^2) = O(p(n)^4)$.

So there is a polynomial reduction from any language in **NP** to SAT. So SAT is **NP**-complete.

11.6 NP-complete problems

Reading: Sipser 7.5

- How to prove Π is **NP**-complete:
  - Show that $\Pi \in$**NP**.
  - Choose a known **NP**-complete Π'.
  - Construct a reduction f from Π' to Π.
  - Prove that f is a polynomial reduction by showing that (1) f can be computed in polynomial time and (2) $\forall I'$ for Π', I' is a yes-instance for Π' if and only if $f(I')$ is a yes-instance for Π.

- Seven basic **NP**-complete problems.
  - 3-Satisfiability (3SAT):
    INSTANCE: A formula α in CNF with each clause having three literals.
    QUESTION: Is α satisfiable?
  - 3-Dimensional Matching (3DM):
    INSTANCE: $M \subseteq X \times Y \times Z$, where X, Y, Z are disjoint and of the same size.
    QUESTION: Does M contain a matching, which is $M' \subseteq M$ with $|M'| = |X|$ such that no two triples in M' agree in any coordinate?
  - PARTITION:
    INSTANCE: A finite set A of numbers.
    QUESTION: Is there $A' \subseteq A$ such that $\sum_{a \in A'} a = \sum_{a \in A - A'} a$?
  - Vertex Cover (VC):
    INSTANCE: A graph $G = (V, E)$ and $0 \leq k \leq |V|$.
    QUESTION: Is there a vertex cover of size $\leq k$, where a vertex cover is $V' \subseteq V$ such that $\forall (u, v) \in E$, either $u \in V'$ or $v \in V'$?
  - Hamiltonian Circuit (HC):
    INSTANCE: A graph $G = (V, E)$.
    QUESTION: Does G have a Hamiltonian circuit, i.e., a tour that passes through each vertex exactly once?
  - CLIQUE:
    INSTANCE: A graph $G = (V, E)$ and $0 \leq k \leq |V|$.
    QUESTION: Does G contain a clique (complete subgraph) of size $\geq k$?

- A proof that 3SAT is NP-complete:

  First, 3SAT is obvious in **NP**.

  Next, we show that SAT$\propto_p$3SAT.

  Given any instance of SAT, $f(x_1, \ldots, x_n) = c_1 \wedge \cdots \wedge c_m$, where c_i is a disjunction of literals. To construct an instance for 3SAT, we need to convert any c_i to an equivalent c_i', a conjunction of clauses with exactly 3 literals.

  Case 1. If $c_i = z_1$ (one literal), define y_i^1 and y_i^2. Let $c_i' = (z_1 \vee y_i^1 \vee y_i^2) \wedge (z_1 \vee y_i^1 \vee \neg y_i^2) \wedge (z_1 \vee \neg y_i^1 \vee y_i^2) \wedge (z_1 \vee \neg y_i^1 \vee \neg y_i^2)$.

  Case 2. If $c_i = z_1 \vee z_2$ (two literals), define y_i^1. Let $c_i' = (z_1 \vee z_2 \vee y_i^1) \wedge (z_1 \vee z_2 \vee \neg y_i^1)$.

  Case 3. If $c_i = z_1 \vee z_2 \vee z_3$ (three literals), let $c_i' = c_i$.

  Case 4. If $c_i = z_1 \vee z_2 \vee \cdots \vee z_k$ ($k > 3$), define $y_i^1, y_i^2, \ldots, y_i^{k-3}$. Let $c_i' = (z_1 \vee z_2 \vee y_i^1) \wedge (\neg y_i^1 \vee z_3 \vee y_i^2) \wedge (\neg y_i^2 \vee z_4 \vee y_i^3) \wedge \cdots \wedge (\neg y_i^{k-3} \vee z_{k-1} \vee z_k)$.

  If c_i is satisfiable, then there is a literal $z_l = T$ in c_i. If $l = 1, 2$, let $y_i^1, \ldots, y_i^{k-3} = F$. If $l = k-1, k$, let $y_i^1, \ldots, y_i^{k-3} = T$. If $3 \leq l \leq k-2$, let $y_i^1, \ldots, y_i^{l-2} = T$ and $y_i^{l-1}, \ldots, y_i^{k-3} = F$. So c_i' is satisfiable.

  If c_i' is satisfiable, assume $z_l = F$ for all $l = 1, \ldots, k$. Then $y_i^1, \ldots, y_i^{k-3} = T$. So the last clause $(\neg y_i^{k-3} \vee z_{k-1} \vee z_k) = F$. Therefore, c_i' is not satisfiable. Contradiction.

  The instance of 3SAT is therefore $f'(x_1, \ldots, x_n, \ldots) = c_1' \wedge \cdots \wedge c_m'$, and f is satisfiable if and only if f' is satisfiable.

- A proof that CLIQUE is **NP**-complete.

  INSTANCE: $G = (V, E)$ and $k \leq |V|$.

  QUESTION: Does G contain a clique (a complete subgraph) of size k or more?

  Proof. CLIQUE is obviously in **NP**. Now we show that SAT$\propto_p$CLIQUE. For any Boolean expression $f(x_1, \ldots, x_n) = c_1 \wedge \cdots \wedge c_m$, we define a graph G, where $V = \{< y, i > | y \text{ is a literal in } c_i\}$ and $E = \{< y, i >< z, j > | i \neq$

j and $y \neq \neg z\}$. Finally, let $k = m$. Since $|V| = O(mn)$ and $|E| = O(m^2n^2)$, the graph can be constructed in polynomial time. Next we show that f is satisfiable iff there is a clique in G with size m.

Assume there is a truth assignment for f such that at least one literal, say l_i, in c_i ($i = 1, \ldots, m$) is true. Consider a subgraph of G consisting of m nodes $< l_i, i >$. For any $i \neq j$, we have $l_i \neq \neg l_j$. So edge $< l_i, i >< l_j, j >$ is in G. Therefore the subgraph is a clique of size m.

Assume G contains a clique of size m or more. Let V' be the set of any m nodes in the clique. Since no edge in G connects $< y, i >$ and $< z, j >$ when $i = j$, and nodes in V' are fully connected, then V' contains $< l_1, 1 >, \ldots, < l_m, m >$, where l_i is a literal in clause c_i. Further $l_i \neq \neg l_j$ for any $i \neq j$. Then there is a truth assignment such that $l_i = true$ for $i = 1, \ldots, m$. Obviously, $f(x_1, \ldots, x_n) = c_1 \wedge \cdots \wedge c_m$ is satisfiable under the truth assignment.