Chap. 10 Intractable Problems

Efficient vs inefficient
polynomial vs exponential
tolerable vs intolerable amount of time

Cook's Theorem

    satisfiability of boolean formula

        *can **not** be decided in polynomial time*

    Reduce this problem to other problems

        ***polynomial** time reduction*

Assumption $P \neq NP$ *controvertible*

    Nondeterministic polynomial time

        $^?$No deterministic polynomial time

10.1 The Classes P and NP

A TM(algorithm) M is said to be of ***time complexity*** *T(n)*

running time *T(n)*

if M with input w of length n, M halts after ***at most*** *T(n) moves.*

A problem P is in P, if there exist an ***deterministic*** *algorithm(program) of* ***polynomial*** *time complexity.*

Is there a ***path*** *from s to t in a directed graph G*

```
1. place a mark on node s
2. Repeat 3 until no additional nodes are marked
3.     for edge (a, b) in G
               if a is marked and b is unmarked then mark b
4. If t marked accept, otherwise reject.
       At most n(number of nodes) marks.
path problem is P.
```

A problem P is in NP, if there exist an ***nondeterministic*** *algorithm*
of ***polynomial*** *time complexity*

$P \subseteq NP$

$P = NP$ or $P \neq NP (P \subset NP)$

${}^{?\exists}P$.϶. $P \in NP$, $P \notin P$.

Is there a ***hamiltonian path****(circuit in this text)*

Travelling Salesman Problem: An NP Example

Verify ***all*** *paths, if it is* ***hamiltonian****.*

at most n! paths

Exponential

Verify n! paths in ***parallel***

O(n) in parallel, NP

Polynomial time reduction*(PTR)*

Reduce *all instances of P_1 to P_2 in* ***polynomial time*** *($P_1 \leq_P P_2$)*

If P_2 is P, then P_1 is P.

If P_1 is not P, then P_2 is not P

*P is NP-****complete problems****, if*

1. P is NP.

*2. For $^\forall P' \in NP$, $^\exists$****polynomial time reduction*** *of P' to P.($P' \leq_P P$).*

*NP-****complete*** *is the hardest problems among NP.*

Theorem 10.4 *If P_1 is NP-complete, and $P_1 \leq_P P_2$,*

then P_2 is NP-complete.

proof *Since P_1 is NP-complete, $^\forall P' \in NP$, $P' \leq_P P_1$., and $P_1 \leq_P P_2$.*

$\therefore$ $^\forall P' \in NP$, $P' \leq_P P_2$. $\therefore$ P_2 is NP-complete.

Theorem 10.5 *If* $^{\exists}P \in$ NP-***complete*** *and* $P \in$ P, *then* P = NP.

proof *Since* $^{\forall}P' \in$ NP, $P' \leq_P P$. NP $\subseteq$ P, *and* P = NP.

We don't resolve that P = NP *neither* P ≠ NP *but*

NP-***complete*** *problems are the* ***hardest*** *ones among* NP *to be* P.

If $P \in$ NP-***complete*** *and* P *is proven to be* P, *then* P = NP.

Otherwise we don't know.(Now!!)

P *is* NP-***hard***, *if*

2. *For* $^{\forall}P' \in$ NP, $P' \leq_P P$.

We can say P = NP, *if we found* $P \in$ NP-***hard*** *is* P.

Furthermore $P \in$ NP-***hard*** *is at least as hard as* $P' \in$ NP-***complete***.

10.2 An NP-complete Problem

$e \rightarrow e \vee e \mid e \wedge e \mid \neg e \mid v \mid T \mid F$

the value for variables(v) are either T or F.

Let E be a boolean expression.

***truth assignment** of E, denoted T,*

***assigns** either T or F for variable in E*

$x_1, \ldots, x_n$ *variable* 2^n *assignments*

E(T) the result of the true assignment T

*E is **satisfiable**, if* $^{\exists}$*truth assignment T such that E(T) = T.*

*The **satisfiability**(SAT) **problem***

*Given a boolean expression, is it **satisfiable**?*

*SAT is the **first** NP-complete problem(Cook's theorem)*

i) SAT ∈ NP, ii)$^{\forall}P \in NP$*,* $P \leq_P$ *SAT, ∴ SAT ∈ NP-**complete** (first).*

If $^{\exists}P \in NP$*, SAT* $\leq_P P$ *then P ∈ NP-**complete**.(Thm 10.5)*

Theorem 10.9 *(Cook's Theorem) SAT is NP-**complete**.*

proof *1) SAT is NP.*

It is trivial. 2^n assignments, we can determine the result of each assignment in polynomial time in NTM. $\therefore$ SAT $\in$ NP.

*2) For $^{\forall}P \in NP$, $^{\exists}$**polynomial time reduction** of P to SAT.($P' \leq_P P$)*

Since $P \in NP$, we assume polynomial p(n) moves in NTM.

$$\alpha_0 \Rightarrow \alpha_1 \Rightarrow^* \ldots \Rightarrow \alpha_{p(n)}.$$

where $\alpha_i \in \Gamma^* \times Q \times \Gamma^*$ and $|\alpha_i| \leq p(n)$ for $0 \leq {}^{\forall}i \leq p(n)$.

$\therefore$ We write $\alpha_i = X_{i0}\, X_{i1} \ldots X_{i\, p(n)}$.

a) Assume X_{ij} two-dimensional array($0 \leq {}^{\forall}i, {}^{\forall}j \leq p(n)$)

where X_{ij} is the symbol (in $Q \cup \Gamma$) for j-th position of i-th ID.

$(p(n)+1)^2$ cells see fig 10.4(p443)

*b) Consider a **boolean** variable y_{ijA} to denote the **proposition** that X_{ij} =A.*

c) Given $M \in TM$ and $w \in \Sigma^*$, consider a ***boolean expression****,*

$$E_{M,\,w} = U \wedge S \wedge N \wedge F'.$$

1. Unique

$\wedge_{i,j} \neg(y_{ijA} \wedge y_{ijB})$ *where* $A \neq B \in Q \cup \Gamma$

2. Starts right: S ***initial*** *configuration*

Assume $w = a_1 \ldots a_n$.

$S = y_{00q_0} \wedge y_{01a_1} \wedge y_{02a_2} \wedge \cdots \wedge y_{0na_n} \wedge y_{0,n+1,B} \wedge \cdots \wedge y_{0,p(n),B}.$

4. Finishes right: F' ***final*** *configuration*

Assume final state in $\alpha_{p(n)}$ *and* $F = \{f_0, \ldots, f_k\}$.

Repeat accepting configuration until p(n)

$F' = F_0 \vee \ldots \vee F_{p(n)}$ *where* F_j *means* $X_{p(n),j} \in F$.

$0 \leq \forall j \leq p(n),\ F_j = y_{p(n),j,f_0} \vee \ldots \vee y_{p(n),j,f_k}.$

3. Next move is right: N ***legal moves*** *in TM*

$N = N_0 \wedge \ldots \wedge N_{p(n)-1}$ *where* N_i *assures that* $\alpha_i \Rightarrow \alpha_{i+1}$ *and*

$0 \leq {}^{\forall} i \leq p(n) - 1$, $N_i = (A_{i0} \vee B_{i0}) \wedge \ldots \wedge (A_{i\,p(n)} \vee B_{i\,p(n)})$.

$0 \leq {}^{\forall} j \leq p(n)$, *two cases (a)* $X_{i,j} \in Q(A_{ij})$ *or (b)* $X_{i,j} \notin Q(\in \Gamma)$ (B_{ij}).

Window: 3 cells

(a) Assume $X_{i,j-1}X_{i,j}X_{i,j+1} = DqA$ *where* $q \in Q$, $D, A \in \Gamma$.

1) Move left: If $\delta(q, \underline{A}) = (\underline{p}, \underline{C}, \mathbf{R})$ *then* $X_{i+1,j-1}X_{i+1,j}X_{i+1,j+1} = DCp$.

$\ldots Dq\underline{A}\ldots \Rightarrow \ldots D\underline{C}\underline{p}\ldots$

$A_{ij} = y_{i,j-1,D} \wedge y_{i,j,q} \wedge y_{i,j+1,A} \wedge y_{i+1,j-1,D} \wedge y_{i+1,j,C} \wedge y_{i+1,j+1,p}$.

2) Move right: If $\delta(q, \underline{A}) = (\underline{p}, \underline{C}, \mathbf{L})$ *then* $X_{i+1,j-1}X_{i+1,j}X_{i+1,j+1} = pDC$.

$\ldots Dq\underline{A}\ldots \Rightarrow \ldots \underline{p}D\underline{C}\ldots$

$A_{ij} = y_{i,j-1,D} \wedge y_{i,j,q} \wedge y_{i,j+1,A} \wedge y_{i+1,j-1,p} \wedge y_{i+1,j,D} \wedge y_{i+1,j+1,C}$.

(b) Assume $Q = \{q_1, \ldots, q_m\}$ *and* $\Gamma = \{Z_1, \ldots, Z_r\}$

State is at left(i-1) or at right(i+1) or not in the window.

$$B_{ij} = (X_{i,j-1} \in Q) \vee (X_{i,j+1} \in Q) \vee (X_{i,j} \in \Gamma \wedge X_{i,j} = X_{i+1,j})$$

$$\begin{array}{ll}
= (y_{i,j-1,q_1} \vee \ldots \vee y_{i,j-1,q_m}) & X_{i,j-1} \in Q, \\
\vee (y_{i,j+1,q_1} \vee \ldots \vee y_{i,j+1,q_m}) & X_{i,j+1} \in Q, \\
\vee ((y_{i,j,Z_1} \vee \ldots \vee y_{i,j,Z_r}) \wedge & X_{i,j} \in \Gamma, \\
\quad ((y_{i,j,Z_1} \wedge y_{i+1,j,Z_1}) \vee \ldots \vee (y_{i,j,Z_r} \wedge y_{i+1,j,Z_r}))) & X_{i,j} = X_{i+1,j}.
\end{array}$$

$N = N_0 \wedge \ldots \wedge N_{p(n)-1}$

A_{ij} *and* B_{ij} *are large but independent of n, the length w.*

The length of N_i *is* $O(p(n))$ *and the length of N is* $O(p^2(n))$.

Conclusion of Cook's theorem

We can reduce ***any*** *problem in NP to SAT in* ***polynomial*** *time.*

If $P \in NP$ *and* $SAT \leq_P P$*, then P is also an NP-**complete** problem.*

Let me finish my lecture CS322 at here!
Details will be covered in CS300 algorithms

My conclusion is

NP ***complete*** *problems are important in practice.*

NP alogrithms are nature of thinking or programming

E. Dijkstra

Computables are more important than decidables!

Exponetials are ***not intractable****, but they are* ***tractable****!*

Example: AlphaGo

Opposite to Cook!

See "NP complete 는 수학인가 *?"*

Turing-Church's thesis!

Turing machine and partial-μ-recursive functions are ***computable****!*

최 and Other's thesis

NP-complete is ***exponetial*** *and* ***tractable****.*

10.3 Restricted Satisfiability Problem
10.3.1 Conjunctive normal form

literal	*a variable or negated variable*	$x, \neg y = \overline{y}$
clause	*OR(**conjunction**) of the literals*	$x \vee \overline{y} \vee z$

*an boolean expression is **conjunctive normal form***
*if it is the AND(**disjunction**) of **clauses***

*An expression is k-**conjunctive normal form**(k-CNF), if **every** clause has **exactly** k distinct **variables**.*

$(x+\overline{y})(x+y+\overline{z})$	*CNF*
$(x+\overline{y})(y+\overline{z})(z+\overline{x})$	*2-CNF*
$(x+\overline{y}+z)(x+y+\overline{z})$	*3-CNF*

*CSAT: **satisfiability** problem of CNF*
*kSAT: **satisfiability** problem of k-CNF*

10.3.2 Converting Expression to CNF

*1. Push all the ¬ in the boolean expression down to the variable(**literal**)*

$\neg(E \wedge F) = \neg E \vee \neg F$

$\neg(E \vee F) = \neg E \wedge \neg F$

$\neg(\neg(E)) = E$

Every literal has at most 1 negation.

*2. Write an CNF by introducing **new variables** and **its complements**.*

*New expression F is **not equivalent** to the old one E, but*

*F is satisfiable **if and only if** E is.*

*3. S is a **extension** of T, if*

1) S assigns the same value as T for the old variables

2) S may assign a value to new variables that T does not mention.

A truth assignment T for E is true, if and only if,

*the **extension** S of T for F is true.*

Theorem 10.12 *Every boolean expression E is equivalent to an expression F in CNF. Moreover the length of F is linear in number of symbols of E, F can be constructed in polynomial time.*

Proof *Induction on number of operators($\wedge$, $\vee$, $\neg$)*

basis *If E has one operator($x \wedge y$, $x \vee y$, $\neg x$), it is trivial.*

induction

1) $E = E_1 \wedge E_2$, $E = E_1 \vee E_2$.

2) $E = \neg E_1$.

2.1) $E = \neg(\neg(E_2)) = E_2$.

2.2) $E = \neg(E_2 \vee E_3) = \neg(E_2) \wedge \neg(E_3)$

2.3) $E = \neg(E_2 \wedge E_3) = \neg(E_2) \vee \neg(E_3)$

10.3.3 NP-Completeness of CSAT

Theorem 10.13 *CSAT is NP-complete.*

Proof *Reduce SAT to CSAT*

Assume E is a boolean expression of length n. Then

a) F is CNF of at most n clauses.

b) F is constructable from E in time at most $c|E|^2$.

c) A truth assignment T is true for E

iff $\exists$ *extension S of T that makes F true.*

Basis *If E consists of one or two symbols, E is CNF.*

Induction *Two cases*

Case 1: $E = E_1 \wedge E_2$.

(If)

(Only if)

Case 2: $E = E_1 \vee E_2$.

Assume $F_1 = g_1 \wedge g_2 \wedge \ldots \wedge g_p$ *and* $F_2 = h_1 \wedge h_2 \wedge \ldots \wedge h_q$.

$F = (y+g_1) \wedge (y+g_2) \wedge \ldots \wedge (y+g_p) \wedge (\overline{y}+h_1) \wedge (\overline{y}+h_2) \wedge \ldots \wedge (\overline{y}+h_q).$

A truth assignment T for E satisfies E, if and only if,

T can be extended to a truth assignment S for F that satisfies F.

(If)

(Only if) Assume the extension S safisfies F.

Example 10.4

$E = x\overline{y} + \overline{x}(y+z)$

$F \Rightarrow x\overline{y} + \overline{x}(v+y)(\overline{v}+z)$ *introducing v*

$\Rightarrow (u+x)(u+\overline{y})(\overline{u}+\overline{x})(\overline{u}+v+y)(\overline{u}+\overline{v}+z)$ *introducing u*

$T(x) = 0, T(y) = 1,$ *and* $T(z) = 1,$

Extend $S(u) = 1, S(v) = 0$ *or* $S(v) = 1.$

*10.3.4 NP-**Completeness of 3SAT***

Theorem 10.14 *3SAT is NP-complete.*

Proof *Reducing CSAT to 3SAT*

Assume CNF $E = e_1 \wedge e_2 \wedge \dots \wedge e_k$.

(1) $e_i = x \Rightarrow \quad (x+u+v)(x+\overline{u}+v)(x+u+\overline{v})(x+\overline{u}+\overline{v})$

(2) $e_i = x+y \Rightarrow \quad (x+y+z)(x+y+\overline{z})$

(3) $e_i = x+y+z \quad$ 3-CNF

(4) $e_i = x_1 + \dots + x_m (m \geq 4)$ introduce $y_1, \dots, y_{m-3}$ variables

$$\Rightarrow (x_1 + x_2 + y_1)(x_3 + \overline{y}_1 + y_2)(x_4 + \overline{y}_2 + y_3) \dots (x_j + \overline{y}_{j-2} + y_{j-1}) \dots$$
$$(x_{m-2} + \overline{y}_{m-4} + y_{m-3})(x_{m-1} + x_m + \overline{y}_{m-3})$$

A truth assignment T of E must make at least one literal of e_i.

If x_j is true, we make $y_1, \dots, y_{j-1}$ are **false** and $y_j, \dots, y_{m-3}$ are **true**.

If T makes all x's false,

10.4 Additional NP-Completeness Problems

10.4.2 The problem of independent set(IS).

***Def**. Let $G=(V, E)$ be a undirected graph.*

$I = \{a, b \in V \mid (a, b) \notin E\}$ ***independent** set*

***Def**. An independent I set is **maximal**, if $|I| \geq |J|$, J is independent.*

***Theorem 10.18** IS is NP-complete.*

***Proof** IS $\in$ NP. guess k nodes and check they are independent.*

***Reducing** 3SAT to IS.*

Let $E = e_1 e_2 \dots e_m$

$= (x_{11}+x_{12}+x_{13})(x_{21}+x_{22}+x_{23}) \dots (x_{m1}+x_{m2}+x_{m3})$ *be a 3-CNF*

Construct a graph $G = (V, F)$

$V = \{[i, j] \mid 1 \leq i \leq m, j = 1, 2, 3\}$

$F = \{([i, 1], [i, 2]), ([i, 2], [i, 3]), ([i, 3], [i, 1]) \mid 1 \leq i \leq m\}$

$\cup \{([i, j], [k, l]) \mid x_{ij} = x, x_{kl} = \overline{x}\}$

E is satisfiable if and only if G has an independent set of size m.

(If) If [i, j], [k, l] $\in$ I, $i \neq k$.

([i, 1], [i, 2]), ([i, 2], [i, 3]), ([i, 3], [i, 1]) $\in$ E.

$\therefore$ *independent set of size m, exactly one node from each literal.*

If [i, j] $\in$ I and $x_{ij} = x$, then T(x) = 1;

$x_{ij} = \overline{x}$, then T(x) = 0. No contradiction!

If [i, j] $\notin$ I, then pick T(x) arbitrary.

$\therefore$ *E is satisfiable.*

(Only if) Assume E is satisfiable by some truth assignment T.

$I = \{[i, j] \mid T(x_{ij}) = 1\}$

If $|I| > m$, $^{\exists}$[i, j], [i, j'] $\in$ I, then remove [i, j'] from I.

Then $|I| = m$.

I is the independent set.

10.4.3 The node-cover problem

Let $G = (V, E)$ be a graph.

Edge cover set EC of a graph G

$$V = \{b \mid (a, b) \in EC\}$$

Minimal edge cover

Node cover set NC of a graph G

$$E = \{(a, b) \mid b \in NC\}$$

Minimal node cover

Minimal node cover set is the **complement** *of maximal independent set.*

$$NC = \neg I.$$

Theorem 10.20 *IS is NP-complete.*
Proof Reducing *IS to NC*
Let G =(N, E) be a graph with n-vertices(|N|=n).
G has an independent set of size k, if and only if,
G has a a node cover of size n - k.

(If) Let C be a node cover set of size n - k.
If v, w ∈ N - C, then (v, w) ∈ E.
Since v, w ∉ C, (v, w) ∈ E is not covered by the node cover C.
∴ N - C is an independent set of size k.

(Only if) Let I be an independent set of size k.
We claim N - I is a node cover by contradiction.
If ∃(v, w) ∈ E, not covered by N - I, but v, w ∈ I.
∴ I is a independent set by contradiction.