NP

Satisfiability or SAT

Input: a formula ϕ over variables $x_1, x_2, \dots x_n$

Solution: find a satisfying assignment or output no satisfying assignment exists.

Example: $\phi = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x_2}) \wedge (x_2 \vee \bar{x_3}) \wedge (x_3 \vee \bar{x_1})$

try all possible assignments $\rightarrow 2^n$

VERIFY (Instance ϕ , Solution $F: \{x_1 \dots x_n\} \rightarrow \{T/F\}$) $\rightarrow$ ① $a = \phi(F(x_i))$ … ② Output a

P & NP

P: Class of search problems where we can find a solution in polynomial in $|I|$ time.

HornSAT $\in$ P. & 2SAT $\in$ P

NP: Class of all search problems, i.e. the class of all problems where we can verify a solution in polynomial time. SAT $\in$ NP. 2SAT $\in$ NP. P $\subseteq$ NP

Graph 3-Coloring $\in$ NP

Input $G=(V,E)$

Solution: Coloring function $c: V \rightarrow \{R, G, B\}$ s.t. $\forall (u,v) \in E$ $c(u) \neq c(v)$ or output no coloring function exists.

Proof. VERIFY (Instance $G=(V,E)$, Solution $c: V \rightarrow \{R,G,B\}$) ① output if $\forall (u,v) \in E$ $c(u) \neq c(v)$ & $\forall v$, $c(v) \in \{R,G,B\}$ ② else output 0.

Vertex Cover (VC) $\in$ NP

Input: $G=(V,E)$, b

Solution: $A \subseteq V$, s.t. $|A| \leq b$, s.t. $\forall (u,v) \in V$, either $u \in A$ or $v \in A$ or output no solution

proof. VERIFY (Instance $G=(V,E), b$, Solution Set A) ① output 0 if $|A| > b$ ② output 0 if $\exists (u,v) \in E$ s.t. $u \notin A$ & $v \notin A$ ③ output 1 otherwise

Factoring $\in$ NP

Input $N = p \cdot q$ Solution p, q

proof. VERIFY (Instance N, Solution p, q) Output 1 if $N = pq$ & 0 otherwise

TSP $\in$ NP

Instance: n vertices and distances d_{ij} between them & a bound B.

Solution: a permutation $\tau: \{1 \dots n\} \rightarrow \{1 \dots n\}$ s.t. $d_{\tau(1)\tau(2)} + d_{\tau(2)\tau(3)} + \dots + d_{\tau(n)\tau(1)} \leq B$ or output no solution exists

proof: VERIFY (Instance $n, \{d_{ij}\}, B$, Solution $\tau: \{1 \dots n\} \rightarrow \{1 \dots n\}$) output 1 if ① $\sum d_{\tau(i)\tau(i+1)} \leq B$ ② $\forall i, j$ $\tau(i) \neq \tau(j)$ output 0 otherwise

RC/HC $\in$ NP

Instance: $G=(V,E)$

Solution: permutation $\tau: \{1 \dots n\} \rightarrow V$ ($|V|=n$) s.t. $(\tau(\tau_1), \tau(\tau_2)), \dots (\tau(\tau_n), \tau(\tau_1)) \in E$. or output no solution exists

proof. VERIFY (Instance $G=(V,E)$, $\tau: \{1 \dots n\} \rightarrow V$) output 1 if (i) $\forall i, j$ $\tau(i) \neq \tau(j)$ and (ii) $(\tau(\tau_1), \tau(\tau_2)) \dots (\tau(\tau_n), \tau(\tau_1)) \in E$. and 0 otherwise

Composition of Reductions

Lemma: If $A \rightarrow B$ & $B \rightarrow C$, then $A \rightarrow C$.

proof: $f_{AC}(I) = f_{BC}(f_{AB}(I))$

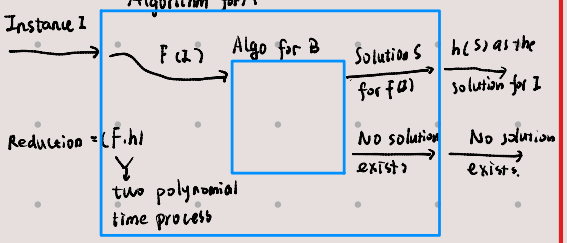
$h_{AC}(S) = h_{AB}(h_{BC}(S))$.

Reductions $A \rightarrow_p B$ $A \leq_p B$

A reduces to B in polynomial time.

An algorithm for B yields an algorithm for A.

B is at least as hard as A.



1) Running time: F & h run in time polynomial in $|I|$.

2) If B outputs S as a solution to f(I) (under problem B) then, h(S) is a solution to I (under prob A)

3) If B outputs "no solution exists" then no solution to I exists either. (If I has a solution, then f(I) also has a solution)

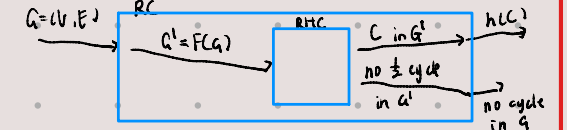
Rudrata Cycle $\rightarrow$ Rudrata Half Cycle

Input: $G=(V,E)$

Solution: cycle visiting each vertex exactly once

Input: $G'=(V',E')$, $|V'| = \frac{|V|}{2}$

Solution: cycle visiting of the vertice exactly once



$f(G): G'=(V',E') \rightarrow E'=E$, $V' = V \cup \{n+1, \dots 2n\}$

proof: ① f & h are polynomial time in $|I|$

② If C is a RHC in G', then h(C) is also RC in G

i) C doesn't contain $n+1 \dots 2n$ (=C)

ii) Number of vertices in C is $n = |V|$. $(|V'| = 2|V|)$

$\Rightarrow$ C contains all of the vertices in $1 \dots n$

$\Rightarrow$ C is a RC in G

③ If G has a RC, then G' has a HRC.

Let C be the RC in G. then (C has $|V|$ vertices)

C is also the HRC in G'.

SAT $\rightarrow$ 3SAT

Input: Formula ϕ ({T/F}) (each clause has $\neq$ vars)

Solution: Assignment $S: \{x_1 \dots x_n\} \rightarrow \{T/F\}$

Input: Formula ϕ

Solution: Assignment $S: \{x_1 \dots x_N\}$, $w = f(\phi)$ $\{T/F\}$

$(a_1 \vee a_2 \cdots a_k) \quad (a_1 \vee a_2 \vee y_1) \wedge (\bar{y_1} \vee a_3 \vee y_2) \dots (\bar{y_{k-3}} \vee a_{k-1} \vee a_k)$

$f: y_1 \dots y_{k-3}$ $h(S)$: recover a solution to ϕ

Lemma: if $w = f(\phi)$ was a satisfying assignment S then $h(S)$ is a satisfying assignment for ϕ ($\exists i$ s.t. $a_i = T$)

Lemma: if ϕ has a satisfying assignment then w also has a satisfying assignment. w be satisfied. Say $a_i = T$. $y_1 \dots y_{i-2}$ to be True and rest to be False

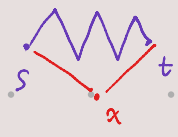
(s,t) Rudrata Path $\rightarrow$ Rudrata Path

Input: $G=(V,E)$, s, t.

Solution: A path starting at s and ending at t that visit each vertex

Input: $G=(V,E)$

Solution: a cycle that visits each vertex exactly once.



$F(G, s, t) = G'$

$G' = (V', E')$ $V' = V \cup \{x\}$

$E' = E \cup \{(x,s), (x,t)\}$

$h(C) = C - \{(x,s), (x,t)\}$

Proof: ① Runtime of F and h

② If S is a RC in G' then h(S) is a RP-(s,t) in G

③ If G has a (s,t)-RP. then G' has a RC.

Circuit SAT $\in$ NP

Instance: a boolean circuit C

Solution: an assignment to unknown input gates s.t. output gate outputs True.

A DAG with 5 kinds of gates

1) AND & OR gates of indegree 2

2) NOT gate of indegree 1.

3) know input gates

4) unknown input gates

Circuit SAT $\rightarrow$ SAT

Instance: a boolean circuit C

Solution: an assignment to unknown input gates s.t. output gate outputs True.

Instance: Formula ϕ

Solution: Assignment $S: \{x_1 \dots x_N\} \rightarrow \{T/F\}$

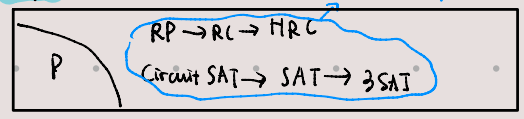
$f(C) \rightarrow \phi$: ∀ gate in the circuit C we will introduce a variable g

true gate (g) false gate $(\bar{g})$

proof: ① polynomial time ② h(S) = S | the unknown input gates

③ if C has a solution, then so does ϕ

So far



NP-Completeness

A search problem is NP-Complete if all other search problems reduce to it.

Lemma: $\forall A \in NP$. $A \rightarrow$ Circuit SAT $\rightarrow$ SAT $\rightarrow$ 3SAT

proof: $\text{VERIFY}_A(I_A, S_A) = 0/1 \rightarrow$ polytime algo in $|I_A|$.

$C_{\text{VERIFY}_A, I_A}(W) = \text{VERIFY}_A(I_A, W)$

$f(I_A) = C_{\text{VERIFY}_A, I_A}$. $h(S) = S$

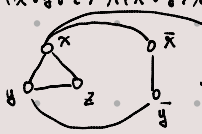
① f & h polytime ② S is a solution to circuit SAT then S is also a solution to I_A. ③ If A has a solution, so does C_{VERIFY_A, I_A}.

3SAT $\rightarrow$ Independent Set

Instance: Formula ϕ on variable $x_1 \dots x_n$

Solution: A satisfying assignment

ϕ: $(x \vee y \vee z) \wedge (\bar{x} \vee \bar{y}) \wedge \dots$



Instance: $G=(V,E)$, g

Solution: $S \subseteq V$. s.t. $|S| = g$ & $\forall u, v \in S$. $(u,v) \notin E$

g = # of clause.

proof: ① polytime

② 1 in G of size g then we can construct a satisfying assignment for ϕ (1 chosen ...)

③ if there is a satisfying assignment for $\phi \rightarrow$ an IS in G

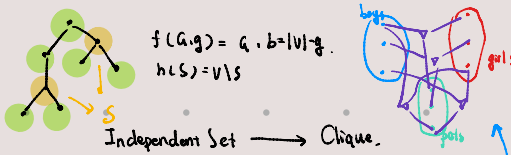
Independent Set $\rightarrow$ Vertex Cover

Instance: $G=(V,E)$, g

Solution: $S \subseteq V$. s.t. $|S| = g$ & $\forall (u,v) \in E$. $u \in S$. $(u,v) \notin E$

Instance: $G=(V,E)$, b

Solution: $S \subseteq V$. s.t. $|S| = b$ & $\forall (u,v) \in E$. either $u \in S$ or $v \in S$



$f(G,g) = G, b = |V| - g$.

$h(S) = V \setminus S$

Independent Set $\rightarrow$ Clique.

Instance: $G=(V,E)$, g

Solution: $S \subseteq V$. s.t. $|S| = g$ & $\forall u, v \in S$. $(u,v) \in E$

Instance: $G=(V,E)$, g.

Solution: $S \subseteq V$. s.t. $|S| = g$ & $\forall u, v \in S$. $(u,v) \in E$

$f(G=(V,E), g) = (G'=(V,E'), g)$ where $E' = V \times V \setminus E$

3D Matching

Input: n boys, girls and pets with preference tuples $\{(b,g,p)\}$

Solution: n- disjoint tuples.

3SAT $\rightarrow$ 3D Matching

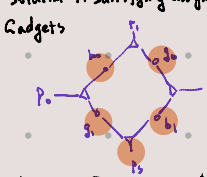
Instance: Formula ϕ on variabe $x_1 \dots x_n$

Solution: A satisfying assignment

Input: n boys, girls and pets with preference tuples $\{(b,g,p)\}$

Solution: n disjoint tuples.

Gadgets



false branch: (b_0, p_1, g_0) & (g_1, p_1, b_1) p_1 & p_2 free

true branch: (p_0, b_0, g_1) & (g_0, p_0, b_1) p_1 & p_2 free

$\forall c = (x \vee \bar{y} \vee z)$. create b_c, g_c.

$\Rightarrow$ x = true or y = false or z = true.

(b_c, g_c, p_{x1}) $(b_c, g_c, p_{\bar{y}2})$ (b_c, p_c, p_{z3})

or (b_c, g_c, p_{x2}) or $(b_c, g_c, p_{\bar{y}3})$ or (b_c, g_c, p_{z3})

∀ c, exists a pet to match

Zero-One Equations

Instance: $A \in \{0,1\}^{m \times n}$

Solution: $x \in \{0,1\}^n$ s.t. $Ax = 1$

$$\left[\begin{array}{c} A \\ \hline \end{array}\right]_{m \times n} \left[x\right]_{n \times 1} = \left[\begin{array}{c} 1 \\ \hline \end{array}\right]_{m \times 1}$$

3D Matching $\rightarrow$ ZOE

Input: t- boys, girls, pets with preference tuples $\{(b,g,p)\}$

Solution: t disjoint tuples

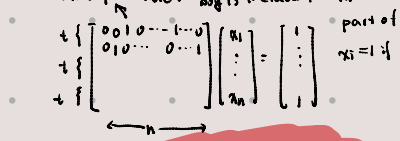
all tuple where 1st boy is included.

Instance: $A \in \{0,1\}^{m \times n}$

Solution: $x \in \{0,1\}^n$ s.t. $Ax = 1$

$x_i = 0$ if T_i is not part of the 3DM solution

$x_i = 1$ if T_i is part of the solution.



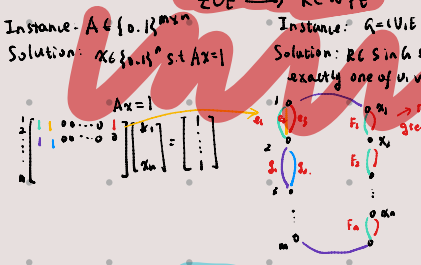
ZOE $\rightarrow$ RC w PE

Instance: $A \in \{0,1\}^{m \times n}$

Solution: $x \in \{0,1\}^n$ s.t. $Ax = 1$

Instance: $G=(V,E)$, $C \in E \times E$

Solution: RC S in G s.t. $\forall (e,e') \in C$ exactly one of v_i in S



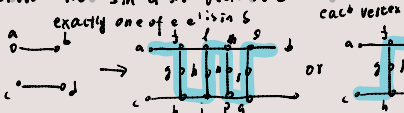
RC w PE $\rightarrow$ RC

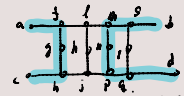
Instance: $G=(V,E)$, $C \in E \times E$

Solution: RC S in G s.t. $\forall (e,e') \in C$ exactly one of e, e' is in S

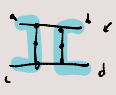
Instance: $G=(V,E)$

Solution: Cycle S visiting each vertex exactly once.





why not



leads to (a,c) (b,d)

none of ab.cd in C

RC → TSP

Instance: $G=(V,E)$

Solution: Cycle S visiting each with exactly once

$G \longrightarrow$ $d_{ij}=1$ if $(i,j)\in E$; $d_{ij}=1+2$ if $(i,j)\notin E$

Instance: distances d_{ij} & a bond B

Solution: a permutation $\tau: \{1\cdots n\} \to \{1\cdots n\}$ s.t. $d_{\tau(n)\tau(1)} + d_{\tau(1)\tau(2)} + \cdots + d_{\tau(n-1)\tau(n)} \le B$

ZOE → Subset Sum

Instance: $A \in \{0,1\}^{m\times n}$

Solution: $x \in \{0,1\}^n$ s.t. $Ax=1$

$A = \begin{bmatrix} \\ \\ \\ \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \longrightarrow a_i = \sum_{i\in S} 2^{i-1}$, $w = \sum_{i=1}^{n} 2^{i-1}$

Instance: $a_1 \cdots a_n, w$

Solution: $x_1 \cdots x_n \in \{0,1\}$ s.t. $\sum a_i x_i = w$

$\begin{bmatrix} 1&0&0&0&0&1 \\ 0&1&0&0&1&0 \\ 0&0&1&0&1&0 \\ 0&1&0&1&0&0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}$

18 5 4 8

$w = \sum_{i=1}^{n} (n+1)^{i-1}$

$q_i = \sum_{j\in S_i} (n+1)^{j-1}$

Up to Now

All of NP

↓ SAT

↓ 3SAT

↙ Independent Set → Vertex cover, Clique. ↘ 3D Matching → RC, ZOE → Subset Sum

NP-Complete problem still need a solution

① "Intelligent" exponential search → running time could be exponential

② Approximation Algorithms → polytime → practical instances → run efficiently.

③ Heuristic → no guarantees on the runtime or optimality of the solution.

Back-tracking

Start with some problem P_0

Let $S=\{P_0\}$ the set of active sub problems

Repeat while S is not empty:

choose subproblem $P_i \in S$, and remove it from S.

expand into smaller subproblems $P_1 \cdots P_k$

For each P_i

if test(P_i) succeeds: halt and announce this solution

test(P_i) fails: discard P_i

otherwise: add P_i to S.

Announce that there is no solution.

Branch and Bound: Generalizing backtracking to search problems

Start with some problem P_0

Let $S=\{P_0\}$ the set of active sub problems

best so far $=\infty$

Repeat while S is not empty:

choose subproblem (partial solution) $P_i \in S$ and remove it from S.

expand into smaller subproblems $P_1 \cdots P_k$

For each P_i

if P_i is a complete solution: update best so far

else if lowerbound(P_i) < best so far: add P_i to S

Return best so far.

TSP - branch and bound

$W_{TSP} = d_{\tau(n)\tau(1)} + d_{\tau(1)\tau(2)} + \cdots + d_{\tau(n-1)\tau(n)}$ is minimized

Lemma: $W_{TSP} \ge W_{MST}$

Approximation Algorithms for an optimization problem

Instance I → $OPT(I)$

(minimization problem eg. TSP) → A → $A(I)$

approx ratio $\alpha = \max_I \frac{A(I)}{OPT(I)}$ (minimization)

$\alpha = \max_I \frac{OPT(I)}{A(I)}$ (maximization)

Set Cover

Input: A set of elements B; sets $S_1, S_2 \cdots S_m \subseteq B$

Output: Smallest selection of S_i, whose union is B

Greedy Algorithm: Repeat until all elements of B are covered

Pick the set S_i with the largest number of uncovered elements.

Claim: say $|B|=n$ and $OPT(I)=k$. Then greedy algorithm uses at most $k \ln n$ sets

proof: let n_t be the number of uncovered elements in B after t iterations of our greedy algorithm.

at most one of these sets has $\ge \frac{n_t}{k}$ uncovered elements.

$\therefore n_{t+1} \le n_t - \frac{n_t}{k} = n_t(1-\frac{1}{k}) \Rightarrow n_t \le n(1-\frac{1}{k})^t \Rightarrow n_t \le ne^{-\frac{t}{k}}$

$\therefore t \Rightarrow k \ln n$

Vertex Cover

Input: $G=(V,E)$ Output: Subset $S \subseteq V$ such that $|S|$ is minimized and S touches every edge.

approx solution: ① find a maximal matching $M \subseteq E$

② Return $S=$ { all endpoint of edges in M }.

$|S| = 2|M| \le 2 \times$ size of any VC

Clustering

Input: Points $X = \{x_1 \cdots x_n\}$ with distance metric $d(\cdot,\cdot)$; integer k.

Output: k clusters $C_1 \cdots C_k$ s.t. max $\max_i \max_{x,y\in C_i} d(x,y)$ minimize the diameter of clusters

(diameter of C_i)

① $\forall x,y$, $d(x,y) \ge 0$

② $d(x,y)=0$ iff $x=y$

③ $d(x,y)=d(y,x)$

④ $d(x,y)+d(y,z) \ge d(x,z)$

Pick any input $\mu_1 \in X$ as the first cluster center

for $i=2 \cdots k$

let $\mu_i \in X$ be the point farthest from $\mu_1 \cdots \mu_{i-1}$

create k clusters. $C_i = \{$all $x \in X$ closet to $\mu_i\}$

1) $\forall i$, $\forall x,y \in C_i$, $d(x,y) \le 2r$

2) $\forall j \in \{1 \cdots k\}$, $d(\mu_i, \mu_j) \ge r$

Lemma: $\forall i$, $\forall x,y \in C_i$, $d(x,y) \le 2r$

proof: $d_A = d(x,y) \le d(x,\mu_i) + d(\mu_i, y) \le r+r \le 2r \le 2 d_{OPT}$

RC → TSP

Input: $G=(V,E)$

Solution: cycle S visiting each vertex exactly one

Instance: distances d_{ij}

Solution: a permutation $\tau: \{1\cdots n\} \to \{1\cdots n\}$ s.t. $d_{\tau(n)\tau(1)} + d_{\tau(1)\tau(2)} + \cdots + d_{\tau(n-1)\tau(n)} \le n$

$G \longrightarrow$ $d_{ij}=1$ if $(i,j)\in E$; $d_{ij}=1+c$ if $(i,j)\notin E$ for some $c>1$

if G had a RC $\Rightarrow G'$ has a TSP solution of cost n

if G does not have a RC $\Rightarrow G'$ has no TSP solution of cost $\le n+c$

RC → 2-TSP

when $c=2n$: if G has a RC → G' has a TSP solution of cost n

if G doesn't have a RC → at least cost $n+(c+2)$

2-TSP

Instance: distances d_{ij}

Solution: a permutation $\tau: \{1\cdots n\} \to \{1\cdots n\}$ s.t. $d_{\tau(n)\tau(1)} + d_{\tau(1)\tau(2)} + \cdots + d_{\tau(n-1)\tau(n)} \le 2 d_{OPT}$

$\forall i,j,k$: $d_{ij} + d_{jk} \ge d_{ik}$.

Lemma: ① MST can be a good starting point

② $d \le 2 d_{MST} \le 2 d^{OPT}_{TSP}$

③ $d^{A}_{TSP} \le d \le 2 d^{OPT}_{TSP}$

d^{A}_{TSP}

Knapsack (without rep) $O(nW)$

Input: n, w_i, v_i Goal: Most valuable combination with total $\le w$

$0<\varepsilon<1$. we will give an approx algorithm s.t. $k \ge (1-\varepsilon)k^*$

Discard any items with weight $> w$

let $V_{max} = \max_i v_i$

Rescal $\hat{v}_i = \lfloor v_i \cdot \frac{n}{\varepsilon V_{max}} \rfloor$

Run DP algorithm with values $\{\hat{v}_i\}$.

Output the resulting choice of items.

$n \times \frac{n}{\varepsilon} = \frac{1}{\varepsilon} n^2$ $\frac{1}{\varepsilon}$-optimal

$O(\frac{n^3}{\varepsilon})$ polytime $(n^3 \frac{1}{\varepsilon})$

(1) $\sum_{i\in S} \hat{v}_i = \sum_{i\in S} \lfloor v_i \frac{n}{\varepsilon V_{max}} \rfloor \ge \sum_{i\in S} (\frac{v_i n}{\varepsilon V_{max}} - 1) \ge (\frac{\sum_{i\in S} v_i}{\varepsilon V_{max}} n) - |S|$

$\ge \frac{k^* n}{\varepsilon V_{max}} - n = (\frac{k^*}{\varepsilon V_{max}} - 1) n$ $k \ge (\frac{k^*}{\varepsilon V_{max}} - 1) n$

(2) $\sum_{i\in S} v_i \ge \sum_{i\in S} \hat{v}_i \frac{\varepsilon V_{max}}{n} \ge (\sum_{i\in S} \hat{v}_i) \frac{\varepsilon V_{max}}{n} \ge (\frac{k^*}{\varepsilon V_{max}} - 1) \frac{n \varepsilon V_{max}}{n}$

$= k^* - \varepsilon V_{max} \ge k^* (1-\varepsilon)$

Local Search Heuristics

Let S be any initial solution

While there is some solution S' in the neighborhood of S for which $cost(S') < cost(S)$: replace S with S'

return S

2-opt ⇒ $O(n^2)$

3-opt ⇒ $O(n^3)$

Graph Partitioning

Input: an undirected graph $G=(V,E)$ with non-negative edge weights.

Output: A partition of vertices into groups A, B s.t. $|A|=|B|$ and minimizing capacity of cut (A,B)

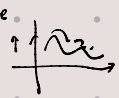
Simulated Annealing. Introduce temperature parameter T).

Let S be any initial solution

Randomly choose a solution S' in the neighborhood of S

if $cost(S') < cost(S)$: replace S by S'

else: replace S by S' with probability $e^{-\frac{(cost(S')-cost(S))}{T}}$



cost

Thinking of NP as a proof (Interactive Proofs)

Prover P → (satisfying assignment) ϕ → Verifier V verifies → polytime.

completeness: If "ϕ is true", then in $P(\phi,w) \longleftrightarrow V(\phi)$ outputs 1

soundness: If "ϕ is true", then in $P(\phi,w) \longleftrightarrow V(\phi)$, V outputs 1 with very small probability eg. 2^{-n} (n is some parameter)

verify $A \vee B = C$

① $r \leftarrow \{1 \cdots q\}$ $\vec{r} = (1, r, \cdots r^{n-1})$

② $C \cdot \vec{r} = (A \vee B) \cdot \vec{r}$ $O(n^2)$ faster than $O(n^3)$ $O(n^2)$

Soundness: $D = A \vee B$ $C \ne D$

then $\exists$ $c_i \ne d_i$ & $c_i \cdot \vec{r} = d_i \cdot \vec{r}$ $\Rightarrow (c_i - d_i) \cdot \vec{r} = 0$

$(p_0 + p_1 \cdots + p_{n-1})(1, r, \cdots r^{n-1}) = 0$. $O(\frac{n^2}{q})$. $p \le \frac{n}{q} \to 0$

$\therefore p(x) = p_0 + p_1 x + \cdots + p_{n-1} x^{n-1} \Rightarrow p_0 + p_1 r + \cdots + p_{n-1} r^{n-1} = 0 \Leftrightarrow$

Graph Isomorphism

Instance $G_0 = (V, E_0)$ $G_1 = (V, E_1)$

Solution: $\pi: V \to V$ s.t. $be \in E_0$ iff $(\pi(u), \pi(v)) \in E_1$

$H \cong_{G_0} P$ $\xleftarrow{H = \sigma(G_0)}$ b'

① $\sigma: \{V \to V\}$

② $b \leftarrow \{0,1\}$

$b=b'$ then output 1 else output 0.

Completeness: V

soundness: $G_0 \not\cong G_1 \not\cong H$

$P \xleftarrow{H=\sigma(G_b)} b' = b'(G_1, b)$

$P[b=b'] = \frac{1}{2}$