NP Complete Problems

Decision vs. Optimization problems

Algorithm (I) → actual solution, with the optimal cost

instance

Input

Optimization

\[
\begin{align*}
\text{Alg}(I, b) & \quad \begin{cases}
\text{Yes/True/1} \\
\text{No/False/0}
\end{cases} \\
\text{Instance/}
9-PSVR \\
\text{bound on the cost/value}
\end{align*}
\]

polynomial time Reductions

P: poly time solvable

NP: (non-deterministic poly time)

verified in poly time

given a solution, we can check if it is correct in polynomial time
$d_{ij} = \text{dist}(i, j)$

$\text{opt-TSP} : \quad \text{Given all the pair-wise distances between vertices, return the optimal path that visits each vertex.}$

Does $\text{opt-TSP} \in \text{NP}$?

$\rightarrow \quad \text{Can we verify a solution in p time}$

Given a solution $S = \langle v_0 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_0 \rangle$

$s = d_{v_0 v_2} + d_{v_2 v_3} + d_{v_3 v_4} + d_{v_4 v_5} + d_{v_5 v_0}$

$\text{End of TSP}$

Is $s$ optimal?

Looks like no efficient way of answering this in poly time.

$\text{p-TSP} : (I, S, b)$

domain

Given $S$, verification means: Is $s_{uv} \leq b$?

\[
\begin{cases}
\text{1)} \quad \text{Compute } s = d_{v_0 v_2} + d_{v_2 v_3} + \ldots \quad \text{from } S \\
\text{2)} \quad \text{Is } s \leq b \\
& \quad \text{YES} \quad \text{NO}
\end{cases}
\]

$O(|V| + |E|)$

we can verify in p time.
we can vary in 1 hour.

Q1: \[
d - TSP: \quad \frac{\text{G}}{\text{di}_j} \Rightarrow 0
\]

Is there a TSP with value \( v \leq b \)?

\( \text{NP} \)-Complete

Q2: \[
\text{opt-} TSP: \quad \frac{\text{G}}{\text{di}_j}
\]

give actual TSP solution with optimal \( v \)

\( \text{NP} \)-hard

\[d - TSP \equiv \text{opt-} TSP\]

poly-time reduction/transformation

bit-complexity of \( b \) is Important

transformation better be \( \log b \)

Reduce \( d - TSP \) to \( \text{opt-} TSP \)

\[T = \begin{cases} \text{G} & \text{if } \text{di}_j \leq b \\ \text{no} & \text{otherwise} \end{cases}\]
Using the optimal TSP to solve \( d \)-TSP is trivial \( O(1) \) time.

How to use \( d \)-TSP to answer \( \text{opt-TSP} \)?

How can we find the optimal TSP value \( v \), using only \( d \)-TSP.

\[ I = \left\{ \begin{array}{l}
J = \{ G \}

\text{Opt-TSP}

\text{Use double binary search}

\Rightarrow \text{Yes or No}

\text{actual optimal value for TSP}

0(\log v)

Naive:

\[ b = 1, b = 2, b = 3, \ldots, b = \text{101}, b = \text{102} \]

No No No No No No Yes

When we get the first yes, that's the optimal value.

\[ \Rightarrow \text{Check the v} \]

\[ b = \text{102} = v \]

\[ 0(v) \]

\[ v \text{ is value } b \text{ for TSP} \]

binary-search:

( double binary search )

a)

\[ b = 2^0 = 1 \]

\[ b = 2^1 = 2 \]

\[ b = 2^2 = 4 \]
\[ b = 2^v \]

\[ \begin{array}{ccccccc}
1 & 2 & 4 & 8 & 16 & 32 & \cdots \\
No & No & No & No & No & No & No
\end{array} \]

At most \( \lceil \log_2 v \rceil \) steps to find the first interval.

b) \( v \in \left[ 2^{k-1}, 2^k \right) \)

Do a regular binary search on this interval to find the actual \( v \).

\[ \frac{2^{k-1}}{2^k} = \frac{1}{2} \]

Q: \( v \leq \text{mid} \)

No -

Only \( 2^{k-1} \) numbers

\[ \log (2^{k-1}) = k-1 \text{ steps} \]
$k = \log_2(n)$

$P$: poly-time solvable

$NP$: poly-time verifiable $\rightarrow d = T^p \in NP$ $\rightarrow \text{opt} - T^s \notin NP$

$NP$-Complete: $NPc$

$\Rightarrow$ hardest problems in $NP$

all these problems are "the same"

poly-time transformation from one to the other

Is $P = NP$?

Is $P = NPc = \emptyset$?

$\Omega(?) = T^p = O(2^n \cdot n^2)$ time

$P$? or $NP$?

DP solution $|V| = n$

$NP$ Complete Problems:

A problem $L$ belongs to $NP$-complete if
1) \( L \in \text{NP} \iff \text{verifiable in poly-time} \)

\[
\begin{cases}
\text{for all problem } L' \text{ in NP} \\
\text{if } L' \text{ can be reduced to } L, \text{ denoted as } L' \rightarrow L
\end{cases}
\]

\( L' \rightarrow L \): \( L' \) can be poly-time reduced to \( L \)

- \( x \) is a solution to \( L' \) if and only if \( f(x) \) is a solution to \( L \),
- and \( f \) is computable in poly-time
- \( f \): transformation function

\[ \text{SAT} \in \text{NP Complete} \]

- Boolean satisfiability problem

\[
\begin{cases}
\text{Variables: } x, y, z \\
\text{Literals: } \text{pur or neg} \\
\text{Vocable: } x, \bar{x} \\
\text{Clause: } \text{OR of literals} \\
\text{Disjunction: AND of clauses} \\
\text{Expression: AND of clauses} \\
\text{SAT: Is the expression satisfiable?}
\end{cases}
\]

Is there an assignment of 0/1 to the variables, such that expression is true?
1) $\text{SAT} \in \text{NP}$

In $O(N)$ time we can verify a solution.

$N = \sum d_j$ all be clause lengths

$N = \text{poly in input size}$

2) All problems in $\text{NP}$ reduce to $\text{SAT}$

$\xrightarrow{\text{poly-time transformation}}$

3) $L'$ can be coded as a digital circuit (boolean expression)

(who coded $\leftarrow$ hardware solution)

\[
\begin{array}{c}
\text{NP} \\
\text{d-TSP} \in \text{NP} \\
\text{digital circuit} \rightarrow \text{boolean expression}
\end{array}
\]

4) We have to show that the digital circuit is poly in size of $L'$ encoding.

Proof exists!

Instead of using poly-size digital circuit for all $\text{NP}$ problem.
Use reduction trick!

\[ \text{SAT} \rightarrow \text{d-TSP} \]

prove to be NPC

---

How to show that a problem \( L' \) is NP complete?

1) \( L' \in \text{NP} \)
2) Find some other NPC problem \( L \)
   then show there is a polynomial-time transformation from \( L \) to \( L' \)

Original: \( \forall L \in \text{NP} \)
\( L \rightarrow L' \)

\[ \text{SAT} \rightarrow \text{3SAT} \]

Known problem

\[ \text{NPC} \]
\[ \text{SAT} \]
Show 3-SAT ∈ NP

\[ L : 3\text{-SAT} \text{ is \ \text{NP}} \]

Specialization of general SAT

3-SAT \leftarrow \text{no clause can have more than 3 literals}

\[
\text{SAT} : (x_1 \lor y \lor z \lor \overline{w} \lor \overline{v}) \land (c) \land (c)
\]

5 literals

\[
\text{3-SAT} : (x_1 \lor y \lor z) \land (\overline{y} \lor \overline{w}) \land (x \lor \overline{z})
\]

\[ |G| \leq 5 \]

Show 3-SAT ∈ NP?

1) 3-SAT ∈ NP \leftarrow \text{obviously true, verify in } P \text{ time}

2) SAT → 3-SAT

\[ \text{SAT} = 3\text{-SAT} \]

\[ \text{Poly time} \]

\[
(c \lor \ell \lor \ell \lor \ell \lor \ell \lor \ell \lor \ell \lor \ell) \land (c)
\]
\( (c_1 \lor c_2 \lor (e_3 \lor (e_4 \lor (e_5 \lor c_6))) \land (\overline{y}_1 \lor y_2 \lor y_3) \land (\overline{y}_2 \lor y_4 \lor y_5) \land (\overline{y}_3 \lor y_5 \lor y_6) \)

\[ y_1, y_2, y_3 \text{ : dummy variables} \]