NP-Completeness


General Problems, Input Size and Time Complexity

- Time complexity of algorithms:
  polynomial time algorithm ("efficient algorithm")
  v.s.
  exponential time algorithm ("inefficient algorithm")
“Hard” and “easy’ Problems

- Sometimes the dividing line between “easy” and “hard” problems is a fine one. For example
  - Find the shortest path in a graph from X to Y. (easy)
  - Find the longest path in a graph from X to Y. (with no cycles) (hard)
- View another way – as “yes/no” problems
  - Is there a simple path from X to Y with weight <= M? (easy)
  - Is there a simple path from X to Y with weight >= M? (hard)
- First problem can be solved in polynomial time.
- All known algorithms for the second problem (could) take exponential time.

Decision problem: The solution to the problem is "yes" or "no". Most optimization problems can be phrased as decision problems (still have the same time complexity).

Example: Assume we have a decision algorithm X for 0/1 Knapsack problem with capacity M, i.e. Algorithm X returns “Yes” or “No” to the question “is there a solution with profit ≥ P subject to knapsack capacity ≤ M?”
We can repeatedly run algorithm X for various profits (P values) to find an optimal solution. Example: Use binary search to get the optimal profit, maximum of \( \log \sum p_i \) runs.

(Where M is the capacity of the knapsack optimization problem)

<table>
<thead>
<tr>
<th>Min Bound</th>
<th>Optimal Profit</th>
<th>Max Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Search for the optimal solution</td>
<td>( \sum p_i )</td>
</tr>
</tbody>
</table>

The Classes of P and NP

- The class P and Deterministic Turing Machine
  - Given a decision problem X, if there is a polynomial time Deterministic Turing Machine program that solves X, then X is belong to P
  - Informally, there is a polynomial time algorithm to solve the problem
• The class NP and Non-deterministic Turing Machine
  
  • Given a decision problem X, if there is a polynomial time Non-deterministic Turing machine program that solves X, then X belongs to NP
  
  • Given a decision problem X. For every instance I of X, (a) guess solution S for I, and (b) check “is S a solution to I?”. If (a) and (b) can be done in polynomial time, then X belongs to NP.

• Obvious : \( P \subseteq NP \), i.e. A problem in P does not need “guess solution”. The correct solution can be computed in polynomial time.

• Some problems which are in NP, but may not in P :
  
  • 0/1 Knapsack Problem
  
  • PARTITION Problem : Given a finite set of positive integers Z.
    Question : Is there a subset \( Z' \) of Z such that
    \[
    \text{Sum of all numbers in } Z' = \text{Sum of all numbers in } Z-Z' \, ?
    \]
    i.e. \( \sum Z' = \sum (Z-Z') \)
• One of the most important open problem in theoretical compute science: Is \( P = NP ? \)
Most likely “No”. Currently, there are many known problems in NP, and there is no solution to show anyone of them in \( P \).

NP-Complete Problems

• Stephen Cook introduced the notion of NP-Complete Problems. This makes the problem “\( P = NP ? \)” much more interesting to study.

• The following are several important things presented by Cook:
1. Polynomial Transformation ("\(\propto\")

- \(L_1 \propto L_2\) : There is a polynomial time transformation that transforms arbitrary instance of \(L_1\) to some instance of \(L_2\).

- If \(L_1 \propto L_2\) then \(L_2\) is in P implies \(L_1\) is in P (or \(L_1\) is not in P implies \(L_2\) is not in P)

- If \(L_1 \propto L_2\) and \(L_2 \propto L_3\) then \(L_1 \propto L_3\)

2. Focus on the class of NP – decision problems only. Many intractable problems, when phrased as decision problems, belong to this class.

3. \(L\) is NP-Complete if \(L \in NP\) and for all other \(L' \in NP\), \(L' \propto L\)

   - If a problem in NP-complete can be solved in polynomial time then all problems in NP can be solved in polynomial time.
   - If a problem in NP cannot be solved in polynomial time then all problems in NP-complete cannot be solved in polynomial time.

   **Note that an NP-complete problem is one of those hardest problems in NP.**
• So, if an NP-complete problem is in P then \(P=NP\)

• If \(P \neq NP\) then all NP-complete problems are in NP-P

Question: how can we obtain the first NP-complete problem \(L\)?

4. Cook Theorem: SATISFIABILITY is NP-Complete. (The first NP-Complete problem)

Instance: Given a set of variables, \(U\), and a collection of clauses, \(C\), over \(U\).

Question: Is there a truth assignment for \(U\) that satisfies all clauses in \(C\)?

Example:

\[U = \{x_1, x_2\}\]

\[C_1 = \{(x_1, \neg x_2), (\neg x_1, x_2)\}\]

\[= (x_1 \text{ OR } \neg x_2) \text{ AND } (\neg x_1 \text{ OR } x_2)\]

If \(x_1 = x_2 = \text{True}\) \(\Rightarrow\) \(C_1 = \text{True}\)

\[C_2 = (x_1, x_2) (x_1, \neg x_2) (\neg x_1) \Rightarrow \text{not satisfiable}\]

“\(\neg x_i\)” = “not \(x_i\)” “OR” = “logical or” “AND” = “logical and”

This problem is also called “CNF-Satisfiability” since the expression is in CNF – Conjunctive Normal Form (the product of sums).
• With the Cook Theorem, we have the following property:

Lemma: If $L_1$ and $L_2$ belong to NP, $L_1$ is NP-complete, and $L_1 \propto L_2$ then $L_2$ is NP-complete.

i.e. $L_1, L_2 \in \text{NP}$ and for all other $L' \in \text{NP}$, $L' \propto L_1$ and $L_1 \propto L_2 \Rightarrow L' \propto L_2$

• So now, to prove a problem $L$ to be NP-complete problem, we need to

  • show $L$ is in NP
  • select a known NP-complete problem $L'$
  • construct a polynomial time transformation $f$ from $L'$ to $L$
  • prove the correctness of $f$ (i.e. $L'$ has a solution if and only if $L$ has a solution) and that $f$ is a polynomial transformation
• P: Problems solvable by deterministic algorithms in polynomial time

• NP: Problems solved by non-deterministic algorithms in polynomial time

• A group of problems, including all of the ones we have discussed (Satisfiability, 0/1 Knapsack, Longest Path, Partition) have an additional important property:
  
  If any of them can be solved in polynomial time, then they all can!

• These problems are called NP-complete problems.

NP-Complete

NP

P

Winter-2005 Young CS 599

NP-Completeness 17