NP-Completeness (part 2 of 3)

Outline

1. Introduction
2. P and NP
3. NP-complete (NPC): formal definition
4. How to prove a problem is NPC
5. How to solve a NPC problem: approximate algorithms

II. P and NP

- An algorithm is said to be *polynomial bounded* if its worst-case complexity $T(n)$ is bounded by a polynomial function of the input size n:

$$T(n) = O(n^k).$$

  Examples: Algorithms for LCS, Shortest path, MST, Euler tour, Circuit value.

- **P** is the class of decision problems that can be *solved* in polynomial time, i.e., they are polynomial bounded

II. P and NP

- **NP** is the class of decision problems that are *verifiable* in polynomial time.[1]

  i.e., if we were given a "certificate" (= a solution), then we could verify that whether the certificate is correct in polynomial time.

  Examples: Certificates for Circuit-SAT, Hamiltonian cycle, graph coloring.

[1] The name "NP" stands for "Nondeterministic Polynomial time".

II. P and NP

- P $\subseteq$ NP, since if a problem is in P then we can solve it in polynomial time without even being given a certificate.
- Open question[2]: Does P $\subset$ NP or P $=$ NP?

[2] http://www.claymath.org/millennium-problems

II. P and NP

- The size of the input can change the classification of P or NP.
- Examples:
  - Prime-testing problem:

$$O(n) \quad \xrightarrow{n=10^m} \quad O(10^m)$$

  - Knapsack problem

$$O(nW) \quad \xrightarrow{W=10^m} \quad O(n \cdot 10^m)$$

- Knowing the effect on complexity of the size of the input is important.
- Unfortunately, even with strong restrictions on the inputs, many NPC problems are still NPC.

  Example: 3-Conjuntive Normal Form (3-CNF) SAT problem

III. NP-complete

- NP-complete (NPC) is the term used to describe decision problems that are the hardest ones in **NP** in the following sense

  > *If there were a polynomial-bounded algorithm for an NPC problem, then there would be a polynomial-bounded time for each problem in NP.*

III. NP-complete

Formal definition:

- A decision problem A is **NP-complete (NPC)** if

  (1) $A \in$ NP and

  (2) every other problems B in NP is *polynomially reducible* to A, denoted as
  $$B \leq_T A$$

If a problem satisfies the property (2), but not necessarily the property (1), we say the problem is NP-hard.

III. NP-complete

Polynomial reduction $B \leq_T A$

- Let A and B be two decision problems, B is polynomially reducible to A, if there is a poly-time computable transformation T such that

$$\text{Yes-instance of } A \quad \stackrel{\text{iff}}{\Longleftrightarrow} \quad \text{Yes-instance of } B$$

III. NP-complete

- Cook's theorem (1971): Circuit-SAT is NPC.

  First result deomonstrating that a specific problem is NPC.

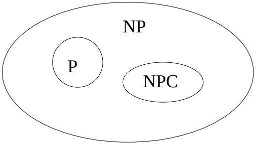
- Known NPC problems:
  - Graph coloring
  - Hamiltonian cycle
  - TSP
  - Knapsack
  - Subset sum
  - ......
  - ......

Subset sum decision problem: Given a positive integer c, and the set $S = \{s_1, s_2, \ldots, s_n\}$ of positive integers s_i for $i = 1, 2, \ldots, n$. Assume that $\sum_{i=1}^{n} s_i \geq c$. Is there a $J \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in J} s_i = c$.

III. NP-complete

P, NP and NPC:

- How most theoretical computer scientists **view** the relationships among P, NP and NPC:
  - Both P and NPC are wholely contained within NP
  - $P \bigcap NPC = \emptyset$



II–III recap

1. P and NP
2. NP-complete: formal definition
3. Polynomial reduction