NP-Hard and NP-Complete

Ming-Hwa Wang, Ph.D.
COEN 279/AMTH 377 Design and Analysis of Algorithms
Department of Computer Engineering
Santa Clara University

Basic Concepts
- tractable: polynomial-time-bound algorithms, generally through deeper insight of the structure of a problem
- intractable: algorithms with at least exponential run time, needs exhaustive searching
- extreme large power or extreme large coefficient
- reasonable encoding scheme: without unnecessary information, differ at most polynomially from one another
- reasonable computer model: Turing machine
  polynomial bound on the amount of work that can be done in a single unit of time
  the intractability of a problem is independent of reasonable encoding scheme and computer model
  proving inherent intractability can be just as hard as finding efficient algorithm
- deterministic: unique next state
- nondeterministic: always choose one of the next states which leads to a solution, imaging an oracle that magically advises the algorithm to make the correct choice at every choice point
- nondeterministic algorithms: use choice function and 2 statements success and failure. All in O(1)
- nondeterministic programming language: ICON, Prolog
- nondeterministic can be simulated by unbounded parallelism
- many optimization problems can be recast into decision problems (minimum \( \rightarrow \leq \), maximum \( \rightarrow \geq \)) with the property that the decision problem can be solved in polynomial time if the corresponding optimization problem can
- decision problems can be represented as languages, solving a decision problem is equivalent to recognizing the corresponding language
- Undecidable problems are recursively undecidable
  - undecidable problems are still undecidable, even if nondeterministic is allowed
  - halting problem, infinite loop-checking program, finitely presented groups, Hilbert’s tenth problem (solvability of polynomial equations in integers), tiling the plane

NP-Complete and NP-Hard
- Nondeterministic polynomial-time (NP) [Cook 1971]
- NP-complete problems are roughly equivalent in complexity. Either all these problems have polynomial-time solutions or none of them do.
- All NP-complete problems are NP-hard, but all NP-hard problems are not NP-complete.
- An NP-complete problem has the property that any problem in NP can be polynomially reduced to (partial order \( \leq \)) it. Any instance of \( P_1 \) can be polynomially reduced/transformed to an instance of \( P_2 \). Solve \( P_2 \), and then map the answer back to the original. [Karp 1972]

Languages and Turing Machines
- For any finite set \( \Sigma \) of symbols, \( \Sigma^* \) is the set of all finite strings of symbols from \( \Sigma \).
- If \( L \) is a subset of \( \Sigma^* \), \( L \) is a language over \( \Sigma \).
- Standard encoding scheme maps instances into structured strings over the alphabet \( \Psi = \{0, 1, -, [ ], ( ), \}, \) where ":" for minus, ":[" and "]" for structured name, "(" and ")" for sequence.
- The deterministic one-tape Turing machine (DTM) consists of a finite state control, a read-write head, and a tape made up of a two-way infinite sequence.
- A program for a DTM specify the following information:
  - a finite set of tape symbols \( \Gamma = \{\Sigma, b\} \), where \( \Gamma \) for input symbols and \( b \) for blank symbol
  - a finite set \( Q \) of states, including distinguished start-state \( q_0 \) and halt-state \( q_f \) and \( q_h \)
  - a transition function \( \delta : (Q - \{q_f, q_h\}) \times \Gamma \rightarrow Q \times \{1, -\}^* \)
- A nondeterministic Turing machine (NDTM) has an extra guessing module and its write-only head to write down the guess
- language recognized by \( M \) is \( L_M = \{x \in \Sigma^*, M \text{ accepts } x\} \), where accept means terminate at \( q_f \)
- Polynomial time algorithm: \( \forall n \in \mathbb{Z}^+ \), \( T_M(n) \leq p(n) \)
- class \( P = \{L : \exists \text{ polynomial time DTM program } M, L = L_M\} \)
- class \( NP = \{L : \exists \text{ polynomial time NDTM program } M, L = L_M\} \)

NP-Complete Problems
- Satisfiability problem, first proven NP-complete problem
  - takes as input a boolean expression and asks whether the expression has an assignment to the variables that gives a value of 1
  - Cook theorem: satisfiability is in \( P \) iff \( P = NP \)
- Order of transformations for NP-complete problems:
  - CNF-satisfiability \( \leq_1 3\)-satisfiability
- Traveling salesman problem
  - Given a complete graph $G = (V, E)$, with edge costs, and an integer $K$, is there a simple cycle that visits all vertices and has total cost $\leq K$?
  - reduce the Hamiltonian cycle problem to traveling salesman problem by constructing a new graph $G'$. $G'$ has the same vertices as $G$. For $G'$, each edge $(v, w)$ has a weight of 1 if $(v, w) \in G$, and 2 otherwise. Choose $K = |V|$. $G$ has a Hamiltonian cycle iff $G'$ has traveling salesman tour of total weight $|V|$.

**Approximation Algorithms**