NP-Complete Problems

- Computationally Difficult Problems
- There is no KNOWN efficient algorithm to solve any of these problems. (e.g., $O(n)$, $O(n^4)$, $O(n^{100})$, $O(n^{1000})$, ...). Therefore, problems are computationally difficult even under this relaxed notion of “efficient” algorithms.
- It is conjectured that no efficient algorithm exists to solve any of these problems
- (Efficient algorithm = worst case time complexity is polynomial wrt the input length).
Intractability

• Intractability is Independent of the Encoding (reasonable encodings) E.g., Adjacency matrix or adjacency lists.

• Intractability is Independent of the Computer Model (as long as it is a reasonable model). Holds for “reasonable” models, e.g., parallel machine with a fixed number of processors.
FOUNDATIONS

- Stephen Cook: “The Complexity of Theorem Proving Procedures”.
  - Polynomial time reducibility.
  - Focussed attention to the class of Decision Problems NP.
  - Showed that Satisfiability is the hardest problem in NP.
  - Suggested other problems in NP share this property (e.g., clique).

- Richard Karp
  - “Reducibility among Combinatorial Problems”.
  - Showed that other problems in NP are as hard as Satisfiability. (equivalence class of “hardest” problems in NP or the class of NP-complete problems). E.g., Knapsack, Traveling Salesperson, Graph Coloration, etc.
NP-Complete (and NP-hard) Problems

- Scheduling
  - One machine with release dates and deadlines.
  - Parallel machines to minimize $\max\{f_i\}$.
  - Open Shops, Flow Shops, Job Shops, etc.

- Mathematical Programming
  - Integer Programming
  - Quadratic Programming
  - Traveling Salesperson
  - Knapsack
  - Flow Problems

- Games
  - NxN Checkers and NxN GO
  - Generalized HEX
  - Generalized Geography
  - Generalized Tetris
• Automata Theory and Formal Languages
  – Finite Automata Inequivalence
  – Regular Expression Inequivalence
  – Finite Automata Intersection
  – Non-LR(k) CFG
  – Context-Sensitive Lang rec.

• Code Generation
  – Code Generation (one register machine)
  – Code Generation (Parallel Assignment)
  – Micro-code bit Optimization
  – Minimizing + in expressions
  – Minimizing + and * in expression

• Programs
  – Inequivalence of Programs (with arrays)
  – Inequivalence of Programs (with assignments)
• Other
  – Deadlock Detection
  – Deadlock Recovery
  – Database Design Problems
  – Data Compression
  – File Allocation
  – Dynamic Storage Allocation
  – Bin Packing
  – Graph Problems
  – Min Page Faults with Complete Information
  – Clustering Problems
  – Wire Routing Problems
  – Via Assignment Problems
  – PLA Folding
  – Placement Problems
  – Robot Motion Planning
  – ETC.
FORMAL DEVELOPMENT

- Restrict to Decision Problems (for convenience).
- A Decision Problem $\pi$ consists of set $D_{\pi}$ of instances, and set $Y_{\pi} \subseteq D_{\pi}$ of yes-instances.
Problem Specification

- **Subgraph Isomorphism**
  - Instance: Two graphs, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$.
  - Question: Does $G_1$ contain a subgraph isomorphic to $G_2$, i.e., a subset $V' \subseteq V_1$ and, a subset $E' \subseteq E_1$ s.t. $|V'| = |V_2|$, $|E'| = |E_2|$, and there exists a 1 - 1 function $f : V_2 \rightarrow V'$ satisfying $u, v \in E_2 \Leftrightarrow f(u), f(v) \in E'$?

- **Traveling Salesperson**
  - Instance: A finite set $C = \{c_1, c_2, ..., c_n\}$ of cities, a distance $d(c_i, c_j) \in Z^+$ for each pair of cities $c_i, c_j \in C$, and a bound $B \in Z^+$
  - Question: Is there a “tour” of all the cities in $C$ having a total length no more than $B$, i.e., an ordering $(c_{\pi_1}, c_{\pi_2}, ..., c_{\pi_n})$ of $C$ s.t. $\sum_{i=1}^{n-1} d(c_{\pi_i}, c_{\pi_{i+1}}) + d(c_{\pi_n}, c_{\pi_1}) \leq B$?
Decision Problems and the Class P

- Decision problem is not harder than the corresponding optimization problem.
- So, showing that the decision problem is NP-complete $\Rightarrow$ the corresponding optimization problem is as hard as the decision problem.
- Class $P$: Set of all decision problems that can be solved in polynomial time with respect to (wrt) the input size.
NonDeterministic Computations

• POLY TIME VERIFICATION FOR TSP
  – Suppose one claims that $I \in Y_\pi$.
  – To prove the claim we need a “valid” tour.
  – The truth or falsity of the claim can be verified by computing the cost of the tour and comparing it to B.
  – This can be done by an algorithm operating in poly time wrt length(I).

• Is Poly time verification $\neq$ Poly time solvability?

• We do not know? Any conjectures?

• YES
Class NP

- NP: Set of all decision problems that can be verified in polynomial time. I.e., we can verify a “yes” answer in polynomial time.

- If $P \neq NP$, $P$ is Poly and $NP - P$ is intractable
Polynomial Transformation

- Polynomial Transformation ($L_1 \alpha L_2$).
- A poly transformation from problem $P_1$ to problem $P_2$ is a function $f$ that transforms any instance of $I_1$ of $P_1$ into an instance $I_2$ of $P_2$ such that
  - function $f$ can be computed in polynomial time wrt size of $I_1$
  - $I_1$ is a “yes” instance of $P_1$ iff $I_2$ is a “yes” instance of $P_2$. 
Example

- Hamiltonian Circuit
  - Instance: \( G = (V, E) \)
  - Does \( G \) contain a HC, i.e., simple circuit that includes all vertices?

- \( HC \propto TSP \)

- Given \( G = (V, E) \) with \( n = |V| \), define \( G' \) as
  - \( v_i \to c_i \) and complete (all edges present) if \( \{v_i, v_j\} \in E \) then \( d\{c_i, c_j\} = 1 \), o.w. 2. \( B \) is \( n \)

\[ \]

- \( f \) takes deterministic polynomial time
- \( G \) contains a HC \( \Leftrightarrow \) there is a tour in \( G' = f(G) \) with length \( \leq B \)
NP-Complete

- $L$ is NP complete if $P_1 \in NP$, and $P_2 \preceq P_1$, for every $P_2 \in NP$.
- “Hardest problem in $NP$” if $P \neq NP$ and $\pi$ is NP-complete then $\pi \in NP - P$.
- Equivalently If $P_2 \in NP$, $P_1$ is NP-complete and $P_1 \preceq P_2$, then $L_2$ is NP-complete.
Satisfiability

- Boolean Variables: $X = x_1, x_2, ..., x_n$.
- Truth assignment function $t : X \rightarrow t, f$.
- $x$ and $\bar{x}$ are literals over $X$.
- Clause over $X$: Set of literals over $X$, e.g., $x_1, \bar{x}_3, x_8$ ("disjunction of literals").
- Clause is satisfied $\iff$ at least one member is true.
- A collection $C$ of clauses over $X$ is satisfied $\iff$ There exists a truth assignment for $X$ that satisfies all clauses in $C$.
- Satisfiability (SAT)
  - Instance: A set $U$ of variables and collection $C$ of clauses over $U$.
  - Question: Is there a satisfying truth assignment for $C$?
Examples

• $x_1, x_2, x_3, x_4, x_5, x_6$.

• $\{x_1, x_2, x_3\}, \{x_2, x_3, \bar{x}_4\}, \{x_4, x_5, x_6\}, \{x_1, x_3, x_5\}$

• Satisfiable ($x_1 = T, x_2 = T, x_4 = T$).

• $x_1, x_2, x_3, x_4$.

• $\{x_1, x_2, x_3\}, \{\bar{x}_1, \bar{x}_2, x_4\}, \{x_1, x_3, \bar{x}_4\}, \{x_1, x_2, x_4\}$

• Satisfiable ($x_1 = T, x_4 = T$).

• $x_1, x_2, x_3$.

• $\{x_1, x_2, x_3\}, \{x_1, x_2, \bar{x}_3\}, \{x_1, \bar{x}_2, x_3\}, \{x_1, \bar{x}_2, \bar{x}_3\}$

• $\{\bar{x}_1, x_2, x_3\}, \{\bar{x}_1, x_2, \bar{x}_3\}, \{\bar{x}_1, \bar{x}_2, x_3\}, \{\bar{x}_1, \bar{x}_2, \bar{x}_3\}$

• Not Satisfiable.
Cook’s Theorem

- Theorem: SAT is NP-complete.