CMSC 28100

Introduction to Complexity Theory

Spring 2025
Instructor: William Hoza

Coping with intractability

The knapsack problem

- Given: Positive integers $w_1, \ldots, w_k, v_1, \ldots, v_k, B$
  - Interpretation: There are k items
  - Item i has weight w_i (in pounds) and value v_i (in dollars)
  - We can carry up to B pounds of stuff in our knapsack
- Goal: Find a set $S \subseteq \{1, 2, \ldots, k\}$ such that $\sum_{i \in S} v_i$ is as large as possible, subject to the constraint $\sum_{i \in S} w_i \leq B$

KNAPSACK is NP-complete

- $\text{KNAPSACK} = \{\langle w_1, \ldots, w_k, v_1, \ldots, v_k, B, V\rangle :$ there exists $S \subseteq \{1, 2, \ldots, k\}$ such that $\Sigma_{i \in S}\, w_i \leq B$ and $\Sigma_{i \in S}\, v_i \geq V\}$

Theorem: KNAPSACK is NP-complete

Knapsack in pseudo-polynomial time

- UNARY-VAL-KNAPSACK $= \{\langle w_1, \ldots, w_k, 1^{v_1}, \ldots, 1^{v_k}, B, 1^V \rangle$: there exists $S \subseteq \{1, 2, \ldots, k\}$ such that $\Sigma_{i \in S} w_i \leq B$ and $\Sigma_{i \in S} v_i \geq V\}$

Theorem: UNARY-VAL-KNAPSACK $\in$ P

Approximation algorithms

- Next approach for coping with intractability: approximation algorithms

Approximation algorithm for Knapsack

- For every $w_1, \dots, w_k, v_1, \dots, v_k, B$, define

$$\mathrm{OPT} = \max\left\{\sum_{i \in S} v_i : S \subseteq \{1, \dots, k\} \text{ and } \sum_{i \in S} w_i \leq B\right\}$$

Theorem: For every $\epsilon > 0$, there exists a poly-time algorithm such that given $w_1, \dots, w_k, v_1, \dots, v_k, B$, the algorithm outputs $S \subseteq \{1, \dots, k\}$ such that $\sum_{i \in S} w_i \leq B$ and $\sum_{i \in S} v_i \geq (1 - \epsilon) \cdot \mathrm{OPT}$

Approximation algorithm for Knapsack

- **Algorithm:** Let $v_i' = \lfloor \alpha v_i \rfloor$, where $\alpha = \frac{k}{\epsilon \cdot \max(v_1, \ldots, v_k)}$, so $v_i' \leq k/\epsilon$
- Output $S \subseteq \{1, \ldots, k\}$ that maximizes $\sum_{i \in S} v_i'$ subject to $\sum_{i \in S} w_i \leq B$
  - Polynomial time, because we can encode v_i' in unary
- **Correctness proof:** Let $S' \subseteq \{1, \ldots, k\}$ be optimal. Then

$$\sum_{i \in S} v_i \geq \frac{1}{\alpha} \sum_{i \in S} v_i' \geq \frac{1}{\alpha} \sum_{i \in S'} v_i' > \frac{1}{\alpha} \sum_{i \in S'} (\alpha v_i - 1) \geq \left(\sum_{i \in S'} v_i \right) - \frac{k}{\alpha} = \mathrm{OPT} - \epsilon \cdot \max(v_1, \ldots, v_k)$$

$$\geq (1 - \epsilon) \cdot \mathrm{OPT}$$

Approximation algorithms are not a panacea

- In some cases, approximation algorithms take some of the sting out of NP-completeness
- However:
  - Approximation is not always applicable
    - E.g., UNDIRECTED-HAM-CYCLE is simply not an optimization problem
  - Even if it's applicable, approximation is not always feasible!

Inapproximability of the clique problem

- For a graph G, let $\omega(G)$ denote the size of the largest clique in G

Theorem: Let $\epsilon > 0$. Suppose there exists a poly-time algorithm such that given a graph $G = (V, E)$, the algorithm outputs a clique $S \subseteq V$ satisfying $|S| \geq \epsilon \cdot \omega(G)$. Then $\mathrm{P} = \mathrm{NP}$.

- (Proof omitted. Not on exercises / exams)

Final exam

- Final exam will be **Wednesday, May 28 from 3pm to 5pm** in STU 105
- The exam is cumulative
- To prepare for the exam, you only need to study the material prior to this point (including exercises 25-28)

Structured inputs

- Let's discuss another approach for coping with $Y \notin \mathrm{P}$
- Idea: Try to identify some additional structure in the instances you care about, beyond the definition of Y
- Example: Initially, you think you need to solve SAT

$$\mathrm{SAT} = \{\langle \phi \rangle : \phi \text{ is a satisfiable CNF formula}\}$$

- SAT is NP-complete 😟

Structured inputs

- However, after studying your situation more closely, you realize that your instances are all “Horn formulas”
- A Horn formula is a CNF formula with at most one positive literal per clause
- $\text{HORN-SAT} = \{\langle \phi \rangle : \phi \text{ is a satisfiable Horn formula}\}$
- Exercise: Prove that $\text{HORN-SAT} \in \text{P}$

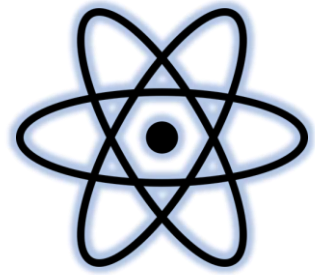
SAT solvers

- Another approach: If your problem is in NP, you can try using a "SAT solver" (practical software for solving SAT)
- For example, many software package managers use SAT solvers to resolve dependencies
- Presumably, the reason this works is that there is some hidden structure in the SAT instances that come up in practice (think Horn formulas)

SAT solvers are not a panacea

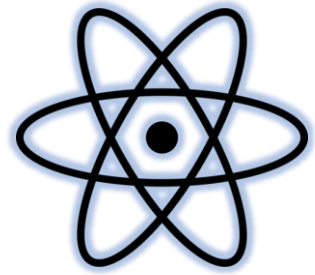
- Note: Despite the practical success of SAT solvers, it is nevertheless conjectured that $\mathrm{P} \neq \mathrm{NP}$
- There are "hard instances" on which practical SAT solvers fail badly
- E.g., good luck using SAT solvers to mine bitcoin…

Quantum computing



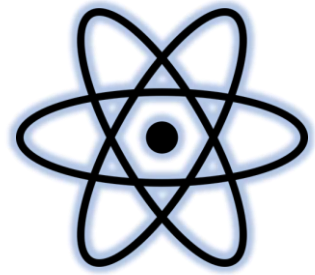
- Another approach for coping with intractability: Quantum Computing
- A quantum computer is a computational device that uses special features of quantum physics
- A detailed discussion of quantum computing is outside the scope of this course
- We will discuss only some key facts about quantum computing

Quantum computing



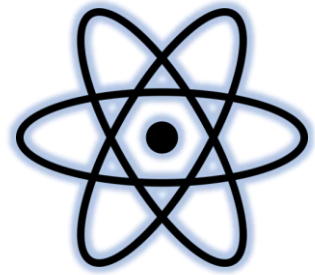
- Quantum computers are, to some extent, hypothetical
- So far, researchers have constructed rudimentary quantum computers
- There are huge ongoing efforts to build fully-functional quantum computers

Quantum complexity theory



- One can define a complexity class, BQP, consisting of all languages that could be decided in polynomial time by a fully-functional quantum computer
- The mathematical definition of BQP is beyond the scope of this course
- One can prove that $\mathrm{BPP} \subseteq \mathrm{BQP} \subseteq \mathrm{PSPACE}$

Shor's algorithm



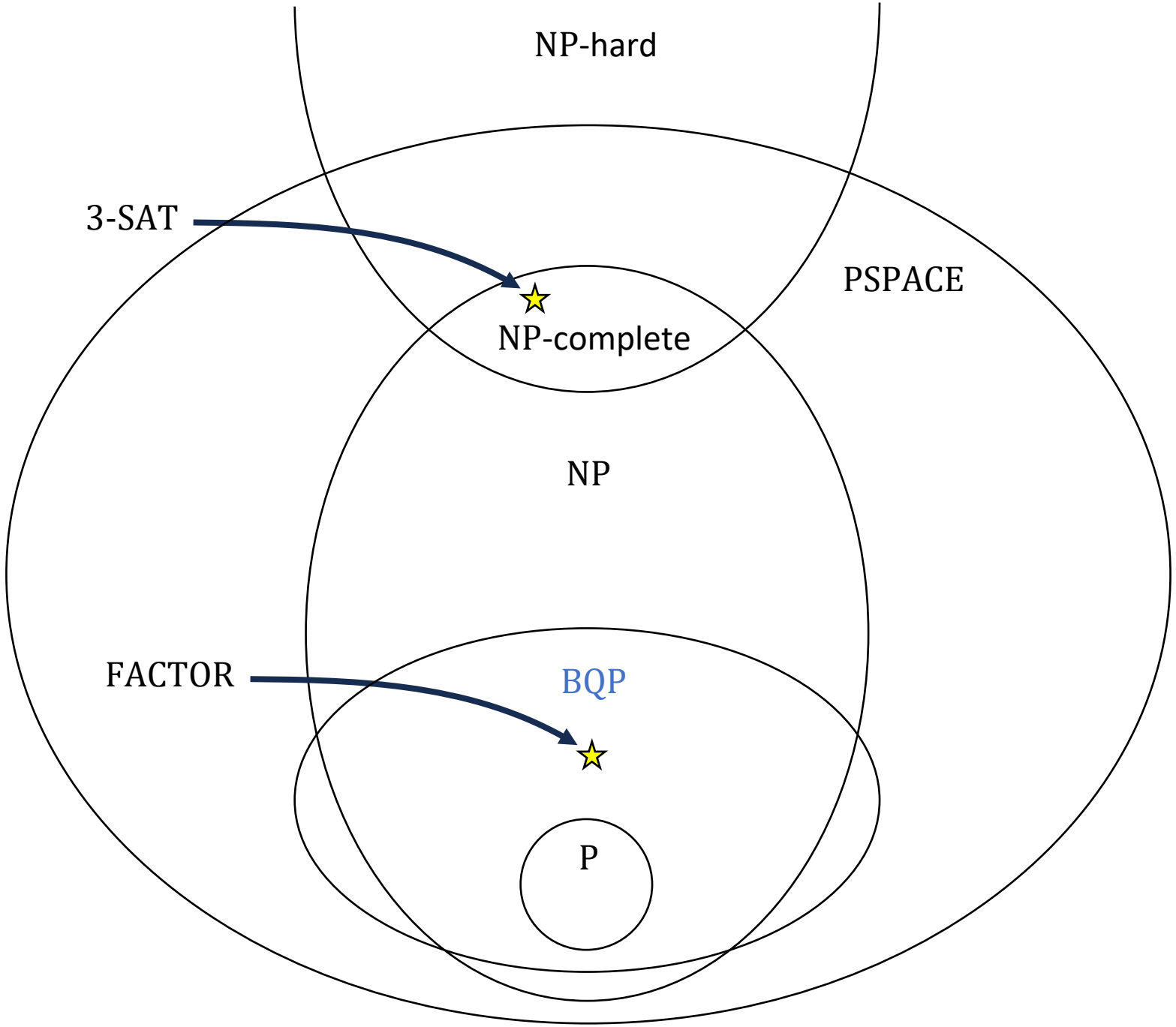
- Recall $\mathrm{FACTOR} = \{\langle N, K\rangle : N \text{ has a prime factor } p \leq K\}$
- **Conjecture:** $\mathrm{FACTOR} \notin \mathrm{P}$

Theorem (Shor's algorithm): $\mathrm{FACTOR} \in \mathrm{BQP}$

- FACTOR is a likely counterexample to the extended Church-Turing thesis!

Quantum computing is not a panacea

- Recall: FACTOR is probably not NP-complete
- In fact, it is conjectured that $\mathrm{NP} \not\subseteq \mathrm{BQP}$
- In this case, even a fully-functional quantum computer would not be able to solve NP-complete problems in polynomial time
- Even quantum computers have limitations


NP-hard
3-SAT
PSPACE
NP-complete
NP
FACTOR
BQP
P

Limitations of quantum computers

- We have developed several techniques for identifying hardness
  - Undecidability
  - EXP-completeness
  - NP-completeness
- Those techniques are all still applicable even in a world with fully-functional quantum computers!
- Complexity theory is intended to be “future-proof” / “timeless”

Which problems

can be solved

through computation?

~~CLASSICAL~~