NP Hardness/Completeness
Overview

Ron Parr
CPS 170

Digression: NP-Hardness

• NP hardness is not an AI topic
• You will not be tested on it explicitly, but

• It’s important for all computer scientists
• Understanding it will deepen your understanding of AI (and other CS) topics
• You will be expected to understand its relevance and use for AI problems

• Eat your vegetables; they’re good for you
## P and NP

- P and NP are about decision problems
- P is set of problems that can be solved in polynomial time
- NP is a superset of P
- NP is the set of problems that:
  - Have solutions which can be verified in polynomial time or, equivalently,
  - can be solved by a non-deterministic Turing machine in polynomial time (OK if you don’t know what that means yet)
- Roughly speaking:
  - Problems in P are tractable – can be solved in a reasonable amount of time, and Moore’s law helps
  - Problems in NP might not be tractable

## NP-hardness

- Many problems in AI are NP-hard (or worse)
- What does this mean?
- These are some of the hardest problems in CS
- Identifying a problem as NP hard means:
  - You probably shouldn’t waste time trying to find a polynomial time solution
  - If you find a polynomial time solution, either
    - You have a bug
    - Find a place on your shelf for your Turing award
- NP hardness is a major triumph (and failure) for computer science theory
Understanding the class NP

- A class of decision problems (Yes/No)
- Solutions can be verified in polynomial time
- Examples:
  - Graph coloring:
  - Sortedness: [1 2 3 4 5 8 7]

What is NP hardness?

- An NP hard problem is at least as hard as the hardest problems in NP
- The hardest problems in NP are NP-complete
- Demonstrate hardness via reduction
  - Use one problem to solve another
  - A is reduced to B, if we can use B to solve A:

  A instance → Poly-time xformation → B Solver

  poly time A solver if B is poly time
Hardness vs. Completeness

- For something to be NP-complete, must be in NP
- If something is NP-hard, it could be even harder than the hardest problems in NP
- Proving completeness is stronger theoretical result – says more about the problem

Why care about NP-completeness?

- Solving any one NP-complete problem gives you the key to all others
- All NP-complete problems are, in a sense, equivalent
- Insight into solving any one gives you insight into solving a vast array of problems of extraordinary practical and economic significance
The First NP Complete Problem  
(Cook 1971)

• SAT:

\[(X_1 \lor \overline{X}_7 \lor X_{13}) \land (\overline{X}_2 \lor X_{12} \lor X_{25}) \land \ldots\]

• Want to find an assignment to all variables that makes this expression evaluate to true
• NP-complete for clauses of size 3 or greater
• How would you prove this?

Hardness w/o completeness?

• NP hardness is a weaker claim (says less about the problem) than NP completeness, but
• NP hard problems might be harder than NP-complete
• NP hard if an NP complete problem is reducible to it
• NP completeness = NP hardness + NP membership
• Consider the problem #SAT
  – How many satisfying assignments to:
    \[(X_1 \lor \overline{X}_7 \lor X_{13}) \land (\overline{X}_2 \lor X_{12} \lor X_{25}) \land \ldots\]
  – Is this in NP? (Not even a decision problem)
  – Is it NP-hard?
#SAT is NP-hard

- Theorem: #SAT is NP hard
- Proof:
  - Reduce SAT to #SAT

\[ \begin{array}{c}
\text{SAT instance} \\
\rightarrow \\
\text{#SAT solver} \quad x \\
\rightarrow \\
\text{If } x > 0 \\
\text{return Y} \\
\text{Else} \\
\text{return N} \\
\end{array} \]

SAT Solver

P=NP?

- Biggest open question in CS
- Can NP-complete problems be solved in polynomial time?
- Probably not, but nobody has been able to prove it yet
How challenging is “P=NP?”

• Princeton University CS department

How To Avoid Embarrassing Yourself

• Don’t say: “I proved that it requires exponential time.” if you really meant:
  • “I proved it’s NP-Hard/Complete”

• Don’t say: “The problem is NP” (which doesn’t even make sense) if you really meant:
  • “The problem NP-Hard/Complete”

• Don’t reduce new problems to NP-hard complete problems if you meant to prove the new problem is hard
• Such a reduction is backwards. What you really proved is that you can use a hard problem to solve an easy one. Always think carefully about the direction of your reductions
NP-Completeness Summary

• NP-completeness tells us that a problem belongs to class of similar, hard problems.
• What if you find that a problem is NP hard?
  – Look for good approximations
  – Find different measures of complexity
  – Look for tractable subclasses
  – Use heuristics