IE418: Integer Programming

Jeff Linderoth

Department of Industrial and Systems Engineering
Lehigh University

21st February 2005

Please Don't Call On Me!

Any questions on the homework?

- Problem #5: Linear Ordering Problem. Some sample code on the course web page.
- Run the code with `minto -s`.

- How do we convert an optimization problem to a decision problem?
- What is "big O"?
- What does it mean when we say an algorithm is *polynomial*?
- Do you know of a polynomial algorithm for the knapsack problem?

Computational Complexity

- The ingredients that we need to build a theory of computational complexity for problem classification are the following
  - A class $\mathcal{C}$ of problems to which the theory applies
  - A (nonempty) subclass $\mathcal{C}_{\mathcal{E}} \subseteq \mathcal{C}$ of "easy" problems
  - A (nonempty) subclass $\mathcal{C}_{\mathcal{H}} \subseteq \mathcal{C}$ of "hard" problems
  - A relation $\lhd$ "not more difficult than" between pairs of problems
- Our goal is just to put some definitions around this machinery
  - Thm: $Q \in \mathcal{C}_{\mathcal{E}}, P \lhd Q \Rightarrow P \in \mathcal{C}_{\mathcal{E}}$
  - Thm: $P \in \mathcal{C}_{\mathcal{H}}, P \lhd Q \Rightarrow Q \in \mathcal{C}_{\mathcal{H}}$

Recall the Formal Definitions

- Given a problem P, and algorithm A that solves P, and an instance X of problem P.
  - $L(X) \equiv$ The length (in a reasonable encoding) of the instance
  - $f_A(X) \equiv$ the number of elementary calculations required to run algorithm A on instance X.
  - $f_A^*(l) \equiv \sup_X \{f_A(X) : L(X) = l\}$ is the *running time* of algorithm A
- If $f_A^*(l) = O(l^p)$ for some positive constant integer p, A is **polynomial**

The problem class $\mathcal{NP}$

- $\mathcal{NP} \neq$ "Non-polynomial"
- $\mathcal{NP} \equiv$ the class of decision problems that can be solved in polynomial time on a non-deterministic Turing machine.
- What the Heck!!?!?!?!?!?!?!?!?
- $\mathcal{NP} \approx$ the class of decision problems with the property that for every instance for which the answer is "yes", there is a short certificate
- The certificate is your "proof" that what you are telling me is the truth

$\mathcal{NP}$: Examples

Example: 0-1IP

$\exists x \in \mathbb{B}^n$ such that $Ax \leq b, c^T x \geq K$?

1. You say the answer is "Yes". I say "prove it."
2. You give me the vector x: This is a "short certificate"
3. The 0-1 vector x can be checked such that $Ax \leq b, c^T x \geq K$?

$\mathcal{NP}$: Examples

Example: Hamiltonian Circuit

Instance: Graph $G = (V, E)$
Question: Does G contain a Hamiltonian Circuit?

- You say the answer is "Yes". I say "prove it."
- You give me the a set of edges $E' \subseteq E$. I check as follows:
  1. Does the degree of each node of $G' = (V, E') = 2$? If not, then return **no**, else go to 2.
     - This takes time $\leq O(|V|^2)$.
  2. Is $G' = (V, E')$ connected. If so, return **yes**, otherwise return **no**.
     - This takes time $O(|E'|)$
- The checking algorithm takes $O(|V|^2 + |E'|)$ time, so it is polynomial. It returns **yes** if and only if the set of edges E' defines a Hamiltonian Circuit in G, so Hamiltonian Circuit $\in \mathcal{NP}$.

$\mathcal{NP}$: Examples

Example: Complement of Hamiltonian Circuit

Instance: Graph $G = (V, E)$
Question: Does G *not* contain a Hamiltonian Circuit?

- You say the answer is "Yes". I say "prove it."
- Equivalently, you say that the answer to Hamiltonian Circuit on G is **no**.
- You give me... ?
  - Careful: Will your answer suffice for *all* graphs G?
  - What you really are giving would be a *characterization* of what graphs are *not* Hamiltonian. G is *not* Hamiltonian if and only if Your Answer.
- No one knows!

The Class co-$\mathcal{NP}$

- The class of problems for which the "complement" problem to P is $\in \mathcal{NP}$
- co-$\mathcal{NP} \approx$ the class of decision problems with the property that for every instance for which the answer is "no", there is a short certficate

Example: 0-1 IP

$\exists x \in \mathbb{B}^n$ such that $Ax \leq b, c^T x \geq K$?

1. You say "no." I say "prove it."
2. You give me what? Is this a short (polynomial length) certificate?

co-$\mathcal{NP}$, More examples

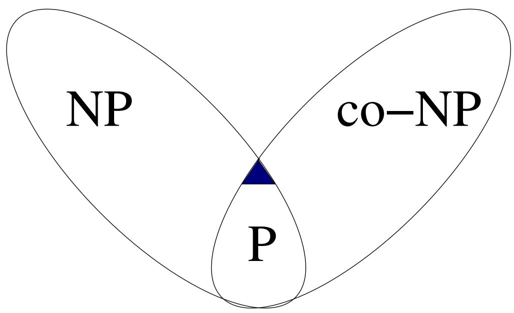
LP

$\exists x \in \mathbb{R}^n_+$ such that $Ax \leq b, c^T x \geq K$?

1. You say "no." I say "prove it."
2. You give me What?.
3. Hint: (x, π) is optimal if and only if $Ax \leq b, x \geq 0, \pi^T A \geq c, \pi \geq 0, c^T x = b^T \pi$
4. $\pi \in \mathbb{R}^m \mid \pi^T A \geq c, \pi \geq 0, \pi^T b < K \Rightarrow \nexists x \in \mathbb{R}^n \mid Ax \leq b, x \geq 0, c^T x \geq K$
5. Is π a short certificate?

The Class $\mathcal{P}$

- $\mathcal{P}$ is the class of problems for which there exists a polynomial algorithm.
- $\mathcal{P} \in \mathcal{NP} \cap$ co-$\mathcal{NP}$



- It is a (very significant) open question as to whether $\mathcal{P} = \mathcal{NP} \cap$ co-$\mathcal{NP}$.
- There are (very few) problems in $\mathcal{NP} \cap$ co-$\mathcal{NP}$ but not (known) to be in P.
  - LP
  - PRIMES
  - Approximating the shortest and closest vector in a lattice to within a factor of $\sqrt{n}$

Where are we?

- We have our class(es) of problems $\mathcal{P}, \mathcal{NP}$, co-$\mathcal{NP}$
- We know class of "easy" problems. (Problems in $\mathcal{P}$)
- We need our class of "hard" problems.
- We need our relation "not (significantly) more difficult than" ($\lhd$)
  - For this we need the concept of problem reductions.

Polynomial Reduction

- If problems $P, Q \in \mathcal{NP}$, and if an instance of P can be converted in polynomial time to an instance of Q, then P is *polynomially reducible* to Q.
  - This is the "not (substantially) more difficult than" relation that we want to use.
  - We will write this as $P \lhd Q$
- Depending on time, I will show a couple reductions here.
  - How could we reduce the assignment problem to a max weighted matching on a bipartite graph?
  - How could we reduce the knapsack to a longest path problem?

The "Hard Problems"—Class $\mathcal{NPC}$

- We want to ask the question—What are the hardest problems in $\mathcal{NP}$?
  - We'll call this class of problems $\mathcal{NPC}$, "$\mathcal{NP}$-Complete".
- Using the definitions we have made, we would like to say that if $P \in \mathcal{NPC}$, then $Q \in \mathcal{NP} \Rightarrow Q \lhd P$
  - If $P \in NP$ and we can convert in polynomial time *every* other problem $Q \in NP$ to P, then P is in this sense the "hardest" problem in $\mathcal{NP}$. $P \in \mathcal{NPC}$
- Is it obvious that such problems exist?
  - **No!** – We'll come to this later...
- Thm: $Q \in \mathcal{P}, P \lhd Q \Rightarrow P \in \mathcal{P}$
- Thm: $P \in \mathcal{NPC}, P \lhd Q \Rightarrow Q \in \mathcal{NPC}$

$\mathcal{P} = \mathcal{NP}$?

- You may be tired of only winning \$1 at a time by answering my questions in class. Here's you chance to win *big* bucks.
    - We've seen some problems in $\mathcal{P}$, and we've seen some problems in $\mathcal{NP}$.
    - We know that $\mathcal{P} \subseteq \mathcal{NP}$.
    - Have we seen any problems in $\mathcal{NP} \setminus \mathcal{P}$?
    - Do such problems exist?
    - No one knows for sure!
- If you can answer this, you will one million dollars!
- `www.claymath.org/Millennium_Prize_Problems/P_vs_NP/`
- I will also give you an A++++++++++++ in the class if you write my name on the paper. `:-)`
- Maybe you can even be on TV: `http://story.news.yahoo.com/news?tmpl=story&u=/usatoday/20050209/ts_usatoday/getoutapieceofpaperandapencil`

The Satisfiability Problem

- This is the first problem to be shown to be NP-complete.
- The problem is described by
    - a finite set $N = \{1, \ldots, n\}$ (the *literals*), and
    - m pairs of subsets of N, $C_i = (C_i^+, C_i^-)$ (the *clauses*).
- An instance is feasible if the set

$$\left\{ x \in \mathbb{B}^n \mid \sum_{j \in C_i^+} x_j + \sum_{j \in C_i^-} (1 - x_j) \geq 1 \ \forall i = 1, \ldots, m \right\}$$

  is nonempty.
- This problem is in $\mathcal{NP}$. **Why?**
- In 1971, Cook defined the class $\mathcal{NP}$ and showed that satisfiability was NP-complete.
- We will not attempt to understand the proof

Proving $\mathcal{NP}$-completeness

- Once we know that satisfiability is $\mathcal{NP}$-complete, we can use this to prove other problems are $\mathcal{NP}$-complete using the "reduction theorem":
  - $P \in \mathcal{NPC}, P \lhd Q \Rightarrow Q \in \mathcal{NPC}$
- Let's prove that **Node Packing** is NP-Complete.
- **Node Packing:**
  - Given a graph $G = (V, E)$ and an integer k
  - Does $\exists\ U \subseteq V$ such that $|U| \geq k$ and U is a *node packing*. $(u \in U \Rightarrow v \notin U\ \forall v \in \delta(u))$

Reduction

Your writing here...

How to Win $1M

- Here's a hint
- Thm: If $P \cap \mathcal{NPC} \neq \emptyset \Rightarrow \mathcal{P} = \mathcal{NP}$
    - Proof: Let $Q \in \mathcal{P} \cap \mathcal{NPC}$ and take $R \in \mathcal{NP}$.
    - $R \lhd Q$
    - $Q \in \mathcal{P}, R \lhd Q \Rightarrow R \in \mathcal{P}$
    - $\mathcal{NP} \subseteq \mathcal{P} \Rightarrow \mathcal{P} = \mathcal{NP}$

QUITE ENOUGH DONE

- To prove $\mathcal{P} = \mathcal{NP}$, you only need to find a polynomial algorithm for any problem that has shown to be $\mathcal{NP}$-complete
    - How good are you at Minesweeper? :-)

The Line Between $\mathcal{P}$ and $\mathcal{NPC}$

- The line between these two classes is very thin!
- Consider a 0-1 matrix A an integer k defining the decision problem

$$\exists \; \{x \in \mathbf{B}^n \mid Ax \leq e, e^T x \geq k\}?$$

    - If we limit the number of nonzero entries in each column to 2, then this problem is known to be in $\mathcal{P}$!
    - If we allow the number of nonzero entries in each column to be 3, then this problem is $\mathcal{NP}$-complete!
    - If we allow at most one '1' per row, the problem is in $\mathcal{P}$
    - If we allow two '1's per row, it is in $\mathcal{NPC}$
- Shortest Path (with non-negative edge weights) is in $\mathcal{P}$.
- Longest Path (with non-negative edge weights) is in $\mathcal{NPC}$

$\mathcal{NP}$-hard Problems

- The class $\mathcal{NP}$-hard extends $\mathcal{NPC}$ to include problems that are not in $\mathcal{NP}$
- If $P \in \mathcal{NPC}$ and $Q \lhd P$, Q is *NP-Hard*
- Thus, all NP-complete problems are NP-hard.
- If a problem P is in $\mathcal{NP}$ and is $\mathcal{NP}$-hard, then $P \in \mathcal{NPC}$
- The primary reason for this definition is so we can classify optimization problems that are not in $\mathcal{NP}$
- It is common for people to refer to optimization problems as being $\mathcal{NP}$-complete, but this is technically incorrect.

Theory versus Practice

- In practice, it is true that most problem known to be in $\mathcal{P}$ are "easy" to solve.
- This is because most known polynomial time algorithms are of relatively low order.
- It seems very unlikely that $\mathcal{P} = \mathcal{NP}$
- Although all NP-complete problems are "equivalent" in theory, they are not in practice.
- TSP vs. QAP
  - TSP—Solved instances of size ≈ 17000
  - QAP—Solved instances of size ≈ 30