Engs 104, W2010
Optimization
George Cybenko

Today Complexity hierarchy

Examples,

3-Satisfiability Problem

$x_1, \ldots, x_n$ are Boolean variables (T, F)

$\bar{x}_i$ is negation

$\cdot, \wedge$ is conjunction (and)

$+, \vee$ is disjunction (or)

$\cancel{(x_i +}$

$$P = \underbrace{(x_1 + \bar{x}_2 + x_5) \cdot (x_2 + \bar{x}_4 + \cancel{x}_5) \cdot \ldots}_{\text{Conjunction of disjunctions with 3 terms that are } x \text{ or } \bar{x}}$$

$$P = (x_1 + \bar{x}_2 + x_5)(x_2 + \bar{x}_4 + x_5) \cdots$$

Is there an assignment of T, F to x_i so that P is T?

Observe: P is T if all disjunctions are T.

Let $x_i = 0$ or 1

$$\bar{x}_i = 1 - x_i \quad (= 0 \text{ or } 1)$$

$x_i + \bar{x}_j + x_k \geq 1$ means T

Engs104, L14

#4

So $P = T$ if

Integer LP
$$\begin{cases} x_i + \overline{x}_j + x_k \geq 1 \\ \vdots \\ \text{list all disjunctions} \\ \text{and} \\ x_i = 0 \text{ or } 1 \end{cases}$$

So we can solve 3-Sat if we can find <u>any</u> feasible solution to constraints. Does not matter what min is. Eg min $x_1 + x_2$ works.

Engs 104, L14

Combinatorial Optimization next

Complexity Hierarchy

Hardest	Intractable/Undecidable
	Exponential
	NP-Complete
Easiest	Solve in polynomial time

Complexity of an <u>algorithm</u> vs complexity of a problem.

Note: Many ways to solve a problem. Complexity of an <u>algorithm</u> is a measure of how many "operations" it requires (+, −, ×, ÷, if, for, etc)

Suppose a problem requires n bits to specify. The complexity of an algorithm for solving the problem is # operations (as a function of n) in <u>worst</u> case.

Example

Matrix Multiplication

Two $m \times m$ matrices require

$n = 2m^2$ entries to be specified

Use K bits for each entry

so $n = 2m^2 K$ really

Usual matrix multiplication requires

$\approx m^3 K \log K \le n^2$ operations

Polynomial in n

An <u>algorithm</u> has "polynomial" complexity if it uses $\leq$ polynomial in n operations to solve the problem, <u>worst</u> case.

An <u>algorithm</u> has "exponential" complexity if it uses $\geq \alpha 2^{\beta n}$ operations to solve ~~an~~ ~~order~~ a problem of size n, worst case.

Example: sort a list of n numbers.
Quicksort $n \log n$, Bubblesort n^2, Enumerate $n! \geq 2^n$

The complexity of a problem is the complexity of the most efficient algorithm for solving <u>all</u> instances of the problem.

<u>Caution</u>: LP has polynomial complexity but Simplex Algorithm has exponential complexity (in worst case). There exist polynomial complexity algorithms for LP.

Pages 104, 114

A problem has polynomial complexity if there exists a polynomial algorithm for solving it (worst case and in all instances).

The class of polynomial complexity ~~algorithms~~ problems is called "P".

Eg Matrix multiplication, solution of linear systems, LP, sorting etc.

Curiosity The actual complexity of matrix multiplication is not known (yet).

Example 2×2 matrix multiply by usual algorithm uses 8 scalar multiplies. Strassen algorithm uses only 7 multiplies. Minimal # of mults is not known in general!

Caution: Real arithmetic vs <u>integer</u> or <u>rational</u> arithmetic must be specified. IE a "real" number requires ∞ bits to specify. Integers grow in size when added or multiplied, etc. Typically, people talk about rational/integer complexity.

Informal definition of the class NP (Nondeterministic Polynomial).

Ingredients:

1. Problem stated in n bits
2. Problem is a "decision problem": has 0,1 or T,F answer
3. "Guessing" a solution by oracle but checking it is in P.

<u>Example</u> 3-Sat is in NP

because if we given a solution, (where solution is produced by an "oracle"), then checking can be done in polynomial time by just evaluating it.

<u>Theorem</u> (Cook, 1972) Any problem in NP can be reduced/expressed (using polynomial resources) as an instance of 3-SAT.

<u>Consequence</u> Any "efficient" solution to 3-SAT, say polynomial, will provide an efficient solution to all problems in NP. So 3-SAT is as hard as any NP problem.

<u>Definition</u> A problem is NP-Complete if

a) It is in NP

b) 3-SAT can be (efficiently) expressed as that problem.

So Integer Linear Programming is NP-Complete. Why?

Engs 104, L14

Examples of NP-Complete problems

- 3-SAT
- Integer LP
- 0-1 Integer LP
- Travelling Salesman Problem
- Hamiltonian Circuit
- Partition Problem
- 1000's more . . .

Problems in NP but not known if NP-Complete or not.

1. Graph Isomorphism
2. Integer Factorization

What are some problems known to be harder than NP Complete.

⇒ Presburger Arithmetic (decidable but exponentially hard, provably).

⇒ Some problems are "undecidable" – ie not algorithms are even possible!

Undecidable Problems

- ⇒ Halting Problem
- ⇒ Tensor rank
- ⇒ Spectral radius of matrices.