Efficient computational proofs and inapproximability

Johan Håstad

Royal Institute of Technology
Stockholm, SWEDEN

My area: Complexity Theory

The goal: To design and analyze efficient algorithms for computational problems.

or

To prove that none exist.

Outline

- Quick review of classical concepts

- Some results on approximability

What is a computer?

Formally: Turing machine.

Informally: "Ordinary" computer with fixed word length.

Time = # of elementary operations.

Efficient algorithms

In practice: Anything you can execute.

In theory: Running in time $O(n^c)$ on inputs of size n for constant c.
Polynomial time, P.

Size: Number of bits to specify input.

Computational Problem

Formally: Any mapping from finite binary strings to finite binary strings.

Informally: What you expect. Avoid real numbers with arbitrary precision and make it mathematically well defined.

The complexity class NP

Decision problems where positive instances have proofs that can be verified efficiently.

Example NP; 3SAT

Given a Boolean formula

$$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4)$$

is it satisfiable?

Proof: A satisfying assignment

$$x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 1$$

NP $\neq$ P??

Unknown but strongly believed. A million dollar Clay-problem and very few people are working on it. No real progress the last 10 years.

Basic problem for proving lower bounds

It is very difficult to prove that any (crazy) algorithm that runs fast does not solve the problem we are studying.

Complexity theory, grade school

Is multiplication harder than addition?

Addition of n-bit integers doable in time $O(n) = cn$ for constant c.

Multiplication of n-bit integers doable in time $O(n \log n \log \log n)$, can it be done in time $O(n)$?

Grade school answer

We do not know!

Cannot rule out a general linear time algorithm for multiplication.

Best multiplication algorithm

Works by doing a Fourier transform of each input in a finite ring.

If we did not know about Fourier transforms we would rate this a crazy algorithm and doubt it was solving the problem.

What can complexity theory do?

- Design and analyze efficient algorithms.
- Prove unconditional lower bounds for really hard problems.
- Compare difficulty of problems.

Comparing difficulties?!

If we can solve satisfiability in polynomial time then we can solve any problem in NP in polynomial time.

Satisfiability is NP-complete and cannot be solved in polynomial time unless $P = NP$.

Comparing difficulties II

If satisfiability on instances of size m requires time $T(m)$.

then

Finding largest independent set in graphs with m nodes requires time $T(m) - m^2$.

Theory gives conclusions even if satisfiability is solvable in time $O(n^{64})$.

A way to think

Assume $NP \neq P$ and even that all NP-complete problems require exponential time.

Can make sense of some statements even in other cases.

NP-theory for decision problems

Almost all problems in NP are classified as either NP-complete or in P.

Exceptions: Integer factorization, discrete logarithms in finite fields, graph isomorphism.

Classification almost complete.

NP-complete problems

We have thousands of NP-complete problems, some of the most famous are:

- Satisfiability.
- Independent set problem in graphs.
- Graph-coloring.
- Traveling salesman problem.
- Integer linear programming.

Finding suboptimal solutions efficiently

If there is an independent set of size S, maybe we efficiently can find an independent set of size $S/2$?

Approximation algorithms

C-approximation

Maximization problems:

Found value $\geq$ (Optimal value)/ C.

Minimization problems:

Found value $\leq$ C(Optimal value).

This is required for all inputs!

Over-determined linear equations

$$\left\{\begin{array}{ccccccccc} x_1 & + & x_2 & + & x_3 & & & = & 1 \\ x_1 & + & x_2 & & & & & = & 1 \\ x_1 & + & x_2 & + & & & x_4 & = & 1 \\ & & x_2 & + & & & x_4 & = & 0 \\ x_1 & + & & & x_3 & + & x_4 & = & 0 \\ & & x_2 & + & x_3 & + & x_4 & = & 1 \\ x_1 & + & & & x_3 & & & = & 0 \end{array}\right. \quad \text{mod } 2$$

Satisfy as many equations as possible or ($\geq K$).

Max-Lin-2, m equations n variables.

Obvious algorithms

Try to satisfy all using Gaussian elimination.

Time $\approx mn^2$.

Try all possible assignments and take the best.

Time $\approx mn2^n$.

Easy heuristic

Pick values for variables randomly.

Satisfies $m/2$ equations on average.

Simple fact

Theorem: Max-Lin-2 in its decision form is NP-complete.

Consequence: We cannot hope to have a polynomial time algorithms that always finds the best solution.

Simple theorem

Theorem: Max-Lin-2 admits an efficient 2-approximation algorithm.

Proof: We can satisfy $m/2$ equations and the optimum satisfies at most all m equations.

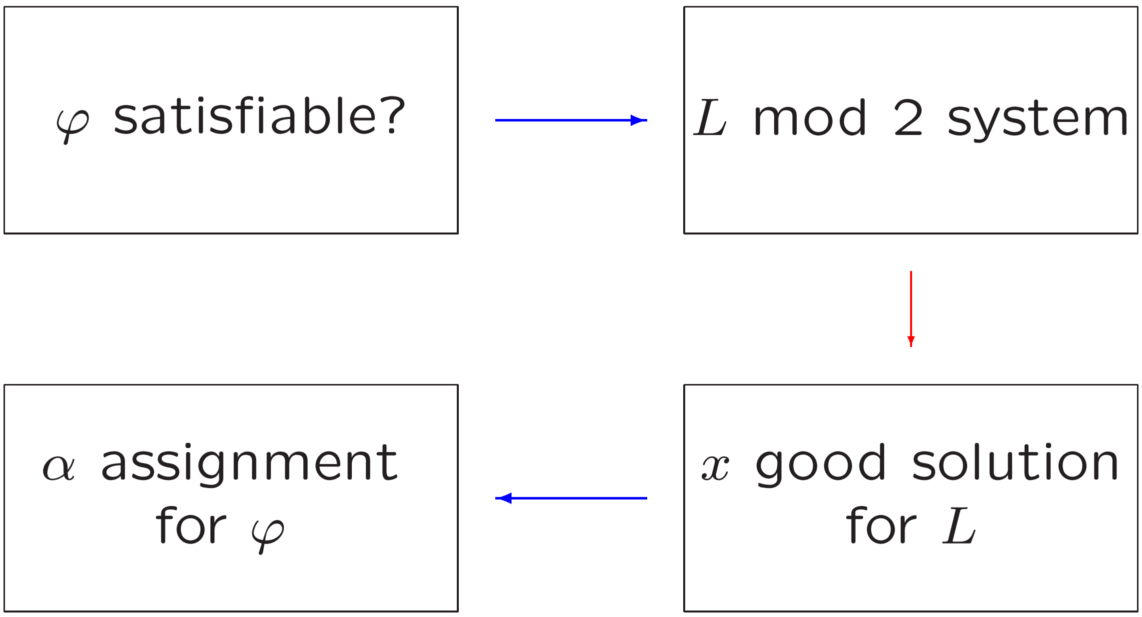
First real theorem

Theorem: [Hå97] $\forall\delta > 0$, unless NP=P, Max-Lin-2 does not admit an efficient (2-δ)-approximation algorithm.

Even when we allow only 3 variables in each equation.

Proof outline

Use a supposed good approximation algorithm to solve an NP-hard problem like SAT in polynomial time.



Probabilistically Checkable Proofs (PCP)

Statement to be verified: Formula φ is satisfiable.

A written proof π, Verifier: V.

Completeness c: If indeed φ is satisfiable then exists π that convinces V with probability c.

Soundness s: If φ is not satisfiable then no π convinces V with probability $\geq s$.

Normally: $c = 1$ and s is a small constant.

A comparison

Traditional proofs: Perfect completeness and soundness ($c = 1$ and $s = 0$) and the verifier reads the entire proof.

New situation: We want reasonable values of c and s when the verifier only reads very small portion of the proof, like 3 bits.

PCP optimization problem

Given a formula φ find the proof π that maximizes the probability that the verifier accepts.

Easy observation: If we can solve this optimization problem within a factor c/s, then we can determine whether φ is satisfiable!

Design the PCP to make this optimization problem equal to Max-Lin-2!

The PCP-theorem [ALMSS]

Is φ satisfiable?

There is a PCP which

- Reads a constant number of bits

- Always accepts a correct proof for a correct statement

- Rejects any proof of a false statement with probability $\geq \frac{1}{2}$

Combinatorial PCP-theorem

There is a polynomial time transformation f taking 3CNF to 3CNF such that:

φ satisfiable	$f(\varphi)$ satisfiable
φ not satisfiable	no assignment satisfies fraction $\geq 9/10$ of clauses of $f(\varphi)$

Amplification of unsatisfiability!

Efficient proofs

φ satisfiable?

Proof: an assignment that satisfies $f(\varphi)$.

Verification: pick 7 random clauses and check that they are satisfied.

Perfect completeness, soundness $(9/10)^7 \leq 1/2$, reads 21 bits.

Components of proof

Coding by polynomials, extending range to complete field. Recursion

Analyze by discrete Fourier transform.

For best inapproximability constants, special inner code.

The long code [BGS]

To code $x \in \{0,1\}^v$ write down $f(x)$ for any $f : \{0,1\}^v \mapsto \{0,1\}$.

We code v bits by 2^{2^v} bits.

Extremely useful for verifying since any property of x can be read in one bit. Useful for very small v.

Theorem again

Theorem: $\forall\delta > 0$, it is NP-hard to (2-δ)-approximate Max-Lin-2.

Even when we allow only 3 variables in each equation.

Remains true in any group, inapproximability is almost size of group.

Approximation properties of other NP-hard optimization problems

Problems do come in different flavors.

Some hardness results follow from old reductions. Sharp results often require special purpose PCPs.

Algorithmic result are special purpose. Key new technique: Semidefinite programming [GW94].

Other problem 1: Max-3-SAT

Problem: Given a 3-CNF formula satisfy the maximal number of clauses.

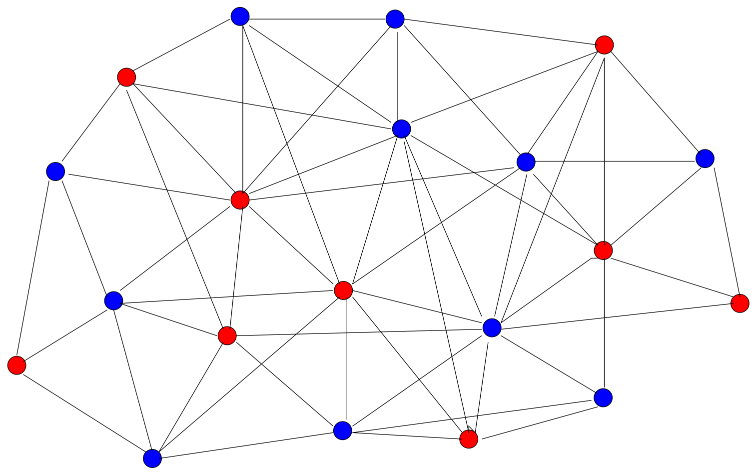
$$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4)$$

Upper bound: 8/7 . Easy when all clauses are of length exactly 3 [J74], otherwise semi-definite programming [KZ99].

Lower bound: $8/7 - \epsilon$. Reduction from Max-Lin-2.

Other problem 2: Max-Cut

Problem: Given a graph G partition the vertices into two sets to cut the maximal number of vertices.



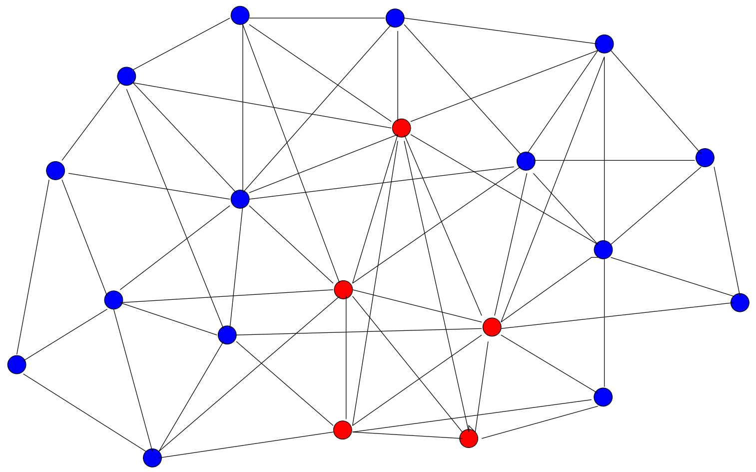
Trivial upper bound: 2 . A random partition cuts half the edges.

Good upper bound:
$\max_\theta \frac{\pi(1-\cos\theta)}{2\theta} \approx 1.138$. Semidefinite programming [GW94].

Lower bound: $17/16 - \epsilon$. Reduction from Max-Lin-2 found by computer, verified by hand.

Other problem 3: Clique (Independent set)

Problem: Given a graph G with n vertices, find the largest number of pairwise connected vertices.

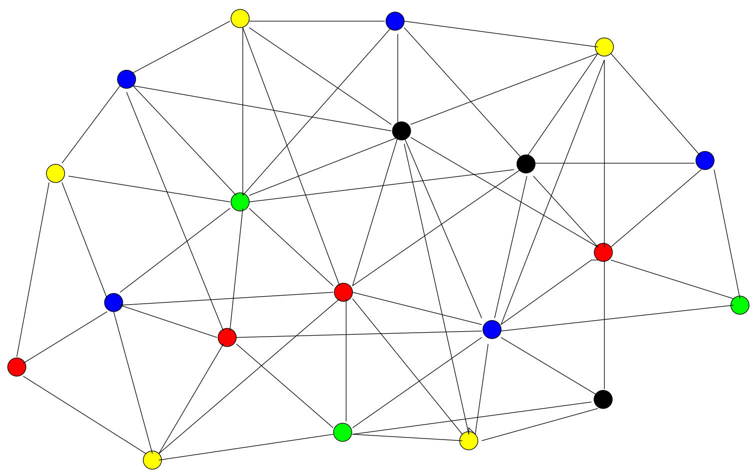


Upper bound: $O\left(\frac{n}{(\log n)^2}\right)$. Combinatorial algorithm removing subgraphs.[BH92]

Lower bound: $n^{1-\epsilon}$. Special purpose PCP [Hå96].

Other problem 4: Graph-coloring

Problem: Given a graph G with n vertices, color the vertices with the minimal number of colors



Upper bound: $O\left(\frac{n(\log\log n)^2}{(\log n)^3}\right)$. Combinatorial algorithm. [Ha93]

Lower bound: $n^{1-\epsilon}$. An additional idea on top of clique lower bound. [FK96]

Other problem 5: Set-cover

Problem: Given a collection of subset of $[n]$, cover the union with as few sets as possible.

$$(1,2,3),(4,5,6)(7,8,9),(1,4,7,8),$$
$$(2,5,9),(3,5,8),(2,6,7)$$

Upper bound: $\ln n + 1$. Greedy algorithm [J74].

Lower bound: $(1 - o(1)) \ln n$. Special purpose PCP [F96].

Other problem 6: Traveling Salesman Problem

Problem: Given n "cities" find the shortest tour that visits all.

Upper bound: Euclidean plane $(1+\epsilon)$ Geometric dynamic programming [A96]. Only Δ-inequality, 3/2. Based on maximal spanning tree [C76].

Lower bound: Cannot be approximated without the Δ-inequality. With Δ-inequality ≈ 1.005 [KP0?].

Open Problems

How many colors are needed to efficiently color a 3-colorable graph?

Upper bound $\tilde{O}(n^{3/14})$, lower bound 5.

How hard is it to approximate TSP on asymmetric graphs?

Upper bound $O(\log n)$, lower bound ≈ 1.01.

Final words

Classification of approximation problems getting more complete.

Uses very efficient proofs that are checked probabilistically.