Solving systems of linear equations over finite fields

Johan Håstad

KTH Numerical Analysis
and Computer Science

June 1, 2007

Basic problem

$$\begin{cases} x_1 + x_2 + x_3 = 1 \\ x_1 + x_2 = 1 \\ x_1 + x_2 + x_4 = 1 \\ x_2 + x_4 = 0 \\ x_1 + x_3 + x_4 = 0 \\ x_2 + x_3 + x_4 = 1 \\ x_1 + x_3 = 0 \end{cases} \mod 2$$

Satisfy as many equations as possible (or $\geq K$).
Max-k-Lin-2, m equations n variables and k variables in each equation. $(m = 7, n = 4, k = 3)$.
Abstract claimed that I would discuss general p instead of 2 but this will happen only briefly at the end.

My angle: efficient computation

- Efficient in theory: Time polynomial in the input size. Used most of the time. Input size mk.
- Efficient in practice. What we can run on our machines, about $2^{50} \approx 10^{15}$ operations and $2^{35} \approx 30 \cdot 10^9$ memory.

Motivation

1. Basic computational problem.

2. Applications in cryptography:
   - Factoring large integers, breaking RSA.
   - Removing non-linearity by making some equations incorrect breaking symmetric encryption.

Obvious algorithms

Try to satisfy all using Gaussian elimination.
Time $\approx mn^2$.

Try all possible assignments and take the best.
Time $\approx mk2^n$.

Satisfying all equations

Gaussian elimination in time $O(mn^2)$ closes the theoretical question but problems remain in practice.

Recent factorization of $2^{1039} - 1$ used
$m \approx n \approx 66 \cdot 10^6$ and $k \approx 150$.

Cannot even store the matrix in dense form and time $mn^2 \approx 2^{78}$ is much too large!

Different algorithms running in time $O(mnk)$ and memory $O(mk)$ are used.

Learning parity with noise

If you want to space out and think consider

$m = 10000$, $n = 200$, $k = n/2$.

Construct left hand side of each equation randomly.

Have a hidden solution x^0 and set right hand side to be correct for x^0 with probability $\frac{2}{3}$.

Given resulting system, find x^0 (the key to a cryptosystem).

One approach in theory

Use equations that behave in the same way on a large set of variables to eliminate these variables simultaneously.

Gets equations in fewer variables that are less likely to be correct.

In theory $m = 2^{n/\log n}$ can be solved in time $2^{O(n/\log n)}$ [BKW].

Beats 2^n time of exhaustive search but not by much....

Can we prove impossibility results?

Is $O(nmk)$ best possible when all equations can be simultaneously satisfied?

For over-determined systems can we find optimal solutions in polynomial time?

Can we prove impossibility results?

Is $O(nmk)$ best possible when all equations can be simultaneously satisfied?

No lower bound higher beyond reading the input is known and seems beyond current technique.

For over-determined systems can we find optimal solutions in polynomial time?

Not known, but NP-complete.

NP and P

NP: Decision problems where a positive solution can be verified in polynomial time.

P: Decision problems that can be solved in polynomial time.

Believed that $NP \neq P$ but we are very far from proving this.

Typical example of NP, satisfiability of Boolean formulas:

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

NP-complete and NP-hard

A problem X$\in$NP is NP-complete if X$\in$ P is equivalent to $P = NP$.

A problem Y is NP-hard if $Y \in P$ implies $P = NP$.

Computational problems that are not decision problems cannot be NP-complete as they do not belong to NP, but can be NP-hard.

Satisfiability (Sat) was the first problem to be proved NP-complete by Cook in 1971.

Max-Lin-2 is NP-hard

Finding the maximal number of simultaneously satisfiable equations is NP-hard by a very simple reduction (exercise for undergraduates).

If NP$\neq$ P, Max-k-Lin-2 cannot be solved optimally in polynomial time, for any $k \geq 2$.

If Sat requires time 2^{cn} then so does Max-Lin-2.

Approximation

Idea: If we cannot find the best solutions efficiently maybe we can find the second best or a reasonably good solution.

Approximation

Idea: If we cannot find the best solutions efficiently maybe we can find the second best or a reasonably good solution.

An algorithm has approximation ratio α if for any instance

$$\frac{\text{Value of found solution}}{\text{Value of optimal solution}} \geq \alpha$$

(trivial) Positive results

Can easily find a solution that satisfies $m/2$ equations, in fact a random assignment does this on the average.

No solution satisfies more than all m equations.

This gives an approximation ratio of $1/2$.

Real results

Theorem [H] : For any $\epsilon > 0$ and any $k \geq 3$ it is NP-hard to approximate Max-k-Lin-2 within $\frac{1}{2} + \epsilon$.

Real results

Theorem [H] : For any $\epsilon > 0$ and any $k \geq 3$ it is NP-hard to approximate Max-k-Lin-2 within $\frac{1}{2} + \epsilon$.

Theorem [GW] : It is possible to approximate Max-2-Lin-2 within $\approx .8785$ in polynomial time.

Real results

Theorem [H] : For any $\epsilon > 0$ and any $k \geq 3$ it is NP-hard to approximate Max-k-Lin-2 within $\frac{1}{2} + \epsilon$.

Proof by a reduction (with several steps).

Theorem [GW] : It is possible to approximate Max-2-Lin-2 within $\approx .8785$ in polynomial time.

Real results

Theorem [H] : For any $\epsilon > 0$ and any $k \geq 3$ it is NP-hard to approximate Max-k-Lin-2 within $\frac{1}{2} + \epsilon$.

Proof by a reduction (with several steps).

Theorem [GW] : It is possible to approximate Max-2-Lin-2 within $\approx .8785$ in polynomial time.

Proof by an algorithm relying on semidefinite programming.

Hardness proof

Given a formula $\varphi(x)$ produce a system of m linear equations $Ay = b$ such that

If φ satisfiable then can satisfy $(1 - \epsilon)m$ equations.

If φ not satisfiable then can only satisfy $(\frac{1}{2} + \epsilon)m$ equations.

Hardness proof

Given a formula $\varphi(x)$ produce a system of m linear equations $Ay = b$ such that

If φ satisfiable then can satisfy $(1-\epsilon)m$ equations.

If φ not satisfiable then can only satisfy $(\frac{1}{2}+\epsilon)m$ equations.

We can decide whether φ is satisfiable by approximating within

$$\frac{\frac{1}{2}+\epsilon}{1-\epsilon}+\delta.$$

Probabilistically checkable proof

Details are complicated and omitted but uses something called on probabilistically checkable proofs (PCPs).

Probabilistically checkable proof

Details are complicated and omitted but uses something called on probabilistically checkable proofs (PCPs).

NP: A computationally limited (P-time) verifier checks a proof of a statement it cannot verify on its own.

For Sat the verifier reads n bits, the number of variables.

Probabilistically checkable proof

Details are complicated and omitted but uses something called on probabilistically checkable proofs (PCPs).

NP: A computationally limited (P-time) verifier checks a proof of a statement it cannot verify on its own.

For Sat the verifier reads n bits, the number of variables.

We want to limit the verifier to reading three bits.

PCP from reduction

Given $\varphi(x)$ to be checked for system create $Ay = b$ as in reduction with $k = 3$.

PCP from reduction

Given $\varphi(x)$ to be checked for system create $Ay = b$ as in reduction with $k = 3$.

φ satisfiable can satisfy $(1 - \epsilon)m$ equations.

φ not satisfiable can only satisfy $(\frac{1}{2} + \epsilon)m$ equations.

PCP from reduction

Given $\varphi(x)$ to be checked for system create $Ay = b$ as in reduction with $k = 3$.

φ satisfiable can satisfy $(1 - \epsilon)m$ equations.

φ not satisfiable can only satisfy $(\frac{1}{2} + \epsilon)m$ equations.

Traditional proof, good assignment to x. Reading n bits to get anything.

PCP from reduction

Given $\varphi(x)$ to be checked for system create $Ay = b$ as in reduction with $k = 3$.

φ satisfiable can satisfy $(1 - \epsilon)m$ equations.

φ not satisfiable can only satisfy $(\frac{1}{2} + \epsilon)m$ equations.

Traditional proof, good assignment to x. Reading n bits to get anything.

PCP, good assignment to y. Pick one random equation and check only this equation.

Parameters of PCP

Reads only three bits.

Parameters of PCP

Reads only three bits.

Completeness: If φ satisfiable, exists proof that makes verifier accept with probability $(1 - \epsilon)$.

Parameters of PCP

Reads only three bits.

Completeness: If φ satisfiable, exists proof that makes verifier accept with probability $(1 - \epsilon)$.

Soundness: If φ not satisfiable, no proof makes verifier accept with probability $\geq (\frac{1}{2} + \epsilon)$.

Extensions

Can make verifier always accept correct proof of correct statement but cannot maintain reading three bits and soundness $\frac{1}{2}$.

Extensions

Can make verifier always accept correct proof of correct statement but cannot maintain reading three bits and soundness $\frac{1}{2}$.

Reading more bits improves completeness and soundness. Can do better than independent repetitions and in fact

q bits read

Completeness one (always accept).

Soundness $2^{-q+O(\sqrt{q})}$ is possible.

Algorithms for $k = 2$

$$x_i + x_j = b$$

modulo 2 with $x_i, x_j \in \{0, 1\}$ is "the same" as

$$\frac{1 + (-1)^b y_i y_j}{2}$$

with $y_i, y_j \in \{-1, 1\}$.

The task is thus

$$\max_{y \in \{-1,1\}^n} \sum_{(i,j) \in Q} \frac{1 + (-1)^{b_{ij}} y_i y_j}{2}$$

The task is thus

$$\max_{y \in \{-1,1\}^n} \sum_{(i,j) \in Q} \frac{1 + (-1)^{b_{ij}} y_i y_j}{2}$$

Relax by setting $z_{ij} = y_i y_j$ and requiring that Z is a positive semidefinite matrix with $z_{ii} = 1$.

Positive semidefinite matrices?

Z symmetric matrix is positive semidefinite iff one of the following is true

- All eigenvalues $\lambda_i \geq 0$.
- $w^T Z w \geq 0$ for any vector $w \in R^n$.
- $Z = V^T V$ for some matrix V.

$z_{ij} = y_i y_j$ is in matrix language $Z = yy^T$.

By a result by Alizadeh we can to any desired accuracy solve

$$\max \sum_{ij} c_{ij} z_{ij}$$

subject to

$$\sum_{ij} a_{ij}^k z_{ij} \leq b^k$$

and Z positive semidefinite.

By a result by Alizadeh we can to any desired accuracy solve

$$\max \sum_{ij} c_{ij} z_{ij}$$

subject to

$$\sum_{ij} a_{ij}^k z_{ij} \leq b^k$$

and Z positive semidefinite.

Intuitive reason, the set of PSD matrices is convex and we should be able to find optimum of linear function as we have no local optima (as is true for LP).

Want to solve

$$\max_{y\in\{-1,1\}^n} \sum_{(i,j)\in Q} \frac{1+(-1)^{b_{ij}} y_i y_j}{2}$$

but as $Z = V^T V$ we do

$$\max_{\|v_i\|=1} \sum_{(i,j)\in Q} \frac{1+(-1)^{b_{ij}} (v_i, v_j)}{2},$$

i.e. optimizing over vectors instead of real numbers.

Going vector to Boolean

The vector problem accepts a more general set of solutions. Gives higher objective value.

Going vector to Boolean

The vector problem accepts a more general set of solutions. Gives higher objective value.

Key question: How to use the vector solution to get back a Boolean solution that does almost as well.

Rounding vectors to Boolean values

Great suggestion by GW.

Given vector solution v_i pick random vector r and set

$$y_i = \mathrm{Sign}((v_i, r)),$$

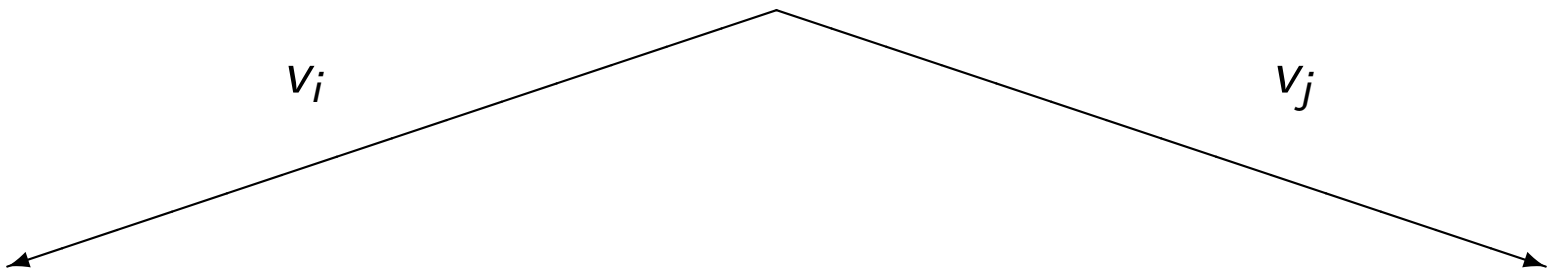
where (v_i, r) is the inner product.

Intuition of rounding

Assume $b_{ij} = 1$. Contribution to objective function large,

$$\frac{1 - (v_i, v_j)}{2}$$

large implying angle between v_i, v_j large,
$\text{Sign}((v_i, r)) \neq \text{Sign}((v_j, r))$ likely



Analyzing GW

Do term by term, θ angle between vectors.
Contribution to semi-definite objective function

$$\frac{1-(v_i,v_j)}{2} = \frac{1-\cos\theta}{2}$$

Probability of satisfying equation

$$Pr[\text{Sign}((v_i,r)) \neq \text{Sign}((v_j,r))] = \frac{\theta}{\pi}$$

Minimal quotient gives approximation ratio

$$\alpha_{GW} = \min_{\theta} \frac{2\theta}{\pi(1-\cos\theta)} \approx .8785$$

Other finite fields

Inapproximability results extend to any field getting ratio $\frac{1}{p} + \epsilon$ for Max-k-Lin-p and $\frac{1}{p^d} + \epsilon$ in $GF[p^d]$ for $k \geq 3$.

SDP can be used to get non-trivial approximation when $k = 2$, again in any field.

Other finite fields

Inapproximability results extend to any field getting ratio $\frac{1}{p} + \epsilon$ for Max-k-Lin-p and $\frac{1}{p^d} + \epsilon$ in $GF[p^d]$ for $k \geq 3$.

SDP can be used to get non-trivial approximation when $k = 2$, again in any field.

Extremely important problem: Consider Max-2-Lin-p. Is it true that for any $\epsilon > 0$ there is a $p = p(\epsilon)$ such it is hard to distinguish cases where we can satisfy $(1 - \epsilon)m$ of the equations from cases where we can only satisfy ϵm of the equations?

Unique games conjecture [K].

Some final words

Possible cases to improve algorithms

- Large sparse systems where we want to satisfy all equations.
- Random linear systems with clear champion satisfying a fraction $p > \frac{1}{2}$ of the equations.

Some final words

Possible cases to improve algorithms

- Large sparse systems where we want to satisfy all equations.
- Random linear systems with clear champion satisfying a fraction $p > \frac{1}{2}$ of the equations.

Other point: PCPs gives a very efficient way to check NP-statements that at least I find surprising.