On the approximability of Contstraint Satisfaction Problems

Johan Håstad

KTH Numerical Analysis
and Computer Science

May 23, 2006

Basic definitions

- Variables x_i ranging over a finite domain $[d] = \{0, 1, \ldots d-1\}$, many times $d = 2$, "Boolean values".
- A set $C_i(x_{i_1}, x_{i_2}, \ldots c_{i_k}), 1 \leq i \leq m$ of k-ary constraints. Usually all of same "type".

We think of d and k as fixed while n and m tend to infinity.

Examples

Max-k-Lin-d Linear equations modulo d, k variables in each equation.

Max-k-Sat Disjunctions of k literals, e.g. $C_i = x_1 \vee \overline{x_7} \vee x_{12}$.

Max-Cut-d Divide nodes of graph in d pieces, $x_i \neq x_j$ $(i,j) \in E$.

Satisfy as many constraints as possible.

Our angle

Efficient algorithms for finding optimal or good solutions.

Probabilistic polynomial time.

NP-hardness from the stone-ages

It is NP-complete to decide if we can satisfy all constraints of Max-k-Sat for $k \geq 3$, Max-Cut-d, $d \geq 3$.

It is NP-hard to find optimal solution to Max-2-Sat, Max-k-Lin-d, and Max-Cut(-2).

Approximation ratio

We try to find good solution. Measure: Approximation ratio

$$\frac{Value(Found\ solution)}{Value(Best\ solution)}$$

worst case over all instances.

For a randomized algorithm we allow expectation over internal randomness, worst case over inputs.

The mindless algorithm

Give each variable a random value.

The mindless algorithm

Give each variable a random value.

Suppose each constraint accepts P out of the d^k possible k-tuples.

The mindless algorithm

Give each variable a random value.

Suppose each constraint accepts P out of the d^k possible k-tuples.

Satisfies, on average mPd^{-k} constraints

Approximation ratio $\geq Pd^{-k}$.

Mindless Max-3-Sat, Max-k-Lin-d

3Sat: 8 possible assignments to three literals, 7 satisfying.

Mindless has approximation ratio $7/8$.

Mindless Max-3-Sat, Max-k-Lin-d

3Sat: 8 possible assignments to three literals, 7 satisfying.

Mindless has approximation ratio $7/8$.

Max-Lin-d: Each equation is satisfied with probability $1/d$, independently of number of appearing variables.

Mindless has approximation ratio $1/d$.

Making mindless algorithm deterministic

Use the method of conditional expectations.

For each value of x_1 calculate expected number of satisfied constraints and use fix x_1 to value that gives maximum.

Now look at x_2, etc.

The key question

For which types of constraints can we beat the random mindless algorithm and on what instances?

- As soon as optimal value is significantly better than $Pd^{-k}m$, i.e. $(1+\epsilon)Pd^{-k}m$.
- When the optimal value is (very) large, i.e. $(1-\epsilon)m$.
- When we can satisfy all constraints, satisfiable instances.

Two branches

- Positive results. Efficient algorithms with provable ratios.
- Negative results. Proving that certain tasks are NP-hard, or possibly hard given some other complexity assumption.

The favorite techniques

Algorithms: Semi-definite programming. Introduced in this context by Goemans and Williamson.

Lower bounds: The PCP-theorem and its consequences. Arora, Lund, Motwani, Sudan and Szegedy.

Max-Cut

The task is to maximize with $x_i \in \{-1, 1\}$ and edges E,

$$\sum_{(i,j)\in E} \frac{1 - x_i x_j}{2}.$$

Max-Cut

The task is to maximize with $x_i \in \{-1, 1\}$ and edges E,

$$\sum_{(i,j)\in E} \frac{1 - x_i x_j}{2}.$$

Relax by setting $y_{ij} = x_i x_j$ and requiring that Y is a positive semidefinite matrix with $y_{ii} = 1$.

Positive semidefinite matrices?

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

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

$y_{ij} = x_i x_j$ is in matrix language $Y = xx^T$.

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

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

subject to

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

and Y positive semidefinite.

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

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

subject to

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

and Y positive semidefinite.

Intuitive reason, set of PSD is convex and we should be able to find optimum of linear function (as is true for LP).

Want to solve

$$\max_{x\in\{-1,1\}^n} \sum_{(i,j)\in E} \frac{1-x_i x_j}{2}.$$

but as $Y = V^T V$ we instead maximize

$$\sum_{(i,j)\in E} \frac{1-(v_i, v_j)}{2}.$$

for $\|v_i\| = 1$, 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

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

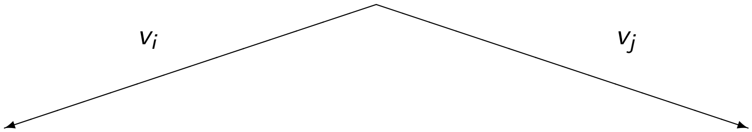
where (v_i, r) is the inner product.

Intuition of rounding

Contribution to objective function large,

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

large implying angle between v_i, v_j large,
$\mathrm{Sign}((v_i, r)) \neq \mathrm{Sign}((v_j, r))$ 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 being cut

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

Minimal quotient gives approximation ratio

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

Immediate other application

Original GW-paper derived same bound for approximating Max-2-Sat.

Improved [LLZ] to $\approx .9401$ (not analytically proved).

"Obvious" semi-definite program. More complicated rounding.

Immediate other application

Original GW-paper derived same bound for approximating Max-2-Sat.

Improved [LLZ] to $\approx .9401$ (not analytically proved).

"Obvious" semi-definite program. More complicated rounding.

Many other applications some using many additional ideas.

Proving NP-hardness results for approximability problems

Want to study problem X.

Proving NP-hardness results for approximability problems

Want to study problem X.

Given a Sat-formula φ, produce an instance, I of X such that:
φ satisfiable $\rightarrow$ $Value(I) \geq c$.
φ not satisfiable $\rightarrow$ $Value(I) \leq s$.

Proving NP-hardness results for approximability problems

Want to study problem X.

Given a Sat-formula φ, produce an instance, I of X such that:
φ satisfiable $\rightarrow$ $Value(I) \geq c$.
φ not satisfiable $\rightarrow$ $Value(I) \leq s$.

It is NP-hard to approximate our problem within $s/c + \epsilon$.

Proving NP-hardness results for approximability problems

Want to study problem X.

Given a Sat-formula φ, produce an instance, I of X such that:
φ satisfiable $\rightarrow$ $Value(I) \geq c$.
φ not satisfiable $\rightarrow$ $Value(I) \leq s$.

It is NP-hard to approximate our problem within $s/c + \epsilon$.

Running approximation algorithm on I tells us whether φ is satisfiable.

Inapproximability for Max-3-Sat

Given a Sat-formula φ, produce a different Sat-formula ψ with m clauses such that:

φ satisfiable $\rightarrow$ ψ satisfiable.

φ not satisfiable $\rightarrow$ Can only simultaneously satisfy only $(1-\epsilon)m$ of the clauses of ψ.

Gives inapproximability ratio $(1-\epsilon)$.

Probabilistically Checkable Proofs (PCPs)

A proof that 3-Sat formula φ is satisfiable.

Traditionally an assignment to the variables.

Checked by reading all variables and checking.

Probabilistically Checkable Proofs (PCPs)

A proof that 3-Sat formula φ is satisfiable.

Traditionally an assignment to the variables.

Checked by reading all variables and checking.

We want to read much less of the proof, only a constant number of bits.

Sought reduction gives PCP!

Proof: An assignment to variables of ψ.
Checking: Pick a random clause and read the variables that appear in the clause and see if it is satisfied.

Sought reduction gives PCP!

Proof: An assignment to variables of ψ.

Checking: Pick a random clause and read the variables that appear in the clause and see if it is satisfied.

Preserving satisfiability: φ satisfiable implies ψ satisfiable and we always accept.

Sought reduction gives PCP!

Proof: An assignment to variables of ψ.
Checking: Pick a random clause and read the variables that appear in the clause and see if it is satisfied.
Preserving satisfiability: φ satisfiable implies ψ satisfiable and we always accept.
Amplifying non-satisfiability: φ not satisfiable implies ψ only $(1-\epsilon)$-satisfiable and we reject with probability $\geq \epsilon$.

Sought reduction gives PCP!

Proof: An assignment to variables of ψ.
Checking: Pick a random clause and read the variables that appear in the clause and see if it is satisfied.
Preserving satisfiability: φ satisfiable implies ψ satisfiable and we always accept.
Amplifying non-satisfiability: φ not satisfiable implies ψ only $(1-\epsilon)$-satisfiable and we reject with probability $\geq \epsilon$.
Repeat a constant number of times to decrease fooling probability.

Thinking more carefully

Our type of reduction is equivalent to a good PCP.

The PCP theorem

PCP theorem: [ALMSS] There is a proof system for satisfiability that reads a constant number of bits such that

- Verifier always accepts a correct proof of correct statement.
- Verifier rejects any proof for incorrect statement with probability $1/2$.

The PCP theorem

PCP theorem: [ALMSS] There is a proof system for satisfiability that reads a constant number of bits such that

- Verifier always accepts a correct proof of correct statement.
- Verifier rejects any proof for incorrect statement with probability $1/2$.

Translates to any NP statement by a reduction.

Proof of PCP theorem

Original proof: Algebraic techniques, properties of polynomials, proof composition, aggregation of queries, etc. Many details.

Proof of PCP theorem

Original proof: Algebraic techniques, properties of polynomials, proof composition, aggregation of queries, etc. Many details.

Interesting new proof by Dinur (2005) that is essentially combinatorial. Relies on recursion and expander graphs.

Proof of PCP theorem

Original proof: Algebraic techniques, properties of polynomials, proof composition, aggregation of queries, etc. Many details.

Interesting new proof by Dinur (2005) that is essentially combinatorial. Relies on recursion and expander graphs.

These basic proofs give BAD inapproximability constants

Improving constants

A long story, one final point:

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

Matches mindless algorithm up to ϵ.

Ingredients in proof/construction

- Two prover games.
- Parallel repetition for two-prover games. [R]
- Coding strings by the long code. [BGS]
- Using discrete Fourier transforms in the analysis. [H]

Classifying CSPs

We have some well defined groups.

1. Hard to approximate better than random mindless algorithm on satisfiable instances.
2. Hard to do better than random mindless algorithm on (almost) satisfiable instances.
3. Have an approximation constant better than achieved by random mindless algorithm.
4. Can beat random mindless algorithm as soon as soon as optimal beats random.

Two first classes we call Approximation resistant.

The case $k = 2$

Predicates that depend on two variables.

Semi-definite programming is universal, for any fixed domain d and any predicate that the depends we can do better than random [H].

Belongs at least to class 4, if optimal significantly better than random, we can efficiently find solution significantly better than random.

Any d, any predicate.

The case $k = 3$ and $d = 2$.

Predicates of three boolean variables.

- Approximation resistant iff we accept either all strings of even parity or all strings of odd parity.
- Fully approximable (class 4) if un-correlated with parity of all three variables.

Other (nontrivial) cases belong to class 3.

The case $k = 3$ and $d = 2$.

Predicates of three boolean variables.

- Approximation resistant iff we accept either all strings of even parity or all strings of odd parity.
- Fully approximable (class 4) if un-correlated with parity of all three variables.

Other (nontrivial) cases belong to class 3.
Max-3-Sat is hard to approximate within $7/8 + \epsilon$, mindless is optimal!

The case $k = 3$ and $d = 2$ unknown.

What happens with the "not two ones" predicate on satisfiable instances.

Could we do better than random?

Not true for just $(1 - \epsilon)$-satisfiable instances!

The case $k = 3$ and $d = 2$ unknown.

What happens with the "not two ones" predicate on satisfiable instances.

Could we do better than random?

Not true for just $(1 - \epsilon)$-satisfiable instances!

Parity is different for satisfiable and almost satisfiable instances!

The case $k = 3$ and $d = 2$ unknown.

What happens with the “not two ones” predicate on satisfiable instances.

Could we do better than random?

Not true for just $(1 - \epsilon)$-satisfiable instances!

Parity is different for satisfiable and almost satisfiable instances!

Adding one more accepting configuration we do get approximation resistance on satisfiable instances.

The case of $k = 4$ and $d = 2$.

Partial classification by Hast.

400 essentially different predicates.

- 79 approximation resistant.
- 275 not approximation resistant.
- 46 not classified.

What can we say in general?

With $d = 2$ and large k.

- Accepts very few inputs, non-trivially approximable.
- Exists rather sparse approximation resistant predicates.
- The really dense predicates are approximation resistant.

General fact?

It seems like the more inputs a predicate accepts the more likely it is to be approximation resistant.

General fact?

It seems like the more inputs a predicate accepts the more likely it is to be approximation resistant.

Approximation resistance is not a monotone property. Have example P, Q,

$$P(x) \rightarrow Q(x)$$

P approximation resistant.

Q not approximation resistant.

Puzzling question

For large k is a random predicate of Boolean variables approximation resistant?

I do not have a strong opinion.

Wide open question

What happens form larger d?

Maybe something nice can be said at least for $k = 3$?

Summing up

We have a huge classification problem ahead of us.

We have only scratched the surface.

Does it have a nice answer, even for $d = 2$?

The question of random predicates might be doable...