KUNGL TEKNISKA HÖGSKOLAN

Every 2-CSP allows nontrivial approximation

Johan Håstad
Royal Institute of Technology
Stockholm, SWEDEN

Constraint satisfaction problems (CSP)

Constraints, $(C_i)_{i=1}^m$, each acting on a constant number of variables taking values in $[d] = 0, 1 \ldots d-1$.

3-Sat: $(x_1 \vee \bar{x}_7 \vee x_{26})$

Equations mod p: $2x_7 + 4x_9 \equiv 4 \bmod 11$.

Max-CSP problem

Given a set of constraints, find the assignment that satisfies the maximum number of constraints.

Almost always NP-hard. Want an efficient C-approximation algorithm.

Find x^0 such that

$Value(x^0) \geq C \cdot OPT$

The “random” algorithm

The mindless approach:

Assign uniformly random values to the variables *ignoring the constraints* and see what we get.

Repeat a few times and take best solution (helps in practice, not in theory).

Approximation ratio of random algorithm

OPT is at most m, i.e. all constraints satisfied. We get approximation ratios

7/8, for 3-Sat

3/4 for 2-Sat

$1/p$ for linear equations mod p

Each constraint is satisfied with the given probability.

Can we improve the approximation ratio?

Gut instinct: We must be able to do better!

Yes: 2-Sat [GW], linear equations with two variables in each equation [AEH].

No: (NP-hard to get additive $\epsilon > 0$): 3-Sat, linear equations with three variables in each equation [H].

Approximation resistance

Problems for which it is NP-hard to do better than random I call *approximation resistant*.

Efficient algorithms cannot do anything useful.

Unless we are willing to spend exponential time we might as well close our eyes as looking at the formula does not help.

Our main result

Theorem: (informal) Constraints that depend on only two variables are *not* approximation resistant.

Theorem: Suppose each C_i depends on two variables with range $[d]$ and rejects t out of the d^2 inputs. Then for some universal $c > 0$ we can, in randomized polynomial time, approximate the CSP within a factor

$$1 - \frac{t}{d^2}\left(1 - \frac{c}{d^2 \log d}\right).$$

Rest of talk

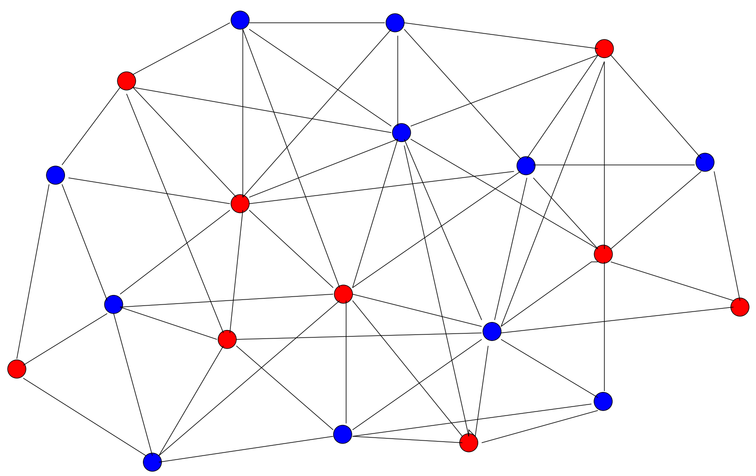
Review history.

Describe ideas needed to prove theorem.

Boolean constraints of width 2

Classical problems:

Max-Cut: Given a graph G partition the vertices into two sets to cut the maximum number of vertices.



Really CSP with $d = 2$, one variable x_i for each node, $x_i \neq x_j$ for each edge (i, j).

Max-2Sat: $(x_i \vee \bar{x}_j)$.

Key technique

Semi-Definite Programming (SDP) introduced by Goemans and Williamson. Relax the combinatorial problem to optimizing a linear function over matrices that are positive semi-definite.

An SDP can be solved in polynomial time to arbitrary accuracy [A].

Max-Cut by SDP

We want to find

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

Relax to

$$\max_{Y} \sum_{(i,j) \in E} \frac{1 - y_{ij}}{2}$$

matrix Y, PSD, symmetric, and $y_{ii} = 1$.

As $Y = V^T V$ this can be viewed as

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

A vector valued solution!

SDP vs Boolean Max-Cut

Boolean solution is a special case of SDP with one-dimensional vectors.

[GW] round a SDP-solution to Boolean by setting

$$x_i = sign((r, v_i))$$

for a random vector r. Each component independently picked from $N(0,1)$.

Analysis Max-Cut

[GW] make an analysis term-wise

$$\forall_{v_i,v_j} \frac{E[1 - x_i x_j]}{1 - (v_i, v_j)} \geq .87$$

giving the Goemans-Williamson constant α.

Max-2Sat by SDP

Again use ± 1 for Boolean values.

$$x_i \vee \bar{x}_j \Rightarrow \frac{3 + x_i - x_j + x_i x_j}{4}$$

Introduce $x_0 \equiv 1$ and replace x_i by $x_0 x_i$ to make bilinear.

More terms and complications.

Basic rounding

$x_i = 1$ if $sign((r, v_i)) = sign((r, v_0))$.

Algorithms for $d > 2$

Better than random for

Max-d-Cut [FJ].

Linear equations mod d [AEH].

Uniform constraints [EG].

Uniform: For each value of one variable equally many accepted values of the other.

Techniques used

Semi-Definite Programming. Code one value in $[d]$ as one or more vectors. Engebretsen and Guruswami and we use

$$x_i \leftrightarrow v_i^0, v_i^1 \ldots v_i^{d-1}$$

Boolean to vector: If $x_i = a$ set $v_i^a = v_0$ and $v_i^{a'} = 0$, $a \neq a'$.

Write objective function as sum of inner products.

Uniformity of constraints implies that we only get natural bilinear terms, good for [EG] analysis.

The idea of rounding

Pick a random vector r with coordinates from $N(0,1)$ and set x_i to value a with a probability $\frac{1}{d} + (r, v_i^a - \frac{1}{d}v_0)/D$.

Works poorly for linear terms but nicely for bilinear terms as

$$E((r, v_i^a)(r, v_j^b)) = (v_i^a, v_j^b).$$

Analysis

Returning to Max-2Sat to avoid notation. Two cases for formula:

- **Balanced:** x_i and $\bar{x}_i$ appears equally often.

- **Unbalanced:** For each i, x_i has a preferred value.

Ignore intermediate cases.

Unbalanced case

Naively: Set x_i to favorite value. Does not work.

Better: Flip x_i with a bias towards favorite value. Works for suitable bias.

Balanced case

Drop all linear terms. They cancel anyway.

$$(x_i \vee \bar{x}_j) \Leftrightarrow \frac{3 - x_i x_j}{4}$$

Do local analysis on these terms. Gives a much better ratio!

Returning to range $[d]$

In the unbalanced case we can find a biased rounding that gives an advantage over random.

In the balanced case, as can be guessed from [EG], essentially their rounding works to beat random.

One of the two techniques always applies.

Always getting something

As stated the algorithm only finds something better than random for almost satisfiable instances.

As proposed by [HS] it makes sense to study by how much we outperform random

$$value'(x_0) = value(x_0) - E[value(x)]$$

General theorem

Theorem Suppose each C_i, $1 \leq i \leq m$ depends on two variables with range $[d]$ and rejects t out of the d^2 inputs and that the optimal solution satisfies

$$\left(1 - \frac{t}{d^2} + \epsilon\right) m$$

constraints. Then, for some universal $c > 0$, we can, in randomized polynomial time, find an assignment that satisfies

$$\left(1 - \frac{t}{d^2} + \frac{c\epsilon}{d^2 \log\left(d/\epsilon\right)}\right) m$$

constraints.

Summary

Semi-definite programming is universal for constraints depending on two variables for any fixed range.

Key new idea: A global view of the the linear terms. Then more or less standard local analysis.

Open question

For Max-2Sat: Is the balanced case the hardest?