Low-rank semidefinite programming for the MAX2SAT problem

Po-Wei Wang J. Zico Kolter

Machine Learning Department
Carnegie Mellon University

School of Computer Science,
Carnegie Mellon University, and
Bosch Center for Artificial Intelligence

Highlights

We will talk about

- Why maximum satisfiability problem matters
- How to **approximate** the MAXSAT problem with semidefinite program
- How to **solve** the above approximation efficiently
- How to **use the approximation** in a combinatorial search procedure

In conclusion, you will find

- The first **continuous solver for MAXSAT**
  that beats all the state-of-the-art solvers in some subclasses
- A new avenue for combinatorial optimization problems

Outline

Maximum satisfiability

Lifting and Goemans-Williamson approximation

SDP solvers and the Mixing method

Branch-and-bound for low-rank SDP

Experimental results

Reasoning with conflicts

Given a knowledge graph with imperfect facts and rules:

- Facts:
  - Spain_Nbr_Portugal, Spain_Nbr_Morocco, Spain_Nbr_France
  - Portugal_In_Eu, France_In_Eu, Morocco_In_Af
- Rules:
  - A country locates in only one region
  - Neighbors tends to in the same region

Say that we want to infer whether S_In_Eu or S_In_Af..

The problem can be translates to a boolean formula

$$(\neg S\_In\_Eu \vee \neg S\_In\_Af) \wedge (S\_Nbr\_M \wedge M\_In\_Af \Rightarrow S\_In\_Af) \ldots$$

Thus, reasoning corresponds to find a model with the least conflicts

- I.e., solving the corresponding **MAXSAT** problem

MAXSAT = Maximizing the # of satisfiable clauses

Given a **Boolean formula**, where variables $v_i \in \{\text{True, False}\}$

$$(\neg v_1) \wedge \underbrace{(v_1 \vee \neg v_2)}_{\text{Clause}} \wedge (v_1 \vee v_2 \vee v_3) \wedge (v_1 \vee v_2 \vee \neg v_3),$$

The **SAT** problem finds a satisfiable $V \in \{T, F\}^n$ if exists.

But in ML, inputs are noisy and models are not perfect
usually cannot find a model that's 100% correct

MAXSAT: maximize the number of satisfiable clauses.
That is, find a model with the least error

Difficulties: MAXSAT is NP-complete

MAXSAT is a binary optimization problem, hard to solve in general.

Modern discrete solvers apply different heuristics:

- Some bound the solution using **SAT solvers** (core-guided methods)
- Some perform greedy **local search** (e.g., CCLS, CCEHC)

There are some **continuous approximations** in theoretical computer science, but doesn't outperform discrete solvers in general

- Underlying optimization problems are expensive comparing to heuristics

We will show an **efficient approximation** that is easy to compute

Outline

Maximum satisfiability

Lifting and Goemans-Williamson approximation

SDP solvers and the Mixing method

Branch-and-bound for low-rank SDP

Experimental results

Relaxation: Lifting binary variables to higher dimensions

The binary optimization problem can be in general written as

$$\underset{V}{\text{minimize}} \; L(V), \quad \text{s.t. } v_i \in \{-1, +1\}, \quad L(\cdot) = \text{Loss func}$$

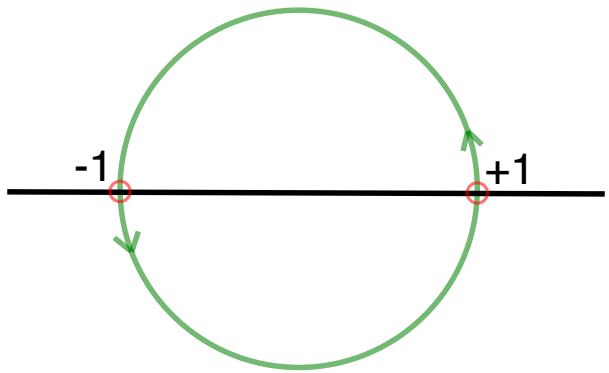
The binary constraint $v_i \in \{-1, +1\}$ are equivalent to the following

$$|v_i| = 1, \quad v_i \in \mathbb{R}^1.$$

To make it smoother, we can **lift** it **to a higher dimension space**

$$\|v_i\| = 1, \quad v_i \in \mathbb{R}^k.$$

That is, relaxing the binary variables to a continuous unit sphere



This lifting from binary variable to the unit sphere is called **Goemans-Williamson approximation**.

The quadratic approximation for MAX-2-SAT

Consider the simplest MAXSAT: the MAX-2-SAT (clause len ≤ 2)

Maximizing the satisfiability is equivalent to minimizing the loss

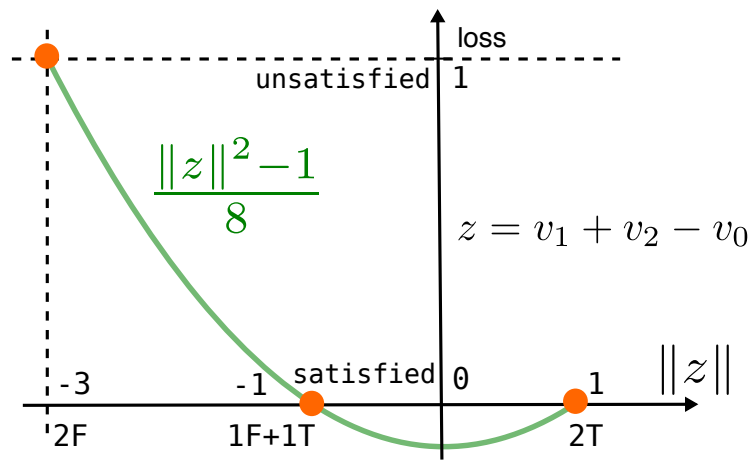
$$\underset{V}{\text{maximize}} \sum \overbrace{(v_1 \vee v_2)}^{\text{merit}} \equiv \underset{V}{\text{minimize}} \sum \overbrace{(\neg v_1 \wedge \neg v_2)}^{\text{loss}}$$

We can cleverly design the loss as

$$\text{loss}(V) = \frac{\|v_1 + v_2 - 1\|^2 - 1}{8}$$

Verify: loss when all literals=false

- $v_1 = -1,\ v_2 = -1$:
  $\text{loss}(V) = (9-1)/8 = 1$
- $v_1 = +1,\ v_2 = \pm 1$:
  $\text{loss}(V) = (1-1)/8 = 0$



Replace the 1 with v_0 as the new "truth" direction.

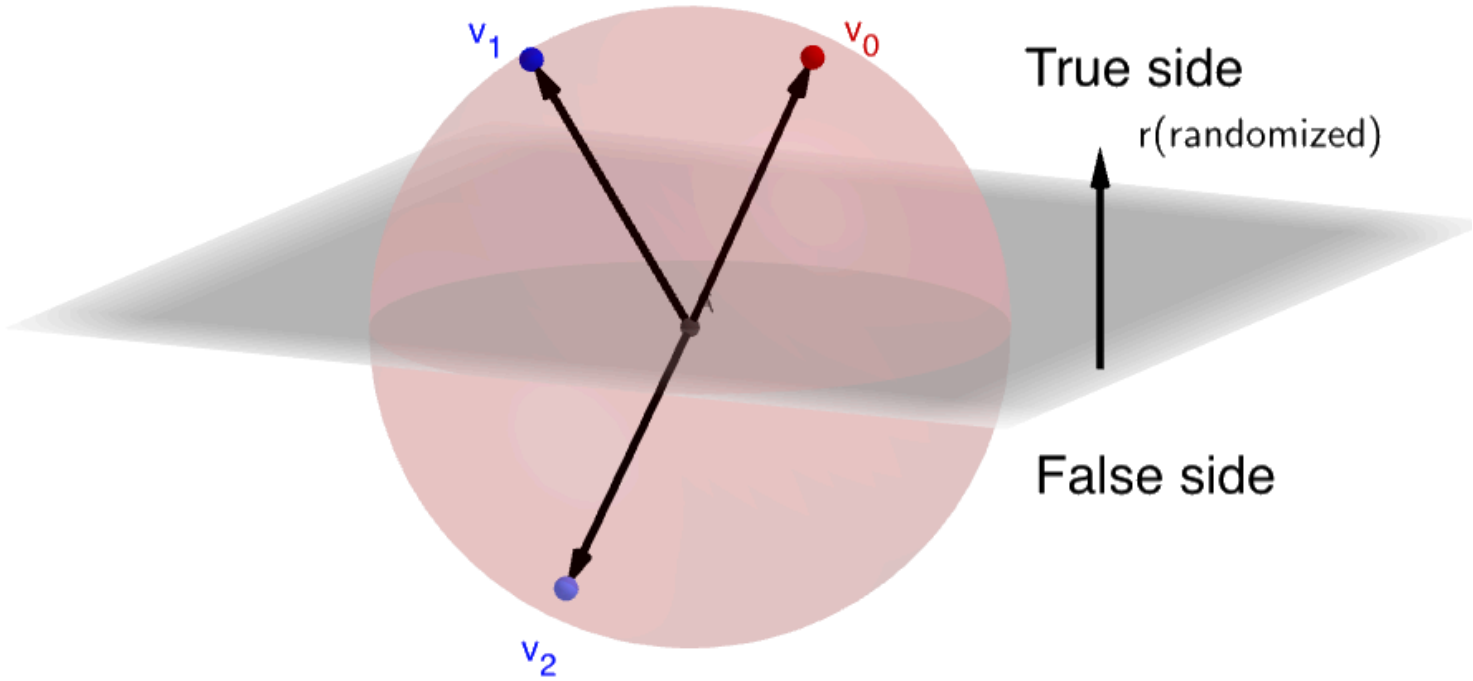
Extendable to general clause with length n_j (quadratic lower bound)

Randomize rounding: from continuous to discrete

After obtaining the optimal v_i, we can recover the discrete assignment:

Randomized rounding: pick a uniform random direction r

$$\text{binary } v_i = \text{sign}(r^T v_0 \cdot r^T v_i),\ \forall i = 1 \cdots n.$$



Goemans & Williamson (1995) proved 0.878 approximation ratio on MAX2SAT in expectation.

Perform multiple times gives surprisingly high approximation rate.

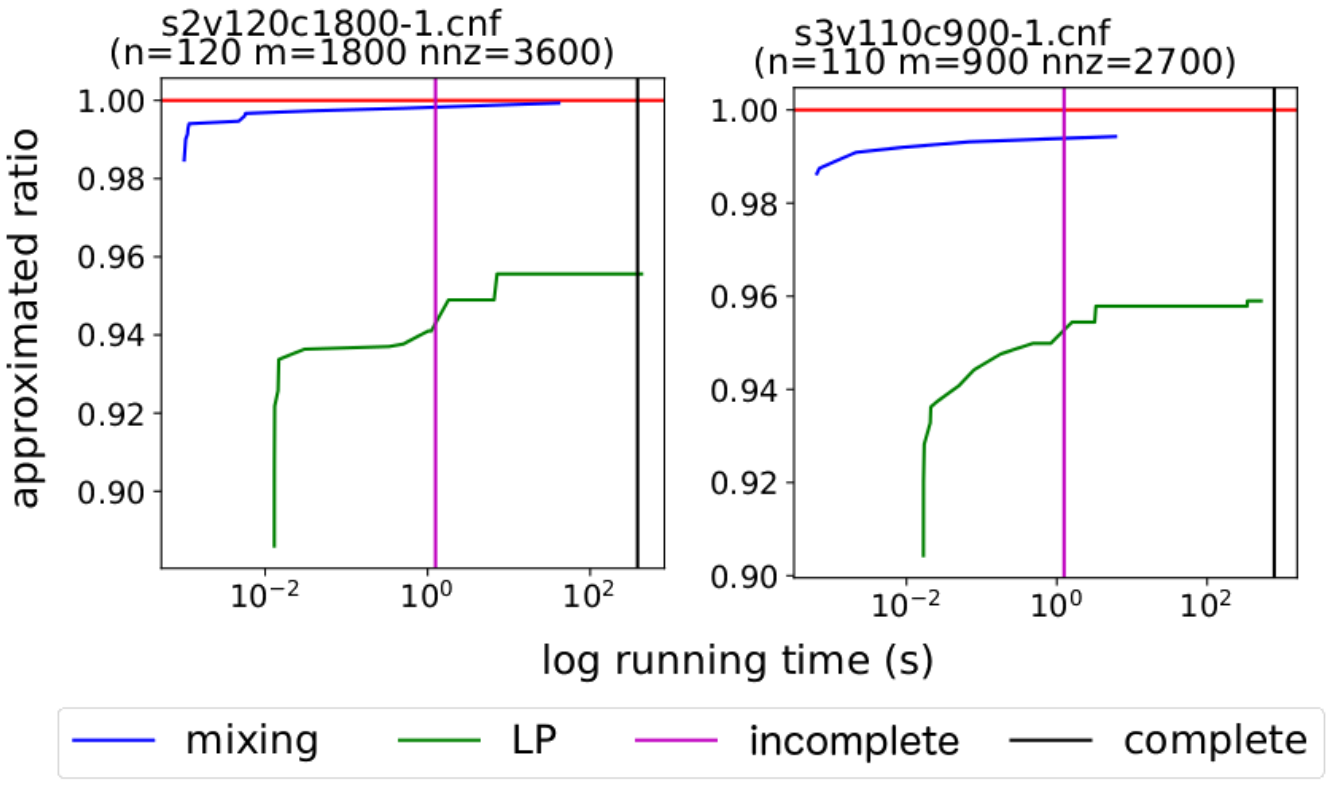
Approximation ratio experiment

- Guaranteed 0.878 approximation:

$$\mathbb{E}(Rounding) \geq 0.878 \cdot \texttt{OPT}$$

- Experiment: 0.978 approximation rate in 2016 MAXSAT dataset.

Key idea: $\mathbb{E}(Rounding) \Rightarrow \sup_{r \sim \texttt{i.i.d.}} \mathbb{E}(\text{r-th } Rounding)$



Outline

Maximum satisfiability

Lifting and Goemans-Williamson approximation

SDP solvers and the Mixing method

Branch-and-bound for low-rank SDP

Experimental results

Reducing vector program to semidefinite program

Recall that the loss is a homogeneous quadratic function to V

$$\text{loss}(V) = \frac{\|v_1 + v_2 - v_0\|^2 - 1}{8}.$$

Thus, the problem is equivalent to the quadratic **vector program**

$$\underset{V}{\text{minimize}} \langle C, V^T V \rangle, \quad \text{s.t. } \|v_i\| = 1, V \in \mathbb{R}^{k \times n}$$

When lifting the solution to a high enough dimension ($k \geq \sqrt{2n}$), the optimal solution in V is equivalent to the following SDP (Pataki, 1998)

$$\underset{X}{\text{minimize}} \langle C, X \rangle, \quad \text{s.t. } X_{ii} = 1, X \succeq 0 \in \mathbb{R}^{n \times n}.$$

Hence the optimal solution $X = V^T V$ in polynomial time.

Biggest problem: SDP solvers are slow

Since SDP is convex, we can solve it in polynomial time. That said, the complexity for SDP solvers (interior point methods) is $O(n^6)$.

- Only scales to $n \approx 10^3$
- Complexity cubic to # of variables ($X \in \mathbb{R}^{n \times n}$) from matrix inversion

We'd like to solve the problem in the low-rank space $V \in \mathbb{R}^{k \times n}$?

- Many fewer variables
- But non-convex (unit sphere shell)

Fortunately, we have a secret weapon that provably recover the optimal solution of SDP in the low-rank space.

- Recover the global optimal solution despite the non-convexity
- At the same time $10 \sim 100$x faster than other state-of-the-art solvers

Secret weapon: the Mixing method for diagonal SDP

For the optimization problem on unit sphere,

$$\underset{V}{\text{minimize}} \, \langle C, V^T V \rangle, \quad \text{s.t. } \|v_i\| = 1, V \in \mathbb{R}^{k \times n}.$$

Observe the objective can be decomposed as

$$\langle C, V^T V \rangle = \sum_i v_i^T (\sum_j c_{ij} v_j)$$

If we solve one vector v_i at a time and fix the others, the sub-problem has a closed-form solution

$$v_i := \text{normalize} \left(-\sum_{j \neq i} c_{ij} v_j \right).$$

Thus, we can **cyclically update all the vectors** for $i = 1 \ldots n$.

- Scales to ten millions of variables, $10 \sim 100$x faster than SOTA
- Provably convergent to global optimum of corresponding SDP!

No convergence proof before for low-rank SDP solvers

Low-rank decomposition first discovered by Burer and Monteiro (2001)

$$\underset{V}{\text{minimize}}\ \langle C,\, V^T V\rangle, \quad \text{s.t. } \|v_i\| = 1,\, V \in \mathbb{R}^{k\times n}.$$

Many working variants, non provably convergent to optimal objective

- Boumal et al. (2018): $H \succeq -\gamma I$ in $O(1/\gamma^3)$, bound $f - f^*$ for $k > n$
- Bhojanapalli et al. (2018): $f_\mu - f_\mu^* \to 0$ (f_μ is extended Lagrangian)

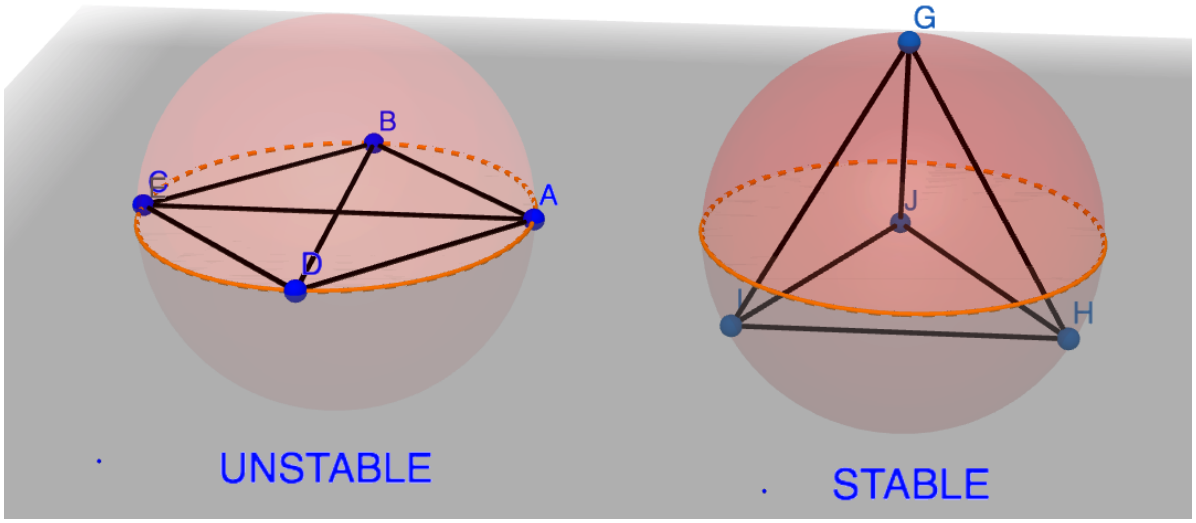
Difficulties

- Non-convex (spherical manifold dom), rotational equivalence.
- Random initialization required.
- Singularity of Jacobian and Hessian.

Main theoretical result

How do we prove it?

- Lyapunov instability in control theory and similarity to Gauss-Seidel



Results:

- First proof of convergence to global optima in 17 years.
- Asymptotic $m\sqrt{n}\log\frac{1}{\epsilon}$ v.s. $O(n^6)$, where m is the # of literals
- Further, it is practically $10 \sim 100$x faster than other low-rank method.

Proof sketch: Lyapunov instability and stable manifolds

Consider one step of the Mixing method $V^{\text{next}} := M(V)$ as operator

Instability: the operator has **expansive** direction $\forall$ non-opt criticals.

- Eigenvalues of Jacobian on manifold contain those in Euclidean

  $$\text{Eigvals}(A \otimes I_k \operatorname{diag}(I - v_i v_i^T) B \otimes I_k) \supseteq \text{Eigvals}(AB) \text{ for } k > \sqrt{2n}.$$

- Jacobian of Gauss-Seidel is unstable when not PSD ($|\lambda_i| > 1$).
- All non-optimal criticals corresponds to a G.-S. on non-PSD system.
- Thus, the Mixing methods are unstable on non-optimal criticals.

Center-stable manifold thm: Existence of **invariant manifolds**.

- Mixing method with step-size is a diffeomorphism (1-to-1 and $\mathcal{C}^1$).
- Apply center-stable manifold thm: basin to unstable is 0 measure.
- That is, random initialization never converges to unstable criticals.
- Converge to critical and never to unstable $\Rightarrow$ converge to global opt.

Outline

Maximum satisfiability

Lifting and Goemans-Williamson approximation

SDP solvers and the Mixing method

Branch-and-bound for low-rank SDP

Experimental results

Branch-and-bound framework for SDP

Despite good approximation, need **tree-search** for discrete optimality

Branch-and-bound. Goal: find the `BEST`=minimum UNSAT solution.

- Initialize a priority queue Q = the unassigned MAXSAT problem
- `While` Q is not empty:
  - $N = Q$.pop()
  - Update BEST base on N if possible.
  - Evaluate heuristic lower bound $f(N) \leq \texttt{UNSAT}(N)$
  - `If BEST` $\leq f(N)$: Prune.
  - `Else`: pick a variable v_i, Q.push($N \mid v_i = T$ and $N \mid v_i = F$).

Since the loss is a quadratic lower bound for the discrete loss,
SDPs is a lower bound in the branch-and-bound framework

Solve SDP for every internal node...

Bounding SDP by initialization

Fortunately, don't need to solve SDPs for every internal node!

- Observation: Child problems will be similar to the parent problem
- Can apply the **primal and dual initialization** to bound the SDP values and only solve SDPs for the unpredictable cases

By the SDP duality, we have the following **primal-dual pairs**

$$\underset{V \in \mathbb{R}^{k \times n}}{\text{minimize}} \; f(V) := \langle C, V^T V \rangle, \quad \text{s.t.} \;\; \|v_i\|^2 = 1.$$

$$\underset{\lambda \in \mathbb{R}^n}{\text{maximize}} \; D(\lambda) := -1^T \lambda, \quad \text{s.t.} \;\; C + D_\lambda \succeq 0.$$

Known fact: $f(V) \geq f^* \geq D(\lambda)$ and $\texttt{BEST} \geq \texttt{UNSAT} \geq \lceil f^* \rceil$.

For any feasible primal dual solution V and λ,

- $f(V) \leq \texttt{BEST} \Rightarrow$ Expand (not prunable by SDP)
- $\lceil D(\lambda) \rceil \geq \texttt{BEST} \Rightarrow$ Prune (not contain optimal solution)

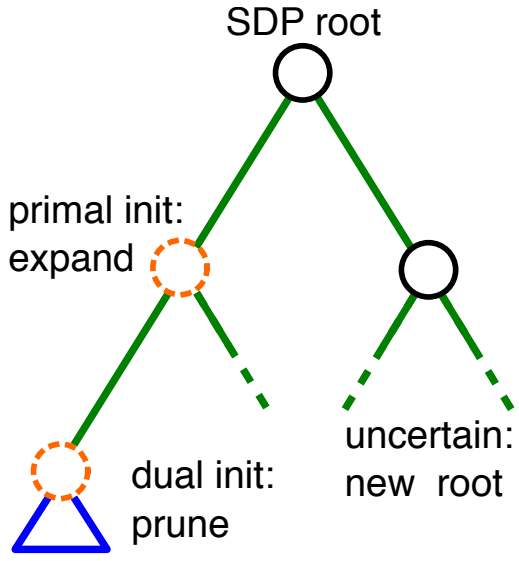
Primal and dual initialization

Primal initialization

- Trivially copy parent solution and evaluate
- Help **expansion**

Dual initialization

- Exploit the difference in C, which exhibits an arrow-head structure
- Calculate a feasible λ satisfying $C + D_\lambda \succeq 0$
- Help **pruning**



Greatly reduce the SDP subproblems that really need to tackle

Different priority queue for different tasks

There are 2 tracks in MAXSAT evaluation with different requirements

The complete track:

- Output the result after verification
- Thus, we set priority queue Q as **stack**
- Equivalent to performing depth-first search at the tree

The incomplete track:

- Output the best result as you get it, no verification
- Thus, we set priority queue Q as a **heap**
- Equivalent to best-first search at the tree

Same algorithm for the two tracks!

Outline

Maximum satisfiability

Lifting and Goemans-Williamson approximation

SDP solvers and the Mixing method

Branch-and-bound for low-rank SDP

Experimental results

Experiment setup

For fairness, we replicate the setup of the 2016 MAXSAT evaluation

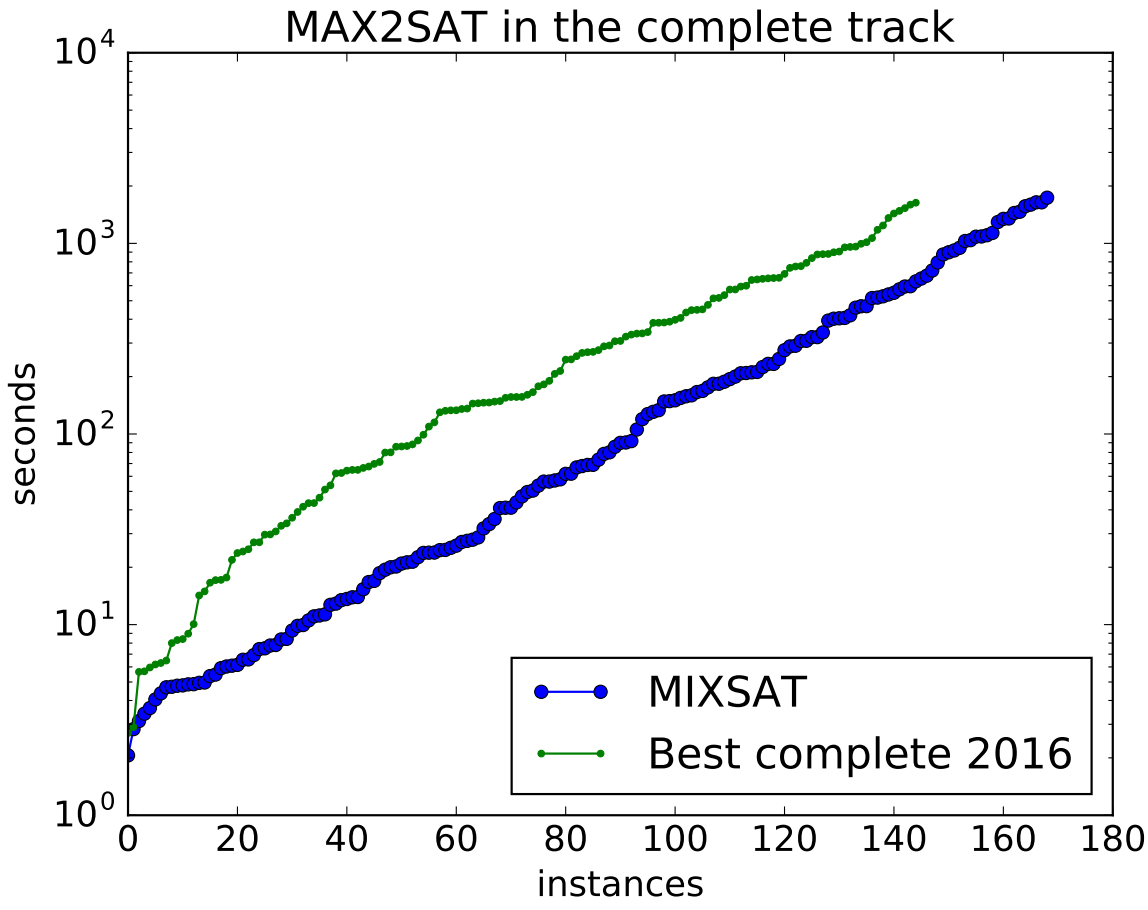
- the same CPU (Intel Xeon E5-2620)
- single core mode
- 3.5 GB memory limit.
- 30 min time limit for complete track, 5 min for incomplete track

We compare our results to the best solvers for each problem instances

- That is, the virtual best solvers

Complete track

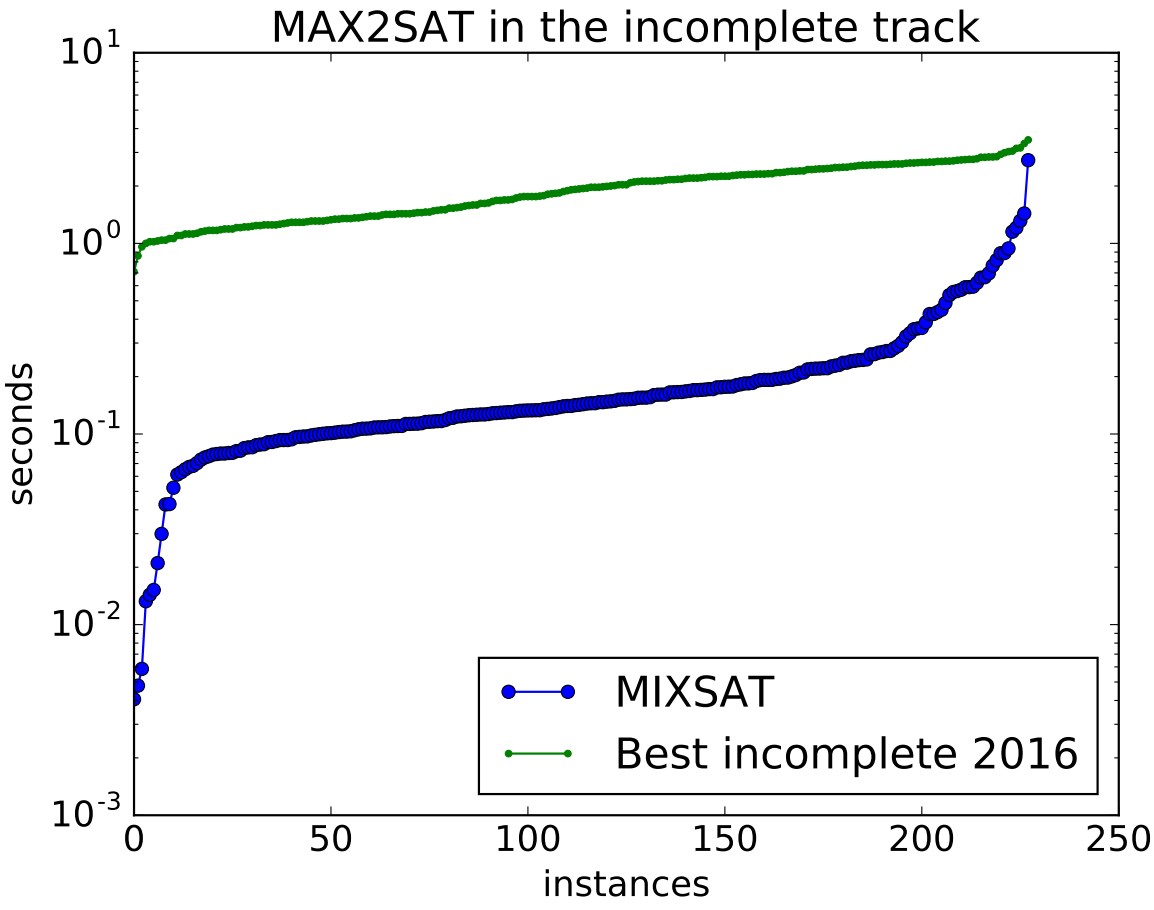
For the complete random categories, we solved 169/228 MAX2SAT instances in avg 273s, while the best solvers solve 145/228 in 341s.
Consistently outperform the VBS!



Incomplete track

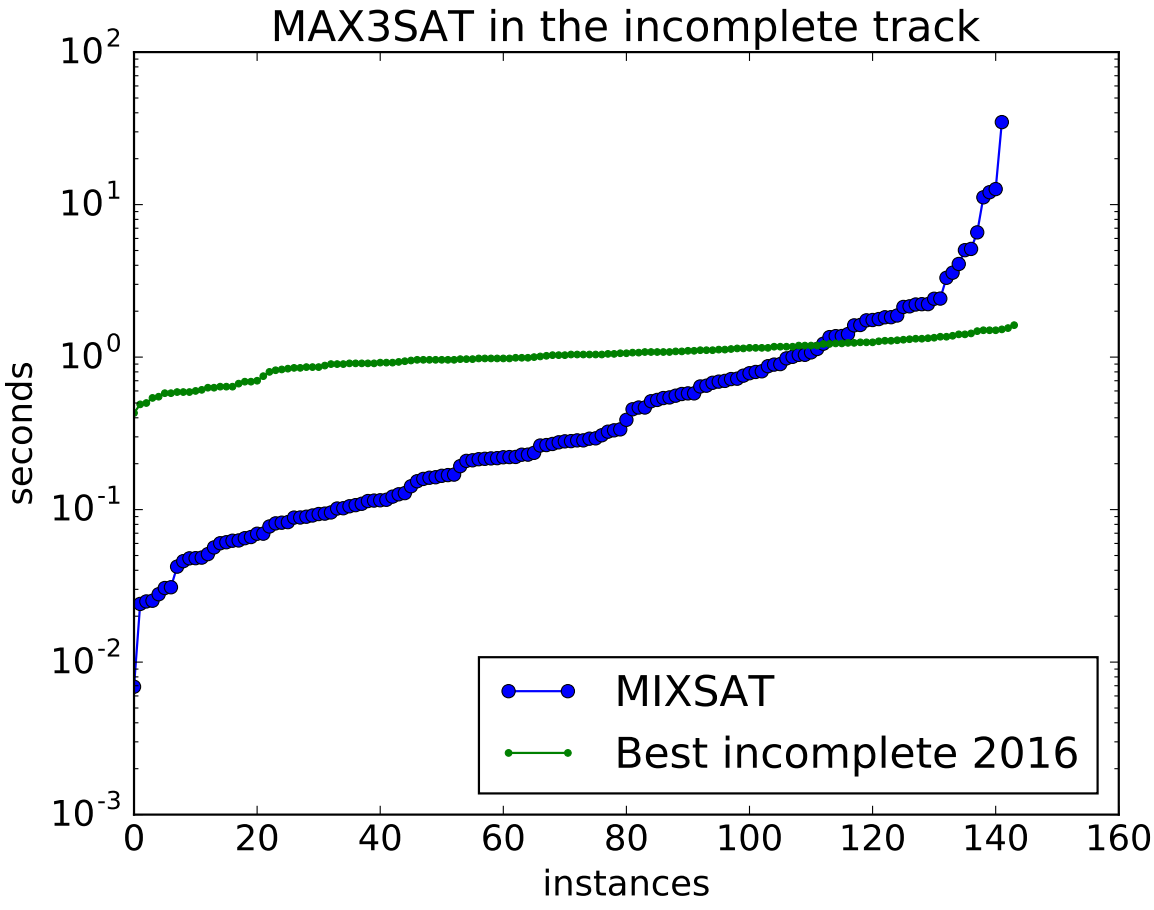
For the incomplete random categories, we solved all the instances in 0.22 sec (best solution), while the 2016 best solvers takes 1.92s

A 8.72x speedup!



MAX3SAT

For all the MAX3SAT instances in the incomplete random track, we solved all except 2 of the instance, and is faster than 2016 best solvers in 114/144 instances.



Summary

In this talk, we

- Presented the first **continuous MAXSAT solver** that outperforms the state-of-the-art discrete solvers in some subclasses
- Developed a lifting approximation and solve it via a low-rank SDP method (Mixing) that provably converges to the optimal
- Recovered the discrete optimal solution via developing the primal-dual initialization in a branch-and-bound framework

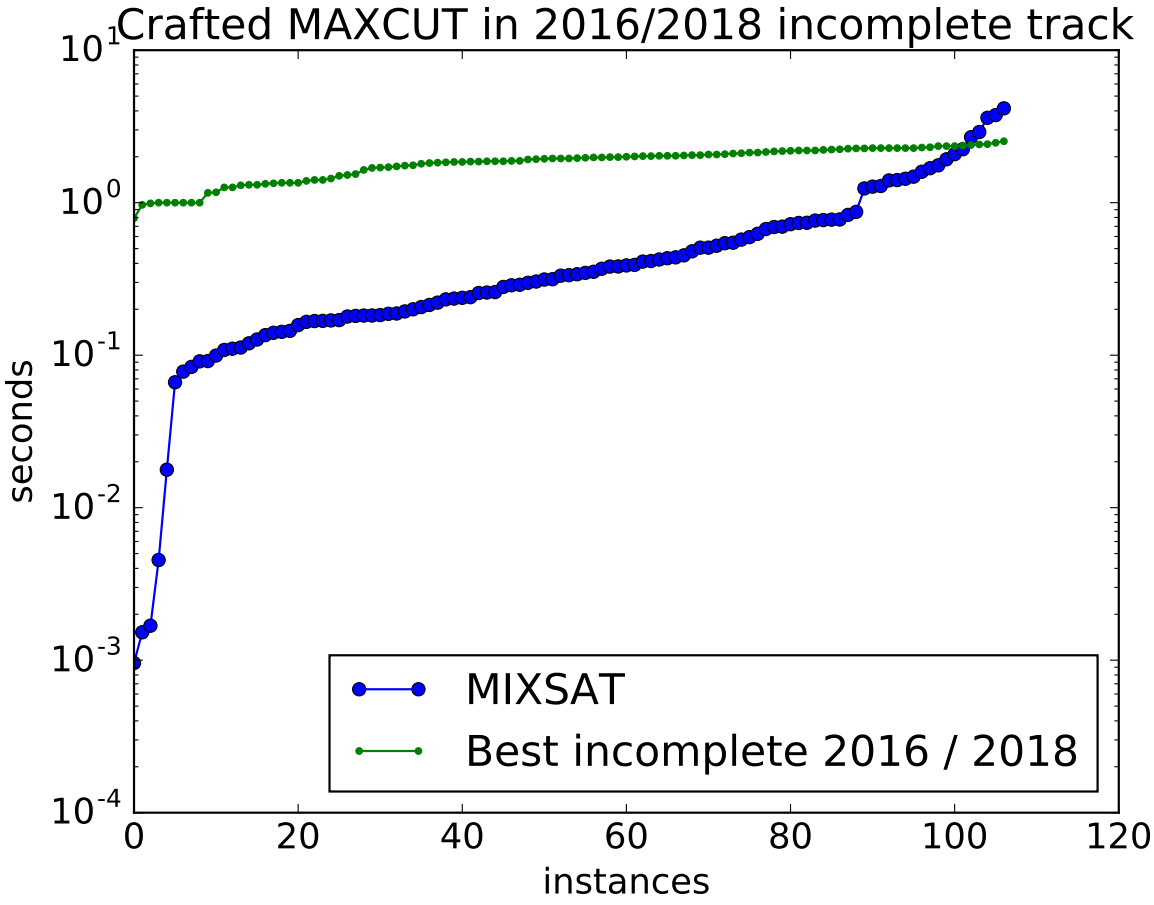
And we showed that

- The solver is many times faster than the best solvers in 2016 in some MAX2SAT categories.
- Also promising in MAX3SAT, but certainly not the best yet.
- It opens a new avenue for combinatorial optimization,
  taking the SDP out of the theoretical world!

Appendix: The MIXSAT algorithm

Initialize a priority queue $Q = \{\text{initial problem}\}$;
while *Q is not empty* **do**
    New SDP root P = Q.pop();
    Solve $f^* := \texttt{SDP}(P)$ with the Mixing method;
    if $\lceil f^* - \epsilon \rceil \geq$ *BEST* **then** continue;
    Update `BEST` and resolution orders by randomized rounding on $V := \arg \texttt{SDP}(P)$;
    foreach *subproblems of* P **do**
        Initialize `PRIMAL` and `DUAL` objective values;
        if *PRIMAL* $\leq$ *BEST* **then** Expand;
        else if $\lceil$*DUAL*$\rceil \geq$ *BEST* **then** Prune;
        else Push the subproblem into the Q;
    end
end

Appendix: Crafted MAXCUT instances



For all the MAXCUT instances in the incomplete crafted track in 2016/2018, we solved all the instances in 0.62s, while the 2016/2018 best solvers takes 1.84s.

Still 3x speedup.