# Hybrid SAT Solving by Continuous Optimization

Presented by Zhiwei Zhang

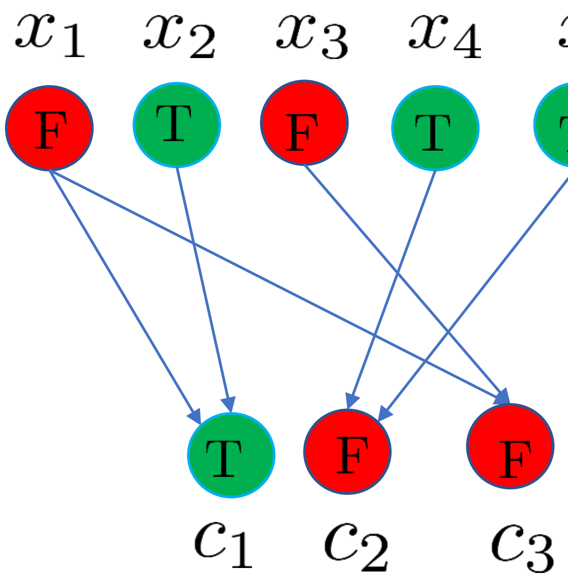
Joint work with Anastasios Kyrillidis, Anshumali Shrivastava, Moshe Vardi

# Boolean SAT and MaxSAT

Boolean variables: $x_1, x_2 \cdots \in \{0,1\}$

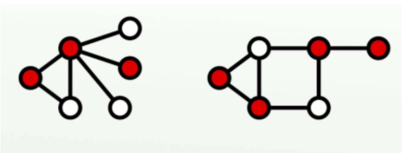
Boolean constraints: $\underline{x_1 \vee \neg x_2}$, $\underline{x_2 \oplus x_3}$, $\underline{x_1 + x_2 + x_3 + x_4 + x_5 \geq 2}, \cdots$

Boolean formula: e.g. $\varphi = (x_1 \vee \neg x_2) \wedge (x_2 \oplus \neg x_3) \wedge (x_2 + x_3 + x_4 \geq 2)$

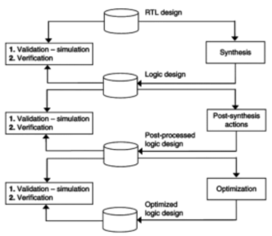



SATisfiability: finding an assignment that satisfies all constraints

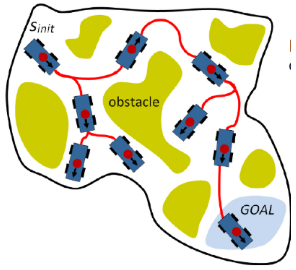
MaxSATisfiability: finding an assignment that satisfies as many constraints as possible



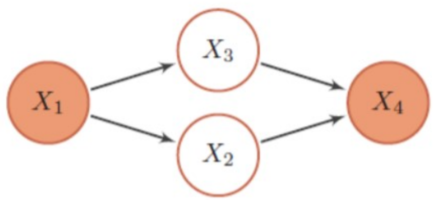
Discrete Optimization



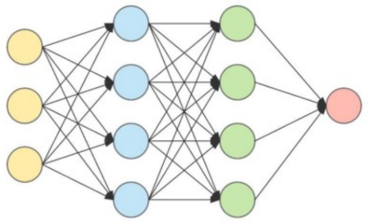

Software Verification




Motion planning




Probabilistic inference



Machine Learning

# CNF and SAT Solvers

- Conjunctive Normal Form (AND of ORs)

$$\text{e.g. } \varphi = \underline{(x_1 \vee x_2)} \wedge \underline{(x_2 \vee \neg x_3)} \wedge \underline{(x_2 \vee x_3 \vee x_4)}$$

- Modern CNF Solvers

CDCL-based SAT solvers: branching on variables with unit propogation, backtracking and clause learning

(discrete) local search SAT solvers: Objective function: $f_\varphi(x) = \#$ of constraints satisfied by $x$

(Greedy) Local Search

Randomly generate a complete assignment $x \in \{0,1\}^n$

while there are unsatisfied constraints

flip the value of the "best" variable to increase the # of satisfied constraints

# The Success of Existing Solvers Rely on Properties of CNF Format

- CNF is the most preferable format currently
  - CDCL solvers:

$$x_1 = T \xrightarrow{\text{constraint simplification}} (x_1 \vee x_2 \vee x_3 \vee x_4) \text{ is satisfied}$$

$$(x_1 \vee x_2 \vee x_3 \vee x_4) \text{ and } x_1 = x_2 = x_3 = F \xrightarrow{\text{unit propogation}} x_4 = T$$

  - Discrete local search solvers:

$$(x_1 \vee x_2 \vee x_3 \vee x_4) \text{ is unsatisfied} \longrightarrow \text{flipping one variable is enough}$$

- non-CNF constraints are harder to handle

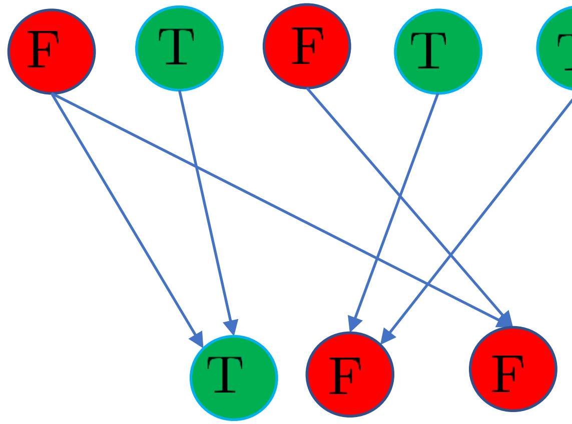
$$(x_1 + x_2 + x_3 + x_4 \geq 2) \text{ is unsatisfied} \longrightarrow \text{flipping one variable might not be enough}$$

The capability to “flip” more than one bits is important

# From Discrete to Continuous Local Search

$x = 0 \quad 1 \quad 0 \quad 1 \quad 1$

$x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5$




$c_1 \quad c_2 \quad c_3$

$a = 0.2 \quad 0.6 \quad 0.1 \quad 0.8 \quad 0.7$

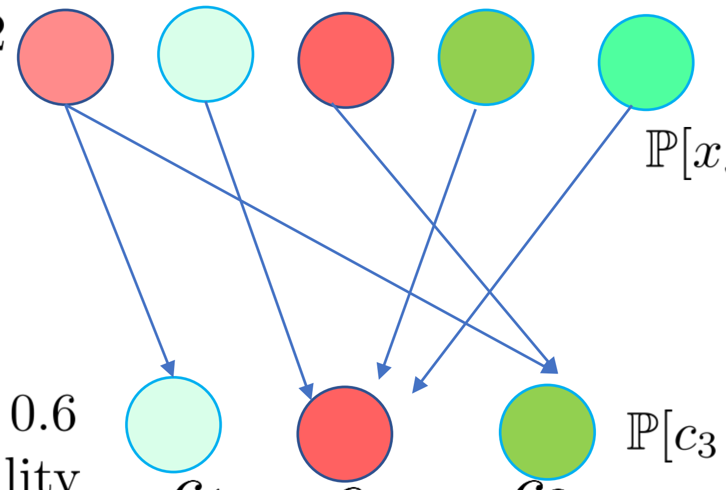
$x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5$

$S_a: \; \mathbb{P}[x_1 = 1] = 0.2$

$\mathbb{P}[x_1 = 0] = 0.8$

input probability

$\mathbb{P}[x_5 = 1] = 0.7$



$\mathbb{P}[c_1 = 1] = 0.6$

output probability

$\mathbb{P}[c_3 = 1] = 0.8$

$c_1 \quad c_2 \quad c_3$

$\mathbb{P}[c_2 = 1] = 0.1$

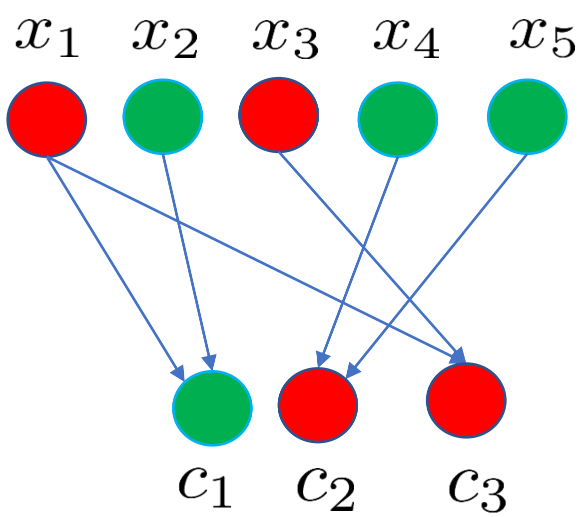
$$\begin{aligned} f_\varphi(x) &= \#\text{ of constraints satisfied by } x \\ &= c_1(x) + c_2(x) + c_3(x) \\ &= 1 + 0 + 0 = 1 \end{aligned}$$

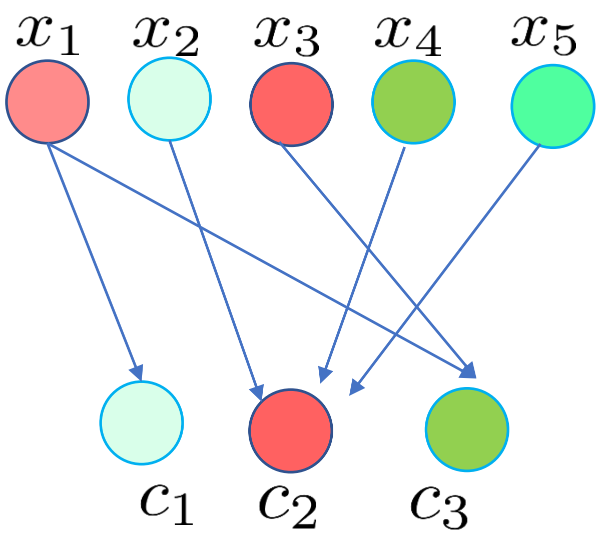
$$\begin{aligned} F_\varphi(a) &= \mathop{\mathbb{E}}_{x \in S_a} f_\varphi(x) \\ &= \sum_c \mathop{\mathbb{P}}_{x \in S_a}[c(x) = 1] \\ &= 0.6 + 0.1 + 0.8 = 1.5 \end{aligned}$$

"flipping" the value of variables to maximize $f_\varphi$

tuning the input probability of variables to maximize $F_\varphi$

# Reduce SAT to Continuous Optimization




"flipping" the value of variables to minimize $f$

tuning the input probability of variables to miminize $F$

$f_\varphi(x) = \#$ of constraints satisfied by $x$

$$F_\varphi(a) = \mathop{\mathbb{E}}_{x \in S_a} f_\varphi(x) = \sum_c \mathop{\mathbb{P}}_{x \in S_a} [c(x) = 1]$$

**Proposition**

$\varphi$ is satisfiable $\Longleftrightarrow \max\limits_{x \in \{0,1\}^n} f_\varphi(x) = \#constraints \Longleftrightarrow \max\limits_{a \in [0,1]^n} F_\varphi(a) = \#constraints$

# The Walsh-Fourier Expansion of Boolean Functions

- Walsh-Fourier transform:

Boolean functions → Multilinear Polynomials

$c : \{1, -1\}^n \to \{0, 1\}$ $\qquad$ $\mathbf{FE}_c : \{1, -1\}^n \to \{0, 1\}$

$\{F, T\}$ $\qquad$ $\{F, T\}$

$x \wedge y$ → $\frac{1}{4} \cdot 1 - \frac{1}{4} \cdot x - \frac{1}{4} \cdot y + \frac{1}{4} \cdot xy$

| $x$ | $y$ | $x \wedge y$ | $\frac{1}{4} - \frac{1}{4}x - \frac{1}{4}y + \frac{1}{4}xy$ |
|---|---|---|---|
| $1$ | $1$ | $0$ | $0$ |
| $1$ | $-1$ | $0$ | $0$ |
| $-1$ | $1$ | $0$ | $0$ |
| $-1$ | $-1$ | $1$ | $1$ |

Theorem
Every Boolean function c has a unique representation in multilinear polynomial that agrees with c on all Boolean assignments

# How to compute expectation: by Walsh-Fourier expansion

$$c = x \wedge y \longrightarrow \texttt{FE}_c = \frac{1}{4} \cdot 1 - \frac{1}{4} \cdot x - \frac{1}{4} \cdot y + \frac{1}{4} \cdot xy$$

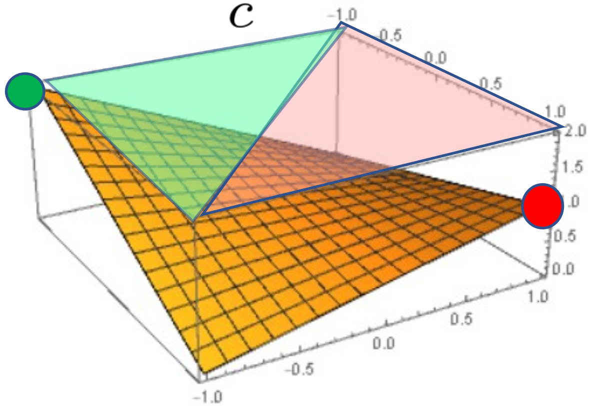
**Proposition**

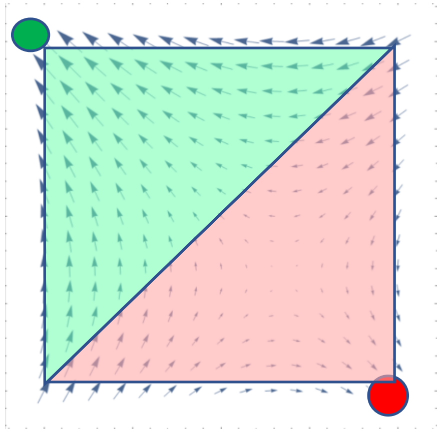
$$\mathop{\mathbb{P}}_{x \in S_a} [c(x) = 1] = \texttt{FE}_c(a) \text{ for all } a \in [-1, 1]^n$$

$$F_\varphi(a) = \mathop{\mathbb{E}}_{x \in S_a} f_\varphi(x) = \sum_c \mathop{\mathbb{P}}_{x \in S_a} [c(x) = 1] = \sum_c \texttt{FE}_c(a)$$

$$\varphi = (x_1 \wedge x_2) \wedge (\neg x_1 \oplus x_2)$$

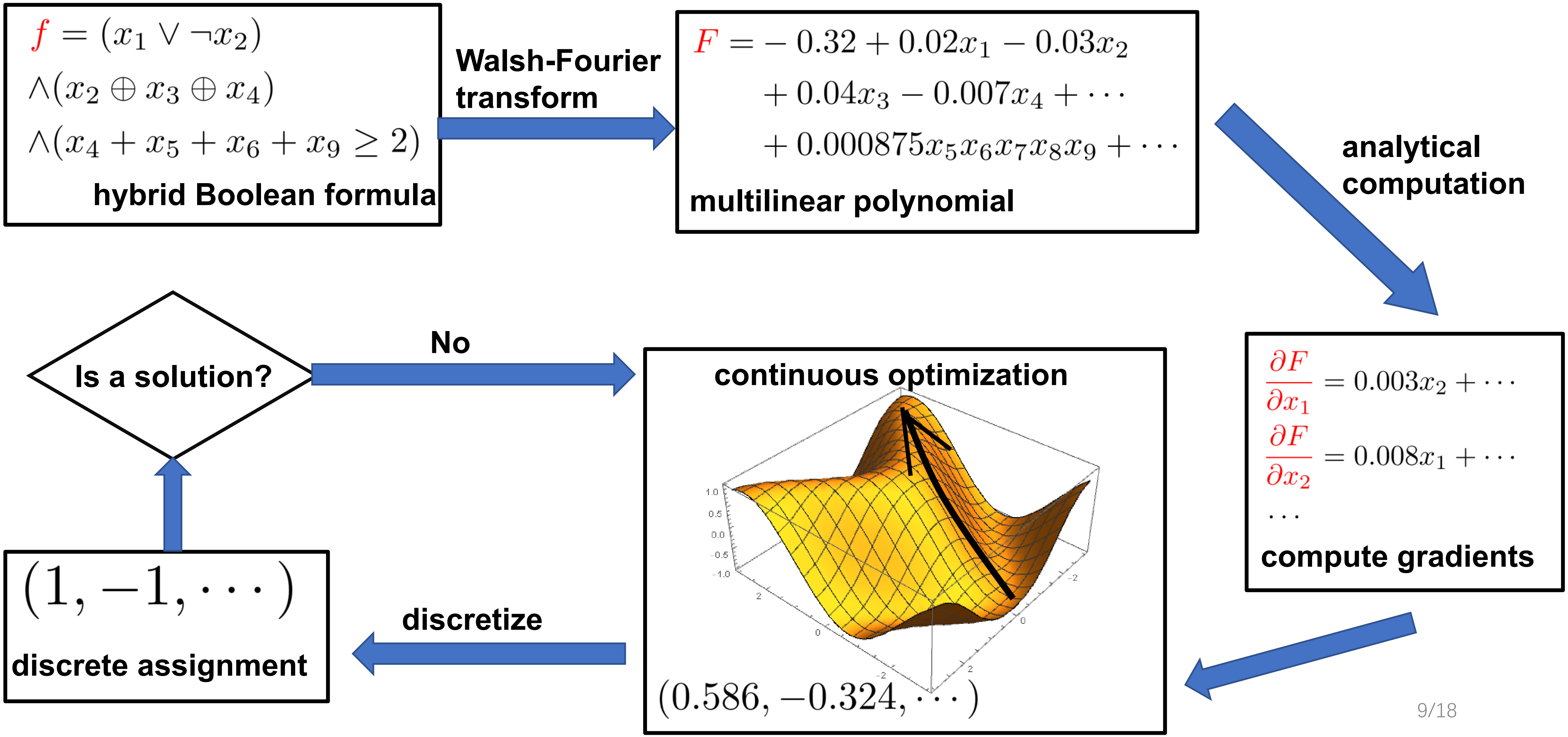
$$F_\varphi = \boxed{\left(\frac{1}{4} \cdot 1 - \frac{1}{4} \cdot x_1 - \frac{1}{4} \cdot x_2 + \frac{1}{4} \cdot x_1 x_2\right)} + \boxed{\frac{1 + x_1 x_2}{2}}$$





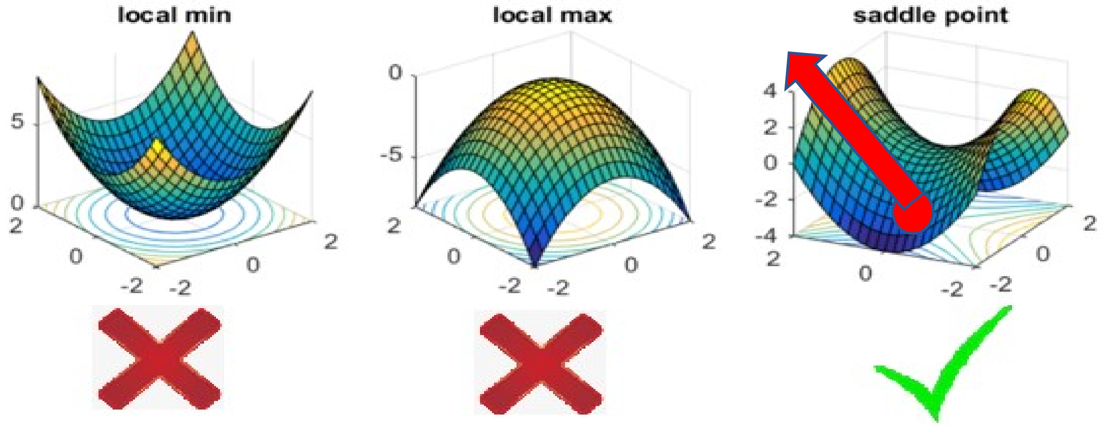
Apply gradient ascent

# Workflow of Our Gradient Ascend-Based Approach



# Where Will We Converge to? The Geometry of Multilinear Polynomials

- Multilinear polynomials are non-convex and non-concave




- Locally, $F_\varphi$ is always convex among some directions while concave on some other directions

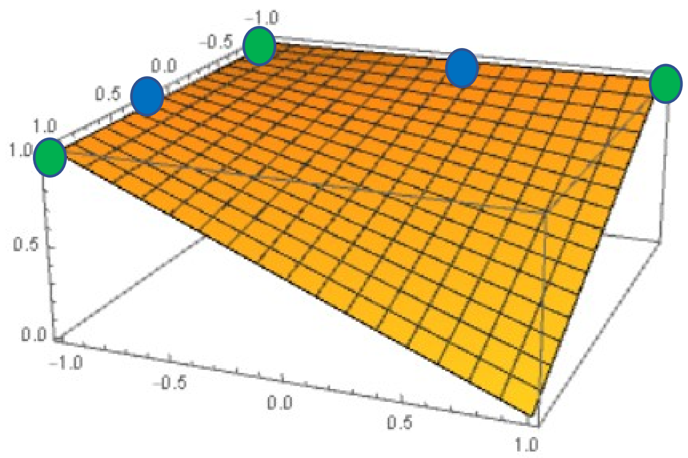
  In unconstrained setting, on every point $a \in \mathbb{R}^n$, there is always a direction that can increase the value of a multilinear polynomial

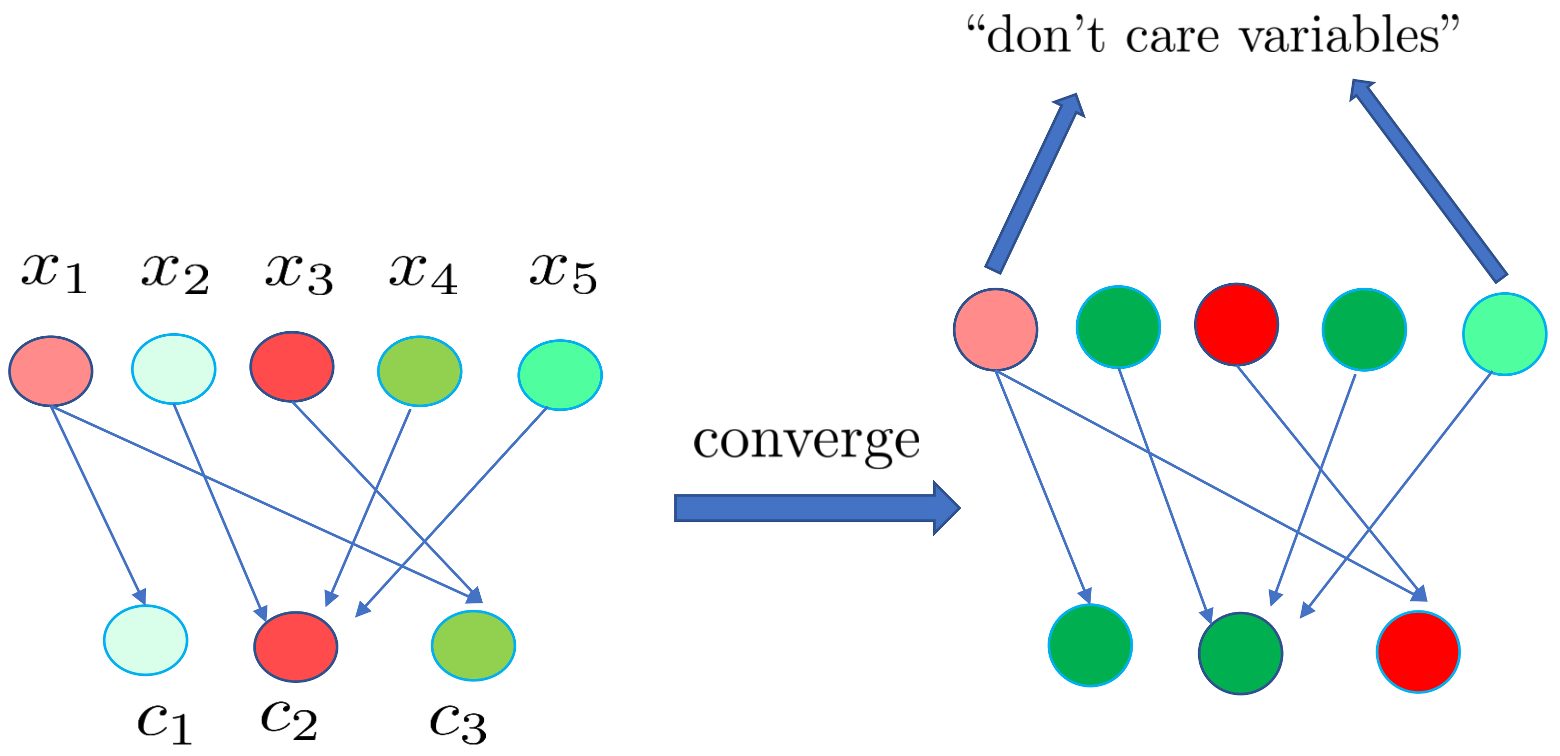
  In the constrained setting where $a \in [-1, 1]^n$, this direction might be towards the boundary. Thus we may still encounter local maxima along the boundary

# Where Will We Converge to? An "Almost" Discrete Assignment

$$\varphi = x \vee y$$

$$F_\varphi = \frac{3}{4} - \frac{1}{4}x - \frac{1}{4}y - \frac{1}{4}xy$$




**Theorem**
Rounding preserves objective value after converging to a local maximum.

# The Versatility of Our Approach

- Modern-SAT solvers are highly CNF-focused

- Other types of constraints are also important

  XOR: $(x_2 \oplus \neg x_3)$

  cardinality constraints: $(x_1 + x_2 + x_3 + x_4 \geq 2)$

  pseudo-Boolean constraints: $(3x_1 - 4x_2 + x_3 + 6x_4 \geq 5)$

> Theorem
>
> Disjunctive clauses (CNF), XOR and cardinality constraints all have closed-form Walsh-Fourier expansions.

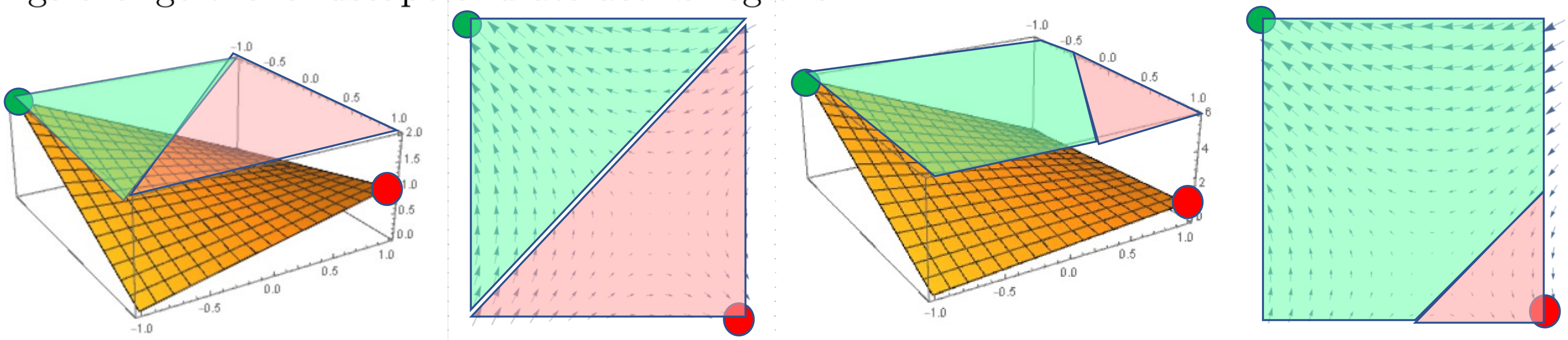
- Our approach treats different types of constraints uniformly as polynomials

# Better Global Convergence--Adding Constraint Weights

$$F_\varphi(a) = \sum_c \underset{x \in S_a}{\mathbb{P}}[c(x)=1] = \sum_c \mathtt{FE}_c(a) \quad \Rightarrow \quad F_{\varphi,w}(a) = \sum_c w(c) \cdot \underset{x \in S_a}{\mathbb{P}}[c(x)=1] = \sum_c w(c) \cdot \mathtt{FE}_c(a)$$

- Different constraints have different relative importance

  Weightings change the landscape and attractive regions



$$\boxed{\left(\tfrac{1}{4} \cdot 1 - \tfrac{1}{4} \cdot x_1 - \tfrac{1}{4} \cdot x_2 + \tfrac{1}{4} \cdot x_1 x_2\right)} + \boxed{\tfrac{1+x_1x_2}{2}} \qquad 5 \cdot \boxed{\left(\tfrac{1}{4} \cdot 1 - \tfrac{1}{4} \cdot x_1 - \tfrac{1}{4} \cdot x_2 + \tfrac{1}{4} \cdot x_1 x_2\right)} + \boxed{\tfrac{1+x_1x_2}{2}}$$

$$\varphi = (x_1 \wedge x_2) \wedge (\neg x_1 \oplus x_2)$$

Adaptative weighting: increase the weight of unsatisfied constraints

# Better Versatility--Beyond Walsh-Fourier Expansions

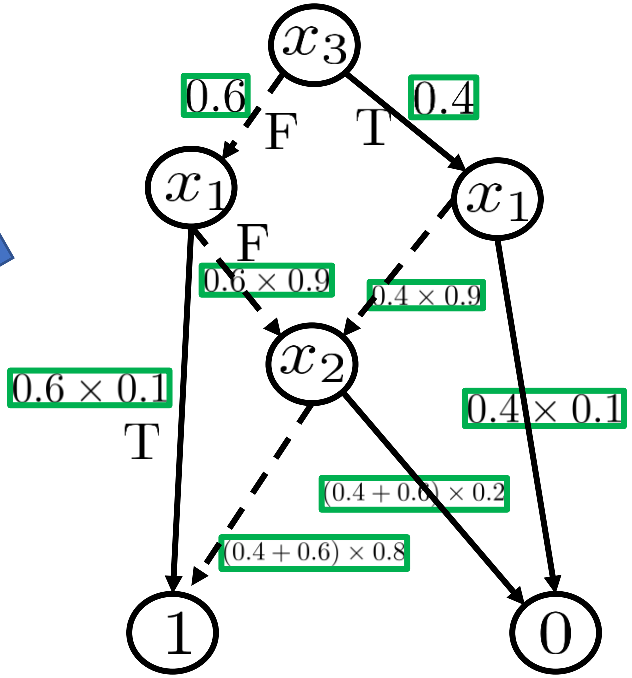
$$x \wedge y$$

| $x$ | $y$ | $x \wedge y$ |
|---|---|---|
| F | F | F |
| F | T | F |
| T | F | F |
| T | T | T |

Truth Table

$$\frac{1}{4} \cdot 1 - \frac{1}{4} \cdot x - \frac{1}{4} \cdot y + \frac{1}{4} \cdot xy$$

Walsh-Fourier Expansion




Binary decision diagram (BDD)

Walsh-Fourier expansions can not handle pseudo-Boolean constraints:
e.g. $(3x_1 - 4x_2 + x_3 + 6x_4 \geq 5)$

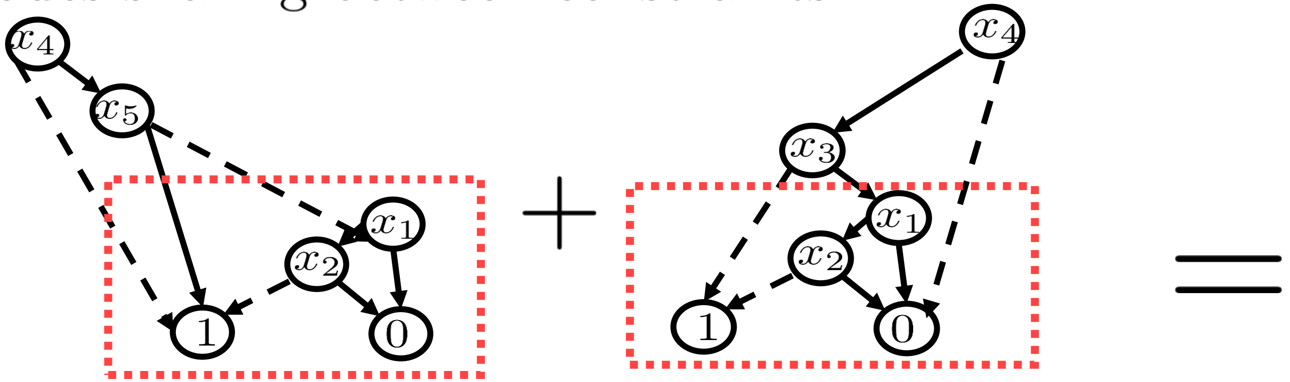
**Proposition**

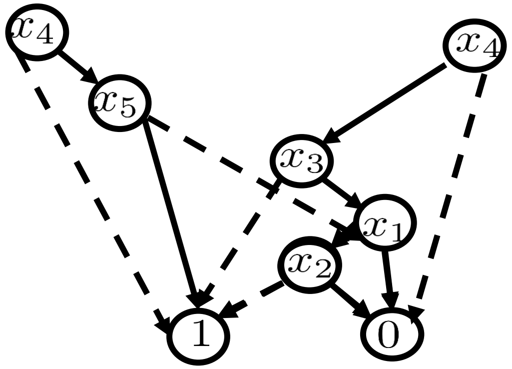
Given the BDD with size $S$ of a constraint, the output probability can be computed in $O(S)$.

- We are able to handle coefficient-bounded pseudo-Boolean constraints.

# Better Efficiency--Computing the Gradient by BDDs

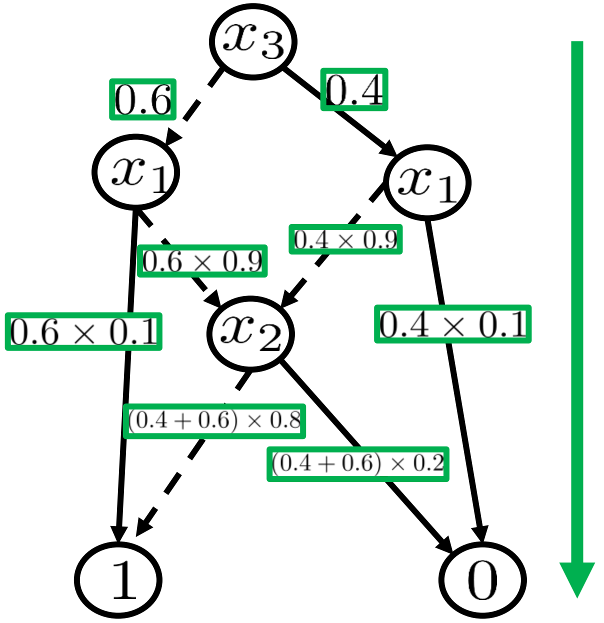
- Nodes sharing between constraints



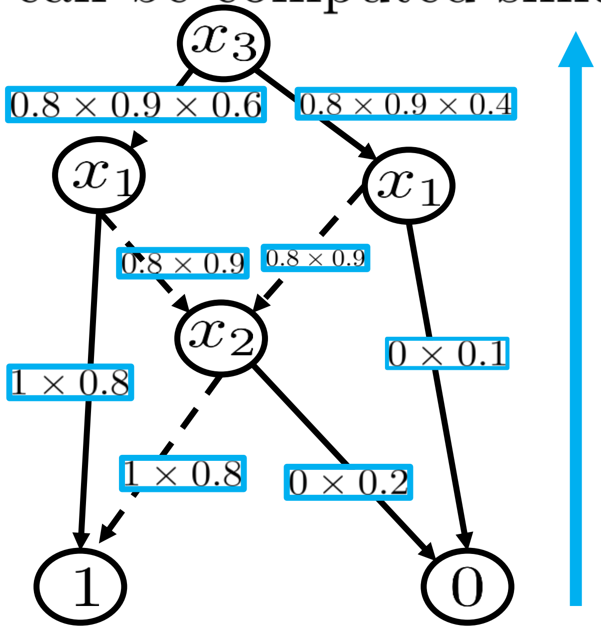




Multi-Rooted BDD

- Gradients of all coordinates can be computed simutaneously




Forward Pass




Backward Pass

**Theorem**

Given the Multi-Rooted BDD with total size $S$, the gradient can be computed in $O(S)$.

# Experimental Results

| Solver | Avg. Score | # of best solutions |
|---|---|---|
| GradSAT (Our approach) | 0.971 | 489 |
| WalkSAT (discrete LS-based) | 0.925 | 124 |
| Mixing Method (SDP-based) | 0.901 | 126 |
| Loandra (SAT-based) | 0.883 | 42 |

Results on 575 small-size instances from MaxSAT Competition

On random CNF-XOR and pseudo-Boolean benchmarks, our solver is better than discrete local-search-based solvers.

On large industrial instances: the cost for differentiation on the real domain is still too expensive

# Summary: Hybrid SAT Solving by Continuous Optimization

- Our motivations are

  handling constraints beyond CNFs

  exploring the potential of continuous methods in SAT/MaxSAT solving

- We find

  nice theoretical results

  interesting problems

  for the continuous optimization approach

- Our method can:

  act as a complement to existing SAT/MaxSAT solvers

  benefit from tractable structures of Boolean functions

  and techniques from continuous optimization

# Challenges and Future Directions

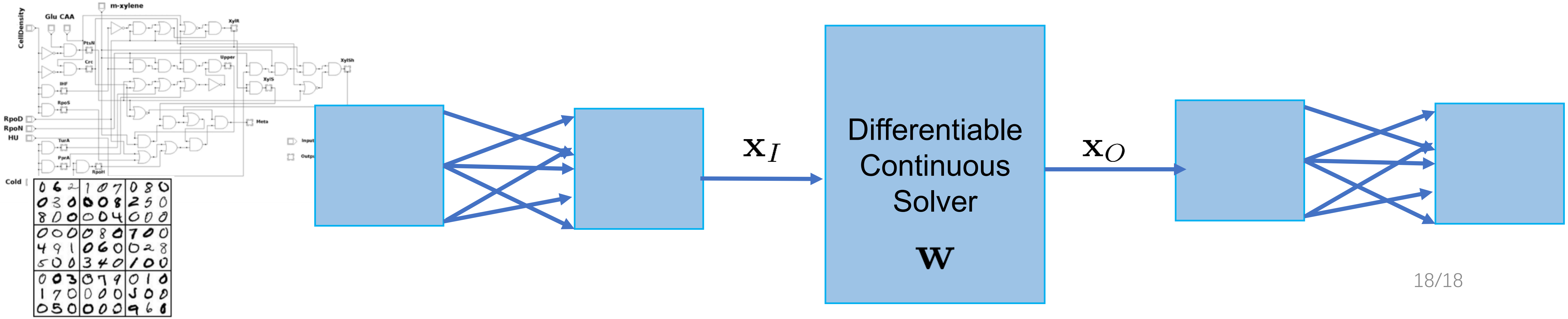
- Full gradient is too expensive

  compute imprecise gradient on large instances

- Balance between Continuous and Discrete Local Search

Fully Discrete Local Search  Fully Continuous Local Search

- Continuous optimizer as a layer of NN



- Our motivations are

  handling constraints beyond CNFs

  exploring the potential of continuous methods in SAT/MaxSAT solving

- We find

  nice theoretical results
  interesting problems
  for the continuous optimization approach

- Our method can:

  act as a complement to existing SAT solvers

  benefit from tractable structures of Boolean functions
  and techniques from continuous optimization

- In the future:

  compute inprecise gradient on large instances

  balance between Continuous and Discrete

  as a layer of neural network