The MaxSAT problem in the real-valued MV-algebra

Zuzana Haniková[1], Felip Manyà[2], Amanda Vidal[2]

[1]Institute of Computer Science of the CAS, Prague, Czech Republic

[2]Artificial Intelligence Research Institute, CSIC, Bellaterra, Spain

Tableaux 2023, Prague, September 21

Boolean maximum satisfiability

Consider a CNF formula

$$\varphi \coloneqq C_1 \wedge C_2 \wedge \ldots C_m$$

where C_i are clauses.

- Is φ satisfiable?
- How many clauses can be satisfied by a single assignment?

Boolean maximum satisfiability

Consider a CNF formula

$$\varphi := C_1 \wedge C_2 \wedge \dots C_m$$

where C_i are clauses.

- Is φ satisfiable?
- How many clauses can be satisfied by a single assignment?

Classical-MaxSAT

Instance: multiset $\langle C_1, \dots, C_m \rangle$ of Boolean clauses (variables $x_1, \dots, x_n$)

Output: maximum integer $k \leq m$ such that there is assignment v in the two-element Boolean algebra $\{0, 1\}$ to $\{x_1, \dots, x_n\}$ that satisfies k clauses.

MV-algebras

Language $\mathcal{L}$ of Łukasiewicz logic.

Basic function symbols $\{\oplus, \neg\}$.

$Fm(\mathcal{L})$ set of well-formed formulas of $\mathcal{L}$.

Some definable symbols: $x \rightarrow y$ is $\neg x \oplus y$; $x \odot y$ is $\neg(\neg x \oplus \neg y)$;
$x \vee y$ is $(x \rightarrow y) \rightarrow y$; 1 is $x \rightarrow x$; 0 is $\neg 1$; ...

nx is $\underbrace{x \oplus \cdots \oplus x}_{n \text{ times}}$.

MV-algebras

Language $\mathcal{L}$ of Łukasiewicz logic.

Basic function symbols $\{\oplus, \neg\}$.

$Fm(\mathcal{L})$ set of well-formed formulas of $\mathcal{L}$.

Some definable symbols: $x \to y$ is $\neg x \oplus y$; $x \odot y$ is $\neg(\neg x \oplus \neg y)$;
$x \vee y$ is $(x \to y) \to y$; 1 is $x \to x$; 0 is $\neg 1$; ...

nx is $\underbrace{x \oplus \cdots \oplus x}_{n \text{ times}}$.

Real-valued MV-algebra $[0,1]_Ł$.

Domain: interval $[0,1]$ of the reals (usual order).

Interpretation of function symbols: for an assigment v,

$$v(x \oplus y) = \min(1, v(x) + v(y))$$
$$v(\neg x) = 1 - v(x)$$

Intended semantics of Łukasiewicz logic.

The subalgebra on $\{0,1\}$ is isomorphic to the two-element Boolean algebra.

[Łukasiewicz 1922; Łukasiewicz and Tarski 1930; Chang 1958, 1959]

Satisfiability in $[0,1]_Ł$

Consider an MV-algebra A. The only designated value is 1^A.

$$\mathrm{SAT}(A) = \{\varphi \in Fm(\mathcal{L}) \mid \exists v_A \, (v_A(\varphi) = 1^A)$$

(Notice φ is arbitrary.)

Write just SAT in case A is $[0,1]_Ł$.

Satisfiability in $[0,1]_Ł$

Consider an MV-algebra A. The only designated value is 1^A.

$$\mathrm{SAT}(A) = \{\varphi \in Fm(\mathcal{L}) \mid \exists v_A\, (v_A(\varphi) = 1^A)$$

(Notice φ is arbitrary.)

Write just SAT in case A is $[0,1]_Ł$.

SAT

Instance: formula φ of $\mathcal{L}$.

Output: (Boolean) Is φ satisfiable in $[0,1]_Ł$?

Satisfiability in $[0,1]_Ł$

Consider an MV-algebra A. The only designated value is 1^A.

$$\mathrm{SAT}(A) = \{\varphi \in Fm(\mathcal{L}) \mid \exists v_A\, (v_A(\varphi) = 1^A)$$

(Notice φ is arbitrary.)

Write just SAT in case A is $[0,1]_Ł$.

SAT

Instance: formula φ of $\mathcal{L}$.

Output: (Boolean) Is φ satisfiable in $[0,1]_Ł$?

SAT is NP-complete:

- bounding the denominator of assignments [Mundici 1987; (Aguzzoli, Ciabattoni, . . .)]
- reduction to mixed integer programming [Hähnle 1994; Olivetti 2003; . . .]

Maximum satisfiability in $[0,1]_{Ł}$

MaxSAT-OPT

Instance: multiset $\langle \varphi_1, \ldots, \varphi_m \rangle$ of formulas of $\mathcal{L}$ in variables $x_1, \ldots, x_n$.

Output: maximum integer $k \leq m$ such that there is assignment v to $\{x_1, \ldots, x_n\}$ in $[0,1]_{Ł}$ that satisfies at least k formulas in the multiset.

MaxSAT-DEC

Instance: multiset $\langle \varphi_1, \ldots, \varphi_m \rangle$ of formulas of $\mathcal{L}$ in variables $x_1, \ldots, x_n$, and integer $1 \leq k \leq m$.

Output: (Boolean) Is **MaxSAT-OPT**$\langle \varphi_1, \ldots, \varphi_m \rangle$ at least k?

Decision version of **MaxSAT**

Theorem

MaxSAT-DEC *is* NP*-complete.*

Decision version of **MaxSAT**

Theorem

MaxSAT-DEC *is* NP*-complete.*

1. For $k = m = 1$ the problem coincides with **SAT**.
2. For $2 \leq k \leq m$,

$$(\langle \varphi_1, \ldots, \varphi_m \rangle, k) \in \textbf{MaxSAT-DEC} \text{ iff } \{\rho_{1/k}\} \cup \Phi_k \cup \{\bigoplus_{i=1}^{m} y_{i,k}\} \in \textbf{SAT}$$

where

$$\rho_{1/k} := \quad y \leftrightarrow \neg((k-1)y)$$

and Φ_k collects all the formulas

$$\{ \ (\varphi_i \leftrightarrow k\, y_{i,k}) \vee \neg y_{i,k} \quad , \quad (y_{i,k} \leftrightarrow y) \vee \neg y_{i,k} \ \}$$

for $1 \leq i \leq m$.

Oracle computation of MaxSAT

Assume an algorithm **A** for **MaxSAT-DEC** ("oracle").
Recall: input $\langle \varphi_1, \ldots, \varphi_m \rangle$ and k; output: 'yes' / 'no'.

Search space: $\{0, 1, \ldots, m\}$.

Oracle computation of MaxSAT

Assume an algorithm **A** for **MaxSAT-DEC** ("oracle").
Recall: input $\langle \varphi_1, \ldots, \varphi_m \rangle$ and k; output: 'yes' / 'no'.

Search space: $\{0, 1, \ldots, m\}$.

Binary search using $O(\log m)$ oracle calls.

Lemma

MaxSAT-OPT *is in* $\mathrm{FP}^{\mathrm{NP}}[\log m]$.
(♯ of calls to NP*-complete oracle is logarithmic in ♯ of formulas).*

Inside $\mathrm{FP}^{\mathrm{NP}}$

$\mathrm{FP}^{\mathrm{NP}}[z(n)]$ is the class of functions computable in P-time with NP oracle with at most $z(|x|)$ oracle calls on input x.

(In particular, $\mathrm{FP}^{\mathrm{NP}} = \mathrm{FP}^{\mathrm{NP}}[n^{O(1)}]$.)

Inside $\mathrm{FP}^{\mathrm{NP}}$

$\mathrm{FP}^{\mathrm{NP}}[z(n)]$ is the class of functions computable in P-time with NP oracle with at most $z(|x|)$ oracle calls on input x.

(In particular, $\mathrm{FP}^{\mathrm{NP}} = \mathrm{FP}^{\mathrm{NP}}[n^{O(1)}]$.)

Let $f, g : \Sigma^* \to \mathbb{N}$.
A metric reduction of f to g is a pair (h_1, h_2) of P-time functions
(with $h_1 : \Sigma^* \to \Sigma^*$ and $h_2 : \Sigma^* \times N \to N$)
such that $f(x) = h_2(x, g(h_1(x)))$ for each $x \in \Sigma^*$.

Inside $\mathrm{FP}^{\mathrm{NP}}$

$\mathrm{FP}^{\mathrm{NP}}[z(n)]$ is the class of functions computable in P-time with NP oracle with at most $z(|x|)$ oracle calls on input x.

(In particular, $\mathrm{FP}^{\mathrm{NP}} = \mathrm{FP}^{\mathrm{NP}}[n^{O(1)}]$.)

Let $f, g : \Sigma^* \to \mathbb{N}$.
A metric reduction of f to g is a pair (h_1, h_2) of P-time functions (with $h_1 : \Sigma^* \to \Sigma^*$ and $h_2 : \Sigma^* \times N \to N$) such that $f(x) = h_2(x, g(h_1(x)))$ for each $x \in \Sigma^*$.

Theorem [Krentel 1988]

Assume $\mathrm{P} \neq \mathrm{NP}$. *Then*
$\mathrm{FP}^{\mathrm{NP}}[O(\log\log n)] \neq \mathrm{FP}^{\mathrm{NP}}[O(\log n)] \neq \mathrm{FP}^{\mathrm{NP}}[n^{O(1)}]$.

No metric reductions from, e.g.,
$\mathrm{FP}^{\mathrm{NP}}[O(\log n)]$-complete problems to problems in $\mathrm{FP}^{\mathrm{NP}}[O(\log\log n)]$.

[Krentel: Complexity of optimization problems. J. Comp. System Sci. 36, 1988]

Complexity of **MaxSAT-OPT**

Theorem

MaxSAT-OPT *is complete for* $\mathrm{FP}^{\mathrm{NP}}[O(\log m)]$ *under metric reductions.*

Complexity of **MaxSAT-OPT**

Theorem

MaxSAT-OPT *is complete for* $\mathrm{FP}^{\mathrm{NP}}[O(\log m)]$ *under metric reductions.*

Proof idea:

Classical-MaxSAT reduces to **MaxSAT-OPT** via
a pair of identity functions.

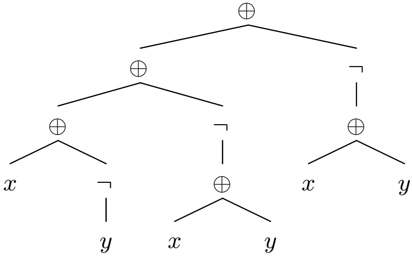
Boolean language is $\{\neg, \vee, \odot\}$ and CNF's define convex functions in $[0, 1]_{Ł}$.

Tseitin transformation

$\varphi := ((x \oplus \neg y) \oplus \neg(x \oplus y)) \oplus \neg(x \oplus y)$. Is $\varphi \in \mathbf{SAT}$?

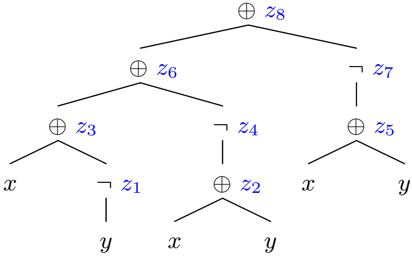
Tseitin transformation

$\varphi := ((x \oplus \neg y) \oplus \neg(x \oplus y)) \oplus \neg(x \oplus y)$. Is $\varphi \in \mathbf{SAT}$?



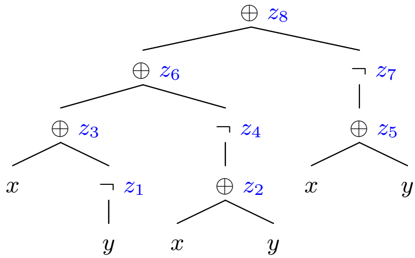
Tseitin transformation

$\varphi := ((x \oplus \neg y) \oplus \neg(x \oplus y)) \oplus \neg(x \oplus y)$. Is $\varphi \in \mathbf{SAT}$?



Tseitin transformation

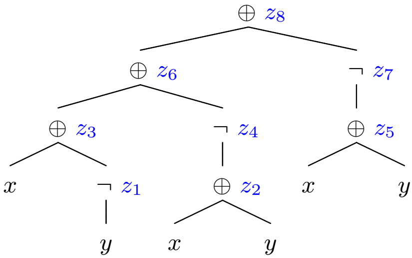
$\varphi := ((x \oplus \neg y) \oplus \neg(x \oplus y)) \oplus \neg(x \oplus y)$. Is $\varphi \in \mathbf{SAT}$?



$z_1 = \neg y$; $z_2 = x \oplus y$; ....; $z_8 = z_6 \oplus z_7$;

Tseitin transformation

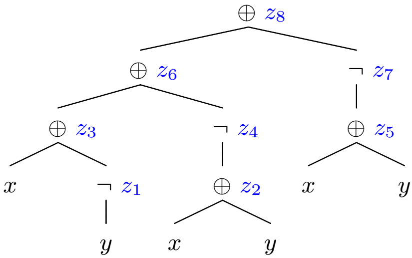
$\varphi := ((x \oplus \neg y) \oplus \neg(x \oplus y)) \oplus \neg(x \oplus y)$. Is $\varphi \in \mathbf{SAT}$?



$z_1 = \neg y$; $z_2 = x \oplus y$; ....; $z_8 = z_6 \oplus z_7$; $z_8 = 1$.

Tseitin transformation

$\varphi := ((x \oplus \neg y) \oplus \neg(x \oplus y)) \oplus \neg(x \oplus y)$. Is $\varphi \in$ **SAT**?



$z_1 = \neg y$; $z_2 = x \oplus y$; ....; $z_8 = z_6 \oplus z_7$; $z_8 = 1$.

Polynomial increase in length. New variables. Preserves satisfiability.

Example: repeating subformulas

$$\alpha := \underbrace{y \oplus y \oplus y \oplus \quad \ldots \quad \oplus y}_{k \text{ times}}$$

Example: repeating subformulas

$$\alpha := \underbrace{y \oplus y \oplus y \oplus \quad \ldots \quad \oplus y}_{k \text{ times}}$$

Let $||\varphi||$ be $\sharp$ of pairwise distinct subformulas in φ.

$$||\alpha|| \text{ proportional to: } \begin{cases} k \text{ if brackets nest to the right / left} \\ \log_2 k \text{ if brackets form a balanced tree} \end{cases}$$

Example: repeating subformulas

$$\alpha := \underbrace{y \oplus y \oplus y \oplus \quad \dots \quad \oplus y}_{k \text{ times}}$$

Let $||\varphi||$ be $\sharp$ of pairwise distinct subformulas in φ.

$$||\alpha|| \text{ proportional to: } \begin{cases} k \text{ if brackets nest to the right / left} \\ \log_2 k \text{ if brackets form a balanced tree} \end{cases}$$

Subformulas will be taken as a set.

Decision method for **SAT**

Input: $\varphi(x_1, \ldots, x_n)$.

1. List **L** of pairwise distinct subformulas in φ.
2. **New variables** z_i for i-th **subformula** in **L**.
3. **Tseitin equations**: list **S** of equations of the form $z_i = x$ or $z_i = \neg z_j$ or $z_i = z_j \oplus z_k$. Notice $|S| = |\mathbf{L}|$.

Decision method for SAT

Input: $\varphi(x_1, \ldots, x_n)$.

1. List **L** of pairwise distinct subformulas in φ.
2. **New variables** z_i for i-th **subformula** in **L**.
3. **Tseitin equations**: list **S** of equations of the form $z_i = x$ or $z_i = \neg z_j$ or $z_i = z_j \oplus z_k$. Notice $|S| = |\mathbf{L}|$.
4. Initialize **rooted linear tree T**. From root down, label each node of **T** with one equation from **S**.
5. **Boundary constraints** $0 \leq z_i$, $z_i \leq 1$.
6. **Target constraint** $z_l = 1$ where z_l is variable for φ.
7. **Expand** nodes with symbols of $\mathcal{L}$ using the rules:

$$\frac{z_i \oplus z_j = z_k}{\begin{array}{c|c} z_i + z_j \leq 1 & z_i + z_j \geq 1 \\ z_i + z_j = z_k & z_k = 1 \end{array}} \qquad \frac{z_i = \neg z_j}{z_i = 1 - z_j}$$

Each branch of **T** then defines a system of linear constraints in $\mathbb{R}$.

Decision method for **SAT**

Input: $\varphi(x_1, \ldots, x_n)$.

1. List **L** of pairwise distinct subformulas in φ.
2. **New variables** z_i for i-th **subformula** in **L**.
3. **Tseitin equations**: list **S** of equations of the form $z_i = x$ or $z_i = \neg z_j$ or $z_i = z_j \oplus z_k$. Notice $|S| = |\mathbf{L}|$.
4. Initialize **rooted linear tree T**. From root down, label each node of **T** with one equation from **S**.
5. **Boundary constraints** $0 \leq z_i$, $z_i \leq 1$.
6. **Target constraint** $z_l = 1$ where z_l is variable for φ.
7. **Expand** nodes with symbols of $\mathcal{L}$ using the rules:

$$\frac{z_i \oplus z_j = z_k}{\begin{array}{c|c} z_i + z_j \leq 1 & z_i + z_j \geq 1 \\ z_i + z_j = z_k & z_k = 1 \end{array}} \qquad \frac{z_i = \neg z_j}{z_i = 1 - z_j}$$

   Each branch of **T** then defines a system of linear constraints in $\mathbb{R}$.
8. Left to right, **test system on each branch** for solvability in $\mathbb{R}$. If branch is found with solvable system, return **'yes'** and exit.
9. **Default.** Return **'no'** and exit.

cf. [Hähnle 1994]

Linearization

Lemma

Assume $a_1, a_2, a_3 \in [0,1]$. *Then* $a_1 \oplus a_2 = a_3$ *holds in* $[0,1]_{Ł}$ *if and only if there is an* $y \in \{0,1\}$ *such that all of the following constraints hold in* $\mathbb{R}$*:*

(i) $a_1 + a_2 \leq 1 + y$
(ii) $y \leq a_1 + a_2$
(iii) $a_3 \leq a_1 + a_2$
(iv) $a_1 + a_2 \leq a_3 + y$
(v) $y \leq a_3$.

[Hähnle 1994, Hájek 1998]

Computing **MaxSAT-OPT**: first remarks

Consider the multiset $\langle \alpha, \ldots, \alpha \rangle$, with $m > 1$.

If $\alpha \in \mathbf{SAT}$,
a sound and complete method ought to produce answer m on this input.

Computing **MaxSAT-OPT**: first remarks

Consider the multiset $\langle \alpha, \ldots, \alpha \rangle$, with $m > 1$.

If $\alpha \in$ **SAT**,
a sound and complete method ought to produce answer m on this input.

Keep the procedure bringing input to Tseitin normal form.

- Update target constraint: multiset on input.
- Update method of reading output from the tree.

Computing **MaxSAT-OPT**

Input: $\langle \varphi_1, \dots, \varphi_m \rangle$ in variables $x_1, \dots, x_n$.

1. List **L** of pairwise distinct subformulas in $\varphi_1, \dots, \varphi_m$.
2. **New variables** z_i — as before.
3. **Tseitin equations** — as before.
4. Initialize **rooted linear tree** — as before.
5. **Boundary constraints** — as before.

Computing **MaxSAT-OPT**

Input: $\langle \varphi_1, \ldots, \varphi_m \rangle$ in variables $x_1, \ldots, x_n$.

1. List **L** of pairwise distinct subformulas in $\varphi_1, \ldots, \varphi_m$.
2. **New variables** z_i — as before.
3. **Tseitin equations** — as before. **Hard.**
4. Initialize **rooted linear tree** — as before.
5. **Boundary constraints** — as before. **Hard.**

Computing **MaxSAT-OPT**

Input: $\langle \varphi_1, \dots, \varphi_m \rangle$ in variables $x_1, \dots, x_n$.

1. List **L** of pairwise distinct subformulas in $\varphi_1, \dots, \varphi_m$.
2. **New variables** z_i — as before.
3. **Tseitin equations** — as before. **Hard.**
4. Initialize **rooted linear tree** — as before.
5. **Boundary constraints** — as before. **Hard.**
6. **Target constraints** $z_{j_i} = 1$ for each var. z_{j_i} introduced for φ_i, preserving the **multiplicity** of φ_i in the input. **Soft.**

Computing **MaxSAT-OPT**

Input: $\langle \varphi_1, \ldots, \varphi_m \rangle$ in variables $x_1, \ldots, x_n$.

1. List **L** of pairwise distinct subformulas in $\varphi_1, \ldots, \varphi_m$.
2. **New variables** z_i — as before.
3. **Tseitin equations** — as before. **Hard.**
4. Initialize **rooted linear tree** — as before.
5. **Boundary constraints** — as before. **Hard.**
6. **Target constraints** $z_{j_i} = 1$ for each var. z_{j_i} introduced for φ_i, preserving the **multiplicity** of φ_i in the input. **Soft.**
7. **Expand tree.** As before. **Hard.**
8. **Test systems.** For each branch, determine the maximum number of satisfied soft constraints in the system determined by the branch, in $\mathbb{R}$.
9. **Maximize.** Return the maximum number of satisfied soft constraints over all the branches.