Boosting MCSat Modulo Nonlinear Integer Arithmetic via Local Search

Enrico Lipparini[1], Thomas Hader[2], Ahmed Irfan[3], and Stéphane Graham-Lengrand[3]

1. University of Cagliari, Italy
2. TU Wien, Austria
3. SRI International, US

2025 SMT workshop (presentation only)

from the 2025 Conference on Automated Deduction (CADE-30)
Original slides prepared by Enrico Lipparini

Problem definition

The MCSat (Model Constructing Satisfiability) calculus for SMT [dMJ13]:

- Generalizes ideas of CDCL from SAT to the theory level

Problem definition

The MCSat (Model Constructing Satisfiability) calculus for SMT [dMJ13]:

- Generalizes ideas of CDCL from SAT to the theory level
- Constructs a model by assigning concrete values to theory variables

$$M = [(z^2 > 1) \mapsto \top, x \mapsto 1, (x + z \geqslant 3) \mapsto \top, z \mapsto 2]$$

Problem definition

The MCSat (Model Constructing Satisfiability) calculus for SMT [dMJ13]:

- Generalizes ideas of CDCL from SAT to the theory level
- Constructs a model by assigning concrete values to theory variables

$$M = [(z^2 > 1) \mapsto \top, x \mapsto 1, (x + z \geqslant 3) \mapsto \top, z \mapsto 2]$$

- Provides reasoning schema for theory reasoning, originally targeting non-linear real/int arith; variants used by several SMT solvers (e.g. YICES2, SMT-RAT, Z3)

Problem definition

The MCSat (Model Constructing Satisfiability) calculus for SMT [dMJ13]:

- Generalizes ideas of CDCL from SAT to the theory level
- Constructs a model by assigning concrete values to theory variables

$$M = [(z^2 > 1) \mapsto \top, x \mapsto 1, (x + z \geqslant 3) \mapsto \top, z \mapsto 2]$$

- Provides reasoning schema for theory reasoning, originally targeting non-linear real/int arith; variants used by several SMT solvers (e.g. YICES2, SMT-RAT, Z3)

... But how do solvers choose which values to assign?

Problem definition

The MCSat (Model Constructing Satisfiability) calculus for SMT [dMJ13]:

- Generalizes ideas of CDCL from SAT to the theory level
- Constructs a model by assigning concrete values to theory variables

$$M = [(z^2 > 1) \mapsto \top, x \mapsto 1, (x + z \geqslant 3) \mapsto \top, z \mapsto 2]$$

- Provides reasoning schema for theory reasoning, originally targeting non-linear real/int arith; variants used by several SMT solvers (e.g. YICES2, SMT-RAT, Z3)

... But how do solvers choose which values to assign?
Currently (at least in YICES2): $\sim$ randomly 😕

Problem definition

The MCSat (Model Constructing Satisfiability) calculus for SMT [dMJ13]:

- Generalizes ideas of CDCL from SAT to the theory level
- Constructs a model by assigning concrete values to theory variables

$$M = [(z^2 > 1) \mapsto \top, x \mapsto 1, (x + z \geqslant 3) \mapsto \top, z \mapsto 2]$$

- Provides reasoning schema for theory reasoning, originally targeting non-linear real/int arith; variants used by several SMT solvers (e.g. Yices2, SMT-RAT, Z3)

... But how do solvers choose which values to assign?
Currently (at least in Yices2): $\sim$ randomly 😕
Idea: use Local Search (LS) to guide decisions

Contributions

In this work, we:

- Propose a theory-agnostic framework that deeply integrates MCSat and Local Search:
  - Search state of MCSat used to instantiate LS
  - LS output used to guide future MCSat decisions

Contributions

In this work, we:

- Propose a theory-agnostic framework that deeply integrates MCSat and Local Search:
  - Search state of MCSat used to instantiate LS
  - LS output used to guide future MCSat decisions
- Define a LS alg. for (QF-)Non-linear Integer Arithmetic ($\mathcal{NIA}$) using
  - feasible-sets jumping to move between feasible intervals
  - hill-climbing to move inside feasible intervals

Contributions

In this work, we:

- Propose a theory-agnostic framework that deeply integrates MCSat and Local Search:
  - Search state of MCSat used to instantiate LS
  - LS output used to guide future MCSat decisions
- Define a LS alg. for (QF-)Non-linear Integer Arithmetic ($\mathcal{NIA}$) using
  - feasible-sets jumping to move between feasible intervals
  - hill-climbing to move inside feasible intervals
- Implement our method in YICES2 and evaluate over SMT-LIB($\mathcal{NIA}$)
  - Significant improvements for both sat/unsat benchmarks
  - 1st with 24s time limit, 3rd with 300s time limit
  - Great complementarity with state-of-the-art tools

MCSat

MCSat incrementally constructs a partial model consisting of Boolean and theory assignments, preserving the invariant that no constraint (initial or learnt) evaluates to false. The search state is given by a *trail* consisting of:

- decided literals (literals that we assume to be true)
- propagated literals (literals implied to be true by the current state)
- decided assignments of concrete values to theory variables
- propagated assignments of concrete values to theory variables (optional).

MCSat

MCSat incrementally constructs a partial model consisting of Boolean and theory assignments, preserving the invariant that no constraint (initial or learnt) evaluates to false. The search state is given by a *trail* consisting of:

- decided literals (literals that we assume to be true)
- propagated literals (literals implied to be true by the current state)
- decided assignments of concrete values to theory variables
- propagated assignments of concrete values to theory variables (optional).

Propagations, decisions, conflict analysis, etc. handled by theory plugins.

MCSat

MCSat incrementally constructs a partial model consisting of Boolean and theory assignments, preserving the invariant that no constraint (initial or learnt) evaluates to false. The search state is given by a *trail* consisting of:

- decided literals (literals that we assume to be true)
- propagated literals (literals implied to be true by the current state)
- decided assignments of concrete values to theory variables
- propagated assignments of concrete values to theory variables (optional).

Propagations, decisions, conflict analysis, etc. handled by theory plugins.

Important features:

- feasibility sets:
  for each variable, maintain the values "consistent with the trail"
  (unfeasible values make some constraint evaluate to false)
- value cache (optional; implemented in YICES2 with key role for LS):
  for each variable, keep track of last value assigned to variable

Local Search

A local search problem is a triple $(\mu_0, f_c, \mathit{moves})$, where:

- μ_0 is an initial assignment for a set of variables *Vars*,
- f_c is a cost function from the set of assignments to $\mathbb{R}_{\geqslant 0}$,
- *moves* is a neighbor relation between assignments.

Local Search

A local search problem is a triple $(\mu_0, f_c, \mathit{moves})$, where:

- μ_0 is an initial assignment for a set of variables *Vars*,
- f_c is a cost function from the set of assignments to $\mathbb{R}_{\geqslant 0}$,
- *moves* is a neighbor relation between assignments.

A local search algorithm works as follows:

- starts from an initial assignment μ_0
- iteratively explores neighboring assignments according to *moves*
- accepts a move from μ to μ' only if $f_c(\mu') < f_c(\mu)$.
- terminates when:
  - zero-cost assignment is found (*solution*)
  - no more possible moves (*local minimum*)
  - stopping criterion reached, e.g. max number of moves reached (*best-effort value*)

Local Search

A local search problem is a triple $(\mu_0, f_c, \mathit{moves})$, where:

- μ_0 is an initial assignment for a set of variables *Vars*,
- f_c is a cost function from the set of assignments to $\mathbb{R}_{\geqslant 0}$,
- *moves* is a neighbor relation between assignments.

A local search algorithm works as follows:

- starts from an initial assignment μ_0
- iteratively explores neighboring assignments according to *moves*
- accepts a move from μ to μ' only if $f_c(\mu') < f_c(\mu)$.
- terminates when:
  - zero-cost assignment is found (*solution*)
  - no more possible moves (*local minimum*)
  - stopping criterion reached, e.g. max number of moves reached (*best-effort value*)

ϕ is satisfiable $\leftrightarrow$ $f_c(\phi)$ has a zero

Local Search in SAT/SMT

- Widely used in SAT solving (*Bool flips*)
- Extension to bit-vectors (BV-SLS, 2015 [FBWH15])
- More recent adoption for arithmetic theories:
  - Linear Integer Arithmetic (LS-LIA, 2022 [CLZ22])
  - Multi-linear Real Arithmetic (LocalSMT(RA), 2023 [LC23])
  - Non-linear Integer Arithmetic (LocalSMT, 2023 [CLZ23], and LS, 2023 [LXZ23])
  - Non-linear Real Arithmetic (Ours, 2024 [WZLC24])
- Also used as sub-routine for numerical optimization global search:
  - Floating Points (XSat, 2016 [FS16])
  - Non-linear Transcendental Arithmetic (uGotNL, 2022 [LCGS22])

Local Search in SMT(arith): current limitations

Limitations of these methods for arithmetical theories:

- Only able to conclude satisfiability
- When not succeeding to conclude sat, all knowledge is lost

Local Search in SMT(arith): current limitations

Limitations of these methods for arithmetical theories:

- Only able to conclude satisfiability
- When not succeeding to conclude sat, all knowledge is lost

A step forward: HybridSMT (2024) [ZLC24]

- Combines LocalSMT within Z3's CDCL(T) for $\mathcal{NIA}$
- When LocalSMT cannot conclude sat, it returns
  best assignment found and *conflict frequency*
  which are used in the CDCL(T) algorithm for
  (Boolean) *phase selection* and (Boolean) *variable ordering*

Local Search in SMT(arith): current limitations

Limitations of these methods for arithmetical theories:

- Only able to conclude satisfiability
- When not succeeding to conclude sat, all knowledge is lost

A step forward: HybridSMT (2024) [ZLC24]

- Combines LocalSMT within Z3's CDCL(T) for $\mathcal{NIA}$
- When LocalSMT cannot conclude sat, it returns
  best assignment found and *conflict frequency*
  which are used in the CDCL(T) algorithm for
  (Boolean) *phase selection* and (Boolean) *variable ordering*

Limitations of HybridSMT:

- LS input limited to CDCL(T)'s Boolean skeleton solutions
- LS theory variable assignments are not used

(edge case: think of a formula consisting of exactly 1 literal)

Combining MCSat and Local Search

We propose a tight integration of LS within MCsat, such that:

- Current state of MCSat is used to instantiate LS $(\mu_0, f_c, \mathit{moves})$:
  - How to choose initial assignment μ_0?
  - How to choose cost function f_c?
  - How to choose neighbor relation *moves*?

Combining MCSat and Local Search

We propose a tight integration of LS within MCsat, such that:

- Current state of MCSat is used to instantiate LS $(\mu_0, f_c, \mathit{moves})$:
  - How to choose initial assignment μ_0?
  - How to choose cost function f_c?
  - How to choose neighbor relation *moves*?
- LS output is used to guide MCSat decisions
  - When to call LS?
  - How to use LS suggested values?
  - How to use LS variables activity?

Note: Pre-print by Ding et al [DLN+25] (July 1, 2025), integrating Local Search & MCSAT for $\mathcal{NRA}$, in solver called HELMS.

Instantiating LS: initial assignment

Given a MCSat trail M, the initial assignment μ_0 is determined as follows. For every variable x:

- If there is a model assignment $x \mapsto \alpha \in M$, then

  x is treated as the constant α

Instantiating LS: initial assignment

Given a MCSat trail M, the initial assignment μ_0 is determined as follows. For every variable x:

- If there is a model assignment $x \mapsto \alpha \in M$, then

$$x \text{ is treated as the constant } \alpha$$

- Else, if $\mathit{cache}(x) \neq \texttt{undef}$ and $\mathit{cache}(x) \in \mathit{feasible}_M(x)$, then

$$\mu_0(x) \leftarrow \mathit{cache}(x)$$

Instantiating LS: initial assignment

Given a MCSat trail M, the initial assignment μ_0 is determined as follows. For every variable x:

- If there is a model assignment $x \mapsto \alpha \in M$, then

  $$x \text{ is treated as the constant } \alpha$$

- Else, if $\mathit{cache}(x) \neq \texttt{undef}$ and $\mathit{cache}(x) \in \mathit{feasible}_M(x)$, then

  $$\mu_0(x) \leftarrow \mathit{cache}(x)$$

- Otherwise,

  $$\mu_0(x) \leftarrow \text{any } \beta \in \mathit{feasible}_M(x)$$

Instantiating LS: cost function

Given a MCSat trail M, the simplified formula ϕ_s used to construct the cost function f_c is determined as follows.
For any clause $C \in \phi$, s.t. $C = L_1 \vee L_2 \vee \ldots L_k$:

- If $(L_i \mapsto \top) \in M$ for any L_i, replace C with L_i in ϕ_s
- For all L_i s.t. $(L_i \mapsto \bot) \in M$, remove L_i from C in ϕ_s

Instantiating LS: cost function

Given a MCSat trail M, the simplified formula ϕ_s used to construct the cost function f_c is determined as follows.
For any clause $C \in \phi$, s.t. $C = L_1 \vee L_2 \vee \ldots L_k$:

- If $(L_i \mapsto \top) \in M$ for any L_i, replace C with L_i in ϕ_s
- For all L_i s.t. $(L_i \mapsto \bot) \in M$, remove L_i from C in ϕ_s

Then f_c can be constructed following the *Logic-to-Optimization* approach:

$$
\begin{array}{lcll}
\mathcal{L}2\mathcal{O}(t_1 = t_2) & \stackrel{\text{def}}{=} & dist(t_1, t_2) & (\text{e.g. } |t_1 - t_2|) \\
\mathcal{L}2\mathcal{O}(t_1 \leqslant t_2) & \stackrel{\text{def}}{=} & ITE(t_1 \leqslant t_2,\ 0,\ dist(t_1, t_2)) & \\
\mathcal{L}2\mathcal{O}(\phi_1 \wedge \phi_2) & \stackrel{\text{def}}{=} & \mathcal{L}2\mathcal{O}(\phi_1) + \mathcal{L}2\mathcal{O}(\phi_2) & \\
\mathcal{L}2\mathcal{O}(\phi_1 \vee \phi_2) & \stackrel{\text{def}}{=} & \mathcal{L}2\mathcal{O}(\phi_1) \cdot \mathcal{L}2\mathcal{O}(\phi_2) & \\
\ldots & & &
\end{array}
$$

setting $f_c = \mathcal{L}2\mathcal{O}(\phi_s)$

Instantiating LS: neighbor relation

The neighbor relation is, in general, theory-dependent.
...however, feasibility sets give precious information.

Instantiating LS: neighbor relation

The neighbor relation is, in general, theory-dependent.
...however, feasibility sets give precious information.

For (at least) arithmetical theories, we have that

$$feasible_M(x) = \bigcup_{i \in [0:m]} I_i$$

where I_i are intervals.

Instantiating LS: neighbor relation

The neighbor relation is, in general, theory-dependent.
...however, feasibility sets give precious information.

For (at least) arithmetical theories, we have that

$$\mathit{feasible}_M(x) = \bigcup_{i \in [0:m]} I_i$$

where I_i are intervals.

This gives rise to two different kinds of moves:

- Intra-feasible intervals moves: $\mu \sim \mu'$ with $\mu, \mu' \in I_i$
- Inter-feasible intervals moves: $\mu \sim \mu'$ with $\mu \in I_i$, $\mu' \in I_j$, $i \neq j$

Moreover, moves $\mu \sim \mu'$ with $\mu'(x) \notin \mathit{feasible}_M(x)$ *should be avoided*

Interaction between MCSat and LS

- When to call LS? After conflict resolution, according to a polynomially increasing conflict threshold:

$$50 \cdot \text{ls\_calls} \cdot \log_{10}(\text{ls\_calls} + 9)^3$$

where ls_calls represents the number of times LS has been called.

Interaction between MCSat and LS

- When to call LS? After conflict resolution, according to a polynomially increasing conflict threshold:

$$50 \cdot \text{ls\_calls} \cdot \log_{10}(\text{ls\_calls} + 9)^3$$

  where ls_calls represents the number of times LS has been called.

- LS output:
  - Best assignment μ: used to repopulate value cache (used to decide values for future decisions)

Interaction between MCSat and LS

- When to call LS? After conflict resolution, according to a polynomially increasing conflict threshold:

$$50 \cdot \text{ls\_calls} \cdot \log_{10}(\text{ls\_calls} + 9)^3$$

  where ls_calls represents the number of times LS has been called.

- LS output:
  - Best assignment μ: used to repopulate value cache (used to decide values for future decisions)
  - Variables activity: most active variables in LS get chosen first for future decisions

Local Search for Non-linear Integer Arithmetic

We define 3 types of moves (starting from assignment μ):

- Boolean flips: For $b : \texttt{bool}$, a *flip move* from μ is

$$\mu_{\neg b} \stackrel{\text{def}}{=} \mu[b \mapsto \neg \mu(b)]$$

Local Search for Non-linear Integer Arithmetic

We define 3 types of moves (starting from assignment μ):

- Boolean flips: For $b : \texttt{bool}$, a *flip move* from μ is

$$\mu_{\neg b} \stackrel{\text{def}}{=} \mu[b \mapsto \neg\mu(b)]$$

- Feasible-set-jumps: For $x : \texttt{int}$, with $\mathit{feasible}(x) = \bigcup_{i\in[0:m]} I_i$, and $\mu(x) \in I_j$ we consider the moves of the form

$$\mu_{j'} \stackrel{\text{def}}{=} \mu[x \mapsto \mathit{pick\_value}(I_{j'})]$$

for $j' \neq j$ (limited to 1 such move per variable per LS call)

Local Search for Non-linear Integer Arithmetic

We define 3 types of moves (starting from assignment μ):

- Boolean flips: For b : `bool`, a *flip move* from μ is

$$\mu_{\neg b} \stackrel{\text{def}}{=} \mu[b \mapsto \neg\mu(b)]$$

- Feasible-set-jumps: For x : `int`, with $\mathit{feasible}(x) = \bigcup_{i\in[0:m]} I_i$, and $\mu(x) \in I_j$ we consider the moves of the form

$$\mu_{j'} \stackrel{\text{def}}{=} \mu[x \mapsto \mathit{pick\_value}(I_{j'})]$$

  for $j' \neq j$ (limited to 1 such move per variable per LS call)
- Hill-climbing moves: For x : `int`, we consider the moves

$$\mu_{x\pm1} \stackrel{\text{def}}{=} \mu[x \mapsto \mu(x) \pm 1]$$

  In paper and implementation:
  dynamic step size (automatic acceleration/deceleration).

Experiments

- Implemented our method in YICES2
- Experimented on all SMT-LIB benchmarks in QF_NIA (14990 sat, 5183 unsat, 5270 unknown)
- Two time limits: 24s (SMT-COMP short track) and 300s (resource constraints)
- Compare new version YICES2_{LS}:
  - against baseline (YICES2_{base})
  - against other solvers (CVC5, HYBRIDSMT, MATHSAT5, Z3)
- Two portfolios: combining YICES2_{LS} with HYBRIDSMT and Z3

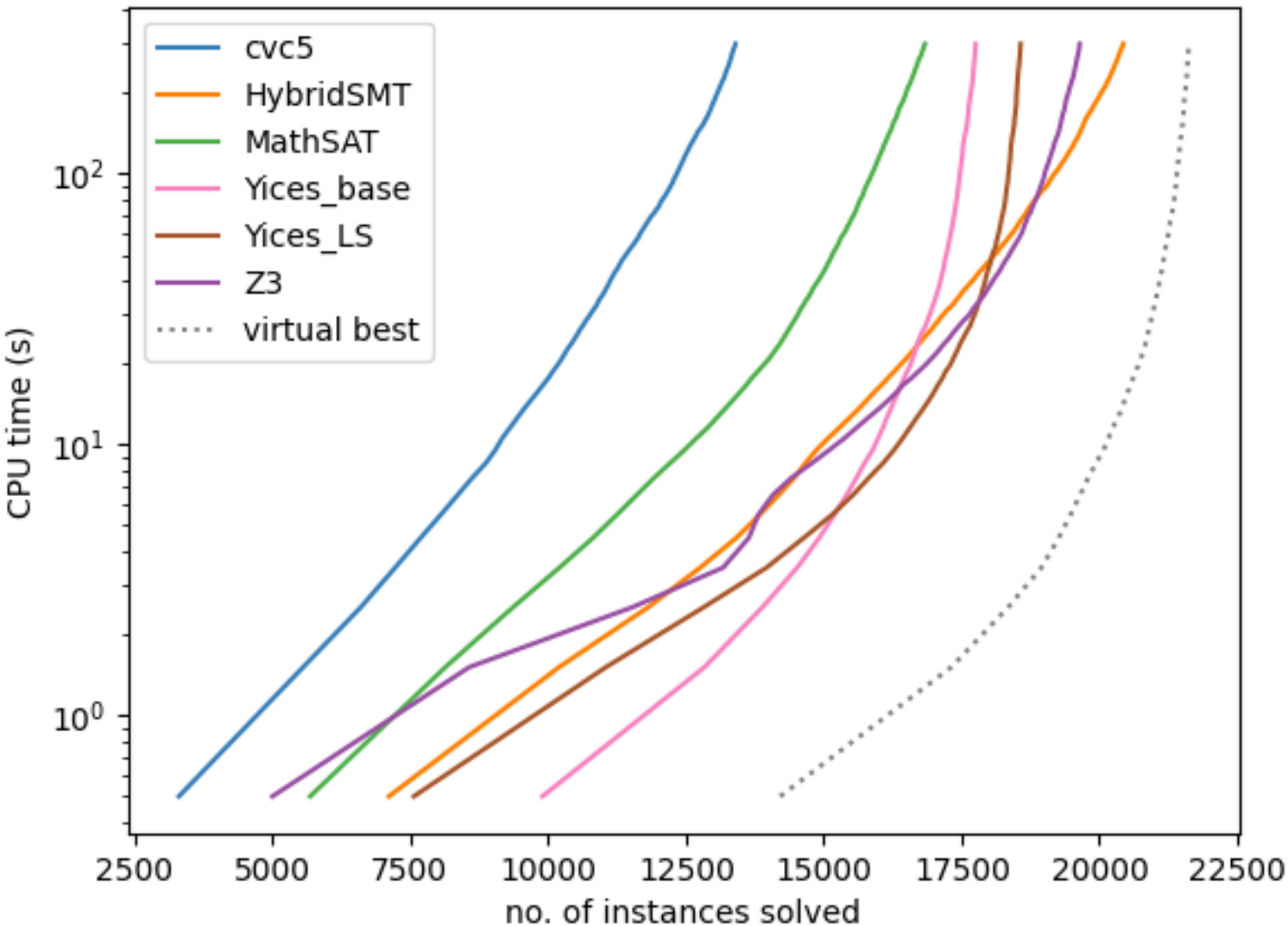
Experiments (time limit 24s)

	Total (sat) (unsat)	VeryMax	calypto	ezsmt	LassoRank	Dartagnan	LCTES	MathProbl	leipgiz	UltAut23	mcm	sqrtmodinv	UltAut	AProVE	UltLasso
CVC5	10457 (7011) (3446)	7799 (5465) (2334)	171 (79) (92)	**8** (8) (0)	97 (4) (93)	320 (11) (309)	**1** (0) (1)	107 (100) (7)	72 (70) (2)	16 (8) (8)	9 (9) (0)	2 (0) (2)	**7** (0) (7)	1816 (1251) (565)	**32** (6) (26)
HybridSMT	16764 (12404) (4360)	13555 (10427) (3128)	174 (78) (96)	**8** (8) (0)	**105** (4) (101)	**354** (10) (344)	**1** (0) (1)	93 (86) (7)	**151** (150) (1)	10 (7) (3)	**56** (56) (0)	7 (0) (7)	**7** (0) (7)	2212 (1572) (640)	31 (6) (25)
MathSAT5	14241 (9536) (4705)	11233 (7615) (3618)	168 (79) (89)	**8** (8) (0)	**105** (4) (101)	327 (12) (315)	**1** (0) (1)	141 (134) (7)	114 (112) (2)	17 (7) (10)	3 (3) (0)	0 (0) (0)	**7** (0) (7)	2085 (1556) (529)	**32** (6) (26)
Yices2$_{base}$	16714 (11403) (5311)	13695 (9461) (4234)	174 (79) (95)	**8** (8) (0)	91 (4) (87)	142 (1) (141)	0 (0) (0)	122 (115) (7)	102 (101) (1)	7 (7) (0)	6 (6) (0)	0 (0) (0)	**7** (0) (7)	2328 (1615) (713)	**32** (6) (26)
Yices2$_{LS}$	**17456** (12177) (5279)	**14269** (10019) (4250)	174 (79) (95)	**8** (8) (0)	96 (4) (92)	87 (0) (87)	0 (0) (0)	**303** (296) (7)	111 (110) (1)	7 (7) (0)	6 (6) (0)	0 (0) (0)	**7** (0) (7)	**2356** (1642) (714)	**32** (6) (26)
Z3	17225 (11561) (5664)	13975 (9599) (4376)	**176** (80) (96)	**8** (8) (0)	104 (4) (100)	352 (9) (343)	**1** (0) (1)	118 (111) (7)	120 (119) (1)	**21** (8) (13)	5 (5) (0)	**17** (0) (17)	**7** (0) (7)	2289 (1612) (677)	**32** (6) (26)
Portfolio (Z3 + Yices2$_{LS}$)	18359 (12459) (5900)	14848 (10271) (4577)	176 (80) (96)	8 (8) (0)	102 (4) (98)	344 (9) (335)	1 (0) (1)	300 (293) (7)	119 (118) (1)	20 (8) (12)	5 (5) (0)	17 (0) (17)	7 (0) (7)	2380 (1657) (723)	32 (6) (26)
Portfolio (HybridSMT +Yices2$_{LS}$)	18988 (13359) (5629)	15428 (11105) (4323)	176 (80) (96)	8 (8) (0)	104 (4) (100)	347 (9) (338)	1 (0) (1)	305 (298) (7)	148 (147) (1)	9 (7) (2)	46 (46) (0)	6 (0) (6)	7 (0) (7)	2371 (1649) (722)	32 (6) (26)

Experiments (time limit 300s)

	Total (sat) (unsat)	VeryMax	calypto	ezsmt	LassoRank	Dartagnan	LCTES	MathProbl	leipgiz	UltAut23	mcm	sqrtmodinv	UltAut	AProVE	UltLasso
CVC5	13398 (8848) (4550)	10489 (7122) (3367)	173 (79) (94)	**8** (8) (0)	98 (4) (94)	350 (17) (333)	1 (0) (1)	177 (170) (7)	89 (87) (2)	16 (8) (0)	17 (17) (0)	2 (0) (2)	**7** (0) (7)	1939 (1330) (609)	**32** (6) (26)
HybridSMT	**20435** (14174) (6261)	**17060** (12093) (4967)	175 (78) (97)	**8** (8) (0)	**106** (4) (102)	**369** (14) (355)	**2** (0) (2)	107 (100) (7)	**156** (155) (1)	10 (7) (3)	**69** (69) (0)	10 (0) (10)	**7** (0) (7)	2352 (1640) (685)	31 (6) (25)
MathSAT5	16836 (11563) (5273)	13651 (9509) (4142)	169 (79) (90)	**8** (8) (0)	105 (4) (101)	347 (18) (329)	1 (0) (1)	183 (176) (7)	126 (124) (2)	17 (7) (10)	10 (10) (0)	0 (0) (0)	**7** (0) (7)	2180 (1622) (558)	**32** (6) (26)
Yices2$_{base}$	17755 (12125) (5630)	14549 (10164) (4385)	174 (79) (95)	**8** (8) (0)	93 (4) (89)	311 (7) (304)	0 (0) (0)	124 (117) (7)	104 (103) (1)	7 (7) (0)	9 (9) (0)	0 (0) (0)	**7** (0) (7)	2337 (1621) (716)	**32** (6) (26)
Yices2$_{LS}$	18572 (12896) (5676)	15160 (10719) (4441)	175 (79) (96)	**8** (8) (0)	97 (4) (93)	288 (3) (285)	0 (0) (0)	**311** (304) (7)	111 (110) (1)	7 (7) (0)	9 (9) (0)	0 (0) (0)	**7** (0) (7)	**2367** (1647) (720)	**32** (6) (26)
Z3	19644 (13059) (6585)	16304 (11044) (5260)	**177** (80) (97)	**8** (8) (0)	**106** (4) (102)	368 (14) (354)	**2** (0) (2)	118 (111) (7)	135 (134) (1)	**21** (8) (13)	10 (10) (0)	**17** (0) (17)	**7** (0) (7)	2339 (1640) (699)	**32** (6) (26)
Portfolio (Z3 + Yices2$_{LS}$)	20281 (13597) (6684)	16698 (11366) (5332)	177 (80) (97)	8 (8) (0)	104 (4) (100)	363 (13) (350)	1 (0) (1)	314 (307) (7)	134 (133) (1)	21 (8) (13)	11 (11) (0)	17 (0) (17)	7 (0) (7)	2394 (1661) (733)	32 (6) (26)
Portfolio (HybridSMT +Yices2$_{LS}$)	20950 (14467) (6483)	17303 (12156) (5147)	177 (80) (97)	8 (8) (0)	105 (4) (101)	367 (13) (354)	1 (0) (1)	327 (320) (7)	154 (153) (1)	10 (7) (3)	64 (64) (0)	8 (0) (8)	7 (0) (7)	2387 (1656) (731)	32 (6) (26)

Experiments (cactus plots)



Observations

- YICES2$_{LS}$ best at 24s; behind HYBRIDSMT and Z3 at 300s
- YICES2$_{LS}$ improves on base YICES2 (+817 benchmarks at 300s)
  (pays off below 10s on average, slightly higher for unsat cases)
- HYBRIDSMT improves on Z3 (+791 benchmarks at 300s)
  (pays off around 100s on average, rarely pays off for unsat cases)
- Portfolios with YICES2$_{LS}$ and virtual best show complementarity with Z3 and HYBRIDSMT

Conclusions

Takeaways:

- MCSat and Local Search fit well together
- We proposed a very flexible framework
- Lots of room for future research:
  - other theories, starting with $\mathcal{NRA}$ (see [DLN+25]), bitvectors
  - other theory-specific LS procedures
  - other heuristics for MCSat-LS interaction

References I

[CLZ22] S. Cai, B. Li, and X. Zhang.
Local Search for SMT on Linear Integer Arithmetic.
In S. Shoham and Y. Vizel, editors, *Computer Aided Verification*, pages 227–248. Springer International Publishing, 2022.

[CLZ23] S. Cai, B. Li, and X. Zhang.
Local search for satisfiability modulo integer arithmetic theories.
ACM Trans. Comput. Logic, 24(4), 2023.

[DLN+25] T. Ding, H. Li, X. Ni, B. Xia, and T. Zhao.
A hybrid smt-nra solver: Integrating 2d cell-jump-based local search, mcsat and opencad, 2025.
https://arxiv.org/abs/2507.00557

[dMJ13] L. M. de Moura and D. Jovanovic.
A model-constructing satisfiability calculus.
In *VMCAI*, 2013.

References II

[FBWH15] A. Fröhlich, A. Biere, C. Wintersteiger, and Y. Hamadi.
Stochastic local search for Satisfiability Modulo Theories.
Proceedings of the AAAI Conference on Artificial Intelligence, 29(1), 2015.

[FS16] Z. Fu and Z. Su.
XSat: A fast floating-point satisfiability solver.
In *CAV*, volume 9780 of *Lecture Notes in Computer Science*, pages 187–209. Springer, 2016.

[LC23] B. Li and S. Cai.
Local search for SMT on linear and multi-linear real arithmetic.
In *2023 Formal Methods in Computer-Aided Design (FMCAD)*, pages 1–10, 2023.

References III

[LCGS22] E. Lipparini, A. Cimatti, A. Griggio, and R. Sebastiani.
Handling polynomial and transcendental functions in SMT via unconstrained optimisation and topological degree test.
In A. Bouajjani, L. Holík, and Z. Wu, editors, *Automated Technology for Verification and Analysis*, pages 137–153. Springer International Publishing, 2022.

[LXZ23] H. Li, B. Xia, and T. Zhao.
Local search for solving satisfiability of polynomial formulas.
In C. Enea and A. Lal, editors, *Computer Aided Verification*, pages 87–109. Springer Nature Switzerland, 2023.

[WZLC24] Z. Wang, B. Zhan, B. Li, and S. Cai.
Efficient local search for nonlinear real arithmetic.
In R. Dimitrova, O. Lahav, and S. Wolff, editors, *Verification, Model Checking, and Abstract Interpretation*, pages 326–349. Springer Nature Switzerland, 2024.

References IV

[ZLC24] X. Zhang, B. Li, and S. Cai.
Deep combination of CDCL(T) and local search for satisfiability modulo non-linear integer arithmetic theory.
In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ICSE '24. Association for Computing Machinery, 2024.