Improving Local Search for Bit-Vector Logics in SMT with Path Propagation

Aina Niemetz, Mathias Preiner, Andreas Fröhlich and Armin Biere

Institute for Formal Models and Verification (FMV)
Johannes Kepler University, Linz, Austria
http://fmv.jku.at/

DIFTS 2015
September 26 - 27, 2015
Austin, TX, USA

Bit-Vectors in Sat Modulo Theories (SMT)

- State-of-the-art: **Bit-Blasting**
  - eager reduction to propositional logic (SAT)
    $\longrightarrow$ offline SAT solver integration
  - efficient in practice
  - relies heavily on rewriting and other simplification techniques

  $\longrightarrow$ does not scale if input size can not be reduced sufficiently

- **DPLL(T)**-based layered approaches [CAV'07, CAV'14]
  - lazy approach with online SAT solver integration

- Recently: **Stochastic Local Search (SLS)** for SMT [AAAI'15]
  - implemented in the SMT solver Z3
  - lifts SLS to the theory level (word-level SLS)
  - without the use of a SAT solver
  - mostly simulates bit-level local search (focus on single bit flips)

  $\longrightarrow$ does not fully exploit the advantage of working on the theory level

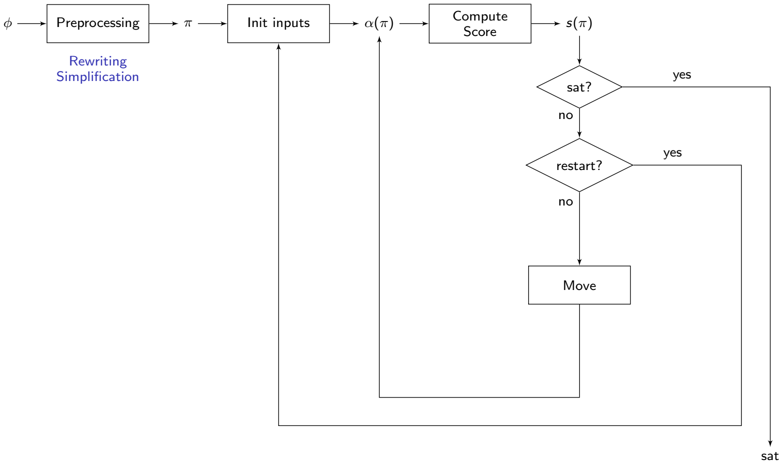
This work

- reimplementation of [AAAI'15] in our SMT solver Boolector
  $\longrightarrow$ confirms its effectiveness
- additional strategy: **propagation moves**
  - do not solely rely on bit flips
  - satisfy lines by propagating assignments from outputs to inputs

  $\longrightarrow$ implemented in Boolector
- extensive experimental evaluation
  $\longrightarrow$ suggests improved performance for **sequential portfolio** of
  - Boolector's bit-blasting engine with
  - propagation-based SLS techniques

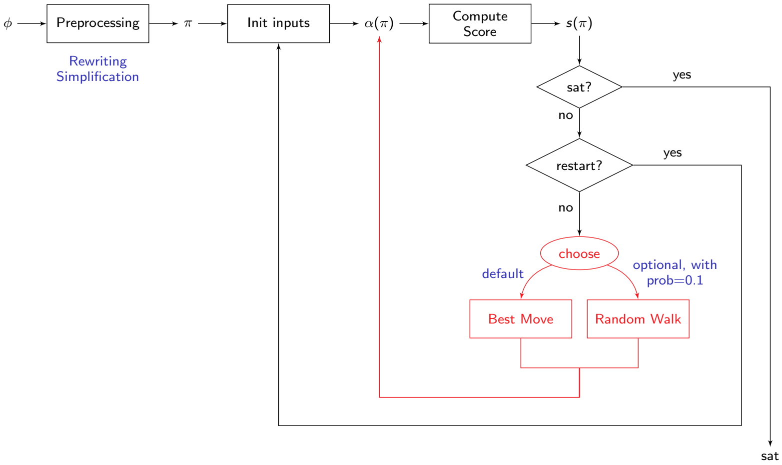
SLS for SMT

A Brief Overview



SLS for SMT

A Brief Overview



Example.

$$\phi \equiv 21_{[7]} * v_{[7]} = 93_{[7]}$$

$21_{[7]} \rightarrow 0010101$, $93_{[7]} \rightarrow 1011101$

Candidate: $v_{[7]} := 0000000$ (initial)

Neighborhood [AAAI'15]

- Single Bit Flips:
  - $v_{[7]} := 0000001$
  - $v_{[7]} := 0000010$
  - $v_{[7]} := 0000100$
  - $v_{[7]} := 0001000$
  - $v_{[7]} := 0010000$
  - $v_{[7]} := 0100000$
  - $v_{[7]} := 1000000$
- Increment
  - $v_{[7]} := 0000001$
- Decrement
  - $v_{[7]} := 1111111$
- Bit-Wise Negation
  - $v_{[7]} := 1111111$

Example.

$$\phi \equiv 21_{[7]} * v_{[7]} = 93_{[7]}$$

$21_{[7]} \rightarrow 0010101$, $93_{[7]} \rightarrow 1011101$

Candidate: $v_{[7]} := 0000000$ (initial)

Assignment $\alpha(\phi) = \{v_{[7]} := 0000000\}$ $\longrightarrow$ Score $s(\phi, \alpha(\phi)) = 0.142857$

Find Best Move [AAAI'15]

- Single Bit Flips:
  - $v_{[7]} := 0000001$ (0.357143)
  - $v_{[7]} := 0000010$ (0.071429)
  - $v_{[7]} := 0000100$ (0.357143)
  - $v_{[7]} := 0001000$ (0.142857)
  - $v_{[7]} := 0010000$ (0.285714)
  - $v_{[7]} := 0100000$ (0.071429)
  - $v_{[7]} := 1000000$ (0.214286)
- Increment
  - $v_{[7]} := 0000001$ (0.357143)
- Decrement
  - $v_{[7]} := 1111111$ (0.214286)
- Bit-Wise Negation
  - $v_{[7]} := 1111111$ (0.214286)

Example. $\phi \equiv 21_{[7]} * v_{[7]} = 93_{[7]}$

$21_{[7]} \rightarrow 0010101$, $93_{[7]} \rightarrow 1011101$

Candidate: $v_{[7]} := 0000000$ (initial)

Assignment $\alpha(\phi) = \{v_{[7]} := 0000000\} \longrightarrow$ Score $s(\phi, \alpha(\phi)) = 0.142857$

Find Best Move [AAAI'15]

- Single Bit Flips:
  - $v_{[7]} := 0000001$ (0.357143)

$\longrightarrow$ (First) Most improvement of score $s(\phi, \alpha(\phi))$

$\longrightarrow$ If no score improving neighbor exists: randomization (e.g. $v_{[7]} := 0110011$)

Example.

$$\phi \equiv \underset{0010101}{\underbrace{21_{[7]}}} * v_{[7]} = \underset{1011101}{\underbrace{93_{[7]}}}$$

Candidate: $v_{[7]} := 0000000$ (initial)

Random Walk [AAAI'15]

- Single Bit Flips:
  - $v_{[7]} := 0100000$

$\longrightarrow$ Picked randomly and immediately

$\longrightarrow$ without trying all other neighbors

Example.

$$\phi \equiv \underset{\downarrow\ 0010101}{\boxed{21_{[7]}}} * v_{[7]} = \underset{\downarrow\ 1011101}{\boxed{93_{[7]}}}$$

Candidate: $v_{[7]} := 0000000$ (initial)

$\longrightarrow$ solution $v_{[7]} = 0101001 \; (41_{[7]})$

$\longrightarrow$ 4 moves, 0 restarts

Example.

$$\phi \equiv 274177_{[65]} * v_{[65]} = 18446744073709551617_{[65]}$$

Candidate: $v_{[65]} :=$ 000000...000000 (initial)

Neighborhood [AAAI'15]

- Single Bit Flips:
  - $v_{[65]} :=$ 000000...000001
  - $v_{[65]} :=$ 000000...000010
  - $v_{[65]} :=$ 000000...000100
  - $v_{[65]} :=$ 000000...001000
  - ...
  - ...
  - ...
  - ...
  - $v_{[65]} :=$ 001000...000000
  - $v_{[65]} :=$ 010000...000000
  - $v_{[65]} :=$ 100000...000000
- Increment
  - $v_{[65]} :=$ 000000...000001
- Decrement
  - $v_{[65]} :=$ 111111...111111
- Bit-Wise Negation
  - $v_{[65]} :=$ 111111...111111

Example.

$$\phi \equiv 274177_{[65]} * v_{[65]} = 18446744073709551617_{[65]}$$

Candidate: $v_{[65]} := 000000...000000$ (initial)

Assume: no preprocessing (rewriting, simplification)

$\longrightarrow$ 355837 moves, 21 restarts

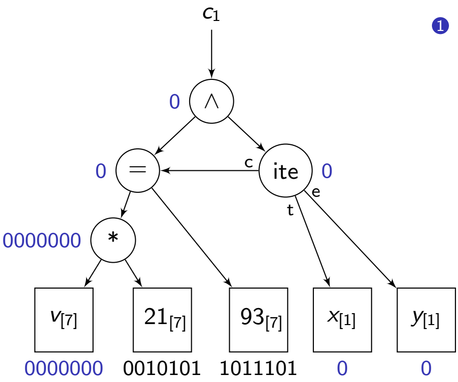
$\longrightarrow$ unable to determine (single) solution $v_{[65]} = 67280421310721_{[65]}$

- within a time limit of 1200 seconds
- on a 3.4GHz Intel Core i7-2600 machine

$\longrightarrow$ solved within **one** single propagation move

Example.

$$\phi \equiv c_1 \wedge c_2 \wedge c_3$$

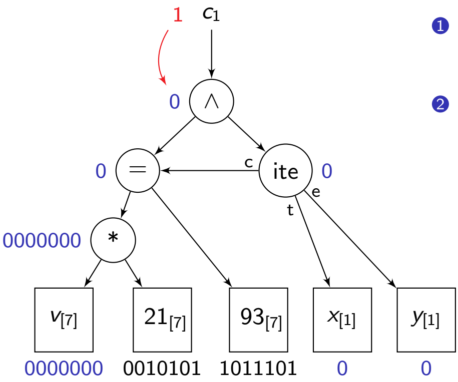


1. initial assignment
   $\{v_{[7]} := 0000000,\ x_{[1]} := 0,\ y_{[1]} := 0\}$

Propagation-Based Move Selection

Example.

$$\phi \equiv c_1 \wedge c_2 \wedge c_3$$

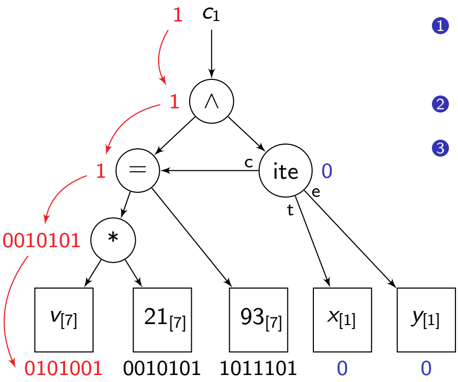


1. initial assignment
   $\{v_{[7]} := 0000000,\ x_{[1]} := 0,\ y_{[1]} := 0\}$
2. force $c_1 = 1$

Propagation-Based Move Selection

Example.

$$\phi \equiv c_1 \wedge c_2 \wedge c_3$$



$\longrightarrow$ Move: $v_{[7]} := 0101001$

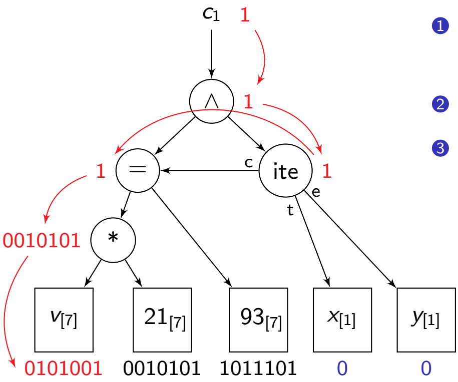
1. initial assignment
   $\{v_{[7]} := 0000000,\ x_{[1]} := 0,\ y_{[1]} := 0\}$
2. force $c_1 = 1$
3. choose path and propagate down
   1. Boolean $\wedge$

      $\longrightarrow$ justification-based selection
      - $0 \rightarrow 1$: choose single controlling input
      - else choose randomly

Propagation-Based Move Selection

Example.

$$\phi \equiv c_1 \wedge c_2 \wedge c_3$$



$\longrightarrow$ Move: $v_{[7]} := 0101001$

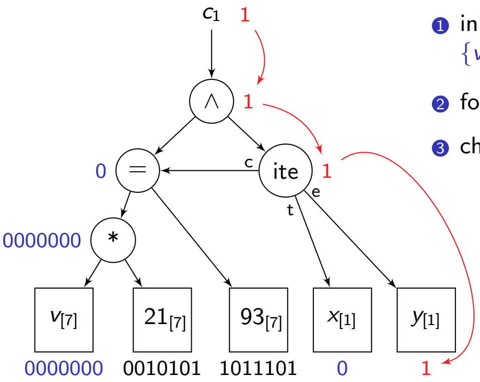
1. initial assignment $\{v_{[7]} := 0000000,\ x_{[1]} := 0,\ y_{[1]} := 0\}$
2. force $c_1 = 1$
3. choose path and propagate down
   1. Boolean $\wedge$

      $\longrightarrow$ justification-based selection
      - $0 \rightarrow 1$: choose single controlling input
      - else choose randomly
   2. if-then-else (ite)
      - a. flip condition

Propagation-Based Move Selection

Example.

$$\phi \equiv c_1 \wedge c_2 \wedge c_3$$



$\longrightarrow$ Move: $y_{[7]} := 1$

1. initial assignment
   $\{v_{[7]} := 0000000,\ x_{[1]} := 0,\ y_{[1]} := 0\}$
2. force $c_1 = 1$
3. choose path and propagate down
   1. Boolean $\wedge$

      $\longrightarrow$ justification-based selection
      - $0 \rightarrow 1$: choose single controlling input
      - else choose randomly
   2. if-then-else (ite)
      - a. flip condition
      - b. follow enabled branch

Down Propagation of Assignments

- via inverse computation
- Restricted set of bit-vector operations
  - **Unary** operations `bvnot` `extract`
  - **Binary** operations `=` `bvult` `bvshl` `bvshr` `bvadd` `bvand` `bvmul` `bvudiv` `bvurem` `concat`
- for some operations no well-defined inverse operation exists

$\longrightarrow$ produce non-unique values

$\longrightarrow$ via randomization of bits or bit-vectors

e.g. $c_{[4]} := a_{[4]} \wedge b_{[4]}$

assume as fixed: $b := 1011$

down propagated: $c := 0001$

selected path, inverse $a := 0X01$

don't care

Propagation-Based Move Selection

Down Propagation of Assignments (cntd.)

- if no inverse found
  - $c_{[n]} := a_{[n]} \text{ op } b_{[n]}$

    $\longrightarrow$ disregard b

    $\longrightarrow$ choose inverse value for a that matches assignment of c

    e.g. $c_{[4]} := a_{[4]} \wedge b_{[4]}$

    disregard $b := 1110$

    down propagated: $c := 0001$

    selected path, choose $a := 0001$

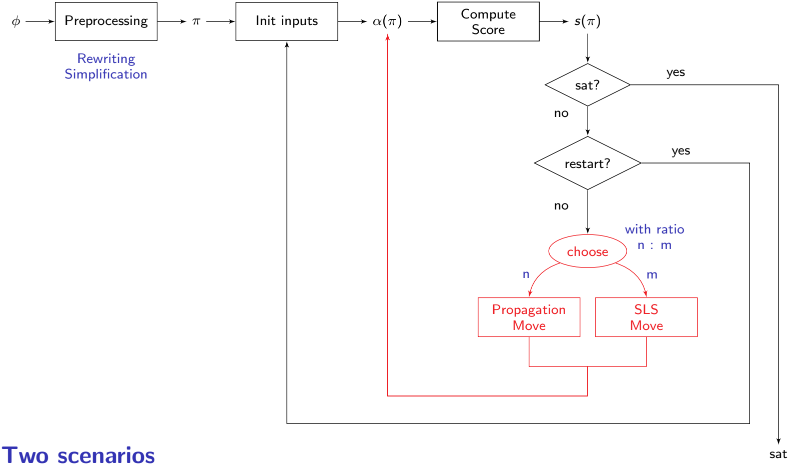
  - $c_{[n]} := a_{[n]} \text{ op } bvconst_{[n]}$

    $\longrightarrow$ assignments of b and c are conflicting

    $\longrightarrow$ no value for a found

    $\longrightarrow$ recover with **regular SLS** move

Propagation-Based Move Selection



Two scenarios

1. Propagation vs. SLS moves with ratio n : m
2. Propagation moves only (ratio ∞ : 0) $\longrightarrow$ default

Boolector Configurations

- **Bb** Core engine (bit-blasting approach)
  - ⟶ winner of the QF_BV main track of SMTCOMP'15
- **Bsls** SLS core engine
  - ⟶ optionally with random walks enabled **(+rw)**
- **Bprop** SLS engine with propagation-based strategy enabled
  - ⟶ **default**: propagation moves only
  - ⟶ **optional**:
    - ratio $n : m$ of propagation to regular SLS moves **(+n:m)**
    - conflict recovery via random walks **(+frw)**

Z3 Configuration

- **Z3sls** SLS engine of Z3 version 4.4.0
  - ⟶ random walks enabled by default

Benchmark Set:

⟶ all QF_BV benchmarks with status sat and unknown in SMT-LIB (18354 benchmarks) **except**

- 449 non-SMT-LIB v2 compliant *Sage2* benchmarks and
- 1469 benchmarks Bb proved to be unsatisfiable within 1200 seconds

⟶ **total**: **16436** benchmarks

All experiments on a cluster with 30 nodes of 2.83 GHz Intel Core 2 Quad machines with 8GB RAM running Ubuntu 14.04.2 LTS.

Experimental Evaluation

	Solved [#]	Time [s]
Bb	**9673**	**80197**
Bsls	7665	91491
Bsls+rw	7630	91635
Z3sls	8791	83262

Time limit: 10 seconds, **Memory limit**: 7GB

- confirms effectiveness and overall gap in performance
- enabling random walks does not improve the performance of Bsls
- performance of Bsls+rw and Z3sls differ on some families

	Bsls+rw		Z3sls	
Family	**Solved [#]**	**Time [s]**	**Solved [#]**	**Time [s]**
sage	5275	10626	**5969**	**5261**
Sage2	620	64289	**1597**	**56890**
spear	**1187**	**6336**	456	13289
uclid	23	2551	**259**	**297**

⟶ random seed almost no influence on Bsls (+rw)

⟶ rewriting and simplification huge impact on performance

Experimental Evaluation

	Solved [#]	Time [s]
Bsls	7665	91491
Bprop+1:100	7651	91505
Bprop+1:10	7622	91759
Bprop+1:1	7623	91690
Bprop+10:1	7711	90015
Bprop+100:1	7706	89983
Bprop	**7755**	**89808**

Time limit: 10 seconds, **Memory limit**: 7GB

- better performance for higher ratio of propagation moves
- configuration Bprop with most improvement in comparison to Bsls

$\longrightarrow$ $>$ 56% solved with propagation moves only

$\longrightarrow$ for $\sim$ 85% instances less than 8 recovery SLS moves required

Experimental Evaluation

	Solved [#]	**Time [s]**
Bsls	6827	10361
Bprop	7185	9807
Bprop+frw	**7270**	**9544**

Time limit: 1 second, **Memory limit**: 7GB

- suggests combination of Bb and Bprop+frw in sequential portfolio

  $\longrightarrow$ run Bprop+frw prior to bit-blasting for 1 second

  $\longrightarrow$ virtual configuration Bb+Bprop+frw

Experimental Evaluation

	Solved [#]	Time [s]
Bsls	6827	10361
Bprop	7185	9807
Bprop+frw	**7270**	**9544**

Time limit: 1 second, **Memory limit**: 7GB

- suggests combination of Bb and Bprop+frw in sequential portfolio

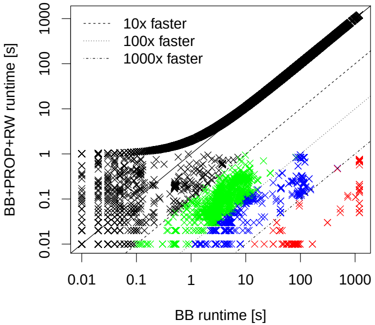
⟶ run Bprop+frw prior to bit-blasting for 1 second

⟶ virtual configuration Bb+Bprop+frw

	Solved [#]	Time [s]
Bb	14806	2623801
Bb+Bprop+frw	**14844**	**2538620**

Time limit: 1200 seconds, **Memory limit**: 7GB

⟶ +38 instances
⟶ in 97% of the total run time



Time limit: 1200 seconds, **Memory limit**: 7GB

$\longrightarrow$ Bb+Bprop+frw orders of magnitude faster than Bb

- x $\geq 1000 \times$ faster (117 instances)
- x $100 - 999 \times$ faster (582 instances)
- x $10 - 99 \times$ faster (1320 instances)

Experimental Evaluation

	Solved [#]	Time [s]		
Bb	14806	2623801		
Bb+Bprop+frw (1s)	14844	2538616	+38	97.1%
Bb+Bprop+frw (2s)	14852	2535600	+46	96.9%
Bb+Bprop+frw (3s)	14858	2534900	+52	96.9%
Bb+Bprop+frw (4s)	14861	2538266	+55	97.1%
Bb+Bprop+frw (5s)	14862	2544488	+56	97.3%
Bb+Bprop+frw (6s)	14862	2551784	+56	97.6%
Bb+Bprop+frw (7s)	14862	2558002	+56	97.9%
Bb+Bprop+frw (8s)	14862	2565357	+56	98.1%
Bb+Bprop+frw (9s)	14862	2572600	+56	98.0%

Time limit: 1200 seconds, **Memory limit**: 7GB

$\longrightarrow$ increasing the runtime for Bprop+frw up to 5s increases performance

Conclusion

- reimplementation of [AAAI'15] in our SMT solver Boolector
  - ⟶ confirms its effectiveness
- propagation-based extension of [AAAI'15] implemented in Boolector
  - ⟶ exploit the advantage of working on the theory level
  - ⟶ SLS recovery tactics not strictly necessary, but current inverse computation introduces short cuts for some conflict cases
  - ⟶ elimination of these short cuts left to **future work**
- **Promising:** combination of Boolector's bit-blasting engine with propagation-based approach in sequential portfolio manner
  - ⟶ considerably improves performance on satisfiable instances
  - ⟶ implementation in Boolector left to **future work**

- R. Bruttomesso, A. Cimatti, A. Franzén, A. Griggio, Z. Hanna, A. Nadel, A. Palti and R. Sebastiani. A Lazy and Layered SMT(BV) Solver for Hard Industrial Verification Problems. In CAV'07, volume 4590 of LNCS, Springer, 2007.
- L. Hadarean, K. Bansal. D. Jovanovic, C. Barrett and C. Tinelli. A Tale of Two Solvers: Eager and Lazy Approaches to Bit-Vectors. In CAV'14, volume 8559 of LNCS, Springer, 2007.
- A. Fröhlich, A. Biere, C. M. Wintersteiger and Y. Hamadi. Stochastic Local Search for Satisfiability Modulo Theories. In AAAI'15, AAAI Press, 2015.
- A. Griggio, Q. Phan, R. Sebastiani and S. Tomasi. Stochastic Local Search for SMT: Combining Theory Solvers with WalkSAT. In FROCOS'11, LNCS, Springer, 2011.