Stochastic Local Search for Satisfiability Modulo Theories

Andreas Fröhlich and Armin Biere

JOHANNES KEPLER UNIVERSITY LINZ

Christoph M. Wintersteiger and Youssef Hamadi

Microsoft Research

Introduction

Satisfiability Modulo Theories (SMT) is essential for many practical applications, e.g., in hard- and software verification, and increasingly also in other scientific areas like computational biology. We present a novel **stochastic local search (SLS)** algorithm to solve SMT problems, especially those in the theory of **bit-vectors**, directly **on the theory level**.

SLS for SMT

SLS for SAT

+

Theory Information

↓

Industrial Applications

SLS solvers for SAT are known to work very well on random benchmarks, but usually perform bad on industrial problems, because they are not aware of structural information.

A user defined neighbourhood relation and a score function to guide the search are sufficient.

We **extend the classical SLS loop** in order to **exploit the structure** that is contained in bit-vector formulas. **Many techniques** used in modern SLS solvers for SAT **can be lifted** to the SMT level. Experimental results show that our approach **outperforms current SLS solvers for SAT** and can **compete with state-of-the-art bit-vector solvers** on many **industrial instances**.

Architecture

- Input formula F as a conjunction of *assertions* in ***Negation Normal Form*** *(NNF)*.
- Given an *assignment* α to all variables and a constant $c \in [0,1]$, we define a **score function for bit-vector expressions** (an extension to bit-vector formulas in NNF is natural):

$$s(t_1^{[n]} = t_2^{[n]}, \alpha) = \begin{cases} 1 & \text{if } t_1|_\alpha = t_2|_\alpha \\ c \cdot \left(1 - \frac{h(t_1|_\alpha, t_2|_\alpha)}{n}\right) & \text{otherwise} \end{cases}$$

$$s(t_1^{[n]} \leq t_2^{[n]}, \alpha) = \begin{cases} 1 & \text{if } t_1|_\alpha \leq t_2|_\alpha \\ c \cdot \left(1 - \frac{t_2|_\alpha - t_1|_\alpha}{2^n}\right) & \text{otherwise} \end{cases}$$

- **Possible moves**: Bit-flips, increment, decrement, negation.
- **Techniques lifted from SAT**: Neighbourhood restriction to pre-selected assertions (similar to *WalkSAT*), additive weighting scheme for assertions (similar to *PAWS*), random walks, restarts (similar to *Luby*).
- **Additional techniques**: *Upper Confidence Bounds (UCB)* selection scheme (as used for bandits), *Variable Neighbourhood Search (VNS)*.

Example

Consider the assertion a: $x + 3 = \neg x$, where x is a bit-vector of size $n = 6$ (in practice, n is often much larger), $\neg$ represents bitwise negation, and the $+$ operation is as usual, i.e., with overflow semantics. The equation has two solutions: $x = [0,1,1,1,1,0]$ and $x = [1,1,1,1,1,0]$. If we initialize the search at $x = [0,\dots,0]$ and use $c = 1$ for computing the score s, the trace of visited states could look as follows:

x: 000000 ✗
a: 000011 = 111111
s: 2/6

$\xrightarrow{\textit{negation}}$

x: 111111 ✗
a: 000010 = 000000
s: 5/6

$\xrightarrow{\textit{bit-flip [0]}}$

x: 111110 ✓
a: 000001 = 000001
s: 6/6

✓

↓ *bit-flip [0]*

x: 000001 ✗
a: 000110 = 111110
s: 2/6

[1] … [4] …

↓ … *[5]*

x: 100000 ✗
a: 100011 = 011111
s: 2/6

All states reachable by a single bit-flip from the initial one, have a score $s < 5/6$.

Experiments

We ran experiments on two different sets of benchmarks. The first benchmark family is the **QF_BV** benchmark set, which consists of **7498 instances** that can be found in the SMT-LIB and are also part of the SMT Competition. The second benchmark family is the **SAGE2** benchmark set, consisting of **8017 instances**. Those problems were generated as part of the SAGE project at Microsoft, describing some testcases for automated whitebox fuzz testing and are known to be hard for state-of-the-art SMT solvers.

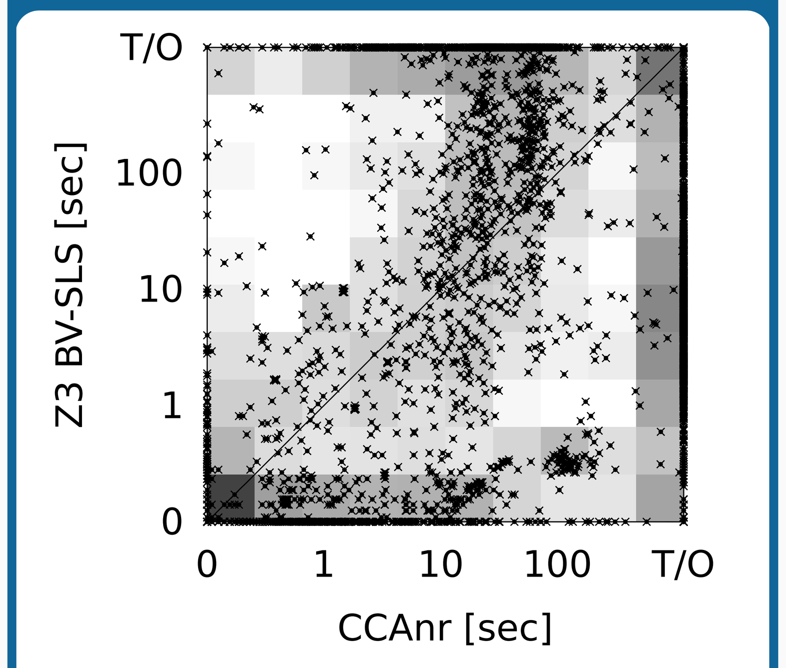
Number of solved instances

	QF_BV	SAGE2
CCAnr	**5409**	64
CCASat	4461	8
probSAT	3816	10
Sparrow	3806	12
VW2	2954	4
PAWS	3331	**143**
YalSAT	3756	142
Z3 (Default)	7173	5821
BV-SLS	**6172**	**3719**

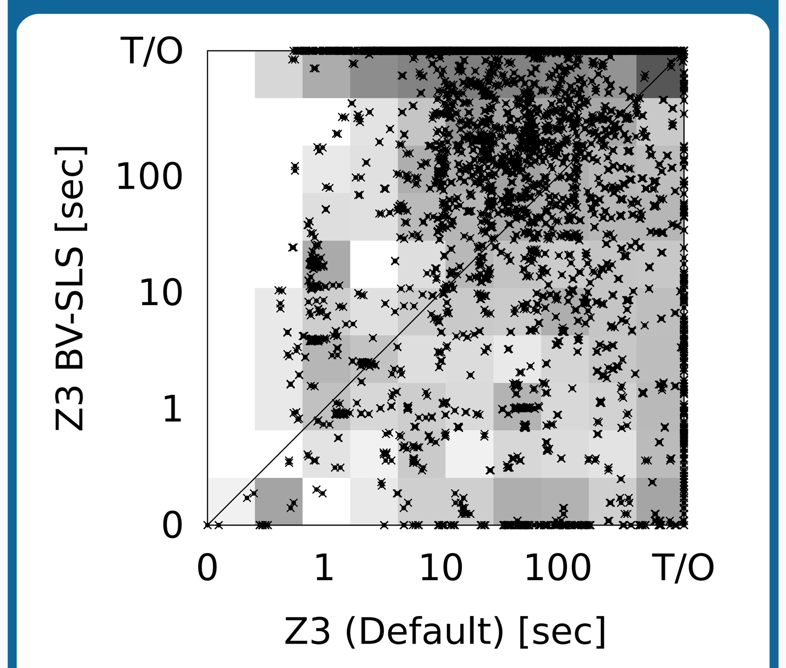
All experiments were run on a Windows HPC cluster of dual Quad-Xeon (E54xx) machines, 16 GB RAM, and used a time limit of 1200 seconds.

We compared our new solver BV-SLS to the most recent version of Microsoft's state-of-the-art SMT solver Z3 (which is based on *bit-blasting*) and also evaluated several SLS solvers for SAT on the propositional encodings of our benchmarks in *Conjunctive Normal Form (CNF)*.

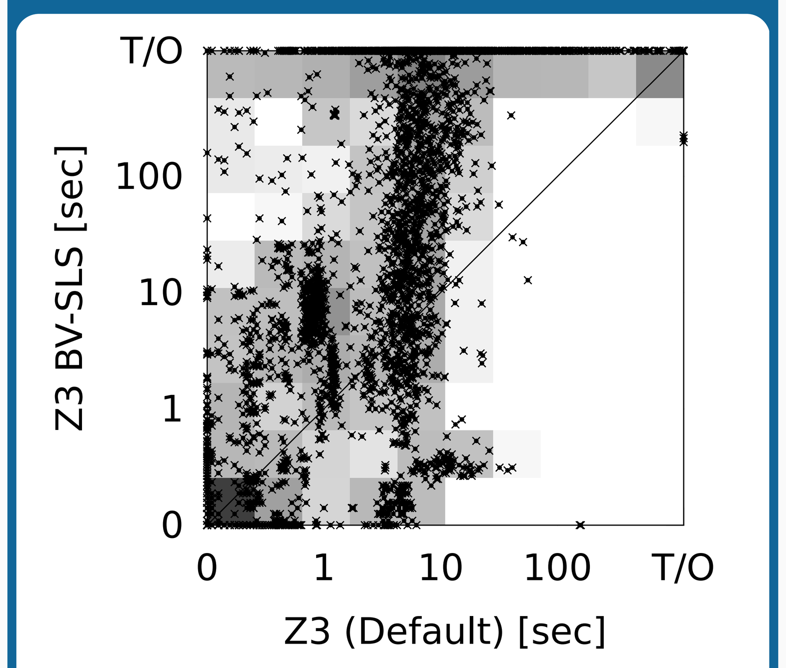
QF_BV, BV-SLS vs CCAnr



SAGE2, BV-SLS vs Z3 (Default)



QF_BV, BV-SLS vs Z3 (Default)



Conclusion

- Novel **SLS algorithm** directly **on the theory level**.
  → Bridging the gap between SMT and SLS.
- **Techniques** used for SAT **can be successfully lifted** to the SMT level.
- Solver BV-SLS **outperforms SLS for SAT** on the propositional encoding.
  → Benefit of using word-level information.
- Insights into the **importance of exploiting problem structure** also in SAT SLS solvers.
- Still a gap in performance compared to state-of-the-art bit-vector solvers in general, but outperforming Z3 on many **industrial instances of practical relevance**.
  → Interesting possibilities in combining our approach with existing techniques.
- Natural extension to **additional theories**.

Additional Information

All source code is available at http://z3.codeplex.com as part of the Z3 project. Contact:

andreas.froehlich@jku.at, biere@jku.at, cwinter@microsoft.com, youssefh@microsoft.com