Evaluating the Hardness of SAT Instances Using Evolutionary Optimization Algorithms

Alexander Semenov[1], Daniil Chivilikhin[1], Artem Pavlenko[1], Ilya Otpushennikov[2], Vladimir Ulyantsev[1], Alexey Ignatiev[3]

[1]ITMO University, Saint Petersburg, Russia
[2]ISDCT SB RAS, Irkutsk, Russia
[3]Monash University, Melbourne, Australia

CP'2021

SAT: Boolean formulas

- Boolean variable $x \in \{0, 1\}$
- Literals: x, $\neg x$
- Boolean formula: literals, braces, and logical connectives, e.g. $\wedge$ (AND), $\vee$ (OR), NOT ($\neg$), $\rightarrow$ (IMPLIES)

Conjunctive normal form

- Clause – disjunction of literals

$$(x \lor \neg y \lor z)$$

- Conjunctive normal form – conjunction of clauses

$$(x \lor \neg y \lor z) \land (\neg x \lor y) \land (y \lor \neg z)$$

- For an arbitrary Boolean formula can construct an equisatisfiable CNF [Tseitin, 1968].

The Boolean satisfiability problem (SAT)

Is a given Boolean formula (in CNF) satisfiable?

SAT is NP-complete [Cook-Levin theorem, 1971, 1973].

Some SAT instances are (very) hard. How to estimate hardness?

Hardness measures for SAT

- Analytical measures for specific classes of CNF formulas
  - Polynomial: 2-CNF, Horn-formulas, Schaefer's class

- Exponential lower bounds for CDCL, Resolution, etc.
  - Pigeon Hole Principle formulas, Parity Principle formulas, Mutilated Chessboard Principle formulas, etc.

- No efficient way to assess hardness of arbitrary CNF formula C w.r.t. arbitrary SAT solver A

Hardness of hard SAT instances

A – arbitrary complete CDCL SAT solver

C – arbitrary CNF formula

- Launch a SAT solver and wait, wait, wait…
  - Heavy-tailed behavior [Gomes C., Sabharwal A. Exploiting runtime variation in complete solvers, 2021]

- Use Strong Backdoor Sets
  - [Williams et al. Backdoors to typical case complexity // IJCAI, 2003]
  - [Ansotegui et al. Measuring the hardness of SAT instances // AAAI, 2008]

Strong Backdoor Set

- X – set of Boolean variables
- C – unsatisfiable CNF formula over X
- $B \subseteq X$ – arbitrary subset of X
- $\{0,1\}^{|B|}$ – set of all assignments to variables from B
- Assignment $\beta \in \{0,1\}^{|B|}$ (e.g. $B = \{\mathrm{x}_1, \mathrm{x}_6, \mathrm{x}_{10}\}, \beta = \{x_1 = 0, x_6 = 1, x_{10} = 0\}$)
- $C[\beta/B]$ derived from C by substitution of β

Definition 1 (Williams et al. // IJCAI 2003). Set $B \subseteq X$ is called a **Strong Backdoor Set** (SBS) for C w.r.t. poly-time sub-solver P, if for $\forall \beta \in \{0,1\}^{|B|}$ the formula $C[\beta/B]$ is reported by P to be unsatisfiable.

We can use this to assess formula hardness

Backdoor-hardness

Definition 2 (b-hardness).

- ❑ C – unsatisfiable CNF formula over the set of variables X
- ❑ P – poly-time sub-solver
- ❑ $B \subseteq X$ – arbitrary SBS for C w.r.t. poly-time sub-solver P
- ❑ $\mu_{B,P}(C)$ – total runtime of P on formulas $C[\beta/B]$, $\forall\beta \in \{0,1\}^{|B|}$

Then, backdoor-hardness of C w.r.t. P:

$$\mu_P(C) = \min_{B \in 2^X} \mu_{B,P}(\mathrm{C})$$

b-hardness in practice

Algorithm for minimum SBS (Williams et al.)

Enumerate all sets B of size 1, 2, 3, …, check if each is an SBS:

- check if P solves all $C[\beta/B]$ for all $\beta \in \{0,1\}^{|B|}$

- If $\exists$ SBS B: $|B| < |X|/2$, runtime complexity is $O\left(p(|C|) \cdot \left(\frac{2|X|}{\sqrt{|B|}}\right)^{|B|}\right)$
- If there is no SBS with $|B| \leq 20$, the algorithm is infeasible
- Moreover, P is polynomial.

What can we do for arbitrary complete SAT solver?

Proposal : decomposition-hardness

Definition (d-hardness).
- C – arbitrary CNF formula over X
- $B \subseteq X$ – arbitrary subset of X
- A – arbitrary deterministic complete SAT solver
- $t_A(C[\beta/B])$ – running time of A on C
- **Decomposition hardness** of C w.r.t. to A:

$$\mu_A(C) = \min_{B \in 2^X} \mu_{B,A}(\mathrm{C}),$$

where

$$\mu_{B,A}(C) = \sum_{\beta \in \{0,1\}^{|B|}} t_A(C[\beta/B])$$

Our contribution

1. Propose new hardness measure for arbitrary CNF formula C w.r.t arbitrary complete deterministic SAT solver A.

2. Algorithm to seek set B with minimal d-hardness.
- ❑ (ε, δ)-approximation of $\mu_{B,A}(C)$
- ❑ Evolutionary optimization on Boolean hypercube

3. Demonstrate practical applicability.

ITsMOre than a UNIVERSITY

Probabilistic expression of $\mu_{B,A}(C)$

1. Connect with $B \in 2^X$ a random variable ξ_B:

- for $\beta \in \{0,1\}^{|B|}$, $\xi_B(\beta)$ = (running time of A on $C[\beta/B]$)

2. ξ_B has finite spectrum $S(\xi_1, \dots, \xi_M)$ and distribution: $P(\xi_B) = \left\{\frac{s_1}{2^{|B|}}, \dots, \frac{s_M}{2^{|B|}}\right\}$, where $s_i, i \in \{1, \dots, M\}$ is the number of $\beta \in \{0,1\}^{|B|}$: $\xi_B(\beta)$ has value ξ_i.

$$\mu_{B,A}(C) = \sum_{\beta \in \{0,1\}^{|B|}} t_A(C[\beta/B]) = \sum_{i=1}^{M} \xi_i \cdot s_i = 2^{|B|} \sum_{i=1}^{M} \xi_i \cdot \frac{s_i}{2^{|B|}} = 2^{|B|} \cdot E[\xi_B].$$

$$0 < E[\xi_B] < \infty$$

Monte Carlo estimation of $\mu_{B,A}(C)$

$\mu_{B,A}(C) = 2^{|B|} \cdot E[\xi_B] < \infty$, so we can estimate it with Monte Carlo[1] method.

Theorem 1 ((ε, δ)-approximation of $\mu_{B,A}(C)$)

- C – arbitrary unsatisfiable CNF formula over X
- A – deterministic complete SAT solver
- $B \subseteq X$ – arbitrary subset of X

For any $\varepsilon, \delta > 0$, for $N > \frac{Var(\xi_B)}{\varepsilon^2 \cdot \delta \cdot E[\xi_B]}$ and $\boxed{\tilde{\mu}_{B,A}(C) = \frac{2^{|B|}}{N} \cdot \sum_{j=1}^{N} \xi^j}$ we have:

$$Pr\left[(1-\varepsilon) \cdot \mu_{B,A}(C) \leq \tilde{\mu}_{B,A}(C) \leq (1+\varepsilon) \cdot \mu_{B,A}(C)\right] \geq 1 - \delta$$

[1] N. Metropolis and S. Ulam. The Monte Carlo Method // J. Amer. Statistical Assoc., 1949.

Estimating d-hardness in practice

Replace $E[\xi_B]$ and $Var(\xi_B)$ with statistical counterparts

1. $E[\xi_B] \rightarrow$ sample mean $\overline{\xi_B} = \frac{1}{N} \cdot \sum_{j=1}^{N} \xi^j$ for sample $\xi^1, \ldots, \xi^N$

2. Var$(\xi_B) \rightarrow$ unbiased sample variance $s^2(\xi_B) = \frac{1}{N-1} \cdot \sum_{j=1}^{N} \left(\xi^j - \overline{\xi_B}\right)^2$

3. Select $N > \frac{s^2(\xi_B)}{\varepsilon^2 \cdot \delta \cdot \overline{\xi_B}^2}$

Estimation of d-Hardness via Evolutionary Optimization Algorithms

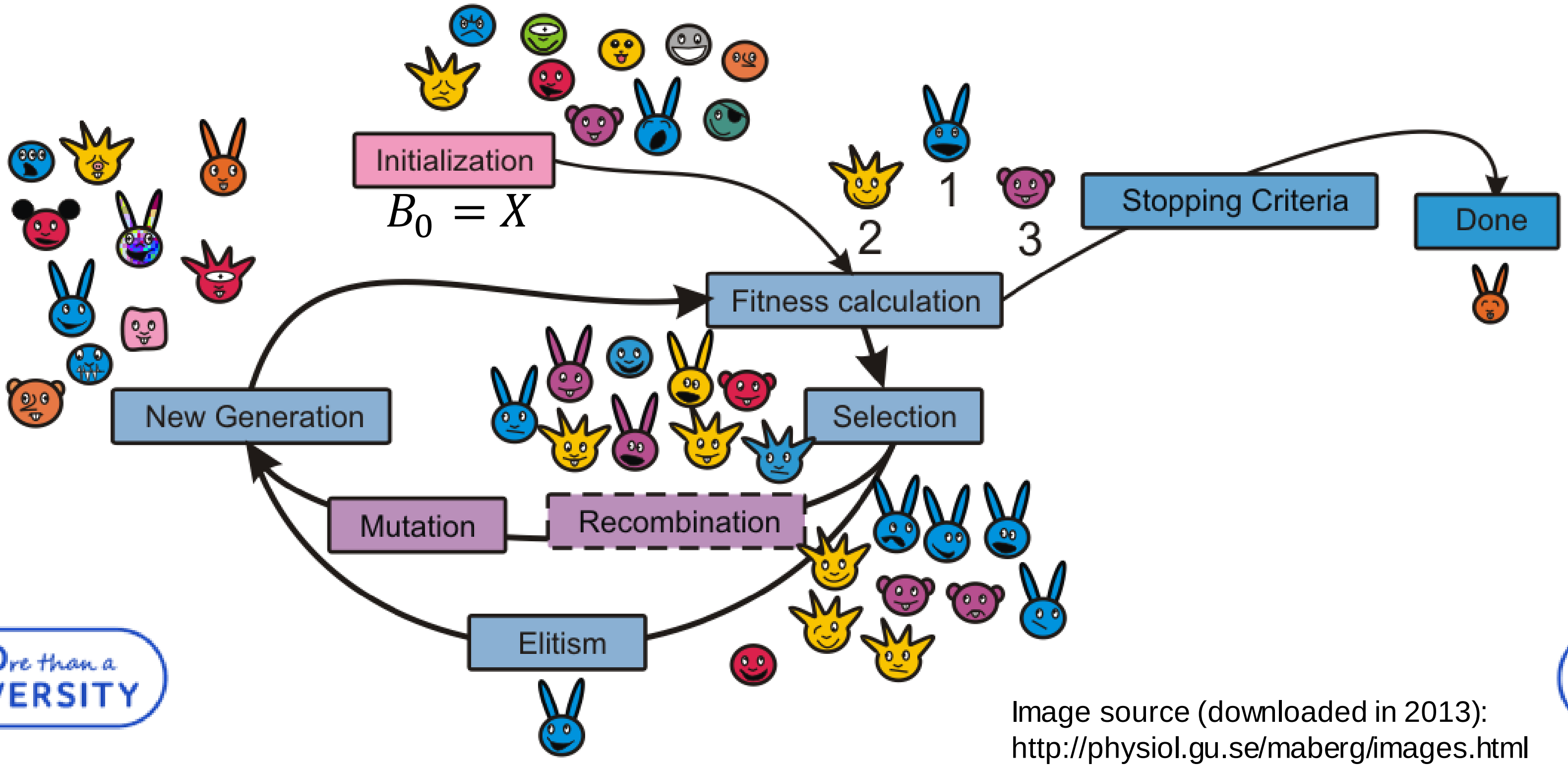
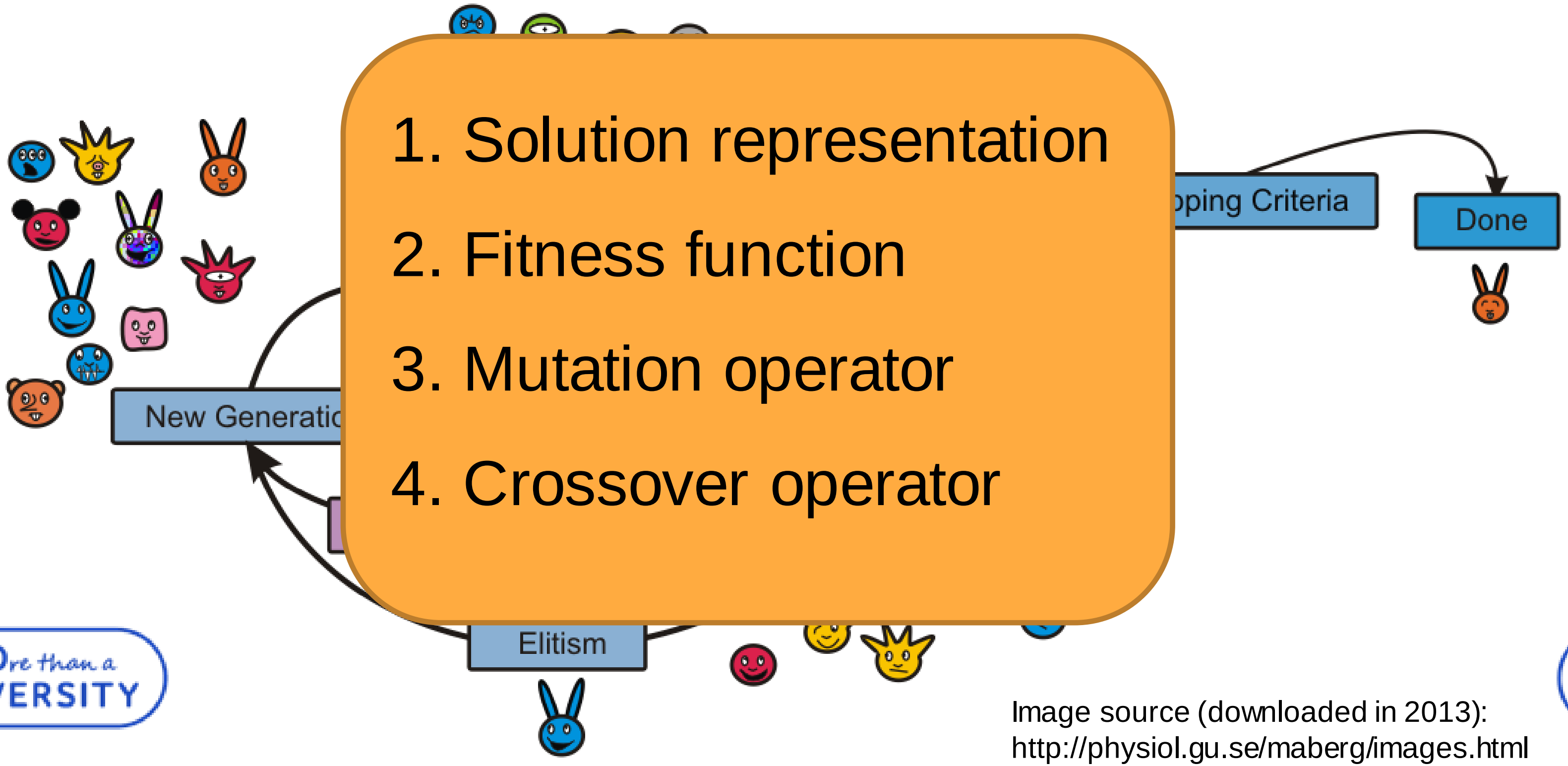


Image source (downloaded in 2013): http://physiol.gu.se/maberg/images.html

Estimation of d-Hardness via Evolutionary Optimization Algorithms



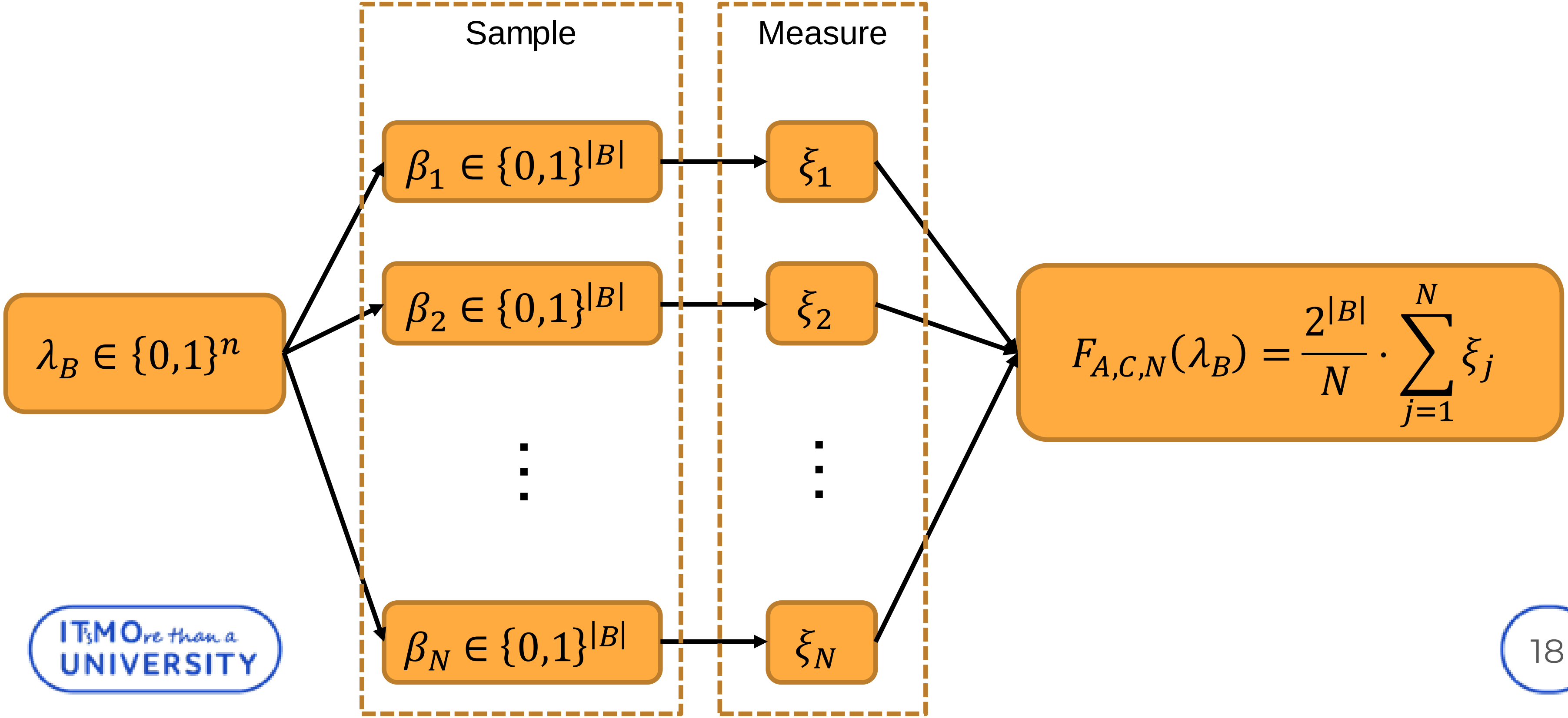
1. Solution representation
2. Fitness function
3. Mutation operator
4. Crossover operator

Image source (downloaded in 2013):
http://physiol.gu.se/maberg/images.html

Solution representation

Candidate solution	Representation
$B \subseteq X$	$\lambda_B \in \{0,1\}^n \quad \lambda_B = (\lambda_1, \dots, \lambda_N) \begin{cases} \lambda_i = 1 \ if \ x_i \in B \\ \lambda_i = 0 \ if \ x_i \notin B \end{cases}$
$X = \{x_1, x_2, x_3, x_4, x_5\}$ $B = \{x_1, x_3, x_5\}$	$\lambda_B = (1, 0, 1, 0, 1)$

Fitness function



Mutation operator

- Flip each bit with probability $1/n$, $n = |X|$

	1	2	3	4	5	6	7
λ_B	1	0	1	1	1	0	1

$$B = \{x_1, x_3, x_4, x_5, x_7\}$$

Heavy-tailed mutation operator[1]

- Flip each bit with probability Λ/n, $n = |X|$
- Λ – value of random variable with Power-Law distribution $D_{n/2}^{\beta}$

[1] B. Doerr et al. Fast genetic algorithms // GECCO 2017

Mutation operator

- Flip each bit with probability $1/n$, $n = |X|$

	1	2	3	4	5	6	7
λ_B	1	0	1	1	1	0	1

$$B = \{x_1, x_3, x_4, x_5, x_7\}$$

Mutation operator

- Flip each bit with probability $1/n$, $n = |X|$

	1	2	3	4	5	6	7
λ_B	1	0	1	1	1	0	1

$$B = \{x_1, x_3, x_4, x_5, x_7\}$$

	1	2	3	4	5	6	7
$\lambda_{B'}$	1	1	1	1	0	0	1

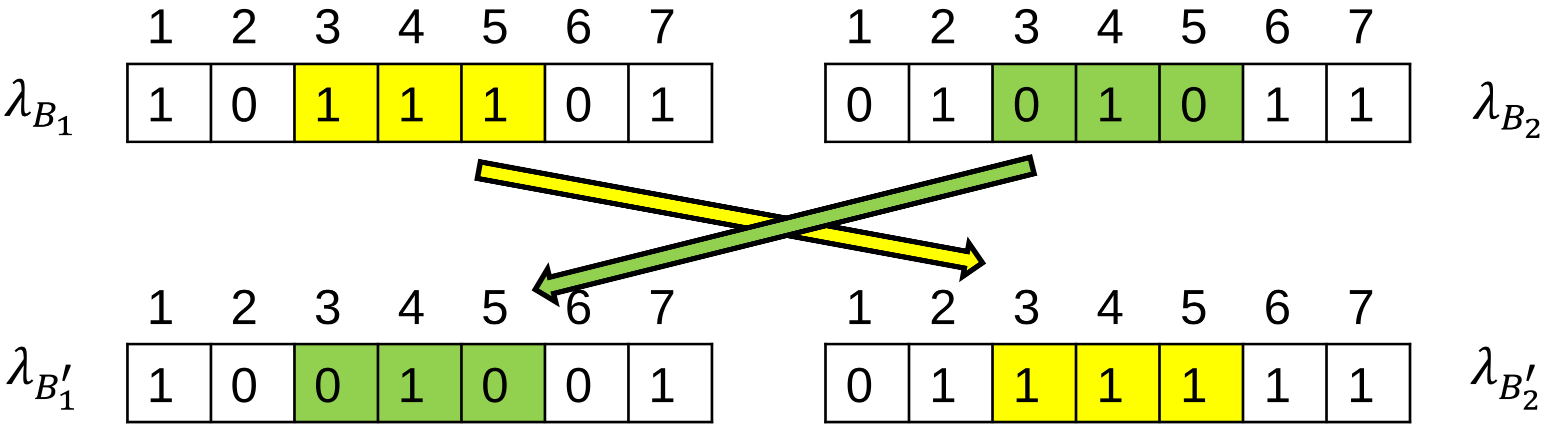
$$B' = \{x_1, x_2, x_3, x_4, x_7\}$$

Two-point crossover operator

	1	2	3	4	5	6	7
λ_{B_1}	1	0	1	1	1	0	1

1	2	3	4	5	6	7	
0	1	0	1	0	1	1	λ_{B_2}

Two-point crossover operator



(1 + 1)-Evolutionary Algorithm

$\lambda_B \leftarrow (1,2,\dots,|X|)$
$f^{\text{best}} \leftarrow F(\lambda_B)$
while $(\neg\text{terminate}())$ **do**
  $\lambda_{B'} \leftarrow \text{mutate}(\lambda_B)$
  $f' \leftarrow F(\lambda_{B'})$
  if $f' < f^{best}$ **then**
    $\lambda_B \leftarrow \lambda_{B'}$
    $f^{best} \leftarrow f'$

#	$\|B\|$	λ_B	$F(\lambda_B)$
1	7	$(1,1,1,1,1,1,1)$	10^{10}
2	6	$(0,1,1,1,1,1,1)$	10^{9}
3	5	$(0,1,1,1,1,1,0)$	10^{8}
4	4	$(0,1,1,0,1,1,0)$	10^{7}
5	3	$(0,0,1,0,1,1,0)$	10^{6}
6	4	$(1,0,0,0,1,1,1)$	10^{3}
		…	
50	4	$(1,0,0,0,1,1,1)$	10^{3}

1. GA: Standard Genetic Algorithm
2. FEA: Fast Genetic Algorithm [Doerr et al.]

EvoGuess: framework for d-hardness estimation and decomposition set search

- https://github.com/ctlab/evoguess
- Python
- PySAT[1,2] for SAT solving

[1] https://pysathq.github.io/

[2] Ignatiev et al. PySAT: A Python toolkit for prototyping with SAT oracles // SAT 2018

Benchmarks

1. sgen [Ivor Spence. Weakening cardinality constraints creates harder satisfiability benchmarks // ACM J. Exp. Algorithmics, 2015].

❑ $|X| \in \{150, 200\}$

2. Equivalence checking tests

Algorithm 1 → Circuit S_1 → $f_1 : \{0,1\}^n \to \{0,1\}^m$

Algorithm 2 → Circuit S_2 → $f_2 : \{0,1\}^n \to \{0,1\}^m$

$$f_1 \cong f_2 \leftrightarrow S_1 \cong S_2$$

Equivalence checking tests

- CNF formula C is constructed[1] from circuits S_1 and S_2 using Tseitin transformations
- $S_1 \cong S_2 \leftrightarrow C$ is unsatisfiable
- Inputs of S_1 and S_2 form the set $X^{\text{in}} \subset X, X^{\text{in}} = \{x_1, \dots, x_n\}$
- X^{in} – SBS w.r.t. Unit Propagation, $|X^{\text{in}}| \leq 200$
- S_1 and S_2: sorting algorithms[2,3] of any d l-bit numbers

S_1	S_2	S_1 **vs** S_2
Bubble sorting	Selection sorting	$\mathrm{BvS}_{l,d}$
Bubble sorting	Pancake sorting	$\mathrm{BvP}_{l,d}$
Pancake sorting	Selection sorting	$\mathrm{PvS}_{l,d}$

ITMOre than a UNIVERSITY

[1]Otpuschennikov et al. Encoding cryptographic functions to SAT using TRANSALG system // ECAI, 2016.
[2]T. Cormen, C. Leiserson, and R. Rivest. Introduction to Algorithms. MIT Press, 1990.
[3]W.H. Gates and C.H. Papadimitriou. Bounds for sorting by prefix reversal. Discret. Math.,1979.

Experimental setup

- Measure $t_A(C)$ in number of Unit Propagations
  - Reproducibility
  - Cross-platform

- SAT solvers
  - Glucose 3 (g3)
  - CaDiCaL (cd)

Experimental setup

1. Search for set B with minimal $\mu_{B,A}(C)$.
   - Up to 12 hours wall-clock time
   - Up to 5 nodes × 36 threads (Intel Xeon E5-2695 2.1 GHz)
2. Solve all $2^{|B|}$ formulas $C[\beta/B]$ defined by found set B.
   - Single-thread
   - Multi-thread

x1
x2
x3
x4
x5
x6
x7
x8

x3
x4
x5

Set B

CNF C

$2^{|B|}$ formulas $C[\beta/B]$

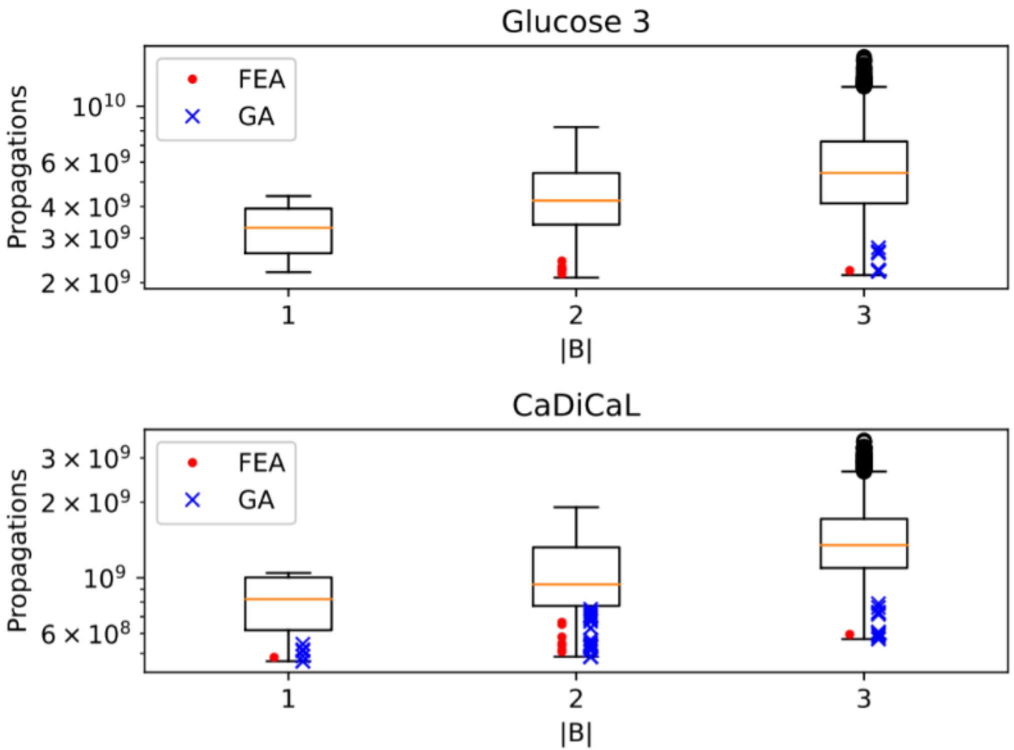
x3	x4	x5	C'
0	0	0	C'_1
0	0	1	C'_2
0	1	0	C'_3
0	1	1	C'_4
1	0	0	C'_5
1	0	1	C'_6
1	1	0	C'_7
1	1	1	C'_8

$2^{|B|}$ runs

Comparison with Williams algorithm with non-polynomial sub-solver

- C: $\mathrm{PvS}_{4,7}$
- $|B| \in \{1, 2, 3\}$

Use Williams algorithm to search for sets B with minimal d-hardness



d-hardness estimations CNF formulas

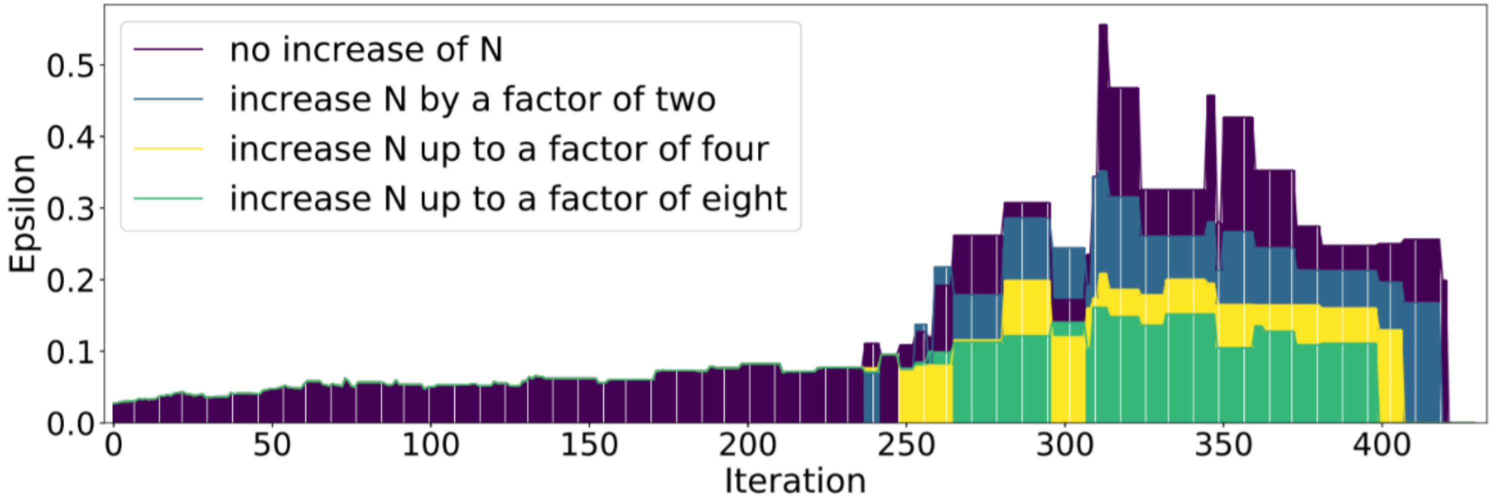
Instance	$\lvert X\rvert$	Solver A	Algorithm	$\lvert B\rvert$	$\mu_{B,A}/10^3$	$r_{B,A}$
$\mathrm{PvS}_{4,7}$	3244	g3	FEA	3	2,190,213	0.792
		g3	FEA	4	2,250,504	0.814
		g3	GA	5	3,319,314	1.201
		g3	GA	6	3,333,915	1.206
		cd	FEA	3	595,695	1.043
$\mathtt{sgen}_{150}^{1001}$	150	g3	FEA	5	101,371	0.424
		cd	FEA	6	244,191	0.763
		g3	GA	6	114,821	0.480
		cd	GA	7	247,947	0.775
$\mathtt{sgen}_{150}^{101}$	150	g3	FEA	8	122,796	0.438
		cd	GA	7	131,557	0.470
$\mathtt{sgen}_{150}^{200}$	150	g3	GA	7	151,275	0.569
		cd	GA	6	229,705	0.541
$\mathrm{BvS}_{4,7}$	2134	g3	GA	3	460,944	1.140
		g3	FEA	3	449,325	1.112
$\mathrm{BvP}_{4,7}$	2060	g3	FEA	3	726,080	1.049
		g3	GA	3	771,521	1.115

Decomposition rate

$$r_{B,A}(C) = \frac{\mu_{B,A}(C)}{t_A(C)}$$

often, $r_{B,A}(C) < 1$

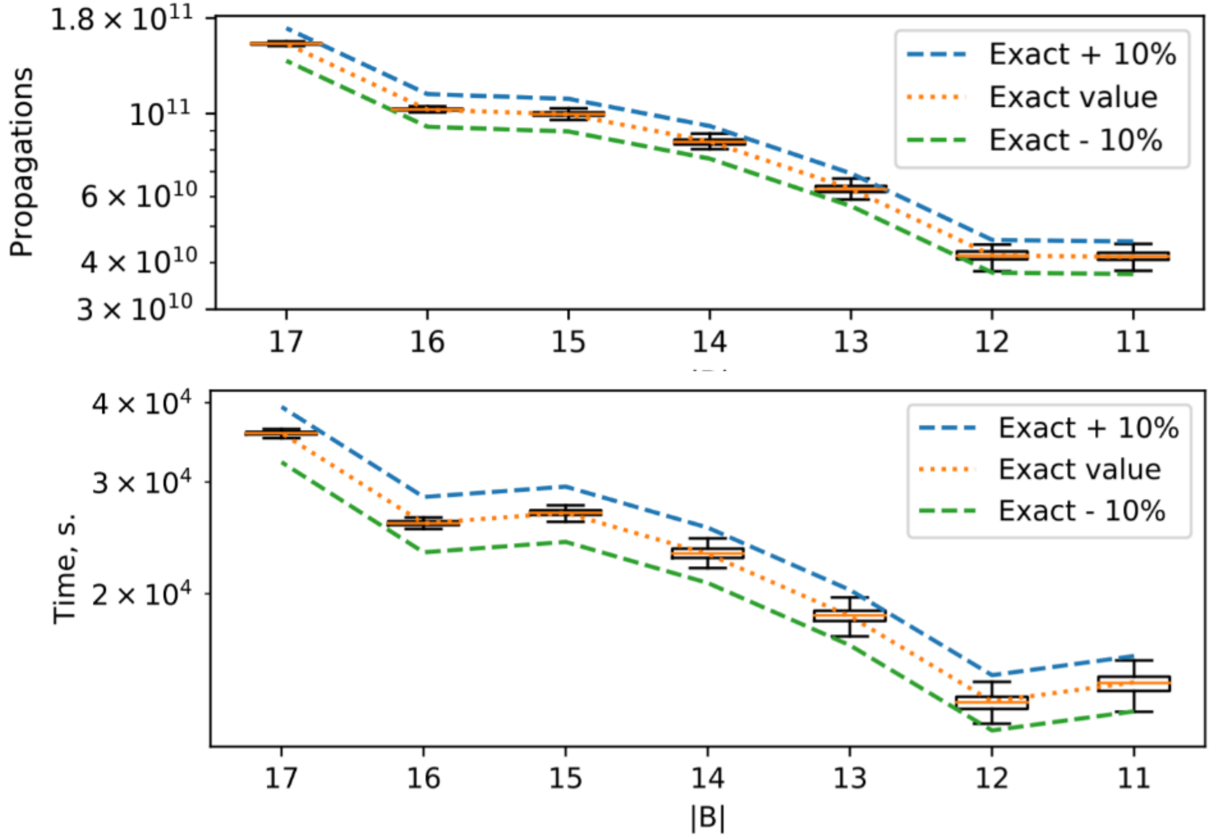
Dependence of ε from iteration (sgen, g3)



- ε < 0.1 most of the time
- When |B| is small, $\mu_{B,A}(C)$ is computed exactly (ε = 0).

d-hardness estimation accuracy ($PvS_{4,7,}$ g3)

$$N = \frac{1}{100} 2^{|B|}$$



Speedup when using set B to solve weakened CNF formulas

Instance	$\lvert B \rvert$	Solver	1 thread	2 threads	4 threads	8 threads	16 threads	32 threads	36 threads
$\mathtt{sgen}_{150}^{101}$	8	g3	$\boxed{2.3}$	$\boxed{4.6}$	$\boxed{8.8}$	$\boxed{16.8}$	$\boxed{31.3}$	$\boxed{37.0}$	$\boxed{37.0}$
	13	cd	$\boxed{1.9}$	$\boxed{3.9}$	$\boxed{7.7}$	$\boxed{14.9}$	$\boxed{29.4}$	$\boxed{56.9}$	$\boxed{62.6}$
$\mathtt{sgen}_{150}^{200}$	8	g3	$\boxed{1.6}$	$\boxed{3.3}$	$\boxed{6.2}$	$\boxed{12.2}$	$\boxed{22.3}$	29.9	29.9
$\mathtt{sgen}_{150}^{200}$	8	g3	$\boxed{1.8}$	$\boxed{3.6}$	$\boxed{7.1}$	$\boxed{13.3}$	$\boxed{25.5}$	$\boxed{36.2}$	$\boxed{36.2}$
	7	g3	$\boxed{2.2}$	$\boxed{4.4}$	$\boxed{8.5}$	$\boxed{15.8}$	$\boxed{28.0}$	28.8	28.8
	8	cd	$\boxed{1.3}$	$\boxed{2.6}$	$\boxed{5.0}$	$\boxed{9.6}$	$\boxed{19.1}$	22.8	22.8

$$\text{speedup} = \frac{\text{\# of UPs used by SAT solver on original CNF}}{\text{maximum \# of UPs used by all threads}}$$

Conclusion & Future Work

- Proposed d-hardness for unsatisfiable SAT formulas w.r.t. an deterministic SAT solver.
- Proposed an (ε, δ)-approximation algorithm for d-hardness estimation.
- Demonstrated effectiveness on several families of SAT formulas.
- Future work
  - Estimate the usefulness of cubes in Cube and Conquer
  - Extend the proposed ideas to existing algorithms based on proof systems strictly stronger than resolution: cutting planes, dual-rail based MaxSAT

Acknowledgements

- This work was supported by the Ministry of Science and Higher Education of Russian Federation, research project no. 075-03-2020-139/2 (goszadanie no. 2019-1339).
- Ilya Otpuschennikov’s research was funded by Ministry of Science and Higher Education of Russian Federation, project with no. of state registration: 121041300065-9.

Thank you.

chivdan@itmo.ru

@chivdan

https://github.com/ctlab/evoguess