On the use of random formulas in solving hard SAT problems

Oliver Kullmann

Computer Science Department
Swansea University

Phase transitions in discrete structures
July 27, 2016

As if they were random

I will concentrate in this talk on a recent success, the solution of the

Boolean Pythagorean Triples problem
(Wikipedia)
from Ramsey theory.

This revived the interest in the older **SAT-heuristics** for so-called

look-ahead solvers

and where considerations of *random formulas*, as training ground, always have played a role

— and where this hopefully can be done much better
(with your help).

It's a feature, not a bug

There will be exactly one `Theorem` in this talk,

and I do NOT show you the proof.

It's a feature, not a bug

There will be exactly one `Theorem` in this talk,

and I do NOT show you the proof.

Better don't ask me about it ... (according to the Daily Mail it will take you 10 billion years to read it).

It's a feature, not a bug

There will be exactly one `Theorem` in this talk,

and I do **NOT** show you the proof.

Better don't ask me about it ... (according to the Daily Mail it will take you 10 billion years to read it).

It should be the most opaque proof
(but kind of clever)
in the history of mankind :-)

Clause-sets

We consider conjunctive normal forms (CNFs), like

$$(a \vee \neg b \vee c) \wedge (\neg a \vee \neg c).$$

The basic task is **SAT decision** (the above is satisfiable, for example via $\langle a \to 1, c \to 0 \rangle$).

I typically prefer the more precise *combinatorial notion* of a *clause-set*, where the above becomes

$$\{ \{a, \overline{b}, c\}, \{\overline{a}, \overline{c}\} \},$$

as a *generalised hypergraph* (with polarities), but for this talk this doesn't matter much.

Consider UNSAT!

Phase transitions and all that are "typically" connected with *satisfiability*.

But the really interesting SAT stuff seems to happen on the UNSAT front.

I made already many years ago the

Conjecture ([10])

All unsatisfiable random instances are exponentially hard for resolution, the harder the lower the density (until very trivial densities).

Experiments confirm this (way below the threshold).

So what to do here??

Helping heuristics

The approach considered here is using the ideas from your community

to help SAT heuristics,
especially *branching heuristics*.

It seems methods related to survey/belief propagation, when used *directly*, are only of restricted use.

I will outline in my talk how
to **integrate** these methods into “ordinary” SAT heuristics.

200

In [7] we show:

Theorem

For every partition of $\mathbb{N} = \{1, 2, \ldots\}$ into two parts, at least one part will contain a Pythagorean Triple (a, b, c), i.e., $a^2 + b^2 = c^2$.

This was an open problem for nearly forty years, and since we derived the "longest proof ever" (200 TB), that created some media buzz, e.g. Lamb [12] (Nature) and in Germany Dambeck [3] (Spiegel Online).

For further links see

http://cs.swan.ac.uk/~csoliver/papers.html#PYTHAGOREAN2016C

For example for the 2-partitioning of $\mathbb{N}$ into odd and even number:

- The odd numbers do not contain a triple: "odd + odd = even".
- But the even numbers contain e.g. $2 \cdot (3, 4, 5) = (6, 8, 10)$.

Translation into SAT I

Due to Compactness, the Theorem holds iff

> there is $n \geq 1$, such that for every 2-partitioning of $\{1, \dots, n\}$, at least one of the two parts contains a Pythagorean triple.

Using boolean variables $1, \dots, n$, thus the Theorem holds iff

$$\bigvee_{a^2+b^2=c^2, c \leq n} (a \wedge b \wedge c) \vee (\overline{a} \wedge \overline{b} \wedge \overline{c})$$

is a tautology iff

$$\bigwedge_{a^2+b^2=c^2, c \leq n} (a \vee b \vee c) \wedge (\overline{a} \vee \overline{b} \vee \overline{c})$$

is unsatisfiable (this is just the usual translation of **hypergraph 2-colouring**, applied to the hypergraph of triples up to n).

Translation into SAT II

Using $n = 7825$, thus the Theorem holds if

$$(3 \wedge 4 \wedge 5) \vee (\overline{3} \wedge \overline{4} \wedge \overline{5}) \vee \cdots \vee \\ (625 \wedge 7800 \wedge 7825) \vee (\overline{625} \wedge \overline{7800} \wedge \overline{7825})$$

is a tautology (listing all triples, twice), i.e., if

$$(3 \vee 4 \vee 5) \wedge (\overline{3} \vee \overline{4} \vee \overline{5}) \wedge \cdots \wedge \\ (625 \vee 7800 \vee 7825) \wedge (\overline{625} \vee \overline{7800} \vee \overline{7825})$$

is unsatisfiable, i.e., if the hypergraph

$$(\{1, \dots, 7825\}, \{\, \{3,4,5\}, \{6,8,10\}, \{5,12,13\}, \dots, \{625,7800,7825\}\, \})$$

is non-2-colourable.

7825

Where does $n = 7825$ come from?

- With the **appropriate choice** of translation/encoding and local-search solver, finding some satisfying assignment for such problems from Ramsey theory is **far easier** than deciding unsatisfiability.
- In this case, `ddfw` from the Ubcsat-suite seems strongest, and finds easily all instances up to $n = 7824$ satisfiable.

We note here that some kind of “phase transition” seems taking place:

1. A backbone (forced assignments) builds up until $n = 7824$, consisting of 2304 of the 3745 variables (after elimination of “blocked clauses”; originally 6494 occurring variables).
2. The clauses added with hypotenuse $c = 7825$ finally contradict these forced assignments.

Some remarks I

The number of clauses is asymptotically $\frac{1}{\pi} n \cdot \ln n$.

In some sense a "randomisation" takes place:

1. The original proof van der Waerden [15] for the existence of van-der-Waerden numbers (hyperedges: arithmetic progressions of fixed length) is purely combinatorial.
2. The refinement Green and Tao [4] to arithmetic progressions in the *prime numbers* uses some randomisation (to make the prime numbers look random).
3. Similarly, the basic proof of Schur [14] on the triples $x + y = z$ is purely combinatorial (a simple application of Ramsey's theorem).
4. Accordingly, for the refinement to *square numbers* (our case) one would expect that some randomisation should be appropriate.

Some remarks II

"Blocked clauses" ?

- This is indeed equivalent here to compute first the

  2-**core** of the hypergraph of triples,

  before translation.
- Perhaps determining n, where the m-core becomes non-empty, is possibly,

  yielding a lower bound on $\mathrm{Ptn}(3; m)$ for m colours.

Numerically, the maximal n where the m-core is empty, $m = 2, 3, 4, 5, 6, 7$, and the successive quotients:

m	2	3	4	5	6	7
n	64	1104	19824	128315	637324	2551020
quot		17.25	17.96	6.47	4.97	4.00

Some remarks III

A few quotients:

1. $2 \cdot 9472/7825 = 2.42\ldots$
2. $2 \cdot 9472/6494 = 2.91\ldots$ (actually occurring vertices)
3. $2 \cdot 7336/3745 = 3.91\ldots$ (the 2-core).

All clauses have length 3.

CDCL killed random

Until around 2000 there were two dominant SAT-solving paradigms:

1. local-search for finding satisfiable assignments ([8]);
2. *look-ahead* especially for unsatisfiable instances ([5]).

Then a new paradigm emerged:

conflict-driven clause-learning (CDCL; [13])

which showed stunning performance on many instances of interest (but not on random ones, sat or unsat).

Random instances were important for both of the older paradigms, but not for CDCL.

So interest in random instances decreased in SAT-solving.

Furthermore, SAT is actually very much about UNSAT.

Overview on CDCL

The basic loop of CDCL for input F is as follows:

- Starting with the empty assignment, extend it,
  - using a variable-choice
  - and variable-value heuristics,
  - only checking for UCP,

  until either a satisfying assignment is found, or, usually, a *conflict* was obtained (the empty clause).
- Analyse the conflicting assignment φ for the "causes" of the conflict, find some $\varphi' \subseteq \varphi$, negate φ', and add this "learnt" clause to the clause-set.

Reminder: UCP is *unit-clause propagation*:

If there is a unit-clause $\{x\} \in F$, set x to true, and simplify.

Further remark: Actually, CDCL solvers are "lazy", so do not actually perform the assignment, but "keep it in mind".

Heuristics for CDCL

There are two main heuristics for a CDCL solver:

1. variable-choice
2. learning.

The main (current!) idea for variable-choice seems to be:

go for variables which recently where involved in conflicts.

Learning is about “making progress” (while staying cheap).

It seems, ideas from the area of BP do not play a role (currently).

Overview on Look-Ahead

Basically, we have "good old DPLL":

For input F, choose a variable v,
and split recursively into
$\langle v \to 0 \rangle * F$ and $\langle v \to 1 \rangle * F$.

Additionally *reductions* are involved.

The "look-ahead" means that:

- for the choice of v,
- the effect of the reduction after splitting
- is partially considered (by actually *running* the reductions).

Why are CDCL solvers often better than look-ahead?

Two approaches to explain the advantage of CDCL:

- Look-ahead is basically tree-like (recursive splitting), while CDCL is dag-like (can *reuse* “lemmas”).
- CDCL is more “optimistic”, looks out for a “weakness”, while look-ahead assume the worst-case.

It seems the instances where look-ahead is better are

“consistently hard”,

(like random formulas), while for CDCL there must be “soft spots”.

Cube and Conquer: The discovery

For experiments with (hard) instances from Ramsey theory (van-der-Waerden; Ahmed, Kullmann, and Snevily [1]), I made the following astounding observation:

1. I just wanted to be able to easily monitor progress, and possibly do parallelisation.
2. So I took my own look-ahead solver, the `OKsolver`, using it to split the instances into a *large number* of instances, cutting off the splitting tree, and at the leaves I ran a CDCL-solver.
3. When the splitting was done reasonably, so that the leaf-instances are roughly of the same hardness, then the total run time, even with a very simple implementation,

   was MUCH LOWER than
   what any single solver could achieve.

Cube and Conquer: Global versus Local I

In Heule, Kullmann, Wieringa, and Biere [6] we made a systematic approach,

- obtaining an algorithmic scheme
- which seems currently the best for really hard problems.

Our current approach for explaining the success is as follows:

1. The worst-case approach of look-ahead is good for splitting, but not for solving.
2. The dag-like structures, exploited by CDCL, are somewhat of a “local” phenomenon.
3. Also, if the instance is too big, then CDCL “looses overview”.

Cube and Conquer: Global versus Local II

Perhaps this has to do with

"short range" versus
"long range" interactions?!

Apparently

- "short range is dag-like",
- while "long-range is tree-like"?!

In any case, this motivates to look again at the

branching heuristics (splitting heuristics)
for look-ahead solvers.

A general theory

The following theory of branching heuristics has been developed (see Handbook chapter [11]):

1. Considered are **branchings** $F \rightsquigarrow (F_1, \dots, F_m)$.
2. A notion of **distance** $d(F, F') \in \mathbb{R}_{>0}$ is needed.
3. We obtain **branching tuples**

$$\big(d(F, F_1), \dots, d(F, F_m)\big) \in \mathcal{BT} := \bigcup_{m \in \mathbb{N}} \mathbb{R}^m_{>0}.$$

4. Finally a **projection** $\rho : \mathcal{BT} \to \mathbb{R}$ is needed.

Choose a branching with minimal projection value.

Canonical projection

I have shown that under rather general conditions

there is exactly one **canonical linear order** on $\mathcal{BT}$.

The naturally associated **canonical projection** is

$$\tau((d_1, \ldots, d_m)) = \text{ that } x > 1 \text{ with } \sum_{i=1}^{m} x^{-d_i} = 1.$$

(To show uniqueness, tuples of arbitrary width are required.)

So the main question is the choice of the distance $d(F, F')$.

Enabling shortcuts

When is look-ahead successful?

When branches are cut off early!

- So $d(F, F')$ should be large if F' has many more *future* reductions.
- In practice that means if F' has many more *expected* UCPs.

Now certain theoretical and all practical evidence says that the

weighted number of new clauses

is a good measure (the more the better).

All direct measure (number of eliminated variables) fail
(they are too pessimistic).

The weighting

The shorter a clause, the closer to unit.

Thus, an exponential decay of the weight with length is the basis.

Especially for **random *k*-SAT**,

- more sophisticated schemes have been developed.
- They treat every literal differently.
- The basic idea is to “estimate” how “likely” it is that the literal becomes `false`.
- From that one can “estimate” how “likely” a clause becomes unit.

I assume now that via adapting the BP equations,
one should be able to do much better.

Behaves like random

Various heuristic schemes for look-ahead solvers have been developed.

The Pythagorean Problems are far best attacked by schemes especially successful on random instances.

(But using different parameter values.)

Summary and outlook

I Hopefully some avenues for new applications to SAT have been outlined.

II They rely on “heuristical” similarity to randomness.

III Hopefully this can be merged with *theoretical* applications to Ramsey theory, where schemes of “pseudo-randomness” have to be developed.

End

(references on the remaining slides).

For my papers see
http://cs.swan.ac.uk/~csoliver/papers.html.

Bibliography I

[1] Tanbir Ahmed, Oliver Kullmann, and Hunter Snevily. On the van der Waerden numbers $\mathrm{w}(2; 3, t)$. *Discrete Applied Mathematics*, 174:27–51, September 2014. doi:10.1016/j.dam.2014.05.007.

[2] Armin Biere, Marijn J.H. Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, February 2009. ISBN 978-1-58603-929-5.

[3] Holger Dambeck. Zahlenrätsel: Der längste Mathe-Beweis der Welt. *Spiegel Online*, May 2016. URL http://www.spiegel.de/wissenschaft/mensch/der-laengste-mathe-beweis-der-welt-umfasst-200-ter html.

Bibliography II

[4] Ben Green and Terence Tao. The primes contain arbitrarily long arithmetic progressions. *Annals of Mathematics*, 167(2):481–547, 2008.

[5] Marijn J. H. Heule and Hans van Maaren. Look-ahead based SAT solvers. In Biere, Heule, van Maaren, and Walsh [2], chapter 5, pages 155–184. ISBN 978-1-58603-929-5. doi:10.3233/978-1-58603-929-5-155.

[6] Marijn J.H. Heule, Oliver Kullmann, Siert Wieringa, and Armin Biere. Cube and conquer: Guiding CDCL SAT solvers by lookaheads. In Kerstin Eder, João Lourenço, and Onn Shehory, editors, *Hardware and Software: Verification and Testing (HVC 2011)*, volume 7261 of *Lecture Notes in Computer Science (LNCS)*, pages 50–65. Springer, 2012. doi:10.1007/978-3-642-34188-5_8. URL http://cs.swan.ac.uk/~csoliver/papers.html#CuCo2011.

Bibliography III

[7] Marijn J.H. Heule, Oliver Kullmann, and Victor W. Marek. Solving and verifying the boolean Pythagorean Triples problem via Cube-and-Conquer. In Nadia Creignou and Daniel Le Berre, editors, *Theory and Applications of Satisfiability Testing - SAT 2016*, volume 9710 of *Lecture Notes in Computer Science*, pages 228–245. Springer, 2016. ISBN 978-3-319-40969-6. doi:10.1007/978-3-319-40970-2_15.

[8] Henry A. Kautz, Ashish Sabharwal, and Bart Selman. Incomplete algorithms. In Biere et al. [2], chapter 6, pages 185–203. ISBN 978-1-58603-929-5. doi:10.3233/978-1-58603-929-5-185.

Bibliography IV

[9] Oliver Kullmann. Towards an adaptive density based branching rule for SAT solvers, using a database for mixed random conjunctive normal forms built upon the advanced encryption standard (AES). In John Franco, Henry Kautz, Hans Kleine Büning, Hans van Maaren, Bart Selman, and Ewald Speckenmeyer, editors, *Fifth International Symposium on Theory and Applications of Satisfiability Testing*, pages 190–200, 2002. University of Cincinnati (USA), May 6, 2002 to May 9, 2002.

[10] Oliver Kullmann. First report on an adaptive density based branching rule for DLL-like SAT solvers, using a database for mixed random conjunctive normal forms created using the advanced encryption standard (AES). Technical Report CSR 19-2002, Swansea University, Computer Science Report Series, March 2002. URL

Bibliography V

http://www-compsci.swan.ac.uk/reports/2002.html. Extended version of [9].

[11] Oliver Kullmann. Fundaments of branching heuristics. In Biere et al. [2], chapter 7, pages 205–244. ISBN 978-1-58603-929-5. doi:10.3233/978-1-58603-929-5-205.

[12] Evelyn Lamb. Maths proof smashes size record: Supercomputer produces a 200-terabyte proof – but is it really mathematics? *Nature*, 534:17–18, June 2016. doi:10.1038/nature.2016.19990.

[13] Joao P. Marques-Silva, Ines Lynce, and Sharad Malik. Conflict-driven clause learning SAT solvers. In Biere et al. [2], chapter 4, pages 131–153. ISBN 978-1-58603-929-5. doi:10.3233/978-1-58603-929-5-131.

[14] Issai Schur. Über die Kongruenz $x^m + y^m = z^m \pmod{p}$. *Jahresbericht der Deutschen Mathematikervereinigung*, 25: 114–117, 1917.

Bibliography VI

[15] B.L. van der Waerden. Beweis einer Baudetschen Vermutung. *Nieuw Archief voor Wiskunde*, 15:212–216, 1927.