Big proofs and SAT

Oliver Kullmann

Computer Science Department
Swansea University

British Logic Colloquium 2017
University of Sussex
September 8, 2017

Harnessing SAT

The power of solving propositional logic, most importantly

- determining a propositional formula to be **satisfiable** or **unsatisfiable**
- resp. to be **falsifiable** or **tautological**,

has increased dramatically over the last two decades.

Especially industrial applications
(safety, correctness)
are impressive.

Also apparently every modern ATP-solver nowadays uses SAT.

Standard use of SAT in ATP

The typical use of SAT in ATP

is based on **abstraction**:
abstract away the parts of the problem
beyond propositional logic.

This leads to two main characteristic features:

1. **heavy interaction** between the non-propositional parts of the solver and the SAT part;
2. the SAT problems are **relatively small**.

A similar approach one finds in SMT ("SAT modulo theory").

Heavier use of SAT

In the direction of Herbrand's Theorem, it is also possible to have

SAT solving as the main part
of the solving process.

Basically

1. **"all instantiations"** of quantifiers are considered (reducing the problem to infinite propositional logic),
2. and **compactness** is applied (creating infinitely many finite problems).

This results naturally in a much heavier use of SAT.

Now very big SAT problems are to be solved.

Our endeavours are going in this direction.

SAT solving different for easy and hard

Not only are the SAT problems arising in this heavy use of SAT very big,

but they are typically also **very hard**.

The success of SAT solving over the last two decades however

concentrates on problems which are
(very) big, but indeed **(relatively) easy** —
these methods go aggressively for the **"easy success"**,
not for the **worst case**.

So new SAT tools are needed:

integrating aggressive solving and worst-case planning.

This talk

In this talk I

focus on the heaviest use of SAT:
the problem is **directly expresses as a SAT problem**
(for efficiency of encoding).

For this I present

improved SAT solving (**Cube-and-Conquer**).

Outline

Ramsey theory

- Many problems of Ramsey theory can be understood easily,
- and they yield great problems for SAT.

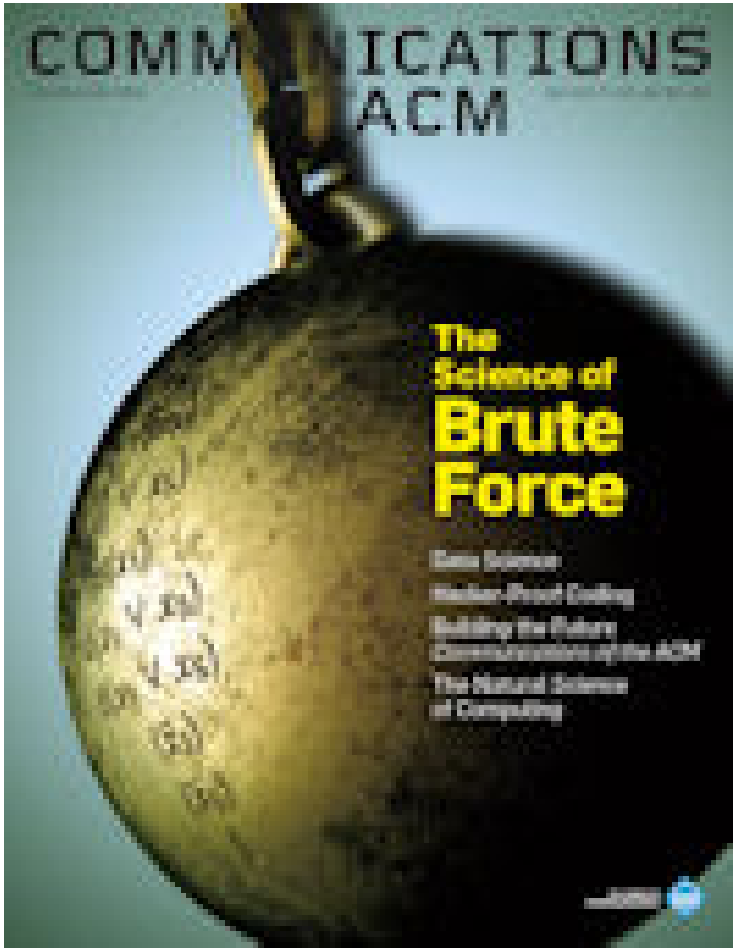
The fundamental theme is:

> For infinite sets X and certain simple "structures" on X:
> how resilient is this structure against finite partitioning?

Remarks:

- inherent difficulty completely open
- see Heule and Kullmann [7] ("The science of brute force", CACM) for an accessible introduction.

The science of Demolition



https://cacm.acm.org/magazines/2017/8

A fundamental problem I

Consider

- a finite set $\mathbb{P}$ of polynomials $P(x_1, \ldots, x_k)$ over k variables,
- with integer coefficients.

Let

$$\boldsymbol{N(\mathbb{P})} := \left\{ \{x_1, \ldots, x_k\} \subset \mathbb{N} : \forall\, P \in \mathbb{P} : P(x_1, \ldots, x_k) = 0 \right\}$$

be the set of (common) zeros (Nullstellen) of $\mathbb{P}$ over **the natural numbers**, where for convenience we don't take the tuple, but just the set of components of the zero-vector.

$$X = \mathbb{N} = \{1, 2, \ldots\}$$

For example

$$N(\{x^2 + y^2 - z^2\}) = \{\{3, 4, 5\}, \{6, 8, 10\}, \{5, 12, 13\}, \ldots\}.$$

A fundamental problem II

Definition

$\mathbb{P}$ is called *m*-**regular** for some $m \in \mathbb{N}$, if the hypergraph $N(\mathbb{P})$ is not m-colourable, that is:

$$\text{For every partition } \mathbb{N} = X_1 \cup \cdots \cup X_m \text{ of } \mathbb{N} \text{ into } m \text{ subsets}$$
$$\text{there is } i \text{ and } \vec{x} \in N(\mathbb{P}) \text{ with } \vec{x} \subseteq X_i.$$

$\mathbb{P}$ is called **regular** if it is m-regular for all $m \geq 1$.

A central problem of Ramsey theory is:

$$\text{Determine the } (m\text{-})\text{regular systems } \mathbb{P}.$$

The linear case was completely solved by Rado [16], generalising work by Schur [17] and van der Waerden [18]; a recent overview was given in Baglini [2].

A fundamental problem III

The general case is wide open.

- By Hilbert's Tenth Problem the case "1-regularity" is undecidable.
- But for all $m \geq 2$ the complexity of "m-regularity" is completely open (from linear-time to undecidable).
- And so is "regularity".

We considered the single equation $x^2 + y^2 = z^2$.

Finitisation

Partitioning into two sets, i.e., $m = 2$, can be directly encoded using boolean variables:

For $n \in \mathbb{N}$ let v_n be a boolean variable.
v_n `true` means the first part,
v_n `false` means the second part.

Now $\mathbb{P}$ is 2-regular iff the (possibly) infinite disjunction

$$\bigvee_{\vec{x} \in N(\mathbb{P})} \left(\bigwedge_{x \in \vec{x}} v_x\right) \vee \left(\bigwedge_{x \in \vec{x}} \neg v_x\right)$$

is a tautology, which by compactness is true iff for some $n \in \mathbb{N}$ the **finite disjunction**

considering only the $\vec{x} \in N(\mathbb{P})$ with $\vec{x} \subseteq \{1, \ldots, n\}$

is a tautology.

The Boolean Pythagorean Triples Problem I

The question about the regularity of

$$x^2 + y^2 - z^2$$

is a long-standing open problem.

- Only 1-regularity was known (since there exist Pythagorean triples, e.g., $3^2 + 4^2 = 5^2$).
- Ron Graham asked in the 80s specifically whether 2-regularity holds (we call this the Boolean Pythagorean Triples Problem).
- There were some opinions (conjectures) that it is not 2-regular.

The Boolean Pythagorean Triples Problem II

We (together with Marijn Heule and Victor Marek)
showed via SAT
the 2-regularity (Heule, Kullmann, and Marek [9], [11, 12]),
and got $100 for it.

Such a solution needs a **proof**, which can be automatically checked.

We provided a proof of 200 TB and checked it.
Meanwhile it has also been checked independently (Cruz-Filipe, Marques-Silva, and Schneider-Kamp [5]).

Concrete (practical) re-formulation I

Call three natural numbers $a, b, c \in \mathbb{N}$ a Pythagorean triple if $a^2 + b^2 = c^2$.

Is it possible
to partition $\mathbb{N}$ into two sets $A \cup B = \mathbb{N}$,
such that neither A nor B contains a Pythagorean triple?

Let's start with partitioning $\mathbb{N}$ into even and odd numbers. Remember:

- even * even = even, odd * odd = odd
- even + even = even, odd + odd = even.

Thus there can be no Pythagorean triple consisting only of odd numbers.
So we are fine on the odd part, but the even part contains e.g.

$$2 \cdot (3, 4, 5) = (6, 8, 10), \quad 6^2 + 8^2 = 10^2.$$

Concrete (practical) re-formulation II

So we need to partition the even numbers further,

to destroy all Pythagorean triples.

For a more experimental approach, it would be easier

instead of partitioning $\mathbb{N}$,
to partition $\{1, \ldots, n\}$ for $n = 5, 6, 7 \ldots$

So we can first destroy all Pythagorean triples with hypotenuse $z \leq 5$, then $z \leq 6$, and so on. At the beginning it's pretty simple, since there are only few Pythagorean triples:

$$(3, 4, 5), (6, 8, 10), (5, 12, 13), (9, 12, 15), (8, 15, 17), \ldots$$

But slowly it gets harder ... will it ever end?!?

Asymptotically, the number of clauses is $\frac{1}{\pi} n \cdot \ln(n)$.

Concrete (practical) re-formulation III

The SAT problem is thus for e.g. $n = 15$:

$$(v_3 \wedge v_4 \wedge v_5) \vee (\neg v_3 \wedge \neg v_4 \wedge \neg v_5) \vee (v_6 \wedge v_8 \wedge v_{10}) \vee (\neg v_6 \wedge \neg v_8 \wedge \neg v_{10}) \vee (v_5 \wedge v_{12} \wedge v_{13}) \vee (\neg v_5 \wedge \neg v_{12} \wedge \neg v_{13}) \vee (v_9 \wedge v_{12} \wedge v_{15}) \vee (\neg v_9 \wedge \neg v_{12} \wedge \neg v_{15}).$$

It is easy to see that this DNF is falsifiable.

The theorem I

https://en.wikipedia.org/wiki/Boolean_Pythagorean_triples_problem

We showed in our SAT 2016 paper ([9, 10]), that one can go up up to $n = 7824$, and then it stops.

Theorem

For all sets A_1, A_2 with $A_1 \cup A_2 = \mathbb{N}$ there is $i \in \{1, 2\}$ and $a, b, c \in A_i$ with $a^2 + b^2 = c^2$. More precisely:

For $n = 7825$, in every partition of $\{1, \ldots, n\}$,
at least one part contains a Pythagorean triple.

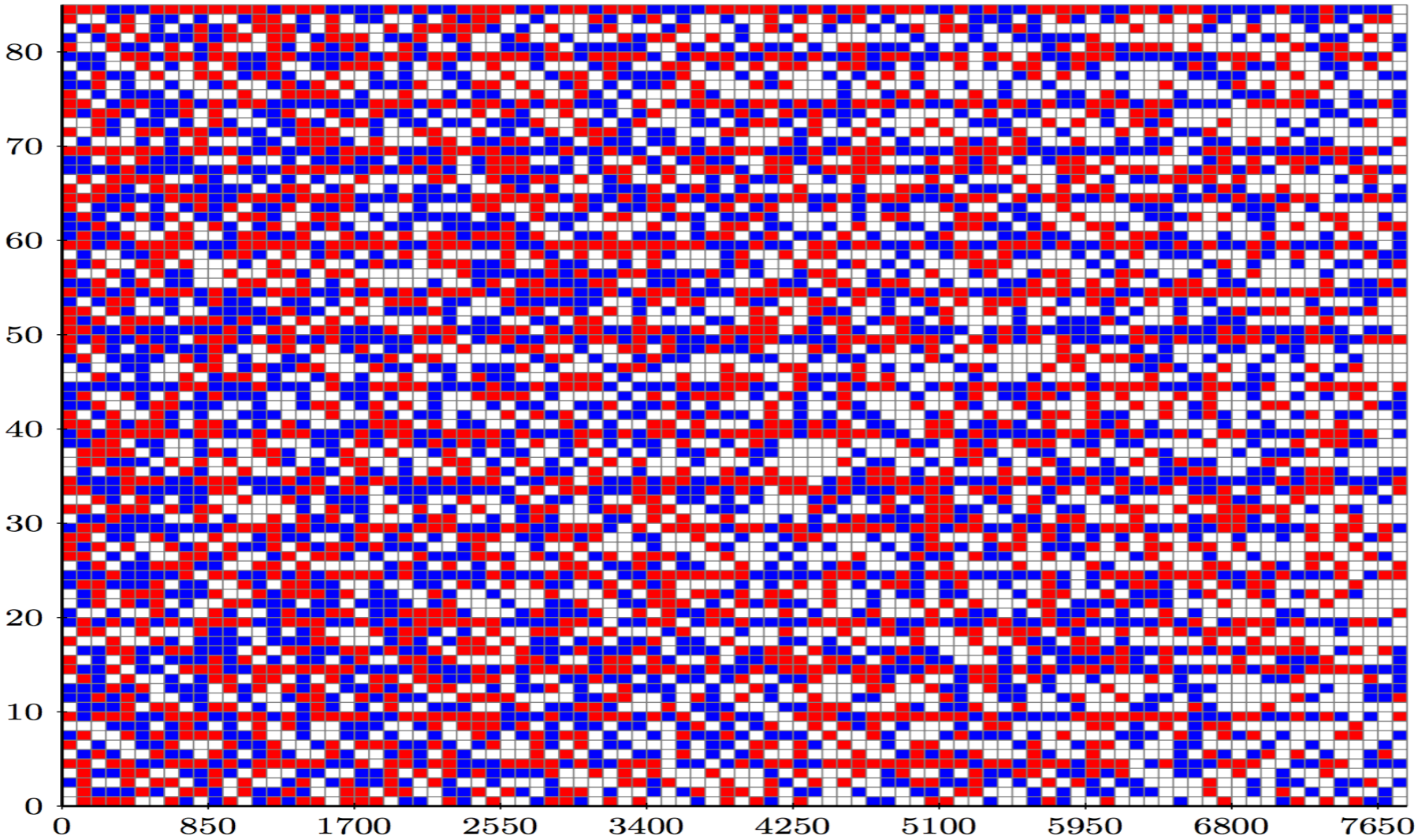
For $n = 7824$ there exist a partition with
no part containing a Pythagorean triple.

The theorem II

This solved a problem open for 30 years, with the longest proof yet.

Different from all previous contributions
to Ramsey theory via SAT solving,
here also the existence problem for n was open.

A solution (white means unassigned)



Actually **proving** it

Just claiming to have solved it — that's not much.

- The total run-time for solving the problem was two days on a supercomputer, where we used only 800 cores.
- Without our novel techniques, just using standard SAT techniques, it would have needed say 1000 times more *time*.
- Still doable in principle (the supercomputer has 10^6 cores).
- But an important point is the **extracted proof** (which we got *down* to "just" 200 TB).
- The total run-time must be small, **AND** the proof format must allow for good compression.
- Without the novel proof format, a blow-up of *space* of at least a factor of 1000 would have occurred.

Remark: without SAT, the age of the universe would be nothing to solve it.

Independent verification

At least one first (published) independent verification took place (Cruz-Filipe et al. [5], Cruz-Filipe and Schneider-Kamp [4]):

- They used the 10^6 **main cases** (still highly complex), as computed by our SAT-solving system (altogether 68 GB).
- These 10^6 SAT problems were solved by some SAT solver.
- For each case extracting a proof using our tools for preprocessing.
- Finally on each of the 10^6 extracted proofs, a verified proof checker was run.
- This checker was extracted by the Coq interactive theorem prover (with some work-arounds for speed and memory).

SAT solving

The previous DNF is negated, yielding a CNF, and the task is to show

unsatisfiability (i.e., inconsistency).

SAT solvers solve CNFs.

- The hybrid SAT-solving method Cube-and-Conquer, whose idea we developed in the context of applications to Ramsey theory, was adopted to the task (various heuristics optimised).
- Due to the nature of "C&C", whether performed on a single computer or on a cluster doesn't matter.

C&C: old and new

SAT is “solving CNFs by brute-force, guided by brute reason”.

- Two main paradigms for “brute reason” have been developed.
- The first and older one, **LOOK-AHEAD**,

  is about logical inference and systematic case distinctions (“systematic and slow”).
- The second and newer one, **CDCL**, is mainly responsible for the SAT revolution in industrial applications.

  CDCL is about making mistakes quickly and learning from them (“quick and dirty, but with magic *local cleverness*”).

C&C combines the two:

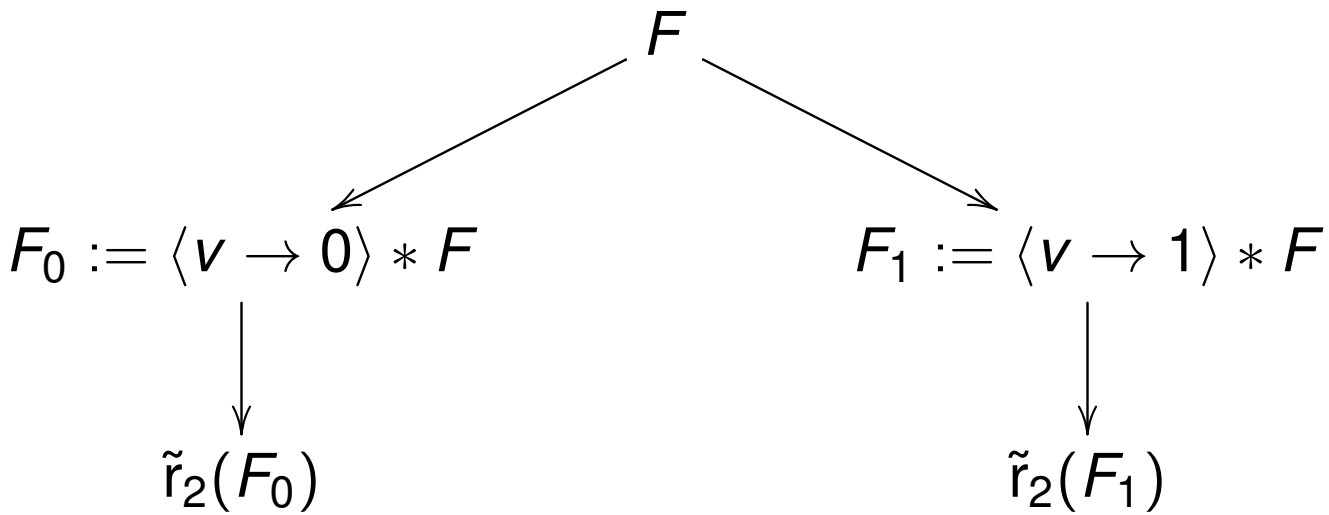
First we build a systematic (and clever) big(!) case distinction.
Then we solve the cases by the quick method (independently!).

Look-ahead solvers

For background see the two Handbook-chapters Heule and van Maaren [6], Kullmann [14]:

Split recursively, applying (strong) reductions,
guided by (strong) heuristics.

That is, for input $F \in \mathcal{CLS}$, choose variable $v \in \mathrm{var}(F)$ and split

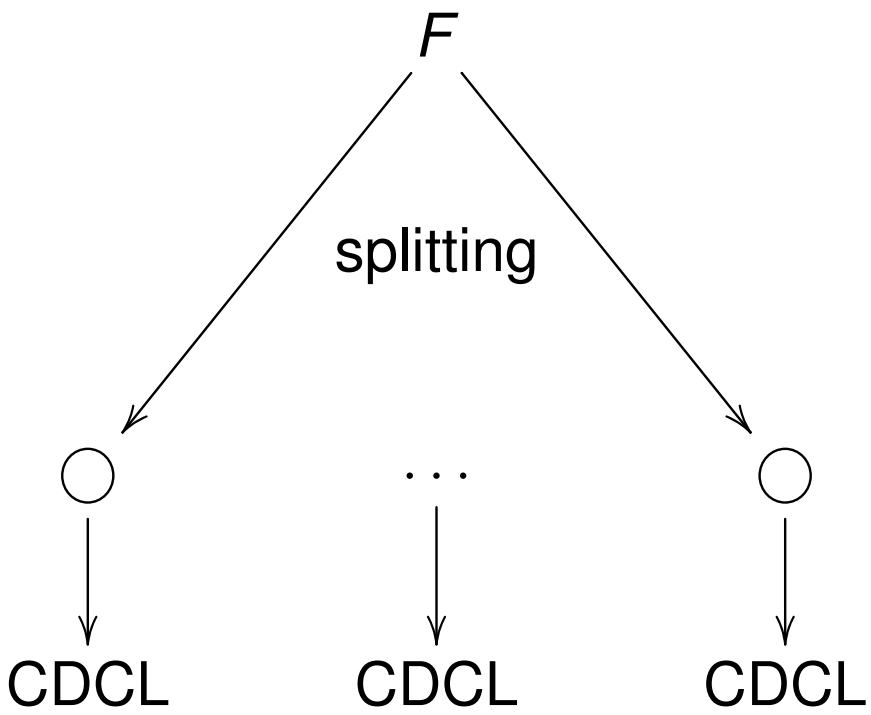


for some reduction $\tilde{r}_2 : \mathcal{CLS} \to \mathcal{CLS}$ (e.g., elimination of failed literals).

A hybrid scheme

The **C&C paradigm** ([8, 12]) has two phases:

1. First a look-ahead solver is employed to split the problem — the splitting tree is cut off appropriately.
2. At the leaves of the tree, CDCL solvers are employed.



First goal: parallelisation

The number N of leaves for the cube-phase is roughly

- in the thousands for relatively easy problems (say one day total run-time);
- in the millions for hard problems (say a month total run-time);
- in the billions for very hard problems.

The sub-problems should be at most **one minute**.

C&C achieves a good equal splitting
(that's what look-ahead is good at!).
The sub-problems (for CDCL) are scheduled independently.

- So a great **linear speed-up** for a large number of processors.
- The cube-phase has also **controlled run-time**.
- And the sub-problems can be **easily uniformly sampled**.

That's it?

So well, that's something:

- Hard problems need distributed solving.
- C&C delivers this, scaling very well.
- That's also quite natural:

The **tree-structure** is optimal for this.
Look-ahead heuristics prefer **equal splittings**.

But that's NOT the end of the story ...

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 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` ([13, 15]), using it to split the instances into a *large number N* of sub-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.

Something's going on I

Consider again N (the number of leaves in the cube-phase):

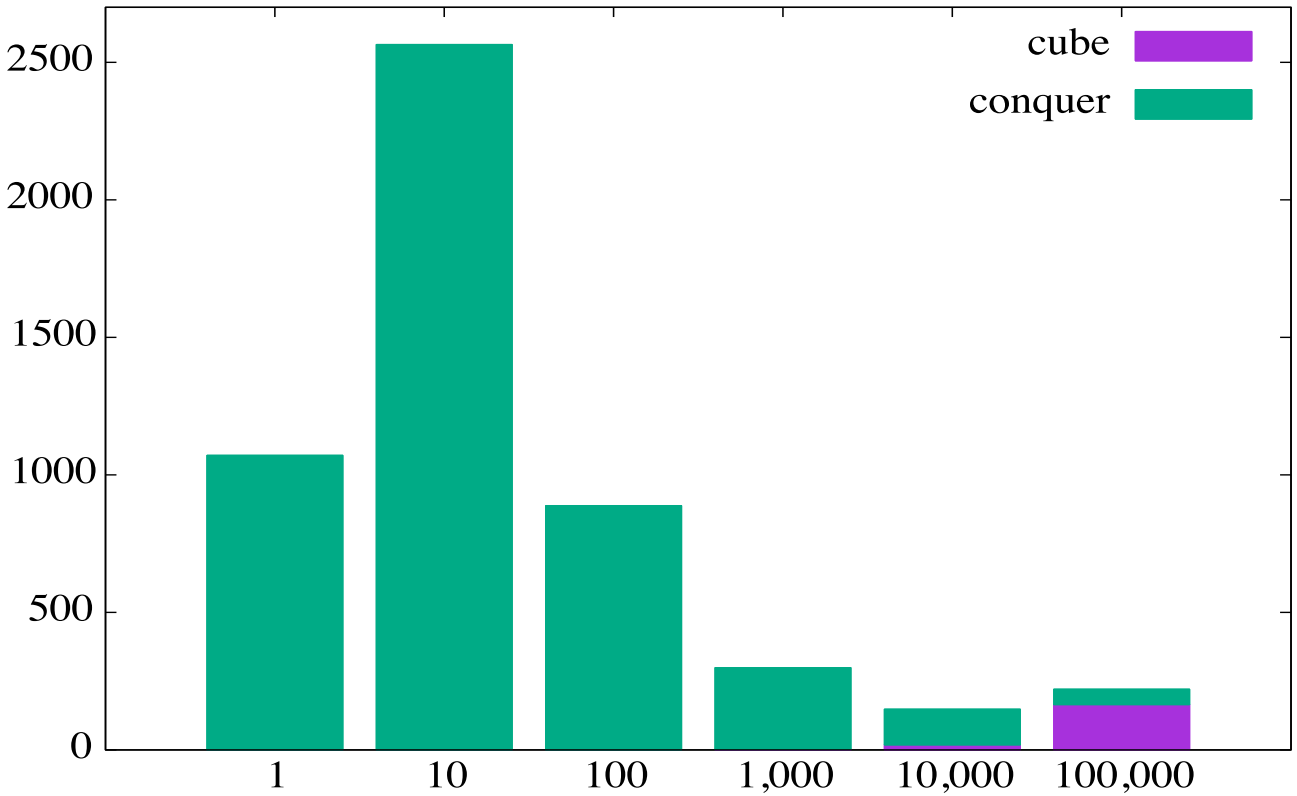
1. $N = 1$ means pure CDCL.
2. Very large N means pure look-ahead.

Now consider the total run-time in dependency on N:

1. Typically, first it increases,
2. then it **decreases** (only for a **large** number of sub-problems!),
3. then it stays for some time at a plateau,
4. and finally it typically increases again (often dramatically, but not for the Pythagorean triples problem).

In the area of optimal N,
the total run-time can be several orders of magnitudes faster
than any single method!!

Something's going on II



Example with Schur triples $a + b = c$
and 5 colours: a clause-set with
708 variables and 22608 clauses.

Why are CDCL solvers often better than look-ahead?

Three 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.
- CDCL is "less intelligent", but "much faster".

It seems the instances where look-ahead is better are

"consistently hard",

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

None of these approaches seems to be able
to explain the C&C phenomenon.

Open Problems I

Explain (theoretically and practically) where look-ahead is best.

Two directions

Two basic opposite hypothesis' about the C&C phenomenon:

I It's a **weakness** of CDCL (resp. current implementation): these solvers have a "point of competence" — you can't run CDCL solvers for a long time.

II It's a **strength** of look-ahead: look-ahead "understands" better the "global structure".

Cube and Conquer: Global versus Local

Our current approach for explaining the success is as follows:

1. In some sense, the heuristic of look-ahead is “global”, the heuristic for CDCL is “local”.
2. The worst-case approach of look-ahead is good for splitting (globally), but not for solving (exploiting local structures).
3. The dag-like structures, exploited by CDCL, are somewhat of a “local” phenomenon.

Open Problems II

Explain (theoretically and practically) why and where C&C is best.

It seems some form of “hybrid” proof theory is needed.

Outlook

I ATP hopefully has still much more to gain via SAT.

II For hard problems the interplay between “old” and “new” in C&C seems crucial.

III The interplay between these paradigms needs to be investigated.

IV “Old” (look-ahead) seems more appropriate for “planning”, “new” (CDCL) more for “solving”.

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] Lorenzo Luperi Baglini. Partition regularity of nonlinear polynomials: a nonstandard approach. *INTEGERS: Electronic Journal of Combinatorial Number Theory*, 14:23, 2014. URL http://www.integers-ejcnt.org/vol14.html. #A30.

[3] 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.

Bibliography II

[4] Luís Cruz-Filipe and Peter Schneider-Kamp. Formally proving the boolean Pythagorean Triples Conjecture. In *LPAR-21: 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 46 of *EPiC Series in Computing*, pages 509–522, 2017. URL https://easychair.org/publications/paper/340348.

[5] Luís Cruz-Filipe, Joao Marques-Silva, and Peter Schneider-Kamp. Efficient certified resolution proof checking. In Axel Legay and Tiziana Margaria, editors, *International Conference on Tools and Algorithms for the Construction and Analysis of Systems: TACAS 2017, Part I*, volume 10205 of *Lecture Notes in Computer Science*, pages 118–135. Springer, 2017. doi:10.1007/978-3-662-54577-5_7.

Bibliography III

[6] Marijn J. H. Heule and Hans van Maaren. Look-ahead based SAT solvers. In Biere, Heule, van Maaren, and Walsh [3], chapter 5, pages 155–184. ISBN 978-1-58603-929-5. doi:10.3233/978-1-58603-929-5-155.

[7] Marijn J.H. Heule and Oliver Kullmann. The science of brute force. *Communications of the ACM*, 60(8):25–34, August 2017. doi:10.1145/3107239.

[8] 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 IV

[9] 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.

[10] Marijn J.H. Heule, Oliver Kullmann, and Victor W. Marek. Solving and verifying the boolean Pythagorean Triples problem via Cube-and-Conquer. Technical Report arXiv:1605.00723v1 [cs.DM], arXiv, May 2016. URL http://arxiv.org/abs/1605.00723.

Bibliography V

[11] Marijn J.H. Heule, Oliver Kullmann, and Victor W. Marek. Solving very hard problems: Cube-and-Conquer, a hybrid SAT solving method. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017)*, pages 4864–4868, 2017. doi:10.24963/ijcai.2017/683.

[12] Marijn J.H. Heule, Oliver Kullmann, and Armin Biere. Cube-and-conquer for satisfiability. In Youssef Hamadi and Lakhdar Sais, editors, *Handbook of Parallel Constraint Reasoning*, chapter 2, pages 31–58. Springer, 2018. ISBN 978-3-319-63515-6. URL http://www.springer.com/us/book/9783319635156.

Bibliography VI

[13] Oliver Kullmann. Investigating the behaviour of a SAT solver on random formulas. Technical Report CSR 23-2002, Swansea University, Computer Science Report Series, October 2002. URL http://www-compsci.swan.ac.uk/reports/2002.html. 119 pages.

[14] Oliver Kullmann. Fundaments of branching heuristics. In Biere et al. [3], chapter 7, pages 205–244. ISBN 978-1-58603-929-5. doi:10.3233/978-1-58603-929-5-205.

[15] Oliver Kullmann. The OKlibrary: Introducing a "holistic" research platform for (generalised) SAT solving. *Studies in Logic*, 2(1):20–53, 2009.

[16] Richard Rado. *Studien zur Kombinatorik*. PhD thesis, Philosophische Fakultät der Friedrich-Wilhelms-Universität, Berlin, 1933. URL http://eudml.org/doc/203249.

Bibliography VII

[17] Issai Schur. Über die Kongruenz $x^m + y^m = z^m \pmod{p}$. *Jahresbericht der Deutschen Mathematikervereinigung*, 25: 114–117, 1917. URL https://eudml.org/doc/145475.

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