The Computational Challenge of Combinations

Jakob Nordström

University of Copenhagen and Lund University

Inaugural Professorial Lecture
Datalogisk Institut, Københavns Universitet
February 3, 2023

This Is Me...

Jakob Nordström

Professor in Computer Science

University of Copenhagen
and Lund University

www.jakobnordstrom.se

… And This Is What I Do for a Living

$(x_{1,1} \lor x_{1,2} \lor x_{1,3} \lor x_{1,4} \lor x_{1,5} \lor x_{1,6} \lor x_{1,7}) \land (x_{2,1} \lor x_{2,2} \lor x_{2,3} \lor x_{2,4} \lor x_{2,5} \lor x_{2,6} \lor x_{2,7}) \land (x_{3,1} \lor x_{3,2} \lor$
$x_{3,3} \lor x_{3,4} \lor x_{3,5} \lor x_{3,6} \lor x_{3,7}) \land (x_{4,1} \lor x_{4,2} \lor x_{4,3} \lor x_{4,4} \lor x_{4,5} \lor x_{4,6} \lor x_{4,7}) \land (x_{5,1} \lor x_{5,2} \lor x_{5,3} \lor x_{5,4} \lor$
$x_{5,5} \lor x_{5,6} \lor x_{5,7}) \land (x_{6,1} \lor x_{6,2} \lor x_{6,3} \lor x_{6,4} \lor x_{6,5} \lor x_{6,6} \lor x_{6,7}) \land (x_{7,1} \lor x_{7,2} \lor x_{7,3} \lor x_{7,4} \lor x_{7,5} \lor x_{7,6} \lor$
$x_{7,7}) \land (x_{8,1} \lor x_{8,2} \lor x_{8,3} \lor x_{8,4} \lor x_{8,5} \lor x_{8,6} \lor x_{8,7}) \land (\neg x_{1,1} \lor \neg x_{2,1}) \land (\neg x_{1,1} \lor \neg x_{3,1}) \land (\neg x_{1,1} \lor \neg x_{4,1}) \land$
$(\neg x_{1,1} \lor \neg x_{5,1}) \land (\neg x_{1,1} \lor \neg x_{6,1}) \land (\neg x_{1,1} \lor \neg x_{7,1}) \land (\neg x_{1,1} \lor \neg x_{8,1}) \land (\neg x_{2,1} \lor \neg x_{3,1}) \land (\neg x_{2,1} \lor$
$\neg x_{4,1}) \land (\neg x_{2,1} \lor \neg x_{5,1}) \land (\neg x_{2,1} \lor \neg x_{6,1}) \land (\neg x_{2,1} \lor \neg x_{7,1}) \land (\neg x_{2,1} \lor \neg x_{8,1}) \land (\neg x_{3,1} \lor \neg x_{4,1}) \land$
$(\neg x_{3,1} \lor \neg x_{5,1}) \land (\neg x_{3,1} \lor \neg x_{6,1}) \land (\neg x_{3,1} \lor \neg x_{7,1}) \land (\neg x_{3,1} \lor \neg x_{8,1}) \land (\neg x_{4,1} \lor \neg x_{5,1}) \land (\neg x_{4,1} \lor$
$\neg x_{6,1}) \land (\neg x_{4,1} \lor \neg x_{7,1}) \land (\neg x_{4,1} \lor \neg x_{8,1}) \land (\neg x_{5,1} \lor \neg x_{6,1}) \land (\neg x_{5,1} \lor \neg x_{7,1}) \land (\neg x_{5,1} \lor \neg x_{8,1}) \land$
$(\neg x_{6,1} \lor \neg x_{7,1}) \land (\neg x_{6,1} \lor \neg x_{8,1}) \land (\neg x_{7,1} \lor \neg x_{8,1}) \land (\neg x_{1,2} \lor \neg x_{2,2}) \land (\neg x_{1,2} \lor \neg x_{3,2}) \land (\neg x_{1,2} \lor$
$\neg x_{4,2}) \land (\neg x_{1,2} \lor \neg x_{5,2}) \land (\neg x_{1,2} \lor \neg x_{6,2}) \land (\neg x_{1,2} \lor \neg x_{7,2}) \land (\neg x_{1,2} \lor \neg x_{8,2}) \land (\neg x_{2,2} \lor \neg x_{3,2}) \land$
$(\neg x_{2,2} \lor \neg x_{4,2}) \land (\neg x_{2,2} \lor \neg x_{5,2}) \land (\neg x_{2,2} \lor \neg x_{6,2}) \land (\neg x_{2,2} \lor \neg x_{7,2}) \land (\neg x_{2,2} \lor \neg x_{8,2}) \land (\neg x_{3,2} \lor \neg x_{4,2}) \land$
$(\neg x_{3,2} \lor \neg x_{5,2}) \land (\neg x_{3,2} \lor \neg x_{6,2}) \land (\neg x_{3,2} \lor \neg x_{7,2}) \land (\neg x_{3,2} \lor \neg x_{8,2}) \land (\neg x_{4,2} \lor \neg x_{5,2}) \land (\neg x_{4,2} \lor \neg x_{6,2}) \land$
$(\neg x_{4,2} \lor \neg x_{7,2}) \land (\neg x_{4,2} \lor \neg x_{8,2}) \land (\neg x_{5,2} \lor \neg x_{6,2}) \land (\neg x_{5,2} \lor \neg x_{7,2}) \land (\neg x_{5,2} \lor \neg x_{8,2}) \land (\neg x_{6,2} \lor \neg x_{7,2}) \land$
$(\neg x_{6,2} \lor \neg x_{8,2}) \land (\neg x_{7,2} \lor \neg x_{8,2}) \land (\neg x_{1,3} \lor \neg x_{2,3}) \land (\neg x_{1,3} \lor \neg x_{3,3}) \land (\neg x_{1,3} \lor \neg x_{4,3}) \land (\neg x_{1,3} \lor \neg x_{5,3}) \land$
$(\neg x_{1,3} \lor \neg x_{6,3}) \land (\neg x_{1,3} \lor \neg x_{7,3}) \land (\neg x_{1,3} \lor \neg x_{8,3}) \land (\neg x_{2,3} \lor \neg x_{3,3}) \land (\neg x_{2,3} \lor \neg x_{4,3}) \land (\neg x_{2,3} \lor \neg x_{5,3}) \land$
$(\neg x_{2,3} \lor \neg x_{6,3}) \land (\neg x_{2,3} \lor \neg x_{7,3}) \land (\neg x_{2,3} \lor \neg x_{8,3}) \land (\neg x_{3,3} \lor \neg x_{4,3}) \land (\neg x_{3,3} \lor \neg x_{5,3}) \land (\neg x_{3,3} \lor \neg x_{6,3}) \land$
$(\neg x_{3,3} \lor \neg x_{7,3}) \land (\neg x_{3,3} \lor \neg x_{8,3}) \land (\neg x_{4,3} \lor \neg x_{5,3}) \land (\neg x_{4,3} \lor \neg x_{6,3}) \land (\neg x_{4,3} \lor \neg x_{7,3}) \land (\neg x_{4,3} \lor \neg x_{8,3})$

We Live in a World of Computation

Computers are everywhere:

- at work
- at home
- in our cars
- in our pockets

We Live in a World of Computation

Computers are everywhere:

- at work
- at home
- in our cars
- in our pockets

But computation is even more wide-spread:

- protein regulation in cells
- neuron interactions in the brain (and artificial neural networks)
- competition in economic markets
- behaviour of elementary particles in quantum mechanics

We Live in a World of Computation

Computers are everywhere:

- at work
- at home
- in our cars
- in our pockets

But computation is even more wide-spread:

- protein regulation in cells
- neuron interactions in the brain (and artificial neural networks)
- competition in economic markets
- behaviour of elementary particles in quantum mechanics

Understanding computation is a foundational challenge with connections to physics, biology, chemistry, economics, social sciences, philosophy. . .

From Philosophy to Mathematics

Computational problem: any task that can be solved by combination of precisely described steps

Computational complexity theory: Mathematical study of efficient methods (algorithms) and limitations on what automated computation can do

From Philosophy to Mathematics

Computational problem: any task that can be solved by combination of precisely described steps

Computational complexity theory: Mathematical study of efficient methods (algorithms) and limitations on what automated computation can do

Ultimate goal: Understand building blocks of digital world we are living in

From Philosophy to Mathematics

Computational problem: any task that can be solved by combination of precisely described steps

Computational complexity theory: Mathematical study of efficient methods (algorithms) and limitations on what automated computation can do

Ultimate goal: Understand building blocks of digital world we are living in

As foundational as particle physics is for understanding the physical world (but comes at a fraction of the cost)

Combinatorial Solving

Combinatorial problems:

- Find solutions by combining objects
- But objects cannot be subdivided

In technical language, this is a discrete problem

Combinatorial Solving

Combinatorial problems:

- Find solutions by combining objects
- But objects cannot be subdivided

In technical language, this is a discrete problem

Continuous problem: Power grid

To get right power distribution, can fine-tune voltages and currents

Discrete problem: Delivery trucks

To distribute packages between delivery trucks, can't fine-tune load balance by assigning 90% of a package to one truck and 10% to another

Combinatorial Solving

Combinatorial problems:

- Find solutions by combining objects
- But objects cannot be subdivided

In technical language, this is a discrete problem

Continuous problem: Power grid

To get right power distribution, can fine-tune voltages and currents

Discrete problem: Delivery trucks

To distribute packages between delivery trucks, can't fine-tune load balance by assigning 90% of a package to one truck and 10% to another

This difference makes combinatorial problems computationally very challenging

Three Questions About Combinatorial Solving

1. Lack of general-purpose algorithms with performance guarantees because
   - We haven't been smart enough / worked hard enough?
   - Or somehow these problems are inherently hard for computers?

Three Questions About Combinatorial Solving

1. Lack of general-purpose algorithms with performance guarantees because
   - We haven't been smart enough / worked hard enough?
   - Or somehow these problems are inherently hard for computers?
2. For the type of combinatorial problems that can be solved in practice
   - Understand when and why algorithms work well?
   - Leverage more advanced mathematics to get even better performance?

Three Questions About Combinatorial Solving

1. Lack of general-purpose algorithms with performance guarantees because
   - We haven't been smart enough / worked hard enough?
   - Or somehow these problems are inherently hard for computers?

2. For the type of combinatorial problems that can be solved in practice
   - Understand when and why algorithms work well?
   - Leverage more advanced mathematics to get even better performance?

3. For problems with life-or-death consequences, can we guarantee that what the computer outputs is in fact a correct solution?

The Challenge of Combinatorial Solving

- In technical language, many combinatorial problems are NP-complete

The Challenge of Combinatorial Solving

- In technical language, many combinatorial problems are NP-complete
- NP-complete problems are widely believed to require exponential-time algorithms in the worst case

The Challenge of Combinatorial Solving

- In technical language, many combinatorial problems are NP-complete
- NP-complete problems are widely believed to require exponential-time algorithms in the worst case
- But we don't know! This is one of the Millennium Prize Problems posed as major challenges for modern mathematics

The Challenge of Combinatorial Solving

- In technical language, many combinatorial problems are NP-complete
- NP-complete problems are widely believed to require exponential-time algorithms in the worst case
- But we don't know! This is one of the Millennium Prize Problems posed as major challenges for modern mathematics
- Can we at least prove that the most popular algorithmic approaches used today require exponential time?

Combinatorial Problems and Logic

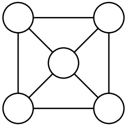
COLOURING

Does the graph $G = (V, E)$ have a colouring with k colours such that all neighbours have distinct colours?

Combinatorial Problems and Logic

Colouring

Does the graph $G = (V, E)$ have a colouring with k colours such that all neighbours have distinct colours?

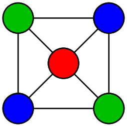


3-colouring?

Combinatorial Problems and Logic

Colouring

Does the graph $G = (V, E)$ have a colouring with k colours such that all neighbours have distinct colours?

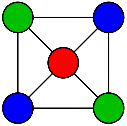


3-colouring? Yes

Combinatorial Problems and Logic

Colouring

Does the graph $G = (V, E)$ have a colouring with k colours such that all neighbours have distinct colours?



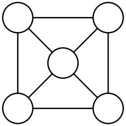
3-colouring? Yes, but no 2-colouring

Combinatorial Problems and Logic

CLIQUE

Is there a clique in the graph $G = (V, E)$ with k vertices that are all pairwise connected by edges in E?

Combinatorial Problems and Logic

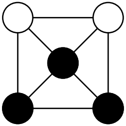


3-clique?

CLIQUE

Is there a clique in the graph $G = (V, E)$ with k vertices that are all pairwise connected by edges in E?

Combinatorial Problems and Logic

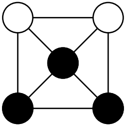


3-clique? Yes

Clique

Is there a clique in the graph $G = (V, E)$ with k vertices that are all pairwise connected by edges in E?

Combinatorial Problems and Logic



3-clique? Yes, but no 4-clique

Clique

Is there a clique in the graph $G = (V, E)$ with k vertices that are all pairwise connected by edges in E?

Combinatorial Problems and Logic

Colouring

Does the graph $G = (V, E)$ have a colouring with k colours such that all neighbours have distinct colours?

Clique

Is there a clique in the graph $G = (V, E)$ with k vertices that are all pairwise connected by edges in E?

Sat

Given propositional logic formula, is there a satisfying assignment?

Combinatorial Problems and Logic

Colouring

Does the graph $G = (V, E)$ have a colouring with k colours such that all neighbours have distinct colours?

Clique

Is there a clique in the graph $G = (V, E)$ with k vertices that are all pairwise connected by edges in E?

Sat

Given propositional logic formula, is there a satisfying assignment?

$$(x \lor z) \land (y \lor \neg z) \land (x \lor \neg y \lor u) \land (\neg y \lor \neg u)$$
$$\land (u \lor v) \land (\neg x \lor \neg v) \land (\neg u \lor w) \land (\neg x \lor \neg u \lor \neg w)$$

Combinatorial Problems and Logic

Colouring

Does the graph $G = (V, E)$ have a colouring with k colours such that all neighbours have distinct colours?

Clique

Is there a clique in the graph $G = (V, E)$ with k vertices that are all pairwise connected by edges in E?

Sat

Given propositional logic formula, is there a satisfying assignment?

$$(x \vee z) \wedge (y \vee \neg z) \wedge (x \vee \neg y \vee u) \wedge (\neg y \vee \neg u)$$
$$\wedge (u \vee v) \wedge (\neg x \vee \neg v) \wedge (\neg u \vee w) \wedge (\neg x \vee \neg u \vee \neg w)$$

- Variables should be set to **false** $(= 0)$ or **true** $(= 1)$

Combinatorial Problems and Logic

COLOURING

Does the graph $G = (V, E)$ have a colouring with k colours such that all neighbours have distinct colours?

CLIQUE

Is there a clique in the graph $G = (V, E)$ with k vertices that are all pairwise connected by edges in E?

SAT

Given propositional logic formula, is there a satisfying assignment?

$$(x \lor z) \land (y \lor \neg z) \land (x \lor \neg y \lor u) \land (\neg y \lor \neg u)$$
$$\land (u \lor v) \land (\neg x \lor \neg v) \land (\neg u \lor w) \land (\neg x \lor \neg u \lor \neg w)$$

- Variables should be set to **false** $(= 0)$ or **true** $(= 1)$
- Constraint $(x \lor \neg y \lor u)$: means x or u should be true or y false

Combinatorial Problems and Logic

COLOURING

Does the graph $G = (V, E)$ have a colouring with k colours such that all neighbours have distinct colours?

CLIQUE

Is there a clique in the graph $G = (V, E)$ with k vertices that are all pairwise connected by edges in E?

SAT

Given propositional logic formula, is there a satisfying assignment?

$$(x \lor z) \land (y \lor \lnot z) \land (x \lor \lnot y \lor u) \land (\lnot y \lor \lnot u)$$
$$\land (u \lor v) \land (\lnot x \lor \lnot v) \land (\lnot u \lor w) \land (\lnot x \lor \lnot u \lor \lnot w)$$

- Variables should be set to **false** $(= 0)$ or **true** $(= 1)$
- Constraint $(x \lor \lnot y \lor u)$: means x or u should be true or y false
- $\land$ means all constraints should hold simultaneously

Combinatorial Problems and Logic

Colouring

Does the graph $G = (V, E)$ have a colouring with k colours such that all neighbours have distinct colours?

Clique

Is there a clique in the graph $G = (V, E)$ with k vertices that are all pairwise connected by edges in E?

Sat

Given propositional logic formula, is there a satisfying assignment?

$$(x \lor z) \land (y \lor \neg z) \land (x \lor \neg y \lor u) \land (\neg y \lor \neg u)$$
$$\land (u \lor v) \land (\neg x \lor \neg v) \land (\neg u \lor w) \land (\neg x \lor \neg u \lor \neg w)$$

- Variables should be set to **false** $(= 0)$ or **true** $(= 1)$
- Constraint $(x \lor \neg y \lor u)$: means x or u should be true or y false
- $\land$ means all constraints should hold simultaneously
- Is there a truth value assignment satisfying all constraints?

Combinatorial Problems and Logic

COLOURING

Does the graph $G = (V, E)$ have a colouring with k colours such that all neighbours have distinct colours?

CLIQUE

Is there a clique in the graph $G = (V, E)$ with k vertices that are all pairwise connected by edges in E?

SAT

Given propositional logic formula, is there a satisfying assignment?

COLOURING:	frequency allocation for mobile base stations
CLIQUE:	bioinformatics, computational chemistry
SAT:	easily models these and many other problems

The Same Problem in Three Different Shapes

$$(x \lor z) \land (y \lor \lnot z) \land (x \lor \lnot y \lor u) \land (\lnot y \lor \lnot u)$$
$$\land\ (u \lor v) \land (\lnot x \lor \lnot v) \land (\lnot u \lor w) \land (\lnot x \lor \lnot u \lor \lnot w)$$

The Same Problem in Three Different Shapes

$$(x \lor z) \land (y \lor \neg z) \land (x \lor \neg y \lor u) \land (\neg y \lor \neg u)$$
$$\land\ (u \lor v) \land (\neg x \lor \neg v) \land (\neg u \lor w) \land (\neg x \lor \neg u \lor \neg w)$$

$$\begin{aligned}
(1-x)(1-z) &= 0 \\
(1-y)z &= 0 \\
(1-x)y(1-u) &= 0 \\
yu &= 0 \\
(1-u)(1-v) &= 0 \\
xv &= 0 \\
u(1-w) &= 0 \\
xuw &= 0
\end{aligned}$$

For **false** $= 0$ and **true** $= 1$, is there a $\{0, 1\}$-valued solution?

The Same Problem in Three Different Shapes

$$(x \lor z) \land (y \lor \neg z) \land (x \lor \neg y \lor u) \land (\neg y \lor \neg u)$$
$$\land\ (u \lor v) \land (\neg x \lor \neg v) \land (\neg u \lor w) \land (\neg x \lor \neg u \lor \neg w)$$

$$\begin{aligned}
1 - x - z + xz &= 0 \\
z - yz &= 0 \\
y - xy - yu + xyu &= 0 \\
yu &= 0 \\
1 - u - v + uv &= 0 \\
xv &= 0 \\
u - uw &= 0 \\
xuw &= 0
\end{aligned}$$

For **false** $= 0$ and **true** $= 1$, is there a $\{0, 1\}$-valued solution?

The Same Problem in Three Different Shapes

$$(x \lor z) \land (y \lor \lnot z) \land (x \lor \lnot y \lor u) \land (\lnot y \lor \lnot u)$$
$$\land\ (u \lor v) \land (\lnot x \lor \lnot v) \land (\lnot u \lor w) \land (\lnot x \lor \lnot u \lor \lnot w)$$

$1 - x - z + xz = 0$	$x + z \geq 1$
$z - yz = 0$	$y + (1 - z) \geq 1$
$y - xy - yu + xyu = 0$	$x + (1 - y) + u \geq 1$
$yu = 0$	$(1 - y) + (1 - u) \geq 1$
$1 - u - v + uv = 0$	$u + v \geq 1$
$xv = 0$	$(1 - x) + (1 - v) \geq 1$
$u - uw = 0$	$(1 - u) + w \geq 1$
$xuw = 0$	$(1 - x) + (1 - u) + (1 - w) \geq 1$

For **false** $= 0$ and **true** $= 1$, is there a $\{0, 1\}$-valued solution?

The Same Problem in Three Different Shapes

$$(x \lor z) \land (y \lor \lnot z) \land (x \lor \lnot y \lor u) \land (\lnot y \lor \lnot u)$$
$$\land\ (u \lor v) \land (\lnot x \lor \lnot v) \land (\lnot u \lor w) \land (\lnot x \lor \lnot u \lor \lnot w)$$

$1 - x - z + xz = 0$	$x + z \geq 1$
$z - yz = 0$	$y - z \geq 0$
$y - xy - yu + xyu = 0$	$x - y + u \geq 0$
$yu = 0$	$-y - u \geq -1$
$1 - u - v + uv = 0$	$u + v \geq 1$
$xv = 0$	$-x - v \geq -1$
$u - uw = 0$	$-u + w \geq 0$
$xuw = 0$	$-x - u - w \geq -2$

For **false** $= 0$ and **true** $= 1$, is there a $\{0, 1\}$-valued solution?

Research on Hardness of Combinatorial Problems

Study methods of reasoning used in different algorithmic approaches

- Resolution (Boolean satisfiability solving)
- Polynomial calculus (algebraic Gröbner basis computations)
- Cutting planes (0-1 integer linear programming)

Research on Hardness of Combinatorial Problems

Study methods of reasoning used in different algorithmic approaches

- Resolution (Boolean satisfiability solving)
- Polynomial calculus (algebraic Gröbner basis computations)
- Cutting planes (0-1 integer linear programming)

Prove exponential lower bounds for such methods

- Consider families of problem instances
- Prove that solving them requires exponential number of steps, even if algorithms combine steps optimally

The Success of Combinatorial Solving in Practice

- Many combinatorial problems are NP-complete and so are widely believed to be exponentially hard in the worst case

The Success of Combinatorial Solving in Practice

- Many combinatorial problems are NP-complete and so are widely believed to be exponentially hard in the worst case
- Revolution last couple of decades in combinatorial solvers for
  - Boolean satisfiability (SAT) solving
  - Constraint programming (CP)
  - Mixed integer linear programming (MIP)

  Solve NP-complete problems (or worse) very efficiently in practice!

The Success of Combinatorial Solving in Practice

- Many combinatorial problems are NP-complete and so are widely believed to be exponentially hard in the worst case
- Revolution last couple of decades in combinatorial solvers for
  - Boolean satisfiability (SAT) solving
  - Constraint programming (CP)
  - Mixed integer linear programming (MIP)

  Solve NP-complete problems (or worse) very efficiently in practice!
- Wide range of applications in, e.g.,
  - logistics
  - airline scheduling
  - computer chip design
  - biology
  - medicine
  - ...

Better Algorithms?

Can we use our mathematical understanding of these methods to

- strengthen the algorithms further?
- combine them in novel ways?

Research on Algorithms for Combinatorial Solving

RoundingSat (`gitlab.com/MIAOresearch/software/roundingsat`)

Solver and optimization engine combining

- Conflict-driven search and learning from SAT solving
- Cutting planes reasoning with 0-1 linear inequalities
- Techniques from SAT-based optimization (MaxSAT solving)
- Linear programming relaxations and cut generation from ILP/MIP

Questioning the Success of Combinatorial Solving

- Many combinatorial problems are NP-complete and so are widely believed to be exponentially hard in the worst case
- Revolution last couple of decades in combinatorial solvers for
  - Boolean satisfiability (SAT) solving
  - Constraint programming (CP)
  - Mixed integer linear programming (MIP)

  Solve NP-complete problems (or worse) very efficiently in practice!

Questioning the Success of Combinatorial Solving

- Many combinatorial problems are NP-complete and so are widely believed to be exponentially hard in the worst case
- Revolution last couple of decades in combinatorial solvers for
  - Boolean satisfiability (SAT) solving
  - Constraint programming (CP)
  - Mixed integer linear programming (MIP)

  Solve NP-complete problems (or worse) very efficiently in practice!

- Except solvers are sometimes wrong... (Even best commercial ones)

What Can Be Done About Solver Bugs?

Software testing

- Hard to get good test coverage for sophisticated solvers
- Limited success in identifying non-trivial defects
- Testing can only detect presence of bugs, not prove absence

What Can Be Done About Solver Bugs?

Software testing

- Hard to get good test coverage for sophisticated solvers
- Limited success in identifying non-trivial defects
- Testing can only detect presence of bugs, not prove absence

Formal verification

- Prove that solver implementation adheres to formal specification
- Provides mathematical guarantees of correctness — very appealing!
- But current techniques cannot scale to state-of-the-art solvers

A Simple but Crucial Change of Perspective

Solution: Design certifying algorithms that

- not only solve problem but also
- provide machine-verifiable proof log certifying that result is correct

A Simple but Crucial Change of Perspective

Solution: Design certifying algorithms that

- not only solve problem but also
- provide machine-verifiable proof log certifying that result is correct

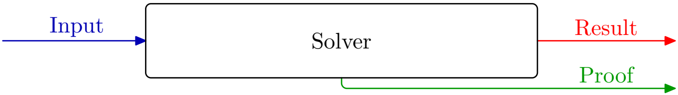
Workflow:

1. Run solver on problem input

A Simple but Crucial Change of Perspective

Solution: Design certifying algorithms that

- not only solve problem but also
- provide machine-verifiable proof log certifying that result is correct



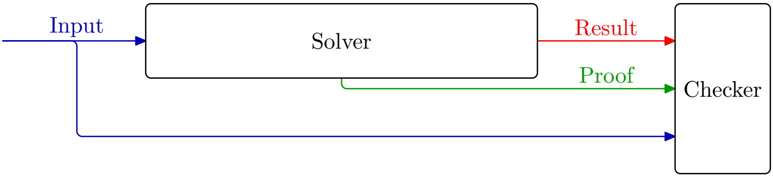
Workflow:

1. Run solver on problem input
2. Get as output not only result but also proof

A Simple but Crucial Change of Perspective

Solution: Design certifying algorithms that

- not only solve problem but also
- provide machine-verifiable proof log certifying that result is correct



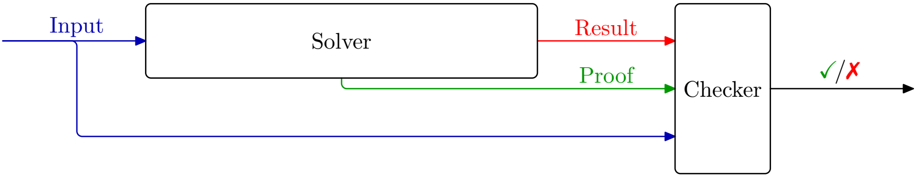
Workflow:

1. Run solver on problem input
2. Get as output not only result but also proof
3. Feed input + result + proof to proof checker

A Simple but Crucial Change of Perspective

Solution: Design certifying algorithms that

- not only solve problem but also
- provide machine-verifiable proof log certifying that result is correct



Workflow:

1. Run solver on problem input
2. Get as output not only result but also proof
3. Feed input + result + proof to proof checker
4. Verify that proof checker says result is correct

Proof Logging Desiderata

Proofs produced by certifying solver should:

- Be powerful enough for proof logging to incur minimal overhead
- Be based on very simple rules
- Not require knowledge of inner workings of solver
- Allow verification by stand-alone proof checker

Proof Logging Desiderata

Proofs produced by certifying solver should:

- Be powerful enough for proof logging to incur minimal overhead
- Be based on very simple rules
- Not require knowledge of inner workings of solver
- Allow verification by stand-alone proof checker

Easier to trust a small, simple checker than a large, complicated solver

- Proof checker should even be simple enough to be formally verified

Does not prove solver correct, but proves solution correct

The Sales Pitch For Proof Logging

1. Certifies correctness of computed results
2. Detects errors even if due to compiler bugs, hardware failures, or cosmic rays
3. Provides debugging support during development
4. Facilitates performance analysis
5. Helps identify potential for further improvements
6. Enables auditability
7. Serves as stepping stone towards explainability

Research on Proof Logging

VeriPB (gitlab.com/MIAOresearch/software/VeriPB)

Versatile proof logging system that in a unified way supports

- Boolean satisfiability (SAT) solving, including advanced techniques
- Graph solving algorithms
- Constraint programming
- Pseudo-Boolean solving
- SAT-based optimization (MaxSAT solving) *[work in progress]*
- 0-1 integer linear programming *[work in progress]*

Summing up

Combinatorial problems

- Show up in wide range of applications
- Appear very challenging in theory
- Can often (but far from always!) be solved efficiently in practice
- But correctness is a huge concern

Summing up

Combinatorial problems

- Show up in wide range of applications
- Appear very challenging in theory
- Can often (but far from always!) be solved efficiently in practice
- But correctness is a huge concern

Research at the intersection of theory and practice can

- Shed light on theoretical power and limitations of applied algorithms
- Identify practically interesting questions for theoretical study
- Lead to new algorithmic ideas to try out in practice
- Provide techniques to produce iron-clad guarantees of correctness

Summing up

Combinatorial problems

- Show up in wide range of applications
- Appear very challenging in theory
- Can often (but far from always!) be solved efficiently in practice
- But correctness is a huge concern

Research at the intersection of theory and practice can

- Shed light on theoretical power and limitations of applied algorithms
- Identify practically interesting questions for theoretical study
- Lead to new algorithmic ideas to try out in practice
- Provide techniques to produce iron-clad guarantees of correctness

Thanks for listening!