FourierSAT: A Fourier Expansion-Based Algebraic Framework for Solving Hybrid Boolean Constraints

Background: SATisfiability Problem

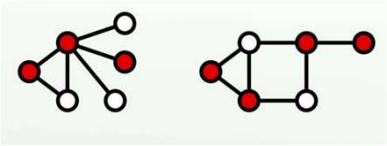
- Variables: $x, y, z... \in \{1, -1\}$
- Connectives: $\neg(Not), \wedge(and), \vee(or), \oplus(xor)...$
- Literals: $\neg x, y, \neg z...$
- Formula: $(x \wedge \neg y) \oplus z$ ...
- Model: an assignment with -1/1s of variables s.t. formula yields -1

- SAT: Does a formula have a model or not?

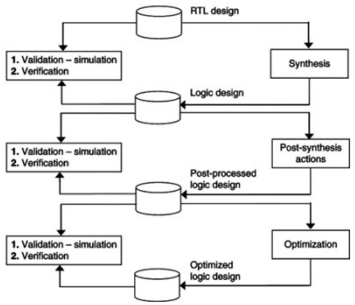
- NP-completeness of SAT was proven in 1971 by Stephen Cook

Applications of SAT

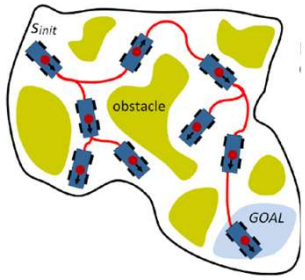
Used by hardware and software designers on a daily basis



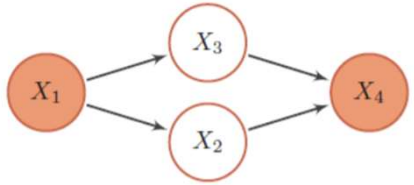
Discrete Optimization
[Ignatiev et al., 2017]



Software Verification
[Velev, 2004]



Motion planning
[Bera, 2017]



Probabilistic inference
[Chavira et al., 2008]

SAT solvers solve industrial SAT instances with millions of variables
[Katebi et al., 2011]

CNF and Hybrid SAT Solving

- Conjunctive Normal Form (CNF)
  - Connectives: $\neg, \wedge, \vee$
  - Clauses: $x_1 \vee x_2 \vee \neg x_3 \ ...$
  - Formulas: $(x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee x_4) \ ...$
  - 3-CNF is NP-complete [Cook, 1971]

- Non-CNF Clauses/Constraints
  - cryptography: XOR [Bogdanov et al., 2011] $x \oplus y \oplus z$
  - graph theory: cardinality constraints [Costa et al., 2009] $x + y + z \geq 2$
    not-all-equal (NAE) [Tomas J, 1978] $NAE(x, y, z) = \neg(x = y = z)$

Contents

- Background and Preliminaries

  Hybrid Boolean satisfiability problem

- Method and Techniques

  Fourier expansions of Boolean functions

  Gradient-based constrained continuous optimization

- Algorithm, Implementation and Empirical Evaluation

  Benchmarks: vertex covering, parity learning, hybrid random formulas

Related Work: Hybrid SAT Solving

- CNF-encoding of Non-CNF constraints

  Involves huge number of new variables and clauses [Wynn, 2018]

  Encodings differ from size, arc-consistency and density of solutions [Prestwich 2009]

- Extensions of CNF solvers

  Cryptominisat (CNF + XOR) [Soos et al., 2009]
  Minicard (CNF + cardinality constraints) [Liffition et al., 2012]
  MonoSAT (CNF + graph properties) [Bayless et al. 2015]
  Pueblo (CNF + pseudo Boolean constraints) [Sheini et al., 2006]

  Need to design algorithms for each specific type of constraints

Contribution: A versatile Boolean SAT Solver

$$f = c_1 \wedge c_2 \wedge \cdots \wedge c_m$$

Each c_i can be a CNF, XOR, Not-all-equal constraint or cardinality constraint

FourierSAT: An incomplete SAT solver based on gradient method to handle different types of constraints uniformly

Theoretical Tool: Fourier Expansion of Boolean Function

Boolean formulas → Multilinear Polynomials

$f : \{1, -1\}^n \to \{1, -1\}$

{F, T}

$p : \{1, -1\}^n \to \{1, -1\}$

x	y	$x \wedge y$	$\frac{1}{2} + \frac{1}{2}x + \frac{1}{2}y - \frac{1}{2}xy$
1	1	1	1
1	-1	1	1
-1	1	1	1
-1	-1	-1	-1

Theoretical Tool: Fourier Expansion of Boolean Function

Boolean formulas → Multilinear Polynomials

Theorem

Given a Boolean function $f : \{-1,1\}^n \to \mathbb{R}$, there is a unique way of expressing f as a multilinear polynomial:

$$f(x) = \sum_{S \subseteq [n]} \left(\widehat{f}(S) \cdot \prod_{i \in S} x_i \right)$$

[O'Donnell, 2014]

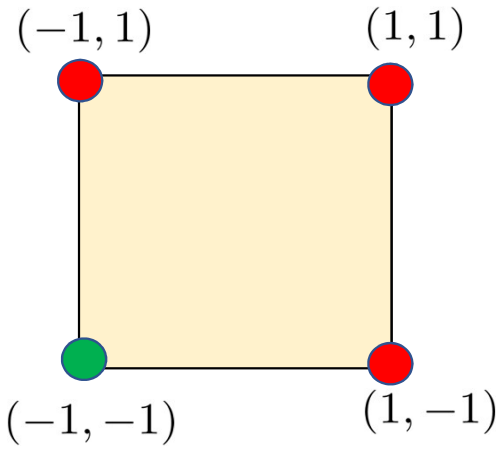
Fourier coefficients on S

multilinear monomials

$$x \wedge y \longrightarrow \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot x + \frac{1}{2} \cdot y - \frac{1}{2} \cdot xy$$

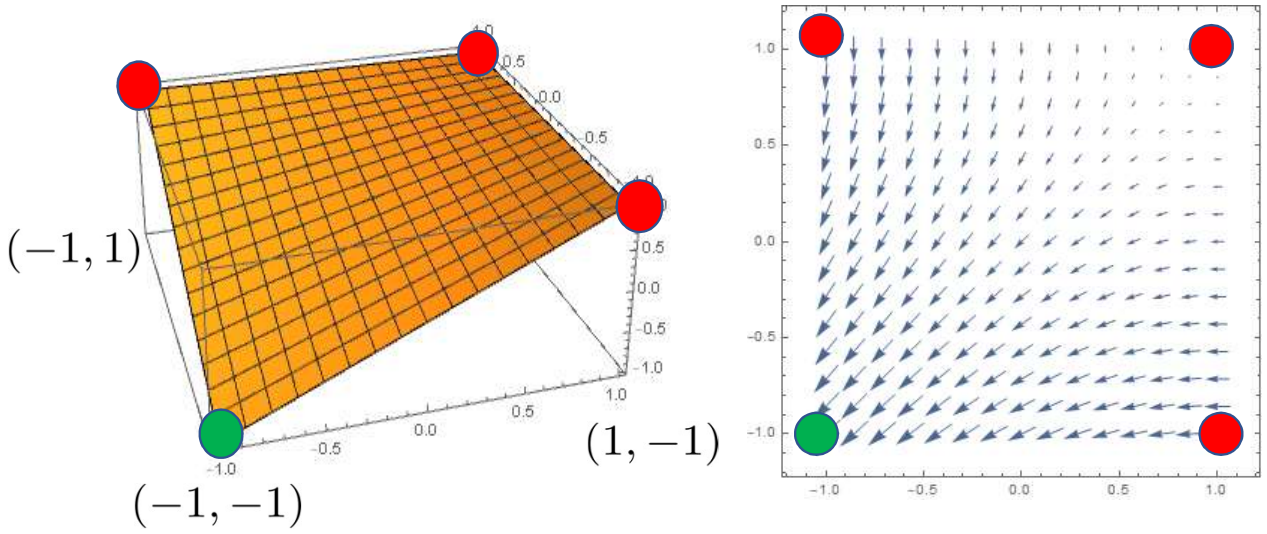
Observation: Polynomials Can Be Evaluated on Real Values

$x \wedge y$



Discrete searching on Boolean formulas

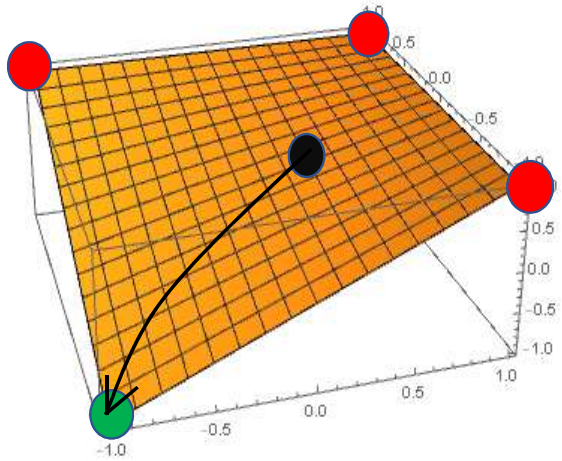
$\frac{1}{2} + \frac{1}{2}x + \frac{1}{2}y - \frac{1}{2}xy$



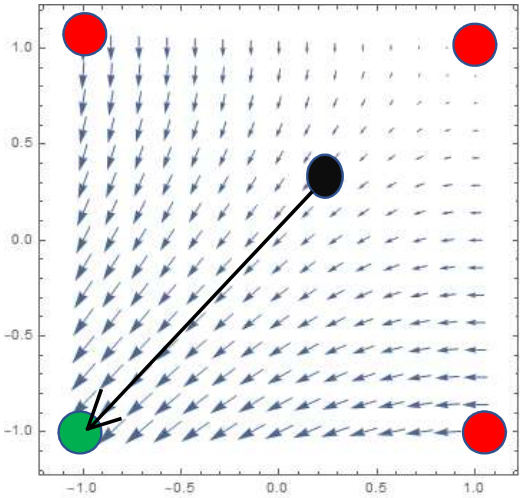
Continuous optimization on multilinear polynomials

Approach: From SAT to Continuous Optimization

Discrete searching on Boolean formulas



Continuous optimization on multilinear polynomials



Problems:

- Fourier Expansion of a Boolean function with n variables has up to 2^n terms
- There exists Boolean function s.t. computing its Fourier Expansion is #P-hard

Factored Representation

$$f = c_1 \wedge c_2 \wedge \cdots \wedge c_m$$

- Many symmetric constraints has closed form Fourier expansion

Type of Constraint	Example	Fourier Expansion
CNF clauses	$x \wedge y$	$\frac{1}{2} + \frac{1}{2}x + \frac{1}{2}y - \frac{1}{2}xy$
XOR	$x \oplus y \oplus z$	$x \cdot y \cdot z$
Cardinality constraints	$x + y + z \geq 2$	$\frac{1}{2}x + \frac{1}{2}y + \frac{1}{2}z - \frac{1}{2}xyz$
Not-all-equal	$NAE(x, y, z)$	$-\frac{1}{2} + \frac{1}{2}xy + \frac{1}{2}yz + \frac{1}{2}xz$

Compute the Fourier Expansion of each clause

- Efficient Objective Function Evaluation
  Objective function can be evaluated in $O(k^2 \cdot m)$ time, where k is the maximum width of all clauses

Objective Function Construction

$$f = c_1 \wedge c_2 \wedge \cdots \wedge c_m$$

$$F = p_1 + p_2 + \cdots + p_m$$

$$F = \sum_{i=1}^{m} p_i$$

Fourier expansions

$$f = (x \vee \neg y) \wedge (x \oplus y)$$

$$F = (-\tfrac{1}{2} + \tfrac{1}{2}x - \tfrac{1}{2}y - \tfrac{1}{2}x \cdot y) + (x \cdot y) = -\tfrac{1}{2} + \tfrac{1}{2}x - \tfrac{1}{2}y + \tfrac{1}{2}x \cdot y$$

Why plus?

- Easy to compute. Avoid maintaining multilinearity in multiplication.
- The function value indicates the number of satisfied clauses.

Multilinear Polynomials: Well-behaved

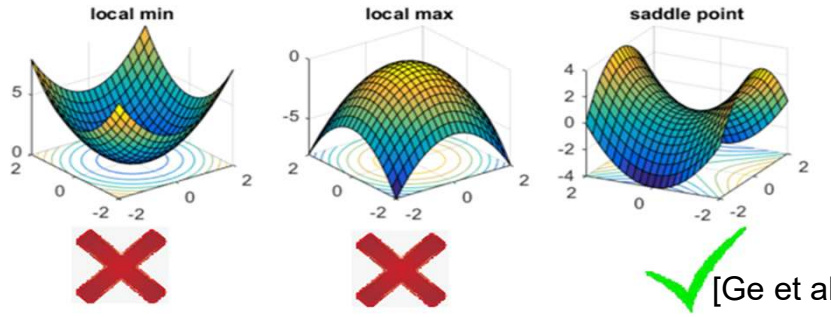
$$F = \sum_{i=1}^{m} p_i$$

$$f = c_1 \wedge c_2 \wedge \cdots \wedge c_m$$

$$F = p_1 + p_2 + \cdots + p_m$$

Bounded: $p_i \in [-1, 1]$ and $F \in [-m, m]$ for every $x \in [-1, 1]^n$

Theorem (Saddle Landscape)
At every $x \in \mathbb{R}^n$, $\exists$ a decreasing direction v and $\epsilon > 0$, $F(x + \epsilon v) < F(x)$



[Ge et al., 2016]

Reduction: from SAT to continuous optimization

$$F = \sum_{i=1}^{m} p_i$$

$$f = c_1 \wedge c_2 \wedge \cdots \wedge c_m$$

$$F = p_1 + p_2 + \cdots + p_m$$

Bounded: $p_i \in [-1, 1]$ and $F \in [-m, m]$ for every $x \in [-1, 1]^n$

Theorem (Saddle Landscape)
At every $x \in \mathbb{R}^n$, $\exists$ a decreasing direction v and $\epsilon > 0$, $F(x + \epsilon v) < F(x)$

Theorem

$$f \text{ is SAT} \Leftrightarrow \min_{x \in [-1,1]^n} (F) = -m$$

Why Not Just Do Discrete Local Search?

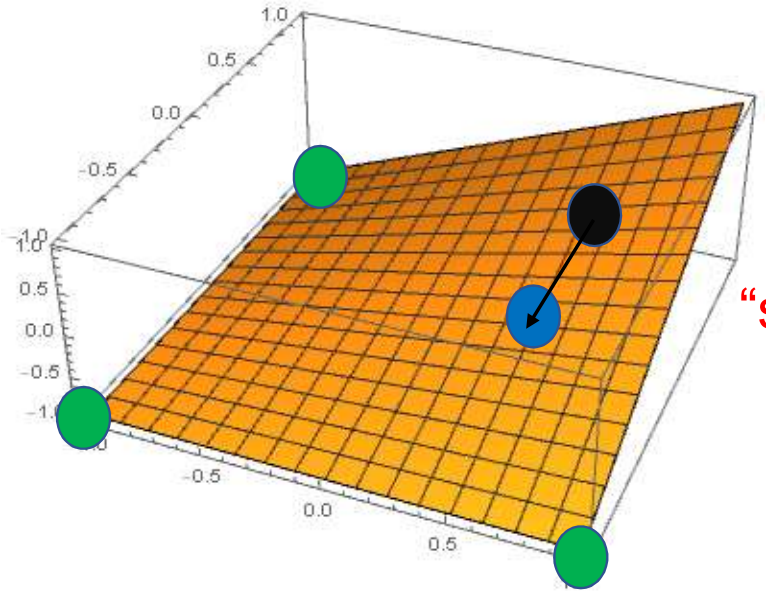
Greedy Local Search Solver (GSAT) [Selman et al., 1992]: per iteration, flip a bit to maximize the number of satisfied clauses

$$x_1 + x_2 + \cdots + x_{100} \geq 70, x_i \in \{0, 1\}$$

$$\text{start from } |x| = 50 \longrightarrow \text{Need to flip at least 20 bits}$$

GSAT will not make progress for a long time!

FourierSAT Always “Makes Progress”



“small step”

Randomized Rounding:

$$P[R(x)_i = -1] = \frac{1}{2} - \frac{1}{2}x_i$$

$$P[R(x)_i = 1] = \frac{1}{2} + \frac{1}{2}x_i$$

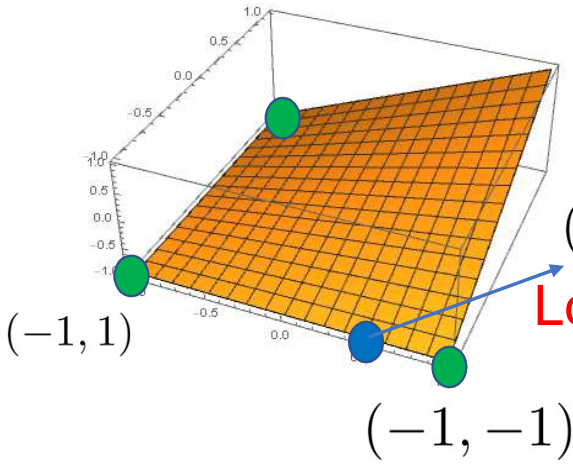
Theorem

$F(x) = -k \Rightarrow \frac{m+k}{2}$ clauses can be satisfied in expectation

Making “small” progress in expectation per iteration

Alternative Application: MaxSAT

$$F = -\tfrac{1}{2} + \tfrac{1}{2}x + \tfrac{1}{2}y + \tfrac{1}{2}xy \;\; (x \vee y)$$



$\mathrm{F}(-1,0.5)=\mathrm{F}(-1,1)=\mathrm{F}(-1,-1)$

Theorem (Deterministically)

$F(x) = -k$

x is a local minimum $\Rightarrow \frac{m+k}{2}$ clauses can be satisfied

Projected Gradient Descent Converges Fast

Multilinear polynomials are smooth: For $F \in [-\alpha, \alpha]$ and $x, y \in [-1, 1]^n$

Lipschitz continuous:

$$|F(x) - F(y)| \leq \alpha\sqrt{n} \cdot \|x - y\|_2$$

Lipschitz gradient:

$$\|\nabla F(x) - \nabla F(y)\|_2 \leq \alpha n \cdot \|x - y\|_2$$

Theorem

For a formula with n variables and m clauses, Projected Gradient Descent converges to a ϵ-projected-critical point in $O(\frac{nm^2}{\epsilon^2})$

Extra algorithms are designed for escaping saddle points

Implementation Techniques

- Weighted version: $F = \sum_{i=1}^{m} p_i \longrightarrow F = \sum_{i=1}^{m} w_i \cdot p_i$

- Analytical Gradient Computation

  Feeding gradient to optimizer significantly accelerated the algorithm

- Random Restart and Parallelization

  Different trials are independent so that parallelization can be applied

Algorithm

Algorithm 1: Projected Gradient Descent for multilinear F

```
for j = 1, ..., J do
    x_0 ~ U[-1,1]^n //choose the initial point randomly
    for t = 0, ... do
        G(x_t) = projected gradient at x_t
        if ||G(x_t)||_2 > 0 then
            x_{t+1} = x_t - η · G(x_t) // Gradient descent
        else
            if x_t is local minimum then
                Break
            else
                Use extra algorithm to escape the saddle point
    Until convergence
return x_j with the lowest F(x) after J iterators
```

Experiments Setting and Benchmarks

Experiments Setting

Rice NOTS Linux cluster:
Hardware: Xeon E5-2650v2 CPUs (2.60-GHz)
Memory limit: 1 GB
Time limit: 60 seconds
Number of CPUs: 24
Optimization Core: SLSQP in Scipy
Weighting function: widths of clauses

Encodings

Cardinality encoding: Sequential Counter [Sinz, 2005], Sorting Network [Batcher, 1968], Totalizer [Bailleux and Boufkhad, 2003], Adder [Een and Sorensson, 2006]
XOR encoding: Linear encoding to 3CNF [Li, 2000]

Solvers for comparison

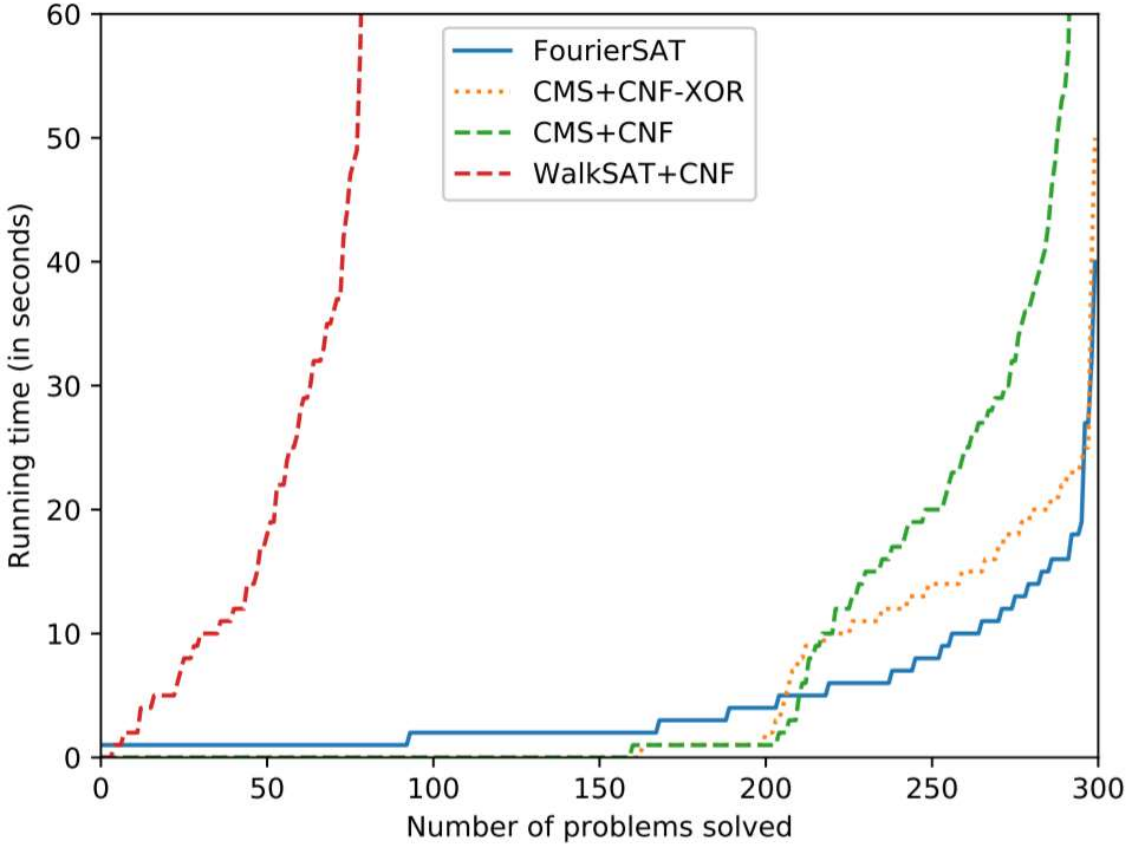
CryptominiSAT (CMS)[Soos et al., 2009],
WalkSAT (Local search SAT solver) [Selman, 1999]
MiniCARD (CNF + cardinality constraints) [Liffiton and Maglalang 2012]
MonoSAT (CNF + graph properties) [Bayless et al. 2015]

Benchmark 1: Parity Learning with Errors

- Solving XOR systems but tolerating up to ¼ of constraints to be violated

- Known as hard instances for both DPLL and local search SAT solvers

$n = 32$

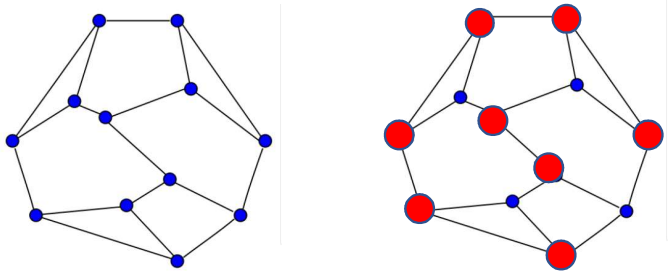


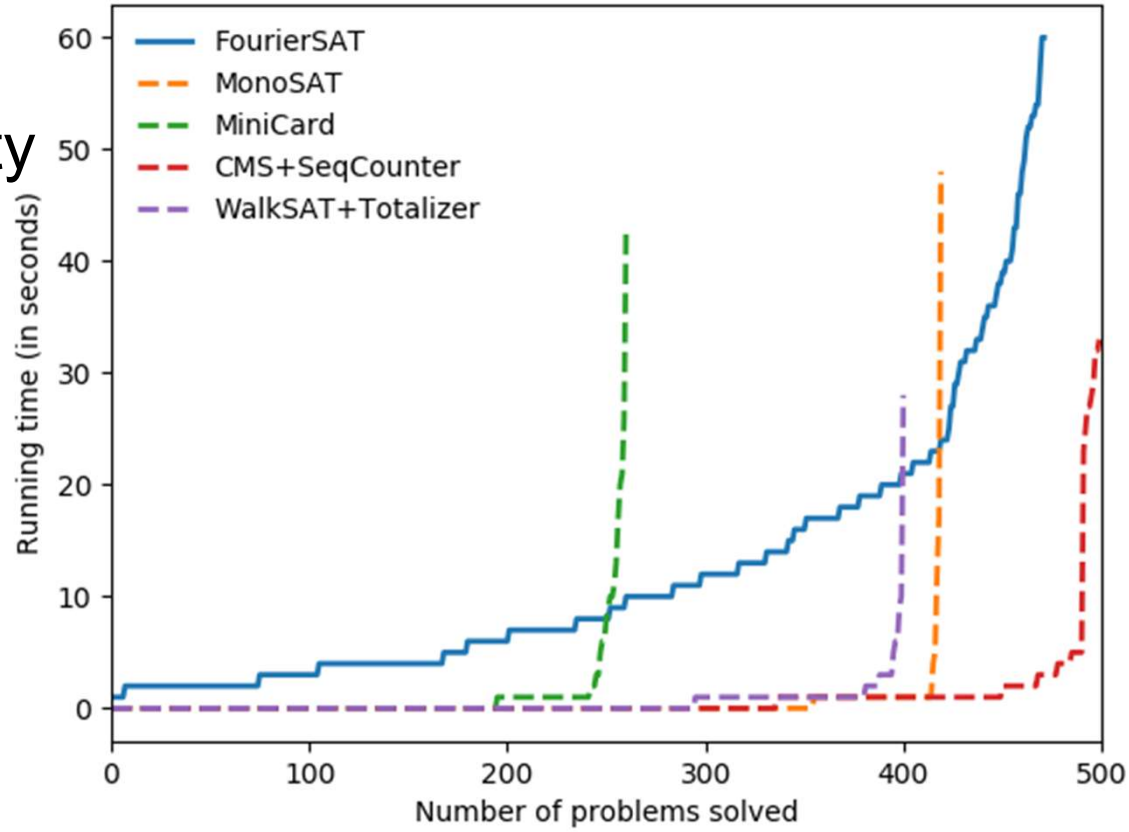
Benchmark 2: Approximate Minimum Vertex Covering

- Finding a vertex set Sol of random cubic graphs s.t. $|Sol| \leq 1.1 \cdot |Opt|$

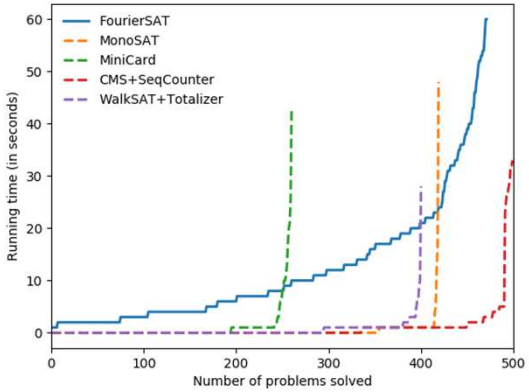
- CNFs + 1 global cardinality Constraint

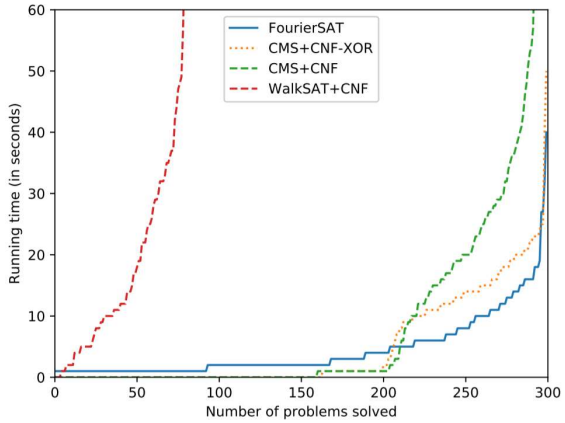
$$50 \leq n \leq 250$$





Experimental Results: FourierSAT Shows Great Potential





- CryptominiSAT with appropriate encodings solved most problems
- FourierSAT is comparable with state-of-the-art solvers even without lots of engineering

Conclusion

- SAT solving beyond CNF is worth studying
- Our work, FourierSAT is a versatile and robust tool for Boolean SAT
- Applications of Fourier analysis and other algebraic techniques for Boolean logic is promising
  - Bridging discrete and continuous optimization
- Future directions:
  - Finding better weighting heuristics
  - Proving unsatisfiability algebraically
  - Deploying FourierSAT with methods from machine learning and local search SAT solvers