SAT Solver and its Application to Combinatorial Problems

Naoyuki Tamura

Kobe University

実験計画法およびその周辺の組合せ構造 2014
December 14th, 2014

Contents

http://bach.istc.kobe-u.ac.jp/papers/pdf/kinosaki2014.pdf

SAT Problems and SAT Solvers

SAT Problems

SAT (Boolean satisfiability testing) is a problem to decide whether a given Boolean formula has any satisfying truth assignment.

- SAT is a central problem in Computer Science both theoretically and practically.
- SAT is the first NP-complete problem [Cook 1971].
- SAT has very efficient implementation (MiniSat, etc.).
- SAT-based approach is becoming popular in many areas.
  - Intel core I7 processor design [Kaivola+, CAV 2009]
  - Windows 7 device drivers verification with Z3 [De Moura and Bjorner, IJCAR 2010]
  - Software component dependency analysis in Eclipse [Le Berre and Rapicault, IWOCE 2009]

SAT-based Systems

- Planning (SATPLAN, Blackbox) [Kautz & Selman 1992]
- Automatic Test Pattern Generation [Larrabee 1992]
- Job-shop Scheduling [Crawford & Baker 1994]
- Software Specification (Alloy) (1998)
- Bounded Model Checking [Biere 1999]
- Software Package Dependency Analysis (SATURN)
- Rewriting Systems (AProVE, Jambox)
- Answer Set Programming (clasp, Cmodels-2)
- FOL Theorem Prover (iProver, Darwin, Paradox)
- First Order Model Finder (Paradox)
- Constraint Satisfaction Problems (Sugar) (Tamura+ 2006)

SAT Instances

SAT instances are given in the conjunctive normal form (CNF).

CNF Formula

- A CNF formula is a conjunction of clauses.
- A clause is a disjunction of literals.
- A literal is either a Boolean variable or its negation.

DIMACS CNF is used as the standard format for CNF files.

p cnf 3 4	; Number of variables and clauses
1 2 3 0	; $p_1 \vee p_2 \vee p_3$
-1 -2 0	; $\neg p_1 \vee \neg p_2$
-1 -3 0	; $\neg p_1 \vee \neg p_3$
-2 -3 0	; $\neg p_2 \vee \neg p_3$

SAT Solvers

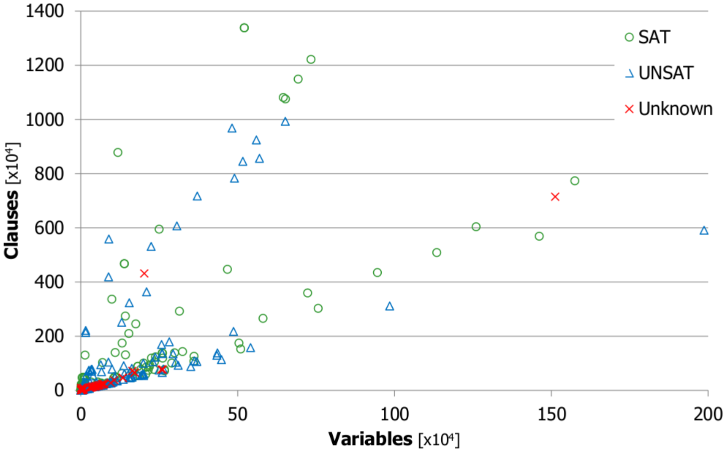
- SAT solver is a program to decide whether a given SAT instance is satisfiable (`SAT`) or unsatisfiable (`UNSAT`).
- Usually, it also returns a truth assignment as a solution when the instance is `SAT`.

- Systematic (complete) SAT solver answers `SAT` or `UNSAT`.
  - Most of them are based on the DPLL algorithm.
- Stochastic (incomplete) SAT solver only answers `SAT` (no answers for `UNSAT`).
  - Local search algorithms are used.

Modern SAT Solvers

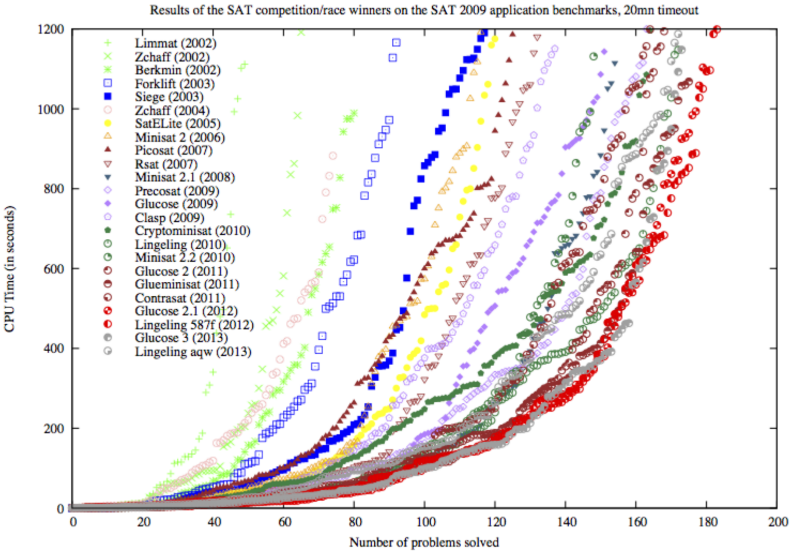
- The following techniques have been introduced to DPLL and they drastically improved the performance of modern SAT solvers.
  - **CDCL** (Conflict Driven Clause Learning) [Silva 1996]
  - Non-chronological Backtracking [Silva 1996]
  - Random Restarts [Gomes 1998]
  - Watched Literals [Moskewicz & Zhang 2001]
  - Variable Selection Heuristics [Moskewicz & Zhang 2001]
- Chaff and zChaff solvers made one to two orders magnitude improvement (2001).
- SAT competitions and SAT races since 2002 contribute to the progress of SAT solver implementation techniques.
- MiniSat solver showed its good performance in the 2005 SAT competition with about 2000 lines of code in C++.
- Modern SAT solvers can handle instances with more than 10^6 variables and 10^7 clauses.

Size of SAT Instances



2011 SAT competition, Applications Track

Performance Progress of SAT Solvers



Famous SAT Solvers

- MiniSat [Eén and Sörensson 2003]
  - http://minisat.se
- Clasp [Gebser+ 2007]
  - http://www.cs.uni-potsdam.de/clasp/
- Glucose [Audemard and Simon 2009]
  - http://www.labri.fr/perso/lsimon/glucose/
- Lingeling [Biere 2010]
  - http://fmv.jku.at/lingeling/
- GlueMiniSat [Nabeshima+ 2011]
  - https://sites.google.com/a/nabelab.org/glueminisat/
- Sat4j [Le Berre 2010]
  - http://www.sat4j.org

Don Knuth's TAOCP

SAT will be a topic of the next fascicle 6A, Volume 4 (Combinatorial Algorithms) of TAOCP (The Art Of Computer Programming) by Don Knuth.

THE ART OF
COMPUTER PROGRAMMING
VOLUME 4 PRE-FASCICLE 6A

A (VERY INCOMPLETE)
DRAFT OF SECTION 7.2.2.2:
SATISFIABILITY

DONALD E. KNUTH *Stanford University*

ADDISON–WESLEY

- Current draft is already 246 pages long!
  - http://www-cs-faculty.stanford.edu/~knuth/fasc6a.ps.gz

Don Knuth's TAOCP

He made an invited talk at SAT 2012 conference, and demonstrated his own SAT solvers!

- Donald E. Knuth: Satisfiability and the Art of Computer Programming, SAT 2012 invited talk, 2012.
  - http://www-cs-faculty.stanford.edu/~uno/sat2012.pdf

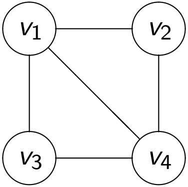
Application to Combinatorial Problems

- Graph Coloring Problem
- Covering Array
- Ramsey Number
- Erdös's Discrepancy Conjecture

GCP (Graph Coloring Problem)

GCP as a decision problem

Find a vertex coloring with c colors of a given graph such that no two adjacent vertices have the same color.



GCP can be formalized as a Constraint Satisfaction Problem (CSP) on integers.

$$\begin{array}{ll} v \in \{1, 2, \ldots, c\} & (v \in V) \\ u \neq v & (\{u, v\} \in E) \end{array}$$

Encoding CSP to SAT

Finite domain (arithmetic) CSP

- Variables
  - **Integer variables** with finite domains
    - $\ell(x)$: the lower bound of x
    - $u(x)$: the upper bound of x
  - **Boolean variables**
- Constraints
  - **Arithmetic operators**: $+$, $-$, $\times$, etc.
  - **Comparison operators**: $=$, $\neq$, $\geq$, $>$, $\leq$, $<$
  - **Logical operators**: $\neg$, $\wedge$, $\vee$, $\Rightarrow$

- How to encode CSP variables to SAT?
- How to encode CSP constraints to SAT?

SAT encodings

There have been several methods proposed to encode CSP.

- **Direct encoding** is the most widely used one [de Kleer 1989].
- **Order encoding** shows a good performance for a wide variety of problems [Tamura+ 2006].
  - It is first used to encode job-shop scheduling problems by [Crawford & Baker 1994].
  - It succeeded to solve previously undecided problems in open-shop scheduling, job-shop scheduling, two-dimensional strip packing, etc.
- Other encodings:
  - *Multivalued encoding* [Selman+ 1992]
  - *Support encoding* [Kasif 1990]
  - *Log encoding* [Iwama+ 1994]
  - *Log-support encoding* [Gavanelli 2007]
  - *Compact order encoding* [Tanjo+ 2010]

Direct encoding

In direct encoding [de Kleer 1989], a Boolean variable $p(x = i)$ is defined as true iff the integer variable x has the domain value i, that is, $x = i$.

Boolean variables for each integer variable x

$$p(x = i) \qquad (\ell(x) \le i \le u(x))$$

For example, the following five Boolean variables are used to encode an integer variable $x \in \{2, 3, 4, 5, 6\}$,

5 Boolean variables for $x \in \{2, 3, 4, 5, 6\}$

$$p(x = 2) \quad p(x = 3) \quad p(x = 4) \quad p(x = 5) \quad p(x = 6)$$

Direct encoding (cont.)

At-least-one and at-most-one clauses are required to make $p(x = i)$ be true iff $x = i$.

Clauses for each integer variable x

$$p(x = \ell(x)) \lor \cdots \lor p(x = u(x))$$

$$\neg p(x = i) \lor \neg p(x = j) \qquad (\ell(x) \le i < j \le u(x))$$

For example, 11 clauses are required for $x \in \{2, 3, 4, 5, 6\}$.

11 clauses for $x \in \{2, 3, 4, 5, 6\}$

$$p(x = 2) \lor p(x = 3) \lor p(x = 4) \lor p(x = 5) \lor p(x = 6)$$

$$\neg p(x = 2) \lor \neg p(x = 3) \qquad \neg p(x = 2) \lor \neg p(x = 4) \qquad \neg p(x = 2) \lor \neg p(x = 5)$$

$$\neg p(x = 2) \lor \neg p(x = 6) \qquad \neg p(x = 3) \lor \neg p(x = 4) \qquad \neg p(x = 3) \lor \neg p(x = 5)$$

$$\neg p(x = 3) \lor \neg p(x = 6) \qquad \neg p(x = 4) \lor \neg p(x = 5)$$

$$\neg p(x = 4) \lor \neg p(x = 6) \qquad \neg p(x = 5) \lor \neg p(x = 6)$$

Direct encoding (cont.)

A constraint is encoded by enumerating its conflict points.

Constraint clauses

When $x_1 = i_1, \ldots, x_k = i_k$ violates the constraint, the following clause is added.

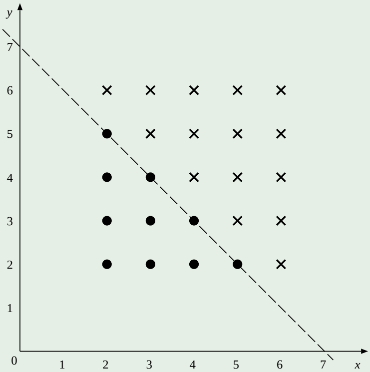
$$\neg p(x_1 = i_1) \lor \cdots \lor \neg p(x_k = i_k)$$

Direct encoding (cont.)

A constraint $x + y \leq 7$ is encoded into the following 15 clauses by enumerating conflict points (crossed points).

Direct encoding of $x + y \leq 7$ when $x, y \in \{2..6\}$

$$\begin{array}{l}
\neg p(x=2) \vee \neg p(y=6) \\
\neg p(x=3) \vee \neg p(y=5) \\
\neg p(x=3) \vee \neg p(y=6) \\
\neg p(x=4) \vee \neg p(y=4) \\
\neg p(x=4) \vee \neg p(y=5) \\
\neg p(x=4) \vee \neg p(y=6) \\
\neg p(x=5) \vee \neg p(y=3) \\
\neg p(x=5) \vee \neg p(y=4) \\
\neg p(x=5) \vee \neg p(y=5) \\
\neg p(x=5) \vee \neg p(y=6) \\
\neg p(x=6) \vee \neg p(y=2) \\
\neg p(x=6) \vee \neg p(y=3) \\
\neg p(x=6) \vee \neg p(y=4) \\
\neg p(x=6) \vee \neg p(y=5) \\
\neg p(x=6) \vee \neg p(y=6)
\end{array}$$



Order encoding

In order encoding [Tamura+ 2006], a Boolean variable $p(x \le i)$ is defined as true iff the integer variable x is less than or equal to the domain value i, that is, $x \le i$.

Boolean variables for each integer variable x

$$p(x \le i) \qquad (\ell(x) \le i < u(x))$$

For example, the following four Boolean variables are used to encode an integer variable $x \in \{2, 3, 4, 5, 6\}$,

4 Boolean variables for $x \in \{2, 3, 4, 5, 6\}$

$$p(x \le 2) \quad p(x \le 3) \quad p(x \le 4) \quad p(x \le 5)$$

- Boolean variable $p(x \le 6)$ is unnecessary since $x \le 6$ is always true.

Order encoding (cont.)

The following clauses are required to make $p(x \le i)$ be true iff $x \le i$.

Clauses for each integer variable x

$$\neg p(x \le i-1) \lor p(x \le i) \quad (\ell(x) < i < u(x))$$

For example, 3 clauses are required for $x \in \{2, 3, 4, 5, 6\}$.

3 clauses for $x \in \{2, 3, 4, 5, 6\}$

$$\neg p(x \le 2) \lor p(x \le 3)$$
$$\neg p(x \le 3) \lor p(x \le 4)$$
$$\neg p(x \le 4) \lor p(x \le 5)$$

Order encoding (cont.)

The following table shows possible satisfiable assignments for the given clauses.

$$\neg p(x \leq 2) \vee p(x \leq 3)$$
$$\neg p(x \leq 3) \vee p(x \leq 4)$$
$$\neg p(x \leq 4) \vee p(x \leq 5)$$

Satisifiable assignments

$p(x \leq 2)$	$p(x \leq 3)$	$p(x \leq 4)$	$p(x \leq 5)$	Interpretation
1	1	1	1	$x = 2$
0	1	1	1	$x = 3$
0	0	1	1	$x = 4$
0	0	0	1	$x = 5$
0	0	0	0	$x = 6$

Order encoding (cont.)

Satisfiable partial assignments

$p(x \le 2)$	$p(x \le 3)$	$p(x \le 4)$	$p(x \le 5)$	Interpretation
–	–	–	–	$x = 2..6$
–	–	–	1	$x = 2..5$
–	–	1	1	$x = 2..4$
–	1	1	1	$x = 2..3$
0	–	–	–	$x = 3..6$
0	0	–	–	$x = 4..6$
0	0	0	–	$x = 5..6$
0	–	–	1	$x = 3..5$
0	–	1	1	$x = 3..4$
0	0	–	1	$x = 4..5$

"–" means undefined.

- Partial assignments on Boolean variables represent bounds of integer variables.

Order encoding (cont.)

A constraint is encoded by enumerating its conflict regions instead of conflict points.

Constraint clauses

- When all points $(x_1, \ldots, x_k)$ in the region $i_1 < x_1 \leq j_1, \ldots, i_k < x_k \leq j_k$ violate the constraint, the following clause is added.

$$p(x_1 \leq i_1) \vee \neg p(x_1 \leq j_1) \vee \cdots \vee p(x_k \leq i_k) \vee \neg p(x_k \leq j_k)$$

Order encoding (cont.)

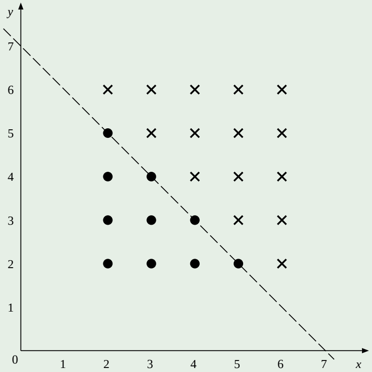
Linear inequality $\sum_{i=1}^{n} a_i x_i \le c$ can be recursively encoded with the following relation.

$$\sum_{i=1}^{n} a_i x_i \le c \iff$$

$$\begin{cases} (x_1 \le \lfloor c/a_1 \rfloor) & (n = 1, a_1 > 0) \\ \neg(x_1 \le \lceil c/a_1 \rceil + 1) & (n = 1, a_1 < 0) \\ \displaystyle\bigwedge_{d \in Dom(x_1)} \Big((x_1 \le d - 1) \vee \sum_{i=2}^{n} a_i x_i \le c - a_1 d \Big) & (n \ge 2, a_1 > 0) \\ \displaystyle\bigwedge_{d \in Dom(x_1)} \Big(\neg(x_1 \le d) \vee \sum_{i=2}^{n} a_i x_i \le c - a_1 d \Big) & (n \ge 2, a_1 < 0) \end{cases}$$

Order encoding (cont.)

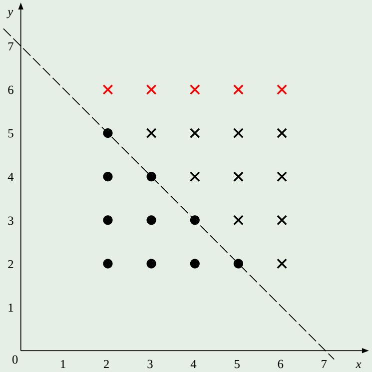
Order encoding of $x + y \le 7$ when $x, y \in \{2..6\}$



Order encoding (cont.)

Order encoding of $x + y \leq 7$ when $x, y \in \{2..6\}$

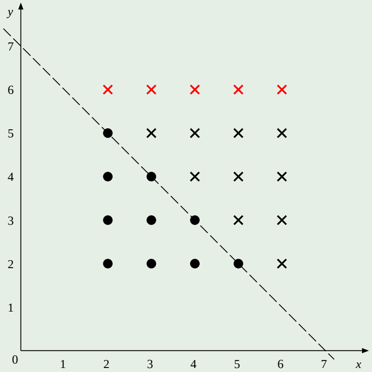
$\neg(y \geq 6)$



Order encoding (cont.)

Order encoding of $x + y \leq 7$ when $x, y \in \{2..6\}$

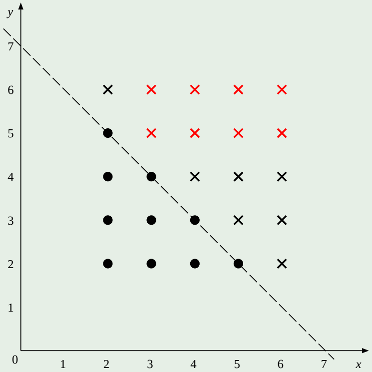
$p(y \leq 5)$



Order encoding (cont.)

Order encoding of $x + y \leq 7$ when $x, y \in \{2..6\}$

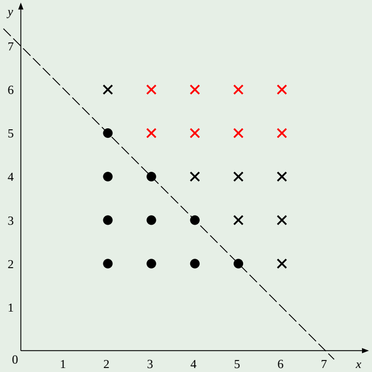
$p(y \leq 5)$
$\neg(x \geq 3 \wedge y \geq 5)$



Order encoding (cont.)

Order encoding of $x + y \leq 7$ when $x, y \in \{2..6\}$

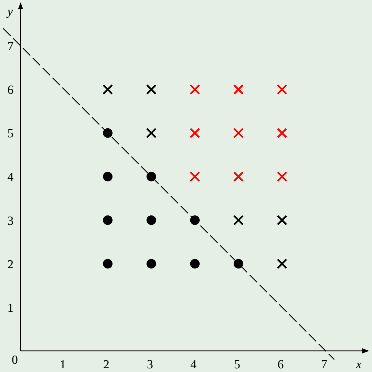
$p(y \leq 5)$
$p(x \leq 2) \vee p(y \leq 4)$



Order encoding (cont.)

Order encoding of $x + y \leq 7$ when $x, y \in \{2..6\}$

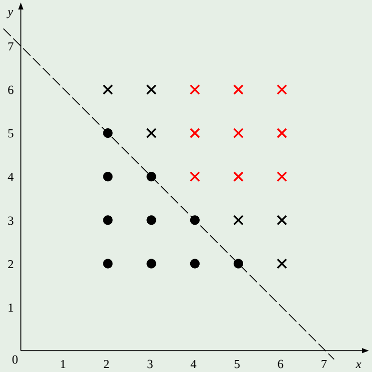
$p(y \leq 5)$
$p(x \leq 2) \vee p(y \leq 4)$
$\neg(x \geq 4 \wedge y \geq 4)$



Order encoding (cont.)

Order encoding of $x + y \leq 7$ when $x, y \in \{2..6\}$

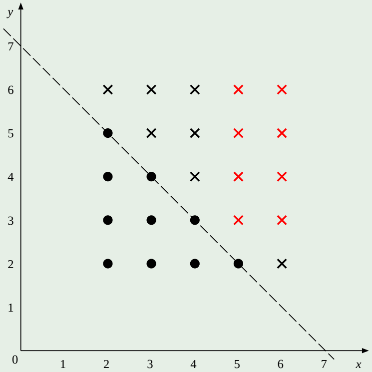
$p(y \leq 5)$
$p(x \leq 2) \vee p(y \leq 4)$
$p(x \leq 3) \vee p(y \leq 3)$



Order encoding (cont.)

Order encoding of $x + y \leq 7$ when $x, y \in \{2..6\}$

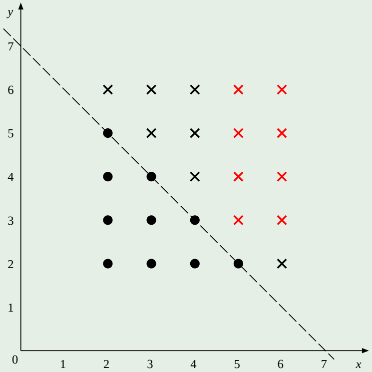
$p(y \leq 5)$
$p(x \leq 2) \lor p(y \leq 4)$
$p(x \leq 3) \lor p(y \leq 3)$
$\neg(x \geq 5 \land y \geq 3)$



Order encoding (cont.)

Order encoding of $x + y \le 7$ when $x, y \in \{2..6\}$

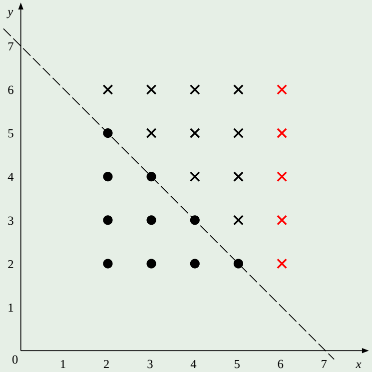
$p(y \le 5)$
$p(x \le 2) \lor p(y \le 4)$
$p(x \le 3) \lor p(y \le 3)$
$p(x \le 4) \lor p(y \le 2)$



Order encoding (cont.)

Order encoding of $x + y \le 7$ when $x, y \in \{2..6\}$

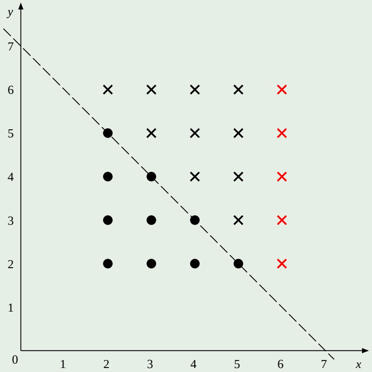
$p(y \le 5)$
$p(x \le 2) \lor p(y \le 4)$
$p(x \le 3) \lor p(y \le 3)$
$p(x \le 4) \lor p(y \le 2)$
$\neg(x \ge 6)$



Order encoding (cont.)

Order encoding of $x + y \le 7$ when $x, y \in \{2..6\}$

$$
\begin{array}{l}
p(y \le 5) \\
p(x \le 2) \lor p(y \le 4) \\
p(x \le 3) \lor p(y \le 3) \\
p(x \le 4) \lor p(y \le 2) \\
p(x \le 5)
\end{array}
$$



Order encoding (cont.)

Order encoding of $x + y \le 7$ when $x, y \in \{2..6\}$

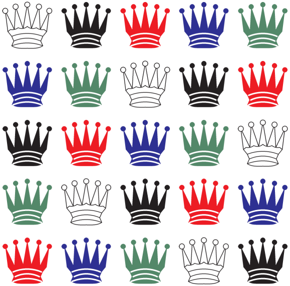
$$
\begin{array}{rc}
C_1: & p(y \le 5) \\
C_2: & p(x \le 2) \lor p(y \le 4) \\
C_3: & p(x \le 3) \lor p(y \le 3) \\
C_4: & p(x \le 4) \lor p(y \le 2) \\
C_5: & p(x \le 5)
\end{array}
$$

- Suppose $p(x \le 3)$ becomes false (i.e. $x \ge 4$), then $p(y \le 3)$ becomes true (i.e. $y \le 3$) by unit propagation on C_3.
- This corresponds to the bound propagation in CSP solvers.
- It is shown that the order encoding is the only one translating some tractable CSP to tractable SAT [Petke+ 2011].

Queens Graph Coloring

The graph is given by a $N \times N$ chess board where any two cells in the same row, column, or diagonal are considered to be adjacent.

- This problem is used in Knuth's TAOCP.
- N colors are sufficient when $N \equiv \pm 1 \pmod{6}$.



Queens Graph Coloring

N	Colors	Encoding	#Vars	#Clauses	#Mems
7	7 (SAT)	Direct	343	4417	416,570
7	7 (SAT)	Order	294	3589	1,339,689
8	8 (UNSAT)	Direct	512	7688	9,534,216,524
8	8 (UNSAT)	Order	448	6222	8,784,182,550

- Memory access count is measured by Knuth's "sat13" solver.

Queens Graph Coloring

N	Colors	Encoding	#Vars	#Clauses	#Mems
7	7 (SAT)	Direct	343	4417	416,570
7	7 (SAT)	Order	294	3589	1,339,689
8	8 (UNSAT)	Direct	512	7688	9,534,216,524
8	8 (UNSAT)	Order	448	6222	8,784,182,550

- Memory access count is measured by Knuth's "sat13" solver.

For each k-clique $\{v_1, v_2, \ldots, v_k\}$ $(1 \leq v_i \leq c)$, we can add extra two clauses to accelerate the solving speed:

$$\bigvee_i v_i \geq k \qquad \bigvee_i v_i \leq c + 1 - k$$

Queens Graph Coloring

N	Colors	Encoding	#Vars	#Clauses	#Mems
7	7 (SAT)	Direct	343	4417	416,570
7	7 (SAT)	Order	294	3589	1,339,689
8	8 (UNSAT)	Direct	512	7688	9,534,216,524
8	8 (UNSAT)	Order	448	6222	8,784,182,550

- Memory access count is measured by Knuth's "sat13" solver.

For each k-clique $\{v_1, v_2, \ldots, v_k\}$ $(1 \le v_i \le c)$, we can add extra two clauses to accelerate the solving speed:

$$\bigvee_i v_i \ge k \qquad \bigvee_i v_i \le c + 1 - k$$

N	Colors	Encoding	#Vars	#Clauses	#Mems
7	7 (SAT)	Order+	294	3661	243,678
8	8 (UNSAT)	Order+	448	6306	30,470,236

Covering Array

$CAN(t, k, g)$

New results	Previously known results
$\mathbf{20} \le CAN(2,7,4) \le 21$	$19 \le CAN(2,7,4) \le 21$
$\mathbf{80} \le CAN(3,8,4) \le 88$	$76 \le CAN(3,8,4) \le 88$
$CAN(3,12,2) = \mathbf{15}$	$14 \le CAN(3,12,2) \le 15$
$\mathbf{15} \le CAN(3,k,2)\ (k \ge 13)$	$14 \le CAN(3,k,2)\ (k \ge 13)$
$\mathbf{50} \le CAN(5,9,2) \le 54$	$48 \le CAN(5,9,2) \le 54$
$CAN(6,8,2) = \mathbf{85}$	$84 \le CAN(6,8,2) \le 85$

- Combination of the order encoding and Hnich's encoding is used.
- Lower-bounds are updated for six instances and the optimum size are decided for two instances [Banbara+, LPAR 2010].

Ramsey Number

Ramsey number $R(s,t)$ is the minimum number n such that any blue-red edge coloring of K_n contains either blue K_s or red K_t.

SAT Encoding for a graph of n vertices $V = \{1, 2, \ldots, n\}$

- Boolean variables: e_{ij} (for $1 \leq i < j \leq n$)
- Clauses:

$$\bigvee_{i,j \in U,\ i<j} \neg e_{ij} \quad (\text{for } U \subset V,\ |U| = s)$$

$$\bigvee_{i,j \in U,\ i<j} e_{ij} \quad (\text{for } U \subset V,\ |U| = t)$$

Fujita showed $R(4,8) \geq 58$ with his own SAT solver named SCSat [Fujita+, SAT 2013].

Erdös's Discrepancy Conjecture

Paul Erdös conjectured that for any positive integer C in any infinite ± 1 sequence (x_n), there exists a subsequence x_d, x_{2d}, x_{3d}, $\ldots$, x_{kd} for some positive integers k and d, such that $\left|\sum_{i=1}^{k} x_{id}\right| > C$.

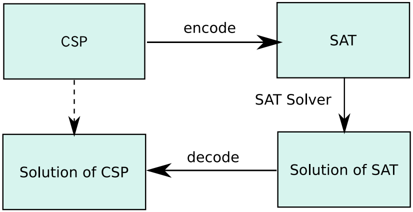
Kovev and Lisista showed in their SAT 2014 paper:

- a sequence of length 1160 satisfying $\left|\sum_{i=1}^{k} x_{id}\right| \leq 2$ for any k and d, and
- a proof of discrepancy conjecture for $C = 2$, claiming that no discrepancy 2 sequence of length 1161 (or more) exists.

They used SAT solvers to find a sequence of 1160, and also a proof (13G bytes) for length 1161.

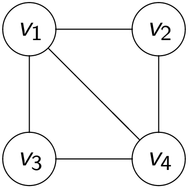
Sugar: a SAT-based Constraint Solver

Sugar: a SAT-based Constraint Solver



- **Sugar** is a SAT-based constraint solver using the **order encoding**.
- It won in global constraint categories of 2008 and 2009 international CSP solver competitions.

Example of Sugar CSP



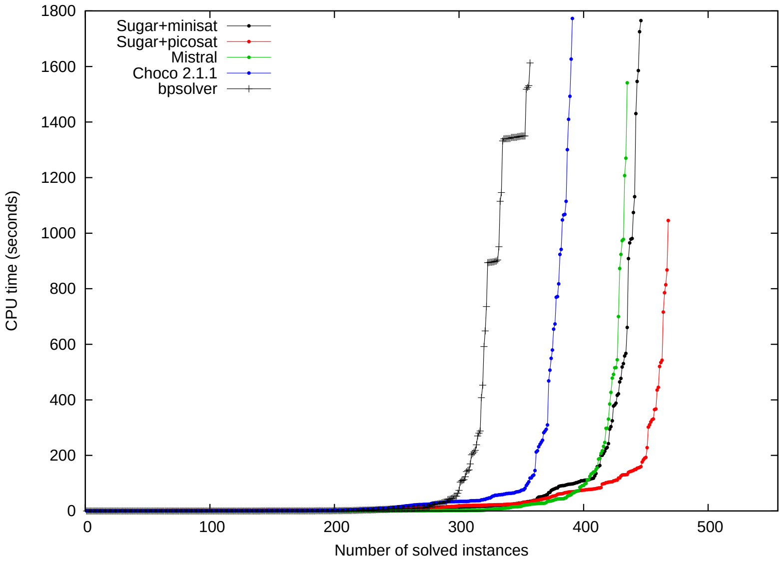

```
(int v1 1 3)
(int v2 1 3)
(int v3 1 3)
(int v4 1 3)
(!= v1 v2)
(!= v1 v3)
(!= v1 v4)
(!= v2 v4)
(!= v3 v4)
```

CSC 2009 (Global Categories)

Series		Sugar+m	Sugar+p	Mistral	Choco	bpsolver
BIBD	(83)	76	77	76	58	35
Costas Array	(11)	8	8	9	9	9
Latin Square	(10)	10	9	5	5	5
Magic Square	(18)	8	8	13	15	11
NengFa	(3)	3	3	3	3	3
Orthogonal Latin Square	(9)	3	3	3	2	3
Perfect Square Packing	(74)	54	53	40	47	36
Pigeons	(19)	19	19	19	19	19
Quasigroup Existence	(35)	30	29	29	28	30
Pseudo-Boolean	(100)	68	75	59	53	70
BQWH	(20)	20	20	20	20	20
Cumulative Job-Shop	(10)	4	4	2	1	0
RCPSP	(78)	78	78	78	77	75
Cabinet	(40)	40	40	40	40	40
Timetabling	(46)	25	42	39	14	1
Total	(556)	446	468	435	391	357

- The number of solved instances in global categories
- Sugar+m : Sugar with MiniSat 2.0 backend
- Sugar+p : Sugar with PicoSAT 535 backend

CSC 2009 (Global Categories)



Summary

1. SAT Problems and SAT Solvers
    - SAT Problems
    - SAT Solvers
    - Don Knuth's TAOCP
2. Applications to Combinatorial Problems
    - Graph Coloring Problem
    - Covering Array
    - Ramsey Number
    - Erdös's Discrepancy Conjecture
3. Sugar: a SAT-based Constraint Solver

References

- Handbook of Satisfiability, IOS Press, 2009.
- **特集「最近の** SAT **技術の発展」**, **人工知能学会誌**, **第** 25 **巻** 1 **号**, 2010.
  - SAT **ソルバーの基礎，高速** SAT **ソルバーの原理，制約最適化問題と** SAT **符号化，**SMT**：個別理論を取り扱う** SAT **技術，モデル列挙とモデル計数，**∗-SAT: SAT **の拡張，**SAT **によるプランニングとスケジューリング，**SAT **によるシステム検証**
- **「私のブックマーク**: SAT **ソルバー」**, **人工知能学会誌**, **第** 85 **巻** 2 **号**, 2013.
  - http://www.ai-gakkai.or.jp/my-bookmark_vol28-no2/
- SAT/SMT Summer School 2014
  - http://satsmt2014.forsyte.at
- Sugar
  - http://bach.istc.kobe-u.ac.jp/sugar/