Complexity theory - when and how does it guide the practitioner

Justyna Petke

When does complexity theory guide the practitioner ?

- During the solving process
- During the modelling process

How does complexity theory guide the practitioner ?

- *'Tractable benchmarks for constraint programming' (Petke & Jeavons, 2009)*
- *'Local consistency & SAT-solvers' (Jeavons & Petke, 2010)*
- *'The order encoding: from tractable CSP to tractable SAT' (Petke & Jeavons, 2011)*

Constraint Satisfaction Problem (CSP)
is a triple (**V**, **D**, **C**)
V : a finite set of variables
D : a finite set of values
C : a finite set of constraints

Boolean Satisfiability problem (SAT)
is a triple (**V**, **D**, **C**)
V : a finite set of variables
D : {0,1}
C : a finite set of propositional clauses

How to find a solution to a CSP instance ?

Given a constraint problem, you can:

- Solve it with a constraint solver
- Encode it as a set of clauses and use a SAT solver

CP solvers

- Gecode
- G12
- Minion
- FznTini (SAT-based constraint solver)
- Many others ..

Q: When does complexity theory guide the practitioner ?

A: When we need to find a solution to a **tractable** instance.

Tractable CSP classes

A class of CSP instances will be called

tractable

if there exists an algorithm

which finds a solution to all instances in that class, or reports that there are no solutions, **whose time complexity is polynomial** in the size of the instance specification.

Types of tractable CSP classes

- Language

  e.g. all constraints are of the form $x_i \geq c$,

  where x_i is a variable and c is a constant

- Structural

  e.g. the constraint graph is a tree

Q: How does complexity theory guide the practitioner ?

A: It provides efficient algorithms for known tractable classes of problems.

'Tractable benchmarks for constraint programming' (Petke & Jeavons, 2009)

Max-closed constraints

$$a_1X_1 + a_2X_2 + \ldots + a_{r-1}X_{r-1} \geq a_rX_r + c$$

- Efficient solving algorithm is based on achieving **arc consistency**.

Max-closed constraints

- We predict that CP solvers will do better than SAT-based solvers

Max-closed constraints

Number of inequalities (k)	Satis-fiable?	Number of variables (n)	Number of values (m)	Gecode Time (secs.)	G12 Time (secs.)	FznTini Time (secs.)	Minion Time (secs.)
10	yes	10	100	0.01	0.32	9.71	0.01
10	yes	10	200	0.01	0.26	3.49	0.01
10	yes	100	10	0.02	0.41	> 15 min	0.05
100	yes	10	100	0.02	0.41	> 15 min	0.06
100	no	10	100	0.02	0.41	> 15 min	0.03
1000	yes	20	100	0.08	1.34	error	0.38
1000	no	20	100	0.03	0.59	error	0.13
1000	yes	30	200	0.08	1.59	error	0.52
1000	no	30	200	0.03	0.57	error	0.13
1000	yes	200	10	0.46	6.99	error	2.88
1000	no	200	10	0.03	0.59	error	0.14

'Tractable benchmarks for constraint programming' (Petke & Jeavons, 2009)

Bounded-width constraints

8 variables : $\{ x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 \}$

2 values : { 0, 1 }

3 constraints :

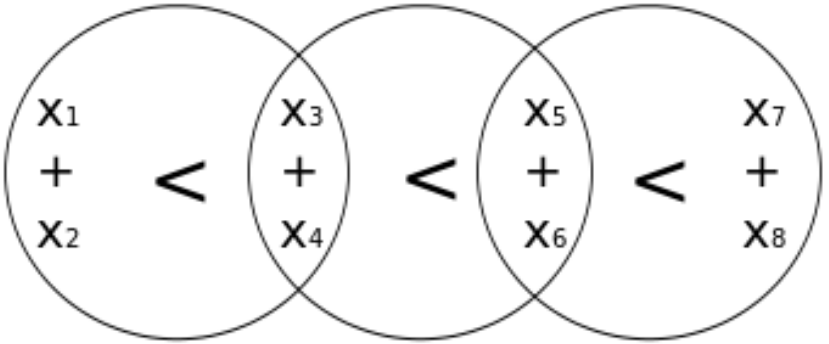
$$x_1 + x_2 > x_3 + x_4$$

$$x_3 + x_4 > x_5 + x_6$$

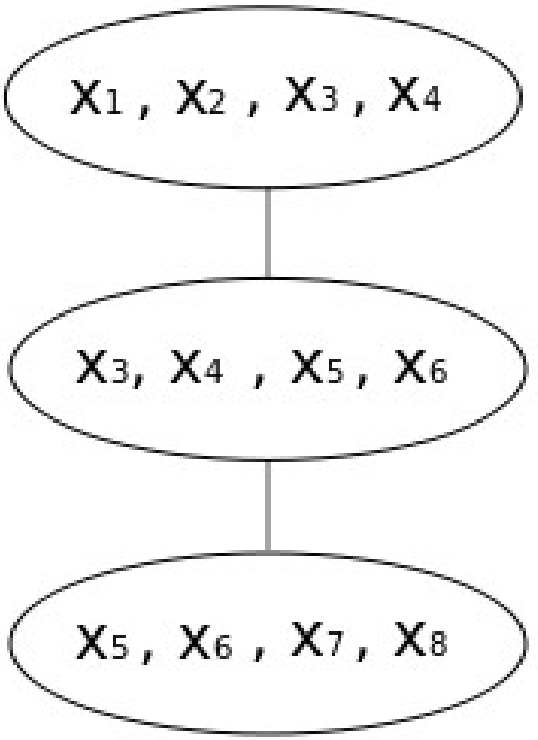
$$x_5 + x_6 > x_7 + x_8$$

Bounded-width constraints

- CSP structure



- Tree decomposition



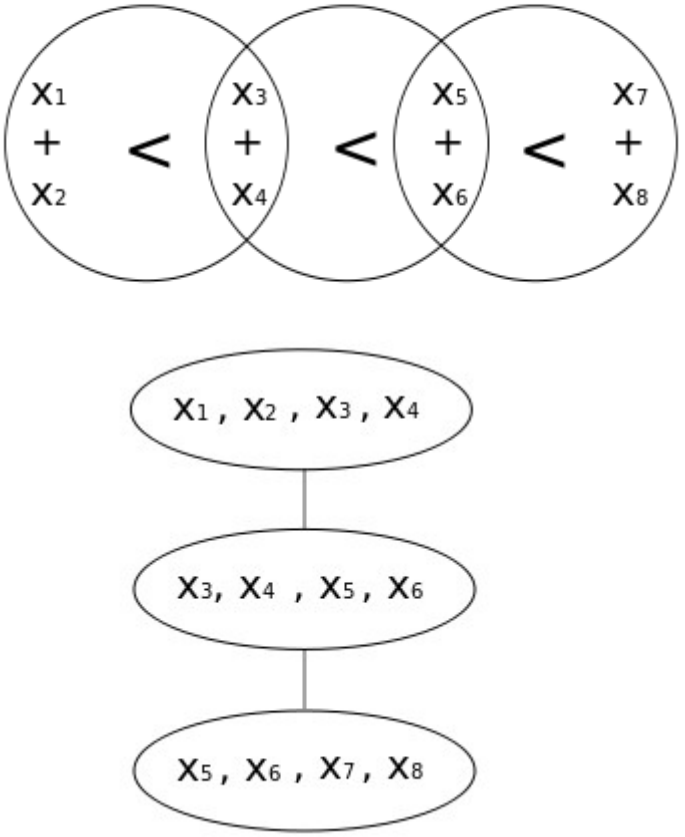
Bounded-width constraints

- the **width** of a tree decomposition =

  = the maximum number of vertices in any node of the tree

- Efficient solving algorithm is based on choosing an appropriate **variable ordering** and imposing **a level of consistency** proportional to the **width**.

Chain of inequalities example



- 2 parameters:
  - d : domain size,
  - w : group size
- unsatisfiable (d=w=2 shown)
- tree-width : $2w - 1$

Bounded-width constraints

Group size (w)	Maximum value (m)	Number of variables $w(wm+1)$	Gecode Time (secs.)	G12 Time (secs.)	FznTini Time (secs.)	Minion Time (secs.)
1	100	101	0.02	0.41	0.26	0.02
2	5	22	0.57	0.22	0.46	0.10
2	6	26	25.07	0.25	0.36	21.22
2	7	30	$>$ 15 min	0.25	0.11	$>$ 15 min
2	12	50	$>$ 15 min	0.40	2.91	$>$ 15 min
3	2	21	0.01	0.23	0.01	1.45
3	3	30	706.59	0.55	0.07	740.55
3	4	39	$>$ 15 min	103.19	0.23	$>$ 15 min
3	5	48	$>$ 15 min	$>$ 15 min	2.31	$>$ 15 min

'Tractable benchmarks for constraint programming'
(Petke & Jeavons, 2009)

Bounded-width constraints

Group size (w)	Maximum value (m)	Number of variables $w(wm+1)$	Gecode Time (secs.)	G12 Time (secs.)	FznTini Time (secs.)	Minion Time (secs.)
1	100	101	0.02	0.37	0.24	0.02
2	5	22	0.01	2.98	0.05	0.06
2	6	26	0.01	166.26	0.03	0.02
2	7	30	0.01	$>$ 15 min	0.12	0.05
2	12	50	0.02	$>$ 15 min	1.66	0.02
3	2	21	0.03	0.49	0.02	0.02
3	3	30	0.07	$>$ 15 min	0.07	0.08
3	4	39	8.51	$>$ 15 min	0.11	10.55
3	5	48	$>$ 15 min	$>$ 15 min	0.15	$>$ 15 min

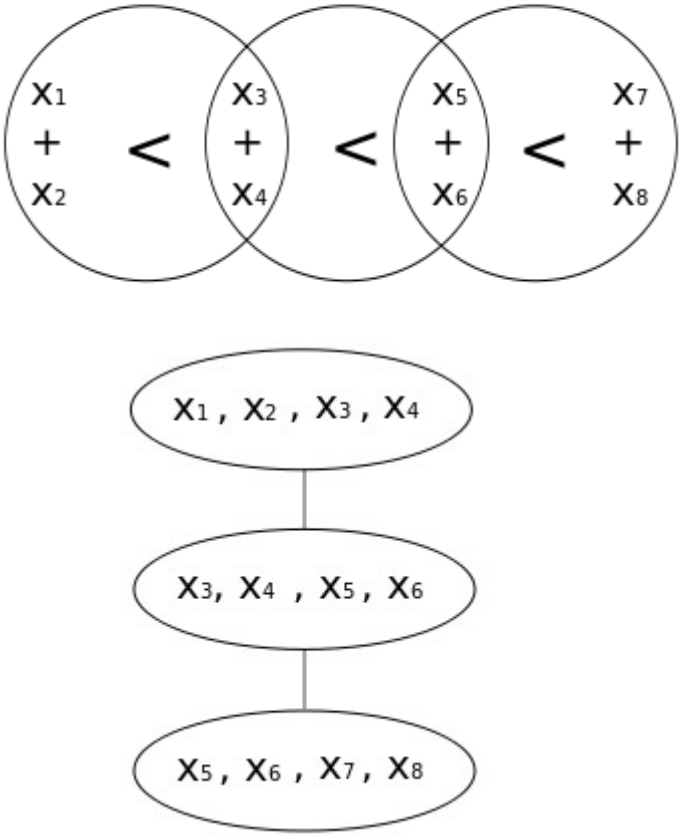
- Results with inequalities reversed.
  'Tractable benchmarks for constraint programming'
  (Petke & Jeavons, 2009)

Q: How does complexity theory guide the practitioner ?

A: By explaining which part of an algorithm is central to solving a problem efficiently.

'Local consistency and SAT-solvers' (Jeavons & Petke, 2010)

Chain of inequalities example



- 2 parameters:
  - d : domain size,
  - w : group size
- unsatisfiable (d=w=2 shown)
- tree-width : $2w - 1$

Direct encoding (tree-width: *2wd-1*)

X1=0 \/ X1=1 , X2=0 \/ X2=1 , X3=0 \/ X3=1 , X4=0 \/ X4=1 , X5=0 \/ X5=1 , X6=0 \/ X6=1 , X7=0 \/ X7=1 , X8=0 \/ X8=1

-X1=0 \/ -X1=1 , -X2=0 \/ -X2=1 , -X3=0 \/ -X3=1 , -X4=0 \/ -X4=1 , -X5=0 \/ -X5=1 , -X6=0 \/ -X6=1 , -X7=0 \/ -X7=1 , -X8=0 \/ -X8=1

-X1=0 \/ -X2=1 \/ -X3=0 \/ -X4=0 , -X3=0 \/ -X4=1 \/ -X5=0 \/ -X6=0 , -X5=0 \/ -X6=1 \/ -X7=0 \/ -X8=0 ,

-X1=0 \/ -X2=1 \/ -X3=0 \/ -X4=1 , -X3=0 \/ -X4=1 \/ -X5=0 \/ -X6=1 , -X5=0 \/ -X6=1 \/ -X7=0 \/ -X8=1 ,

-X1=1 \/ -X2=1 \/ -X3=0 \/ -X4=1 , -X3=1 \/ -X4=1 \/ -X5=0 \/ -X6=1 , -X5=1 \/ -X6=1 \/ -X7=0 \/ -X8=1 ,

-X1=1 \/ -X2=1 \/ -X3=0 \/ -X4=0 , -X3=1 \/ -X4=1 \/ -X5=0 \/ -X6=0 , -X5=1 \/ -X6=1 \/ -X7=0 \/ -X8=0 ,

-X1=1 \/ -X2=0 \/ -X3=1 \/ -X4=0 , -X3=1 \/ -X4=0 \/ -X5=1 \/ -X6=0 , -X5=1 \/ -X6=0 \/ -X7=1 \/ -X8=0 ,

-X1=1 \/ -X2=1 \/ -X3=1 \/ -X4=0 , -X3=1 \/ -X4=1 \/ -X5=1 \/ -X6=0 , -X5=1 \/ -X6=1 \/ -X7=1 \/ -X8=0 ,

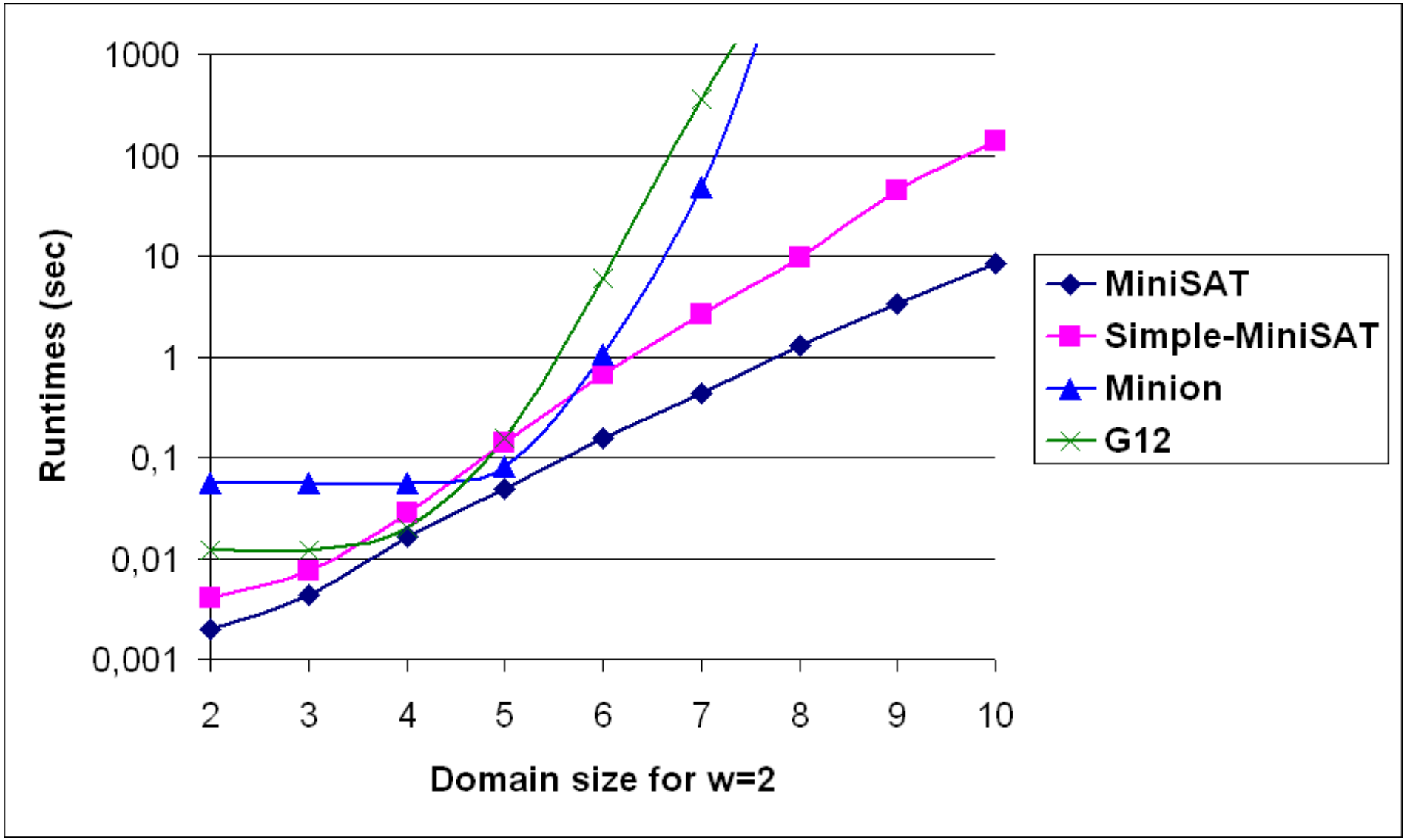
-X1=1 \/ -X2=1 \/ -X3=1 \/ -X4=1 , -X3=1 \/ -X4=1 \/ -X5=1 \/ -X6=1 , -X5=1 \/ -X6=1 \/ -X7=1 \/ -X8=1 ,

-X1=0 \/ -X2=0 \/ -X3=0 \/ -X4=0 , -X3=0 \/ -X4=0 \/ -X5=0 \/ -X6=0 , -X5=0 \/ -X6=0 \/ -X7=0 \/ -X8=0 ,

-X1=0 \/ -X2=1 \/ -X3=1 \/ -X4=0 , -X3=0 \/ -X4=1 \/ -X5=1 \/ -X6=0 , -X5=0 \/ -X6=1 \/ -X7=1 \/ -X8=0 ,

-X1=1 \/ -X2=0 \/ -X3=0 \/ -X4=0 , -X3=1 \/ -X4=0 \/ -X5=0 \/ -X6=0 , -X5=1 \/ -X6=0 \/ -X7=0 \/ -X8=0 ,

-X1=1 \/ -X2=0 \/ -X3=0 \/ -X4=1 , -X3=1 \/ -X4=0 \/ -X5=0 \/ -X6=1 , -X5=1 \/ -X6=0 \/ -X7=0 \/ -X8=1



'Local consistency and SAT-solvers' (Jeavons & Petke, 2010)

Theorem

If the **tree-width** of an unsatisfiable CSP instance is k, then a standard randomised **SAT-solver** will decide the unsatisfiability of the direct encoding of P in a **polynomial** number of restarts.

for the exact statement see *'Local consistency and SAT-solvers' (Jeavons & Petke, 2011)*

Problem Modelling

- CP solvers:
  - no standard input format
- SAT solvers:
  - standard input format (CNF)
  - various CSP-to-SAT encodings

Q: When does complexity theory guide the practitioner ?

A: When an input model for the instance needs to be chosen.

Q: How does complexity theory guide the practitioner ?

A: By providing rules for choosing the right input model of an instance.

'The order encoding: from tractable CSP to tractable SAT'

(Petke & Jeavons, 2011)

SAT encodings

- Sparse encodings (e.g. direct, support)

  $x_{vi} : v = i$

- The log encoding

  x_{vi} : i^{th} bit of variable v

- The order encoding

  $x_{vi} : v \leq i$

Proposition

No sparse encoding of a CSP instance with domain size > 2 **belongs** to a **tractable** language **class** of SAT.

Moreover, **the log encoding** of any CSP instance with domain size > 4 containing certain unary constraints **does not belong** to any **tractable** language **class** of SAT.

'The order encoding: from tractable CSP to tractable SAT'
(Petke & Jeavons, 2011)

Max-closed constraints

$$a_1X_1 + a_2X_2 + \ldots + a_{r-1}X_{r-1} \geq a_rX_r + c$$

a_is are non-negative constants, c is a constant and X_is are variables

Min-closed constraints

Lemma

A constraint defined on the Boolean domain is **min-closed** if and only if it is logically equivalent to a conjunction of **Horn clauses** over literals representing comparisons.

- Efficient algorithm for deciding the satisfiability of Horn clauses is based on **unit propagation**.

Theorem

If a CSP instance contains **max-closed** constraints only, then its **order encoding** will be **min-closed**.

'The order encoding: from tractable CSP to tractable SAT'
(Petke & Jeavons, 2011)

Max-closed constraints

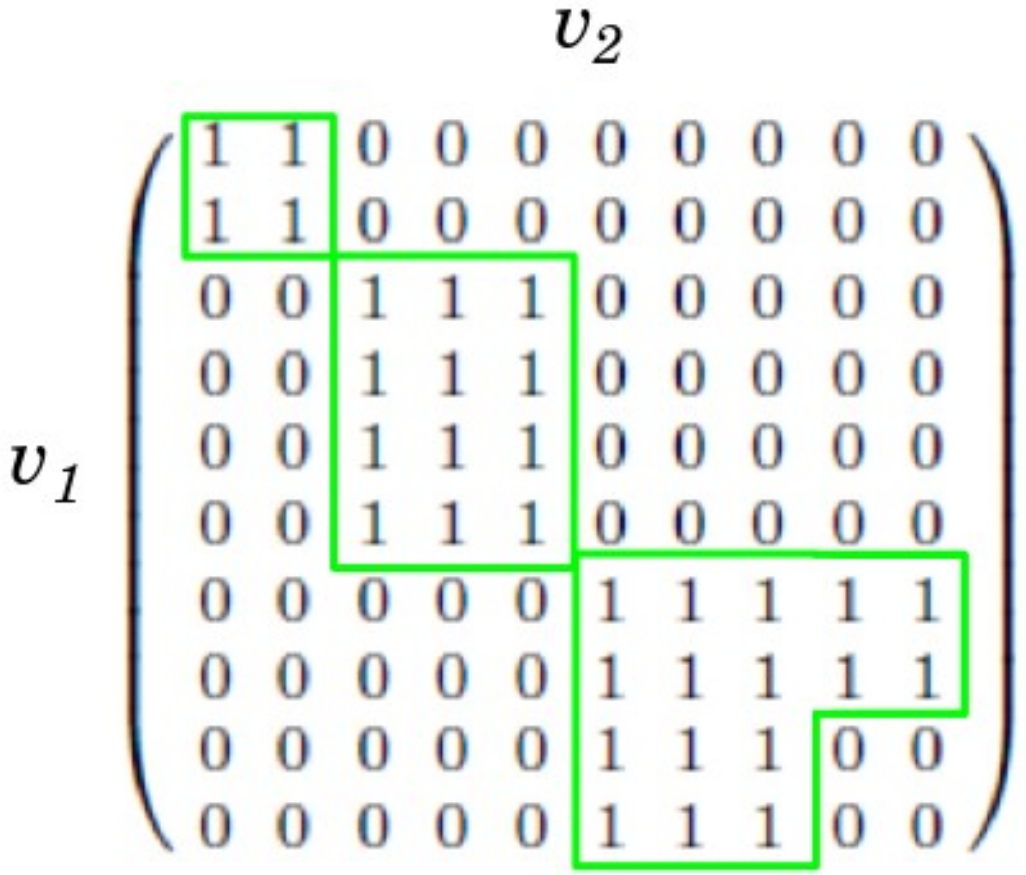
- We predict that SAT solvers will do better with the order encoding

Max-closed constraints

number of CSP variables	number of CSP values	number of constraints	MiniSAT (sec.) direct encoding	MiniSAT (sec.) log encoding	MiniSAT (sec.) order encoding
satisfiable instances					
3	10	100	0.01	0.06	0.00
6	6	10	7.42	96.97	0.07
9	3	10	1.30	4.99	0.01
9	3	100	5.15	23.27	0.59
9	4	10	> 15 min	176.16	0.73
10	3	10	20.08	107.38	0.11
unsatisfiable instances					
3	10	100	0.02	0.06	0.00
6	6	10	26.64	515.90	0.01
9	3	10	7.85	41.12	0.00
9	3	100	7.87	40.15	0.02
9	4	10	> 15 min	299.60	0.02
10	3	10	140.68	907.51	0.01

'The order encoding: from tractable CSP to tractable SAT' (Petke & Jeavons, 2011)

Connected row-convex constraints



domain matrix representing the allowed assignments

Lemma

A constraint is **connected-row-convex** iff it is logically equivalent to a conjunction of **2CNF** clauses over literals representing comparisons.

Theorem

If a CSP instance contains only **connected row-convex** constraints, then its **order encoding** will be **connected-row-convex**.

'The order encoding: from tractable CSP to tractable SAT'
(Petke & Jeavons, 2011)

Connected-row-convex constraints

- We predict that SAT solvers will do better with the order encoding

Connected row-convex constraints

number of CSP variables	number of CSP values	number of constraints	MiniSAT (sec.) direct encoding	MiniSAT (sec.) log encoding	MiniSAT (sec.) order encoding
satisfiable instances					
100	10	100	0.01	0.01	0.00
10	100	10	0.31	6.14	0.00
100	100	10	2.85	11.76	0.02
10	100	100	15.74	645.36	0.00
10	110	100	15.75	696.07	0.00
10	200	100	226.75	> 15 min	0.00
unsatisfiable instances					
100	10	100	0.01	0.02	0.00
10	100	10	0.83	14.97	0.00
100	100	10	3.21	16.67	0.01
10	100	100	5.78	368.79	0.01
10	110	100	9.72	> 15 min	0.01
10	200	100	83.33	> 15 min	0.00

'The order encoding: from tractable CSP to tractable SAT' (Petke & Jeavons, 2011)

Complexity theory guides the practitioner :

- During the solving process
- During the modelling process

Complexity theory guides the practitioner by:

- Providing efficient algorithms for certain classes of CSP problems

- Providing benchmarks for the solvers that are challenging yet easy to analyse

Complexity theory guides the practitioner by:

- Providing rules for the choice of an input format

- Providing explanations for which parts of an algorithm are central for solving a particular class of problems