A Parametric Approach for Smaller and Better Encodings of Cardinality Constraints

CP 2013 - Uppsala

Robert Nieuwenhuis

+ Ignasi Abío, Albert Oliveras, Enric Rodríguez

Barcelogic Research Group, Tech. Univ. Catalonia, Barcelona

Outline of this talk

Outline of this talk

- Modern SAT solvers. Why do they work so well?

Outline of this talk

- Modern SAT solvers. Why do they work so well?
- Encoding a constraint for SAT

Outline of this talk

- Modern SAT solvers. Why do they work so well?
- Encoding a constraint for SAT
- For each constraint: encode it or build it in?

Outline of this talk

- Modern SAT solvers. Why do they work so well?
- Encoding a constraint for SAT
- For each constraint: encode it or build it in?
- In any case, better encodings are crucial

Outline of this talk

- Modern SAT solvers. Why do they work so well?
- Encoding a constraint for SAT
- For each constraint: encode it or build it in?
- In any case, better encodings are crucial
- Can do a lot of work at encoding time!

Outline of this talk

- Modern SAT solvers. Why do they work so well?
- Encoding a constraint for SAT
- For each constraint: encode it or build it in?
- In any case, better encodings are crucial
- Can do a lot of work at encoding time!
- What makes a good encoding for a given problem instance?

Outline of this talk

- Modern SAT solvers. Why do they work so well?
- Encoding a constraint for SAT
- For each constraint: encode it or build it in?
- In any case, better encodings are crucial
- Can do a lot of work at encoding time!
- What makes a good encoding for a given problem instance?
- “Optimal” encodings for cardinality constraints

Outline of this talk

- Modern SAT solvers. Why do they work so well?
- Encoding a constraint for SAT
- For each constraint: encode it or build it in?
- In any case, better encodings are crucial
- Can do a lot of work at encoding time!
- What makes a good encoding for a given problem instance?
- "Optimal" encodings for cardinality constraints
- Experimental results

Outline of this talk

- Modern SAT solvers. Why do they work so well?
- Encoding a constraint for SAT
- For each constraint: encode it or build it in?
- In any case, better encodings are crucial
- Can do a lot of work at encoding time!
- What makes a good encoding for a given problem instance?
- "Optimal" encodings for cardinality constraints
- Experimental results
- Concluding remarks

Why are modern SAT solvers so good?

Decades of academic and industrial efforts in SAT
Lots of $$$ from, e.g., EDA (Electronic Design Automation)

Why are modern SAT solvers so good?

Decades of academic and industrial efforts in SAT
Lots of $$$ from, e.g., EDA (Electronic Design Automation)

Lesson: Real-world problems $\neq$ random or artificial ones !

Why are modern SAT solvers so good?

Decades of academic and industrial efforts in SAT
Lots of $$$ from, e.g., EDA (Electronic Design Automation)

Lesson: Real-world problems $\neq$ random or artificial ones !

SAT gives us complete systematic search solvers:

- outperforming other methods
- on real-world problems from many sources, with a
- single, fully automatic, push-button, var selection strategy!
- Hence modeling is essentially declarative.

Why are modern SAT solvers so good?

Decades of academic and industrial efforts in SAT
Lots of $$$ from, e.g., EDA (Electronic Design Automation)

Lesson: Real-world problems $\neq$ random or artificial ones !

SAT gives us complete systematic search solvers:

- outperforming other methods
- on real-world problems from many sources, with a
- single, fully automatic, push-button, var selection strategy!
- Hence modeling is essentially declarative.

BUT...

- Very low-level language: need modeling and encoding tools
- Sometimes no adequate/compact encodings: arithmetic...
- Answers "unsat" or model. Optimization not as well studied.

DPLL (or CDCL) SAT Solvers

here: DPLL (= Davis-Putnam-Loveland-Logemann) = CDCL

DPLL (or CDCL) SAT Solvers

here: DPLL (= Davis-Putnam-Loveland-Logemann) = CDCL

An Abstract DPLL state has the form $A \| F$ (see [NOT], JACM'06):

Assignment A **:** **Clause set** F **:**

$\emptyset \quad \| \quad \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow$

DPLL (or CDCL) SAT Solvers

here: DPLL (= Davis-Putnam-Loveland-Logemann) = CDCL

An Abstract DPLL state has the form $A \| F$ (see [NOT], JACM'06):

Assignment A :		**Clause set** F :		
$\emptyset$	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(Decide)
1	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(UnitPropagate)

DPLL (or CDCL) SAT Solvers

here: DPLL (= Davis-Putnam-Loveland-Logemann) = CDCL

An Abstract DPLL state has the form $A \| F$ (see [NOT], JACM'06):

Assignment A :		**Clause set** F :		
$\emptyset$	$\|$	$\overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(Decide)
1	$\|$	$\overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2$	$\|$	$\overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(Decide)

DPLL (or CDCL) SAT Solvers

here: DPLL (= Davis-Putnam-Loveland-Logemann) = CDCL

An Abstract DPLL state has the form $A \| F$ (see [NOT], JACM'06):

Assignment A :		**Clause set** F :		
$\emptyset$	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$	$\Rightarrow$	(Decide)
1	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2$	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$	$\Rightarrow$	(Decide)
$1\ 2\ 3$	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$	$\Rightarrow$	(UnitPropagate)

DPLL (or CDCL) SAT Solvers

here: DPLL (= Davis-Putnam-Loveland-Logemann) = CDCL

An Abstract DPLL state has the form $A \| F$ (see [NOT], JACM'06):

Assignment A **:**		**Clause set** F **:**		
$\emptyset$	$\|$	$\overline{1}\lor 2,\ \overline{3}\lor 4,\ \overline{5}\lor\overline{6},\ 6\lor\overline{5}\lor\overline{2}$	$\Rightarrow$	(Decide)
1	$\|$	$\overline{1}\lor 2,\ \overline{3}\lor 4,\ \overline{5}\lor\overline{6},\ 6\lor\overline{5}\lor\overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2$	$\|$	$\overline{1}\lor 2,\ \overline{3}\lor 4,\ \overline{5}\lor\overline{6},\ 6\lor\overline{5}\lor\overline{2}$	$\Rightarrow$	(Decide)
$1\ 2\ 3$	$\|$	$\overline{1}\lor 2,\ \overline{3}\lor 4,\ \overline{5}\lor\overline{6},\ 6\lor\overline{5}\lor\overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2\ 3\ 4$	$\|$	$\overline{1}\lor 2,\ \overline{3}\lor 4,\ \overline{5}\lor\overline{6},\ 6\lor\overline{5}\lor\overline{2}$	$\Rightarrow$	(Decide)

DPLL (or CDCL) SAT Solvers

here: DPLL (= Davis-Putnam-Loveland-Logemann) = CDCL

An Abstract DPLL state has the form $A \| F$ (see [NOT], JACM'06):

Assignment A :		**Clause set** F :		
$\emptyset$	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(Decide)
1	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(UnitPropagate)
1 2	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(Decide)
1 2 3	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(UnitPropagate)
1 2 3 4	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(Decide)
1 2 3 4 5	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(UnitPropagate)

DPLL (or CDCL) SAT Solvers

here: DPLL (= Davis-Putnam-Loveland-Logemann) = CDCL

An Abstract DPLL state has the form $A \| F$ (see [NOT], JACM'06):

Assignment A :		**Clause set** F :		
$\emptyset$	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(Decide)
1	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2$	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(Decide)
$1\ 2\ 3$	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2\ 3\ 4$	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(Decide)
$1\ 2\ 3\ 4\ 5$	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2\ 3\ 4\ 5\ \overline{6}$	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$		

DPLL (or CDCL) SAT Solvers

here: DPLL (= Davis-Putnam-Loveland-Logemann) = CDCL

An Abstract DPLL state has the form $A \| F$ (see [NOT], JACM'06):

Assignment A :		**Clause set** F :		
$\emptyset$	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(Decide)
1	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2$	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(Decide)
$1\ 2\ 3$	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2\ 3\ 4$	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(Decide)
$1\ 2\ 3\ 4\ 5$	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2\ 3\ 4\ 5\ \overline{6}$	$\|$	$\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$	$\Rightarrow$	(Backtrack)

DPLL (or CDCL) SAT Solvers

here: DPLL (= Davis-Putnam-Loveland-Logemann) = CDCL

An Abstract DPLL state has the form $A \| F$ (see [NOT], JACM'06):

Assignment A **:**		**Clause set** F **:**		
$\emptyset$	$\|$	$\overline{1}\lor 2,\ \overline{3}\lor 4,\ \overline{5}\lor\overline{6},\ 6\lor\overline{5}\lor\overline{2}$	$\Rightarrow$	(Decide)
1	$\|$	$\overline{1}\lor 2,\ \overline{3}\lor 4,\ \overline{5}\lor\overline{6},\ 6\lor\overline{5}\lor\overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2$	$\|$	$\overline{1}\lor 2,\ \overline{3}\lor 4,\ \overline{5}\lor\overline{6},\ 6\lor\overline{5}\lor\overline{2}$	$\Rightarrow$	(Decide)
$1\ 2\ 3$	$\|$	$\overline{1}\lor 2,\ \overline{3}\lor 4,\ \overline{5}\lor\overline{6},\ 6\lor\overline{5}\lor\overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2\ 3\ 4$	$\|$	$\overline{1}\lor 2,\ \overline{3}\lor 4,\ \overline{5}\lor\overline{6},\ 6\lor\overline{5}\lor\overline{2}$	$\Rightarrow$	(Decide)
$1\ 2\ 3\ 4\ 5$	$\|$	$\overline{1}\lor 2,\ \overline{3}\lor 4,\ \overline{5}\lor\overline{6},\ 6\lor\overline{5}\lor\overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2\ 3\ 4\ 5\ \overline{6}$	$\|$	$\overline{1}\lor 2,\ \overline{3}\lor 4,\ \overline{5}\lor\overline{6},\ 6\lor\overline{5}\lor\overline{2}$	$\Rightarrow$	(Backtrack)
$1\ 2\ 3\ 4\ \overline{5}$	$\|$	$\overline{1}\lor 2,\ \overline{3}\lor 4,\ \overline{5}\lor\overline{6},\ 6\lor\overline{5}\lor\overline{2}$		

DPLL (or CDCL) SAT Solvers

here: DPLL (= Davis-Putnam-Loveland-Logemann) = CDCL

An Abstract DPLL state has the form $A \| F$ (see [NOT], JACM'06):

Assignment A :		**Clause set** F :		
$\emptyset$	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$	$\Rightarrow$	(Decide)
1	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2$	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$	$\Rightarrow$	(Decide)
$1\ 2\ 3$	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2\ 3\ 4$	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$	$\Rightarrow$	(Decide)
$1\ 2\ 3\ 4\ 5$	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2\ 3\ 4\ 5\ \overline{6}$	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$	$\Rightarrow$	(Backtrack)
$1\ 2\ 3\ 4\ \overline{5}$	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$		model found!

DPLL (or CDCL) SAT Solvers

here: DPLL (= Davis-Putnam-Loveland-Logemann) = CDCL

An Abstract DPLL state has the form $A \| F$ (see [NOT], JACM'06):

Assignment A :		Clause set F :		
$\emptyset$	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$	$\Rightarrow$	(Decide)
1	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2$	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$	$\Rightarrow$	(Decide)
$1\ 2\ 3$	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2\ 3\ 4$	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$	$\Rightarrow$	(Decide)
$1\ 2\ 3\ 4\ 5$	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$	$\Rightarrow$	(UnitPropagate)
$1\ 2\ 3\ 4\ 5\ \overline{6}$	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$	$\Rightarrow$	(Backtrack)
$1\ 2\ 3\ 4\ \overline{5}$	$\|$	$\overline{1}\vee 2,\ \overline{3}\vee 4,\ \overline{5}\vee\overline{6},\ 6\vee\overline{5}\vee\overline{2}$		model found!

More rules: Backjump, Learn, Forget, Restart [M-S,S,M,...]!

Backtrack vs. Backjump

Same example as before. Remember: Backtrack gave $1\ 2\ 3\ 4\ \overline{5}$.

But: decision level $3\ 4$ is irrelevant for the conflict $6 \vee \overline{5} \vee \overline{2}$:

$$\emptyset \quad \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad (\text{Decide})$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$1\ 2\ 3\ 4\ 5\ \overline{6} \quad \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad (\text{Backjump})$$

Backtrack vs. Backjump

Same example as before. Remember: Backtrack gave $1\ 2\ 3\ 4\ \overline{5}$.

But: decision level $3\ 4$ is irrelevant for the conflict $6 \vee \overline{5} \vee \overline{2}$:

$$
\begin{array}{llllll}
\emptyset & \parallel & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow & (\text{Decide}) \\
\vdots & \vdots & \vdots & & \\
1\ 2\ 3\ 4\ 5\ \overline{6} & \parallel & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow & (\text{Backjump}) \\
1\ 2\ \overline{5} & \parallel & \overline{1} \vee 2,\ \ \overline{3} \vee 4,\ \ \overline{5} \vee \overline{6},\ \ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow & \ldots
\end{array}
$$

Backtrack vs. Backjump

Same example as before. Remember: Backtrack gave 1 2 3 4 $\overline{5}$.

But: decision level 3 4 is irrelevant for the conflict $6 \vee \overline{5} \vee \overline{2}$:

$$
\begin{array}{lclcl}
\emptyset & \parallel & \overline{1}\vee 2,\ \ \overline{3}\vee 4,\ \ \overline{5}\vee\overline{6},\ \ 6\vee\overline{5}\vee\overline{2} & \Rightarrow & (\text{Decide}) \\
\vdots & \vdots & \vdots & & \\
1\ 2\ 3\ 4\ 5\ \overline{6} & \parallel & \overline{1}\vee 2,\ \ \overline{3}\vee 4,\ \ \overline{5}\vee\overline{6},\ \ 6\vee\overline{5}\vee\overline{2} & \Rightarrow & (\text{Backjump}) \\
1\ 2\ \overline{5} & \parallel & \overline{1}\vee 2,\ \ \overline{3}\vee 4,\ \ \overline{5}\vee\overline{6},\ \ 6\vee\overline{5}\vee\overline{2} & \Rightarrow & \ldots
\end{array}
$$

Backjump =

1. Conflict Analysis: compute backjump clause $C \vee l$ (here, $\overline{2}\vee\overline{5}$)
   - that is a logical consequence of F: can Learn it!
   - that reveals a unit propagation of l at earlier decision level d (i.e., where its part C is false)
2. Return to decision level d and do the propagation.

CDCL Solvers

CDCL Solvers

- UnitPropagate has priority over Decide.

CDCL Solvers

- UnitPropagate has priority over Decide.
- For Decide, select literal involved in many, recent conflicts (implemented, e.g., as VSIDS, [Chaff 2001]).

CDCL Solvers

- UnitPropagate has priority over Decide.
- For Decide, select literal involved in many, recent conflicts (implemented, e.g., as VSIDS, [Chaff 2001]).
- When a conflict is found, it is analyzed [M-SS 1999]:
  – The derived clause is Learned.
  – Backtrack is replaced by Backjump.

CDCL Solvers

- UnitPropagate has priority over Decide.
- For Decide, select literal involved in many, recent conflicts (implemented, e.g., as VSIDS, [Chaff 2001]).
- When a conflict is found, it is analyzed [M-SS 1999]:
  - The derived clause is Learned.
  - Backtrack is replaced by Backjump.
- Periodically, the solver Restarts [Gomes et al 1998].

CDCL Solvers

- UnitPropagate has priority over Decide.
- For Decide, select literal involved in many, recent conflicts (implemented, e.g., as VSIDS, [Chaff 2001]).
- When a conflict is found, it is analyzed [M-SS 1999]:
  – The derived clause is Learned.
  – Backtrack is replaced by Backjump.
- Periodically, the solver Restarts [Gomes et al 1998].
- Also periodically, Forget non-active learned clauses [GN 2002].

Encoding a constraint for SAT

Example: Cumulative resource constraints [Schutt Et al 2009 CP]:

- A number of tasks $\{1, 2, \cdots, n\}$ must be done.
- Tasks require some (limited) resources.
- Variable $a_{i,t}$ means "task i is active at time t"
- Cardinality Constraint:
  at every timepoint t, no more active tasks than machines:

$$a_{1,t} + a_{2,t} + \cdots + a_{n,t} \leqslant 20$$

Naive (direct) encoding: $\binom{n}{21}$ clauses of the form:

$$\overline{x}_1 \vee \ldots \vee \overline{x}_{21}$$

Encode it or build it in? (see next talk!)

"Build it in" = "Sat Modulo Theories" or "Lazy Clause Generation":

- Example: building in a cardinality constraint $x_i + \cdots + x_n \leqslant K$
- A propagator watches it
- Each time the propagator detects K true variables in $\{x_i, \ldots x_n\}$, it can propagate the remaining ones to false
- To explain a propagation: clause of the form $x_1 \wedge \ldots \wedge x_K \rightarrow \overline{y}$ or equivalently $\overline{x}_1 \vee \ldots \vee \overline{x}_K \vee \overline{y}$
- Explanations are needed at least for conflict analysis
- But someteimes it is useful to Learn them
- Bad situation: end up Learning full (naive) $\binom{n}{K}$ encoding: better to use a compact one with auxiliary variables

Encoding is many times better, especially for simpler constraints, such as Cardinality ones (see next talk).

What's a good encoding for an instance?

"Best encoding" = for this SAT solver on this instance.

But some criteria are usually desirable (for any constraint):

1. the encoding is correct and complete
2. UnitPropagate should preserve generalized arc consistency
3. small number of clauses needed
4. small number of auxiliary variables needed

Here, criteria 1 and 2 will always hold.

But is 3 more important or 4? Depends on solver and instance!

Therefore here we define a single encoding (a much more compact one) that really optimizes wrt. a cost function $\lambda \cdot \#vars + \#clauses$, where λ is decided by the user.
(can in fact optimize wrt. any efficiently computable function).

Our encoding

- [miniSAT+] Sorting network, $O(n \log^2 n)$ clauses and aux vars to sort $(x_1 \ldots x_n)$ into $(y_1 \ldots y_n)$.
- To express $x_i + \cdots + x_n \leqslant K$, add unit clause $\overline{y}_{k+1}$.
- For $\ldots \geqslant K$, add y_k. For $=$, add both.
- [Asin et al 2011] onle need $(y_1 \ldots y_k)$: other recursive approach using $O(n \log^2 K)$ clauses and aux vars.
  Large improvement since frequently $n \gg K$.

This paper:

- For small inputs, the naive direct approach is frequently better.
- For large inputs, we should use the recursive approach.
- Idea: Use recursive until small enough for direct.
- Dynamic programming for optimality wrt. $\lambda \cdot \#vars + \#clauses$

Our encoding (II)

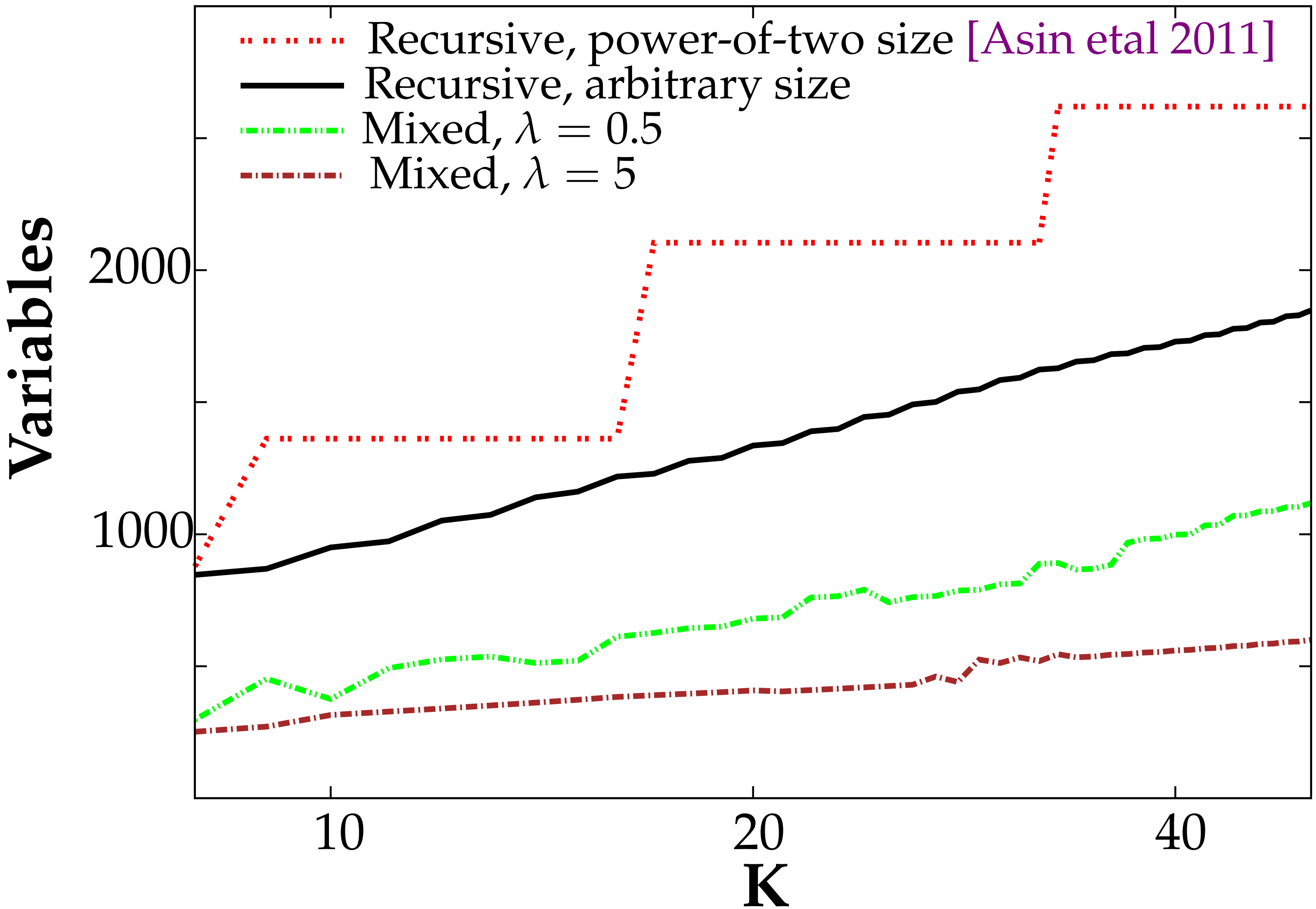
- We first remove the power-of-two restriction of our Cardinality networks of [Asin et al 2011].
  This already has a significant impact (see below).
- The work done at encoding time (dynamic programming) is negligible wrt. runtime.
- Some recursive cases not split into halves, but differently!

Experiments:

- We first compare wrt. number of variables and clauses, only with [Asin et al 2011]: known to be in general better than other previous approaches
- For SAT solver runtime (Lingeling, [Biere]), we also compare Adder [ES 06], BDD [BBR 06], and with our SMT approach.

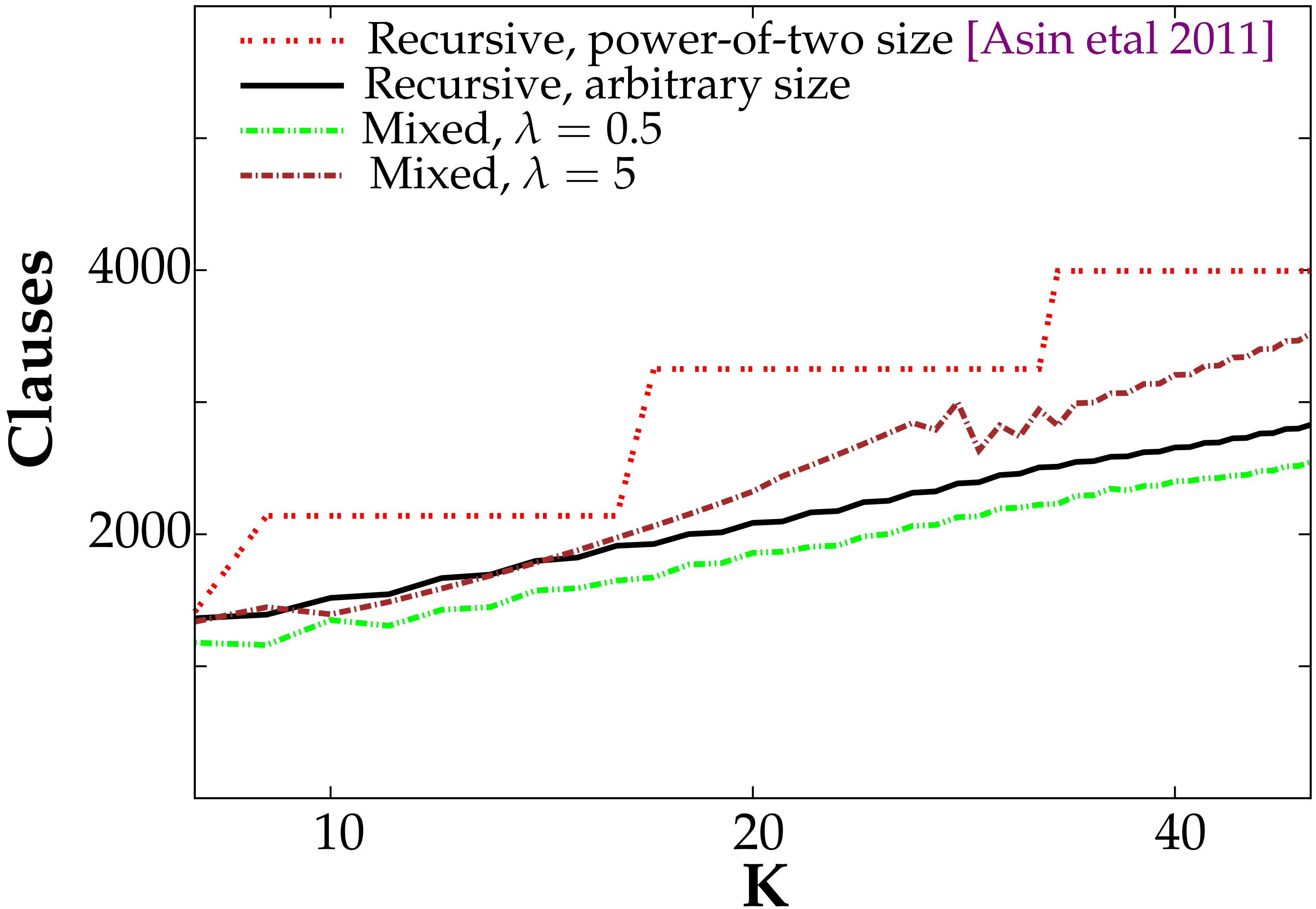
Experimental Results (Variables)

For $1 \leq K \leq 50$ and $n = 100$ (this is representative):



Experimental Results (Clauses)

For $1 \leq K \leq 50$ and $n = 100$ (this is representative):



Experimental Results (SAT Solving times)

MSU4 Suite: 6496 instances taking >5s (see paper for other suites).
Lingeling, TO 600s.

	# insts. w/ Speed-up of Mixed					≈	# insts. w/ Slow-down of Mixed				
	Speed-up factor:							Slow-down factor:			
	∞	>4	>2	>1.5	**total**		**total**	>1.5	>2	>4	∞
Power-of-two CN	43	732	2957	1278	*5010*	1438	*48*	1	23	13	11
Arbitrary-sized CN	10	149	544	726	*1429*	4835	*232*	3	106	43	80
Adder	985	1207	1038	1250	*4480*	1927	*89*	0	13	36	40
BDD	187	1139	1795	1292	*4413*	2002	*81*	4	10	31	36
SMT	1143	323	102	53	*1621*	3184	*1691*	0	1417	211	63

What does this mean? Some examples:
– in 187 instances Mixed did not time out but BDD did
– in 1139 instances Mixed was more than 4 times faster than BDD
– in 36 instances Mixed timed out but BDD did not

Concluding remarks

This kind of pragmatic work has a big impact in practice (Barcelogic.com)

Can do a lot of work at encoding time!

Divide and Conquer: even more expensive search at encoding time could pay off to find the best encoding for a given constraint

Pseudo-Boolean constraints: $a_1 x_1 + \cdots + a_n x_n \leq K$:

- Similar ideas mixing direct encodings and recursive ones
- Explore shared encoding of several constraints together

Build database of encodings for certain frequent constraints?

Thank you!