

The Limits of Constraint Decompositions

Christian Bessiere, **George Katsirelos**, Nina Narodytska,
Claude-Guy Quimper, Toby Walsh

May 28, 2009

Motivation

- Mature and effective technologies in SAT solvers
 - Watched literals
 - Clause learning
 - Restarts
 - VSIDS heuristic
 - ...
- Many successful applications
 - Model checking, planning, ...

Why Decompositions?

Set of “primitive” constraints that enforce the same level of consistency as a propagator.

- Using local reasoning only

Advantages:

- Learning
- Incrementality
- Ease of implementation

Why CNF Decompositions?

Primitive constraints are now just propositional clauses.

- Learning
- Incrementality
- Ease of implementation
- Efficiency
- Generality
 - Fixed-arity constraints

Why CNF Decompositions?

- Many global constraints can be effectively encoded into SAT
 - REGULAR, SEQUENCE, TABLE [Bacchus CP07]
 - GRAMMAR constraints [Quimper & Walsh CP07]

Can it always be done?

Can it always be done?

No.

Can it always be done?

Theorem

A propagator can be decomposed to CNF iff it can be encoded as a monotone Boolean circuit.

→ Lower bounds for monotone circuits

Propagator

A propagator is a function f on sets that is

- contracting: $f(D) \subseteq D$
- idempotent: $f(f(D)) = f(D)$
- *monotone*: $D_1 \subseteq D_2 \implies f(D_1) \subseteq f(D_2)$

Consistency checker

A consistency checker is a monotone Boolean function that is 0 only if the constraint is disentailed.

Polytime propagator



Polytime consistency checker

CNF Decompositions

Map the characteristic function of domains to propositional variables

- $x_{i,j}$ is FALSE if $j \notin D(X_i)$
- Otherwise *unset*

Use *auxiliary variables* to avoid exponential blowup.

CNF Decompositions

CNF decomposition of propagator: prunes domains by setting variables to `FALSE`

CNF decomposition of consistency checker: Sets an *output variable* to `FALSE`, *never fails*

Polysize CNF decomposition of propagator



Polysize CNF decomposition of consistency checker

Circuits

- Generic computation model
- The value problem is P -complete

- Monotone circuits: AND, OR gates only
- Compute exactly monotone Boolean functions
 - But not always in polynomial size
- Value problem still P -complete

Main result

Polysize CNF decomposition of consistency checker

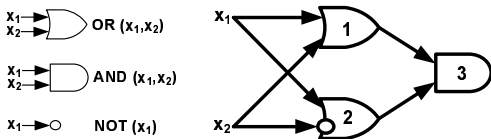


Polysize monotone circuit

Even when there exists polysize general circuit!

Unit propagation and general circuits

UP on Tseitin encoding computes the value, so why is that not a decomposition?



UP on the assignment \bar{x}_1 should compute FALSE but does not. Why? The variables of gates 1 and 2 have not been fixed to a value.

Equivalence to monotone circuits

Two observations

- Negative literals of input variables need not appear in any clause
- Auxiliary variables need not be forced to `TRUE`

Equivalence to monotone circuits

- Negative literals of input variables need not appear in a CNF decomposition

$$(\bar{x}_{i,j}, T) \rightarrow (x_{i,1}, \dots, x_{i,j-1}, x_{i,j+1}, \dots, x_{i,d}, T)$$

Equivalence to monotone circuits

- Negative literals of auxiliary variables need not be forced to `TRUE`
 - Lattice structure of models:
 $f(D_1) = 0 \wedge f(D_2) = 0 \implies f(D_1 \cap D_2) = 0$
 - UP on $D_1 \cap D_2$ makes a superset of the inferences of UP on D_1 and D_2 separately.
 - If an auxiliary variable y is `TRUE` under D_1 and `FALSE` under $D_2 \implies$ both `TRUE` and `FALSE` under $D_1 \cap D_2$, contradiction
- each variable only ever gets forced to a single value. By renaming, we make that value `FALSE` for all variables.

Known lower bounds

- Bipartite matching is superpolynomial for monotone circuits [Razborov85]
- Necessary and sufficient condition for satisfiability of `ALLDIFFERENT`
 - Smallest domain consistency CNF Decomposition of `ALLDIFFERENT` is superpolynomial
- Tardos gives exponential lower bounds for polytime computable graph properties

Possible escape

- Non-extensional representations of domains of auxiliary variables
- Bounds Consistency reasoning

Unlikely to work out.

What can we achieve?

BC on ALLDIFFERENT, GCC

- Based on Hall Intervals
- Interval of m values which contains m variables
 - Other variables must find their bound supports outside of this

ALLDIFFERENT BC

	1	2	3	4	5
X_1			*	*	
X_2	*	*	*	*	*
X_3			*	*	
X_4		*	*	*	*
X_5	*				

ALLDIFFERENT BC

	1	2	3	4	5
X_1			*	*	
X_2	*	*	*	*	*
X_3			*	*	
X_4		*	*	*	*
X_5	*				

$[1,1]$ is a Hall interval with X_5

ALLDIFFERENT BC

	1	2	3	4	5
X_1			*	*	
X_2	*	*	*	*	*
X_3			*	*	
X_4		*	*	*	*
X_5	*				

$[1,1]$ is a Hall interval with $X_5 \Rightarrow X_2 \neq 1$

ALLDIFFERENT BC

	1	2	3	4	5
X_1			*	*	
X_2		*	*	*	*
X_3			*	*	
X_4		*	*	*	*
X_5	*				

$[3,4]$ is a Hall interval with X_1, X_3

ALLDIFFERENT BC

	1	2	3	4	5
X_1			*	*	
X_2		*	*	*	*
X_3			*	*	
X_4		*	*	*	*
X_5	*				

$[3,4]$ is a Hall interval with $X_1, X_3 \Rightarrow X_2, X_4 \notin [3,4]$

ALLDIFFERENT BC

	1	2	3	4	5
X_1			*	*	
X_2		*			*
X_3			*	*	
X_4		*			*
X_5	*				

Constraint is now Range Consistent

ALLDIFFERENT BC Decomposition

$$A_{ilu} \iff X_i \in [l, u]$$
$$\sum_{i=1}^n A_{ilu} \leq u - l + 1$$

- Can be given directly to PB solver like MiniSat+
- or can be easily transformed to SAT
- UP achieves BC on ALLDIFFERENT
- Time complexity $O(nd^2)$ down branch

Other constraints

Similar decompositions for other constraints

- GCC
- NValue
- Same/UsedBy
- Sort
- SoftGCC

Conclusions

- We cannot decompose all polytime propagators to SAT
 - `ALLDIFFERENT` is an example, but hopefully more examples in the future
 - Open Question: will exponentially large domains and BC reasoning help us?
- But some algorithms have surprisingly simple decompositions
 - BC on `ALLDIFFERENT` and many related counting constraints