Lifting Structural Tractability to CSP with Global Constraints

Evgenij Thorstensen

Department of Computer Science
University of Oxford

September 2013

Global vs. table constraints

CSP solvers allow us to represent constraints *implicitly*, using built-in algorithms.

Constraint that specifies how often each domain value is to occur in a set of variables:

V	X	Y	Z
a	b	b	a
a	a	b	b
a	b	a	b
	...		

As GCC:

$K(a) = 2$
$K(b) = 2$

Global constraints

Many different global constraints exist:

- Global Cardinality (different kinds)
- AllDifferent
- NValue
- Regular language membership
- *SAT clauses*!

A lot is known about propagation for such constraints.

Structural tractability results?

For the extensional case, we get tractable classes by bounding

- treewidth
- hypertree width
- fractional hypertree width
- submodular width (FPT)

For problems with global constraints, only treewidth works.

Proof: Take k-colourability as CSP, add a cardinality constraint on V with $K(a) = \{0, \ldots, |V|\}$.

Structural tractability results?

For the extensional case, we get tractable classes by bounding

- treewidth
- hypertree width
- fractional hypertree width
- submodular width (FPT)

For problems with global constraints, only treewidth works.

Bounded treewidth implies bounded arity, so not very useful.

What properties of table constraints do we need?

Interlude: What is a global constraint?

A *global constraint* is a parameterized poly-time algorithm.

It takes a description, and decides which assignments are allowed.

Interlude: What is a global constraint?

A *global constraint* is a parameterized poly-time algorithm.

It takes a description, and decides which assignments are allowed.

This covers the known global constraints.

Various similar definitions exist in the literature.

Properties of structural restrictions

Extensional representation grows linearly with the number of solutions — not the case for global constraints.

On extensional CSP, fractional hypertree width k implies that the number of solutions to instance P is at most $|P|^k$ *for every projection* of P.

Therefore, a simple backtracking search can find all solutions in polynomial time.

Backtracking search

Consider a CSP $P = \langle V, D, C \rangle$.

Pick a variable $v_1 \in V$. For every assignment to it, check whether the projection of P down to this variable allows this assignment.

Pick another variable $v_2 \in P$. For each assignment to v_1 from above, check if any assignment to v_2 extends this assignment to one allowed by the projection of P down to $\{v_1, v_2\}$.

And so on.

Algorithm properties

Runtime depends on

- Time needed to build projection
- Time to check each assignment
- Number of solutions at every step

No need to actually build the projection.

Few solutions

Checking partial assignments can be done in polynomial time for most global constraints.

The number of solutions at every step is not related to hypergraph width.

Few solutions

Checking partial assignments can be done in polynomial time for most global constraints.

The number of solutions at every step is not related to hypergraph width.

Obvious fix: Consider only constraints that have polynomially many solutions in their size.

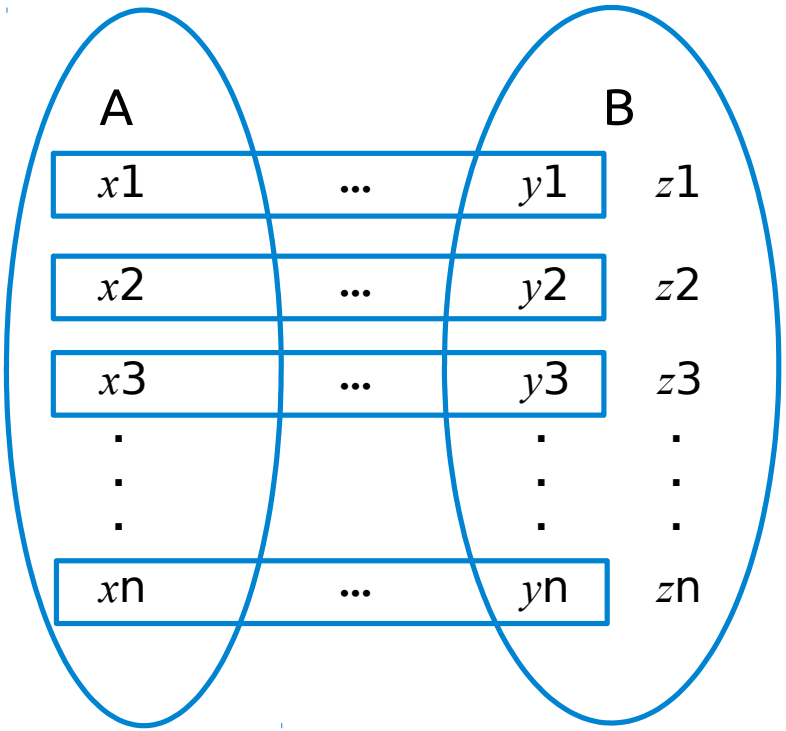
A better idea

Variables that only occur in a single constraint can be safely ignored.

For every constraint, consider set S of intersection variables. If our instance P has few solutions when projected down to any $X \subseteq S$, we can find them.

This gives us a reduction to extensional representation while *preserving the hypergraph*.

Example



A and B having few solutions suffices.

Result statement

- If our global constraints allow poly-time checking of partial assignments, and
- The instance has few solutions for every projection of every set of intersection variables, then

in polynomial time, we can reduce our instance to an extensional one with the same hypergraph.

Therefore, under these conditions every structural tractability result applies.

Bits and pieces

Result can be further generalized (see doctoral thesis). In particular, to Weighted CSP.

What if we have a constraint that is too hard?

If it has a back door set that is covered by other constraints satisfying our properties, we are still tractable.