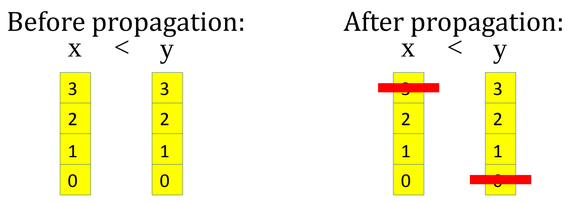
Extending Simple Tabular Reduction with Short Supports

Christopher Jefferson and Peter Nightingale

Introduction – Supports

In Constraint Programming systems, constraint *propagation algorithms* filter values out of variable domains when the values cannot be part of a global solution. For example, consider the following less-than constraint:



Because of the constraint, value 3 of x and value 0 of y cannot take part in any solution, so they are deleted.
If a value is contained in a satisfying assignment for the constraint (*eg x=2, y=3* for <) then it is not deleted, and we call the satisfying assignment a *support* for the value. This concept of support is pervasive in propagation algorithms.

Short Supports

The key concept used in this paper is short supports.
Some constraints can be satisfied by assigning only a few of their variables – after the assignment, the constraint doesn't care about the values of the rest. A short assignment that satisfies the constraint is called a *short support.*

A conventional support will only support the values contained in it.
A short support will support *all values of any variable not mentioned in it.* For example:

Domains *x1:{1,..,11}, x2, x3:{1,..,10}*
Constraint: (*x1 = x2 OR x1 = x3*)
Short support *S: (x1 → 1, x2 → 1)*

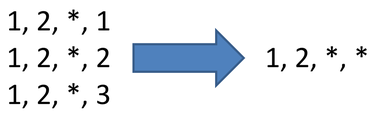
S supports *x1 → 1, x2 → 1* and all values of x3.
S supports *x1 → 1, x2 → 1* explicitly
S supports all values of *x3* implicitly

Compression of Tuple Sets

Arbitrary constraints are typically given to a constraint solver as a list of satisfying tuples. To automatically apply one of the 'short' algorithms, we need a procedure to compress the full-length tuples into short supports. Finding the minimal set is NP-Hard. We propose a fast greedy tuple compression algorithm.

The basic step of the algorithm is to take *d* (short) supports and compress them into one short support. The *d* short supports must be identical apart from one variable.

For example, suppose we have a constraint of arity 4, and all domains are {1,2,3}. We use * to represent any-value (i.e. the variable is not mentioned in the short support).



In this example, the short support set contains three tuples that are identical apart from the last position. In the last position, all values are represented – so we can replace this set with a single short support. This step is iterated until it cannot be applied again.

This algorithm is used for some of the experiments presented in the paper.

STR2+ and ShortSTR2

Simple Tabular Reduction algorithms maintain a sparse set of supports. The supports are traditionally full-length tuples that satisfy the constraint. An example of a sparse set is shown below.

LIMIT → 4

Set Index	Tup Index
1	3
2	4
3	1
4	5
5	6
6	2

Tup Index	Tuple
1	<1,2,3>
2	<1,3,2>
3	<2,1,3>
4	<2,3,1>
5	<3,1,2>
6	<3,2,1>

In this sparse set, tuples 3,4,1,5 are in, and 6,2 are out.
In this paper, we adapt STR2+ to use short supports in place of full-length supports. Short supports are an excellent fit for STR2+ and the changes to STR2+ to accommodate short supports are very minor. We call the resulting algorithm ShortSTR2.

Compared to STR2+, ShortSTR2 benefits from potentially very large compression of the tuple sets so ShortSTR2 can be much faster than STR2+. Even when the tuple set does not compress at all, ShortSTR2 is almost as fast as STR2+.

Experiments

There are five experiments comparing ShortSTR2 against HaggisGAC (an earlier short support algorithm) and STR2+.

In one experiment we compared ShortSTR2 with short supports (obtained using the greedy compression algorithm (left)) against ShortSTR2 with full-length supports, on some 'oscillating life' problems. Short supports give a speed improvement and also a substantial memory saving – we were able to run the largest problem (QuadLife) with short supports but not using full-length supports because it exceeded 4GiB.

Problem	ShortSTR2 node rate Greedy compression	ShortSTR2 node rate Full length supports
Life	4,970	3,960
Brian's Brain	532	75
Immigration	4930	3590
QuadLife	483	>4GiB Memory

ShortSTR2 as a drop-in replacement

In one experiment we evaluated ShortSTR2 (with greedy tuple compression) as a drop-in replacement for STR2+, using XCSP benchmarks. Memory savings vary quite widely depending on whether the constraints are amenable to short supports.

Timings are of the whole solver when only a subset of the constraints may have been compressed, therefore they will underestimate the speed-up of ShortSTR2 compared to STR2+. Even so ShortSTR2 would be a worthwhile replacement for STR2+.

Problem class	Compression ratio	Speed-up ShortSTR2 compared to STR2+
Half	1.87	1.75
modifiedRenault	5.35	0.99
Rand-8-20-5	1.01	1.05
bddSmall	1.90	1.13
Renault	6.31	1.06
bddLarge	1.80	1.21
cril	1.19	1.11