Extending Simple Tabular Reduction with Short Supports

Christopher Jefferson, Peter Nightingale

University of St Andrews

Constraints, GAC

- Suppose we have finite-domain variables x_1, x_2, x_3 with domains x_1:{1,..,11}, x_2, x_3:{1,..,10}
- Constraint: ($x_1 = x_2$ OR $x_1 = x_3$)
- *Generalised Arc-Consistency* (GAC) requires that each value of each variable is contained in a satisfying tuple of the constraint
- To *establish* GAC: $x_1 \neq 11$

Support

- Suppose we have finite-domain variables x_1, x_2, x_3 with domains x_1:{1,..,11}, x_2, x_3:{1,..,10}
- Constraint: ($x_1 = x_2$ OR $x_1 = x_3$)
- Traditional definition of GAC *support*: a satisfying tuple of the constraint
- Value $x_1 \rightarrow 11$ has no support, and is deleted
- Value $x_1 \rightarrow 1$ is *not* deleted because it has support $\langle 1, 1, 3 \rangle$ (for example).

Short Support

- *The key idea used in this paper:*
- Suppose a constraint can be satisfied by an assignment to a small subset of its variables
  - This assignment is a *short support*
- Exploit these short supports to maintain GAC more efficiently

Short Support – Example

- Consider the running example again
- Domains x_1:{1,..,11}, x_2, x_3:{1,..,10}
- Constraint: ($x_1 = x_2$ OR $x_1 = x_3$)
- Short support: ($x_1 \rightarrow 1$, $x_2 \rightarrow 1$)
- Any *extension* of this short support to cover x_3 is a full-length support
  - Assuming we always use values in the domain
- Supports $x_1 \rightarrow 1$, $x_2 \rightarrow 1$, and all values of x_3

Short Support – Explicit and Implicit

- Consider the running example again
- Domains x_1:{1,..,11}, x_2, x_3:{1,..,10}
- Constraint: ($x_1 = x_2$ OR $x_1 = x_3$)
- Short support: ($x_1 \rightarrow 1$, $x_2 \rightarrow 1$)
  - *Explicitly* supports $x_1 \rightarrow 1$, $x_2 \rightarrow 1$
  - *Implicitly* supports all values of x_3

Short Support

- Previously applied in *GAC-Schema*-like algorithms:
  - SHORTGAC (IJCAI 2011), then refined to HAGGISGAC (JAIR 2013)
  - HAGGISGAC is orders of magnitude faster than GAC-Schema when using short supports
  - HAGGISGAC a little faster than GAC-Schema with full-length supports (for an unrelated reason)
- Bigger goal: match the speed of hand-written propagators

ShortSTR2

- A new GAC algorithm extending STR2+ with short supports
  - *Short supports are a perfect fit for STR2(+)*
  - STR2(+) already optimises *fully supported* variables
    - The variable is removed from loops
- For each short support:
  - Variables with implicit support are marked as fully supported
  - Variable-value pairs with explicit support are treated exactly as in STR2+
- Given full-length supports, virtually identical to STR2+

Simple Tabular Reduction

- STR maintains a sparse set of the satisfying tuples

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

Tup Index	<x1, x2, x3>
1	<1,2,3>
2	<1,3,1>
3	<2,1,3>
4	<2,3,2>
5	<3,1,2>
6	<3,2,1>

- Suppose x3, 1 is pruned
- Tuples 3,4,1,5 are in the set and 6,2 are out

Simple Tabular Reduction

LIMIT

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

Tup Index	<x1, x2, x3>
1	<1,2,3>
2	<1,3,1>
3	<2,1,3>
4	<2,3,2>
5	<3,1,2>
6	<3,2,1>

- Now suppose x2, 2 is pruned
- STR algorithms iterate through tuples 3, 4, ...

Simple Tabular Reduction

LIMIT → 3

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

Tup Index	<x1, x2, x3>
1	<1,2,3>
2	<1,3,1>
3	<2,1,3>
4	<2,3,2>
5	<3,1,2>
6	<3,2,1>

- Now suppose x2, 2 is pruned
- STR algorithms iterate through tuples 3, 4, 1, ..
- Set now contains 3, 4, 5

Simple Tabular Reduction

- STR(2)(+) worst case complexity is terrible – $O(n^2d^{n+1})$
- Why are STR algorithms fast for some constraints?
- After just a few calls, set has been reduced enormously
- An extremely eager incremental propagator

Tuple Compression

- Take a set of full-length tuples and create a (non-unique) set of short supports
  - NP-hard to find a minimal set
- We propose a simple, fast greedy algorithm

Tuple Compression

- Using * to represent *any-value*
- Arity 4 constraint, each domain {1,2,3}
- Basic step is to take *d* (short) tuples and compress to one short tuple:

1, 2, *, 1
1, 2, *, 2 → 1, 2, *, *
1, 2, *, 3

- Apply this rule to exhaustion

ShortSTR2 vs STR2+

- ShortSTR2 with tuple compression as a drop-in replacement for STR2+
- Whole solver speed-up– ranges from 0.99 to 1.75

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

Short Supports vs Full Length

- On Conway's Life and similar
- Problems are almost entirely one table constraint repeated
- Benefit of short supports varies

Problem	ShortSTR2 node rate Greedy compression	ShortSTR2 node rate Full length supports
Life	4,970	3,960
Brian's Brain	532	75
Immigration	4,930	3,590
QuadLife	483	>4GiB Memory

ShortSTR2 vs HaggisGAC

- Pigeonhole problem generalised to vectors of variables
- Vector not-equal constraints
- *p* is number of ‘pigeons’, *a* is number of variables per vector

p	a	ShortSTR2	HaggisGAC
30	5	92,500	44,100
30	10	142,000	70,700
30	20	111,000	67,000
30	50	87,200	55,000
30	100	67,600	45,200
30	200	53,700	46,100

ShortSTR2 vs HAGGISGAC

- Pigeonhole problem generalised to vectors of variables
- Vector not-equal constraints
- *p* is number of 'pigeons', *a* is number of variables per vector
- Neither dominates the other - complementary

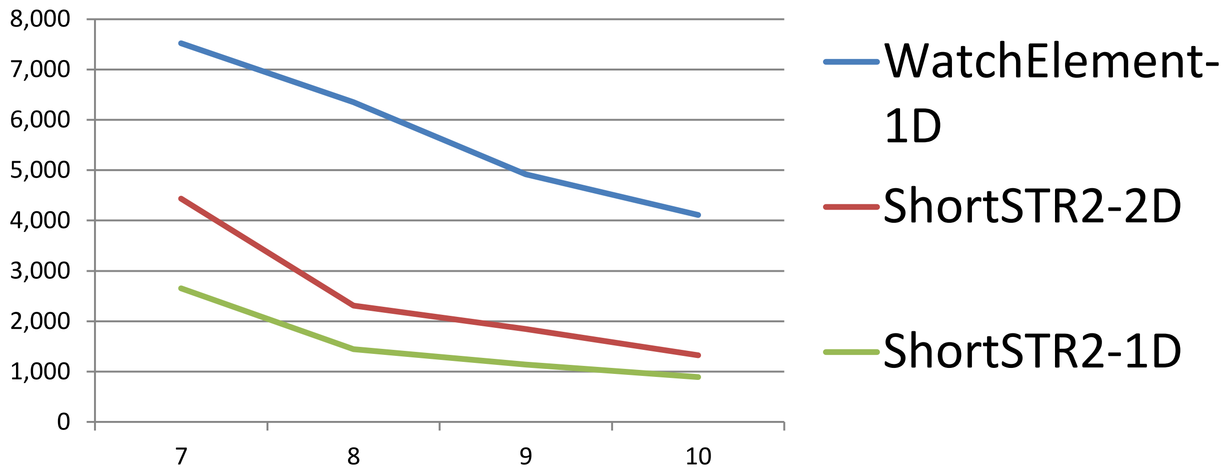
p	a	ShortSTR2	HAGGISGAC
5	100	592,000	1,790,000
10	100	250,000	653,600
20	100	119,000	158,800
30	100	67,600	45,200
40	100	43,700	18,000
50	100	31,900	10,900

ShortSTR2 vs HaggisGAC

- HaggisGAC is orders-of-magnitude faster than Constructive Or and GAC-Schema (JAIR 2013)
  - When constraint is amenable to short supports
  - Element, Lex ordering, Square packing
- HaggisGAC approaches specialised propagators – particularly lex ordering

ShortSTR2 vs specialised propagator

- We compared to the Watched Element propagator on quasigroup problems



- 2x to 4x slower than hand-written propagator

Conclusions

- ShortSTR2 is a new GAC algorithm that extends STR2+ using short supports
  - Could be used as a drop-in replacement for STR2(+)
- Complementary to HAGGISGAC in performance
  - Much simpler than HAGGISGAC
- Generic propagators as fast as specific hand-written ones?
  - Getting closer