Improving AC–Algorithms With Double–Support Checks

M.R.C. van Dongen and J.A. Bowen
University College Cork

August 26, 2000
Outline

- Constraint Networks.
- Arc–Consistency.
- Existing Arc–Consistency Algorithms.
- Double–Support Checks.
- Experimental Results.
- Discussion.
Constraint Networks

Let $X$ be a set of variables. For all $x \in X$ let $D(x)$ denote the domain of $x$. Finally let $S = \{x_{i_1}, \ldots, x_{i_m}\} \in 2^X \setminus \emptyset$.

$C_S$ is called a constraint on $S$ if $C_S \subseteq \times_{x \in S} D(x)$.

If $(v_{i_1}, \ldots, v_{i_m}) \in C_S$ it is said to satisfy $C_S$.

A constraint network is a collection of variables and constraints on those variables.
A constraint–network is called \textit{arc–consistent} iff for every variable, say $A$ it holds that for every value, say $v_A$, in $D(A)$ and for every constraint $C_{\{A,B\}}$ in the constraint network there is a value, say $v_B$, in $D(B)$ s.t. $v_B$ \textit{supports} $v_A$. 
Existing Arc–Consistency Algorithms

**DEE** Uses a queue of edges. Finds support for the values in the domains at both ends of the edge.

**AC-3** Uses a queue of arcs. When processing the arc from A to B it finds support for the values in $D(A)$ with $D(B)$. It has a $O(ed^3)$ time–complexity.

**AC-7** Never repeats a consistency–check. It has an optimal $O(ed^2)$ time–complexity.
AC-7  #CC (1)
AC-7  #CC (1)
AC-7  #CC (2)
AC-7  #CC (3)
AC-7  #CC  (4)
A

\[ \begin{array}{cccc}
0 & 1 & 2 & 3 \\
\end{array} \]

B

\[ \begin{array}{cccc}
0 & 1 & 2 & 3 \\
\end{array} \]

AC-7 \#CC (6)
AC-7  #CC  (10)
Double Support Checks

A *double–support check* is a consistency–check which seeks to find support for *two* values, whose support–statuses before the check are unknown.

**Note 1.** *To minimise the number of consistency–checks the number of successful double–support checks has to be maximised.*
AC-3\textsubscript{b}

- It is a cross-breed between AC-3 and DEE.

- It uses a heuristic which attempts to maximise the number of \textit{successful} double-support checks.

- It has a $O(ed^3)$ time-complexity.
AC-3 b  #CC (0)
AC-3 b #CC (1)
AC-3  \_b  \#CC  (2)
AC-3 \_b \quad \#CC \ (3)
AC-3 \ b \ \#CC \ (5)
AC-3 \ b \ #CC \ (6)
AC-3 \_ b \#CC (7)
AC-3 $b$  
$\#CC (7)$
AC-3 \( b \) \#CC (8)
Experimental Results

For each combination of (density, tightness) in 
\( \{(d/40, t/40) | d \in \{1, 2, \ldots 39\}, t \in \{1, 2, \ldots 39\}\} \) 20 random connected CSPs were generated (30,420 in total).

<table>
<thead>
<tr>
<th>#cc</th>
<th>DEE</th>
<th>AC-3</th>
<th>AC-3(_b)</th>
<th>AC-7</th>
</tr>
</thead>
<tbody>
<tr>
<td>7311</td>
<td>7261</td>
<td>5077</td>
<td>5319</td>
<td></td>
</tr>
</tbody>
</table>

Average Number of Consistency-Checks
$\#cc(AC-3) - \#cc(AC-3_b)$
#cc(AC-7) \textminus #cc(AC-3_b)
Discussion

• To minimise the number of consistency–checks, the number of successful double–support checks has to be maximised.

• For the problem set under consideration and the “usual” ordering heuristics $AC-3_b$ outperforms $AC-7$.

• Trying to maximise the number of successful double-support checks seems to improve arc–consistency algorithms.

• Don’t be too eager!