2010

A First Practical Algorithm for High Levels of Relational Consistency

Shant Karakashian  
*University of Nebraska - Lincoln*, shantk@cse.unl.edu

Robert J. Woodward  
*University of Nebraska - Lincoln*, rwoodwar@cse.unl.edu

Christopher Reeson  
*University of Nebraska - Lincoln*, creeson@cse.unl.edu

Berthe Y. Choueiry  
*University of Nebraska - Lincoln*, choueiry@cse.unl.edu

Christian Bessiere  
*University of Montpellier, France*, bessiere@lirmm.fr

Follow this and additional works at: [http://digitalcommons.unl.edu/cseconfwork](http://digitalcommons.unl.edu/cseconfwork)

Part of the [Computer Sciences Commons](http://digitalcommons.unl.edu/cseconfwork)

Karakashian, Shant; Woodward, Robert J.; Reeson, Christopher; Choueiry, Berthe Y.; and Bessiere, Christian, "A First Practical Algorithm for High Levels of Relational Consistency" (2010). *CSE Conference and Workshop Papers*. 175.

[http://digitalcommons.unl.edu/cseconfwork/175](http://digitalcommons.unl.edu/cseconfwork/175)

This Article is brought to you for free and open access by the Computer Science and Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in CSE Conference and Workshop Papers by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.
A First Practical Algorithm for High Levels of Relational Consistency

Shant Karakashian, Robert Woodward, Christopher Reeson, Berthe Y. Choueiry & Christian Bessiere

Constraint Systems Laboratory, University of Nebraska-Lincoln
LIRMM-CNRS, University of Montpellier

Acknowledgements:
• Kostas Stergiou and anonymous reviewers
• Experiments conducted at UNL’s Holland Computing Center
• NSF CAREER Award #0133568
• Robert Woodward supported by a UNL undergraduate research grant (UCARE) & by a B.M. Goldwater Scholarship
Outline

• Introduction
• Relational Consistency $R(\ast, m)C$:
  – Definition, Naïve algorithm, Properties
• Preliminaries: Dual CSP
• Our Approach
  – Algorithm
  – Index-Tree Data Structure
  – Advantages
• A weakened version of $R(\ast, m)C$: $wR(\ast, m)C$
• Experimental Evaluations
• Conclusions & Future Work
Introduction

• Local consistency techniques are at the heart of solving CSPs
• Low level consistency properties such as GAC are easy to apply & are effective for many problems
• There are problems that require higher levels of consistency for finding a solution in a reasonable amount of time
• We present a practical algorithm for enforcing relational $m$-wise consistency: $R(\ast,m)C$
Definition of $R(\ast,m)C$

- A CSP is $R(\ast,m)C$ iff
  - Every tuple in a relation can be extended to the variables in the scope of any $(m-1)$ other relations in an assignment satisfying all $m$ relations simultaneously.
Naïve Algorithm for $R(\ast,m)C$

- $R(\ast,m)C$ can be enforced on a CSP by
  - joining every combination of $m$ relations and
  - projecting the product on the individual relations

$$\forall R_i \in \{R_1, \ldots, R_m\}, R_i \leftarrow \pi_{\text{scope}(R_i)} (\bowtie_{j=1..m} R_j)$$
Properties of $R(\ast, m)C$

- It does not change the structure of the constraint network
- $R(\ast, m)C \prec RmC$  
  \[\text{[Dechter & van Beek ’97]}\]
- It filters the relations by removing tuples
- It is parameterized
  - We can control the level of consistency ($m$)
Preliminaries

• The **dual graph** of a CSP is a graph where
  – The nodes represent the relations
  – The edges are added between two relations with at least one common variable

• **Connected combination** of \( m \) relations is a set of relations that induce a connected component in the dual graph
The Induced Dual CSP

• Consider $\omega = \{R_1, R_2, \ldots, R_m\}$ a set of $m$ relations
• $P_\omega$ is the dual CSP **induced** by $\omega$ where
  – The dual variables represent the $m$ relations
  – The domains are the tuples of the relations $R_i$
  – The constraints in $P_\omega$ are binary & enforce equality on the CSP variables shared by the two relations
Enforcing R(*,m)C on the Induced Dual CSP P_ω

For each τ in R

Assign τ as a value for R

Solve P_ω (with τ fixed) with forward checking

If no solution found: delete τ

Add <ω', R'> to Q: R_i ≠ R'_i, R_i ∈ ω' and R'_i ∈ ω'
Index-Tree Data Structure

• When solving $P_\omega$, for a tuple $\tau$, Forward checking requires identifying all tuples matching $\tau$ in the neighboring relations

• We propose a new data structure: index-tree
  – Given a tuple $\tau$ of $R_1$ and a relation $R_2$
  – Identifies all the tuples of $R_2$ that match $\tau
Advantages of Our Approach

• The memory requirement of the operation
  \[ \forall R_i \in \{R_1, \ldots, R_m\}, R_i = \pi_{\text{scope}(R_i)} (\bowtie_{j=1..m} R_j) \]
  – \(O(t^m)\), \(t\): max number of tuples in a relation
  – For relations with 10,000 tuples, enforcing \(R(*,3)C\) requires in the order of 1TB of memory

• With our approach, the memory requirement is dominated by the index-tree structures
  – \(O(kte^2)\), \(k\): max arity of relations, \(e\): number of relations
  – While slightly decreasing the time complexity
Some edges are redundant for \( m=2 \)
Removing them reduces the number of combinations
For \( m>2 \), removal of these edges weakens \( R(\ast, m)C \)

Example
- Assume that no assignment satisfies variables A, B & C simultaneously
- To detect this inconsistency, need to consider \( R_1R_2R_4 \) simultaneously
- This inconsistency is not detected because we removed the combination \( R_1R_2R_4 \)
**R(\(*,m\)C versus wR(\(*,m\)C**

**R(\(*,m\)C is defined for** \(m \geq 2\)

<table>
<thead>
<tr>
<th>(m) (\geq 2)</th>
<th>(R(*,2)C \equiv wR(*,2)C)</th>
<th>[Janssen+ ‘89]</th>
</tr>
</thead>
<tbody>
<tr>
<td>(m &gt; 2)</td>
<td>(R(*,2)C \prec wR(*,m)C) (\prec R(*,m)C)</td>
<td></td>
</tr>
<tr>
<td>(m &lt; n)</td>
<td>(R(*,m)C \prec R(*,n)C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(wR(*,m)C \prec wR(*,n)C)</td>
<td></td>
</tr>
</tbody>
</table>

**A \prec B:** A is strictly weaker than B
## Experimental Results

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Algorithm</th>
<th>#Nodes Visited</th>
<th>Time [sec]</th>
<th>#Completed in 1 hour</th>
<th>#Fastest</th>
<th>#Backtrack Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifiedRenault</td>
<td>GAC</td>
<td>1,324,309.8</td>
<td>402.44</td>
<td>26</td>
<td>14</td>
<td>4/50</td>
</tr>
<tr>
<td>Max #tuples: 48,721</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>maxRPWC</td>
<td>2,110.8</td>
<td>305.37</td>
<td>31</td>
<td>3</td>
<td>19/50</td>
</tr>
<tr>
<td></td>
<td>wR(*,2)C</td>
<td>192.5</td>
<td>2.99</td>
<td>46</td>
<td>27</td>
<td>41/50</td>
</tr>
<tr>
<td></td>
<td>wR(*,3)C</td>
<td>82.5</td>
<td>7.55</td>
<td>50</td>
<td>4</td>
<td>48/50</td>
</tr>
<tr>
<td></td>
<td>wR(*,4)C</td>
<td>82.5</td>
<td>33.88</td>
<td>50</td>
<td>2</td>
<td>50/50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rand-8-20-5</td>
<td>GAC</td>
<td>30,501.7</td>
<td>1,795.26</td>
<td>9</td>
<td>2</td>
<td>0/20</td>
</tr>
<tr>
<td>Max #tuples: 78,799</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>wR(*,2)C</td>
<td>941.3</td>
<td>1,162.22</td>
<td>16</td>
<td>14</td>
<td>0/20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dag-rand</td>
<td>wR(*,2)C</td>
<td>0.0</td>
<td>27.21</td>
<td>25</td>
<td>25</td>
<td>25/25</td>
</tr>
<tr>
<td>Max #tuples: 150,000</td>
<td>wR(*,3)C</td>
<td>0.0</td>
<td>37.75</td>
<td>25</td>
<td>0</td>
<td>25/25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aim-200</td>
<td>GAC</td>
<td>1,876,247.6</td>
<td>542.48</td>
<td>8</td>
<td>0</td>
<td>0/24</td>
</tr>
<tr>
<td>Max #tuples: 7</td>
<td>maxRPWC</td>
<td>842,488.8</td>
<td>414.05</td>
<td>8</td>
<td>1</td>
<td>0/24</td>
</tr>
<tr>
<td></td>
<td>wR(*,2)C</td>
<td>2,670.2</td>
<td>35.51</td>
<td>12</td>
<td>7</td>
<td>4/24</td>
</tr>
<tr>
<td></td>
<td>wR(*,3)C</td>
<td>580.2</td>
<td>35.91</td>
<td>14</td>
<td>7</td>
<td>8/24</td>
</tr>
<tr>
<td></td>
<td>wR(*,4)C</td>
<td>443.8</td>
<td>240.13</td>
<td>14</td>
<td>2</td>
<td>9/24</td>
</tr>
</tbody>
</table>
Conclusions & Future Work

• We studied the relational consistency property \( R(*,m)C \)
  – Proposed a weaker variant \( wR(*,m)C \)
  – Presented a parameterized algorithm for enforcing it
  – Designed a new data structure (index tree) for efficiently checking the consistency of tuples between two relations
  – Evaluated it against GAC & maxRPWC

• Future work:
  – Handle relations defined as conflicts or in intension by domain filtering
  – Automatically identify the appropriate consistency level
  – Use \( R(*,m)C \) in a solver to identify tractable classes of CSPs
Thank You for Your Attention

Questions?