A Reformulation Strategy for Multi-Dimensional CSPs: The Case Study of the SET Game

Amanda Swearngin
University of Nebraska-Lincoln, aswearng@cse.unl.edu

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

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

Eugene C. Freuder
University College Cork, e.freuder@cs.ucc.ie

Follow this and additional works at: http://digitalcommons.unl.edu/cseconfwork

Part of the Computer Sciences Commons

http://digitalcommons.unl.edu/cseconfwork/179

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 Reformulation Strategy for Multi-Dimensional CSPs: The Case Study of the SET Game

Presented by: Robert J. Woodward,

Amanda Swearngin\textsuperscript{1} Berthe Y. Choueiry\textsuperscript{2} Eugene C. Freuder\textsuperscript{3}

\textsuperscript{1}ESQuaReD Laboratory, University of Nebraska-Lincoln
\textsuperscript{2}Constraint Systems Laboratory, University of Nebraska-Lincoln
\textsuperscript{3}Cork Constraint Computation Centre, University College Cork

Acknowledgements
\begin{itemize}
  \item National Science Foundation under grants CCF-0747009, CNS-0855139, and RI-1117956
  \item Science Foundation Ireland under grant 05/IN/1886
\end{itemize}
Outline

• General reformulation strategy for CSPs
  – Multidimensional CSPs (MD-CSPs)
  – Problem reformulation by value interchangeability
  – A general reformulation strategy for MD-CSPs

• Game of Set: A new toy problem
  – Game, CSP model
  – Problem reformulation
  – Algorithms & Results

• Conclusions
Multi-Dimensional CSPs

- All variables have the same domain
- Domain is multi-dimensional
  - A set of dimensions
  - Each domain value is described by a combination of dimensions values

- In MD-CSPs, a constraint can be
  - One-dimensional: defined over a single dimension
  - Multi-dimensional, otherwise

- Typical applications
  - Scheduling, resource allocation, configuration, etc.
Reformulation by value interchangeability

• Value interchangeability [Freuder 91]
  – Domain abstraction: equivalent values
  – ‘Perfect’ equivalence rare, small domain partitions
  – Ignoring some constraints yields larger domain partitions, smaller CSPs, smaller search space [Haselboeck 93, Choueiry+ 94]

• Abstraction in MD-CSPs [Freuder+ 95,97]
  – Abstract domains based on a dimension, $P_r$
  – Solve reformulated CSP
  – Use solution of $P_r$ to guide solving original CSP, $P_o$

• How to “use solution of $P_r$ to solve $P_o$”? Hard to automate
Reformulation Strategy for MD-CSPs

• Process
  For each one-dimensional constraint
  Abstract domains using interchangeability
  Enforce one-dimensional constraint
  Solve remaining CSPs with some solver

• Questions
  – Which 1-dim constraint to use first?
  – How to process reformulated problems?

• Case study of the Set game

\[ P_o: \text{Original MD-CSP} \]
• One-dimensional constraints: \( \{C_1, C_2, C_3, \ldots, C_n\} \)
• Other constraints

Exploit approximate symmetries to enforce \( C_1 \)

\[ P_i: \text{A set of reformulated CSPs} \]
• One-dim constraints: \( \{C_2, C_3, \ldots, C_n\} \)

Exploit approximate symmetries to enforce \( C_2 \)

\[ P_n: \text{A set of reformulated CSPs} \]
• One-dim constraints: \( \emptyset \)

Enforce remaining constraints
Outline

• General reformulation strategy for CSPs
  – Multidimensional CSPs (MD-CSPs)
  – Problem reformulation by value interchangeability
  – A general reformulation strategy for MD-CSPs

• Game of Set: A new toy problem
  – Game, CSP model
  – Problem reformulation
  – Algorithms & Results

• Conclusions
Game of Set

- Deck of 81 (=3^4) cards, each card with a unique combination of 4 attributes values
  1. Number ∈ {1, 2, 3}
  2. Color ∈ {green, purple, red}
  3. Filling ∈ {empty, stripes, full}
  4. Shape ∈ {diamond, squiggle, oval}

- Solution set: 3 cards
  ∀ attribute, the 3 cards have either the same value or all different values

- 12 cards are dealt, on table [3, 21]
- Recreational game, favorite of children & CS/math students
- New toy problem for AI: a typical multi-dimensional CSP
Set as an MD-CSP

• Model
  – Three variables
  – Same domain (12 cards)
  – One ‘physical’ constraints
  – Four 1-dimensional constraints

Same domain for all 3 variables

Domain Table

<table>
<thead>
<tr>
<th></th>
<th>(c_1)</th>
<th>(c_2)</th>
<th>(c_3)</th>
<th>(c_4)</th>
<th>(c_5)</th>
<th>(c_6)</th>
<th>(c_7)</th>
<th>(c_8)</th>
<th>(c_9)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Color</td>
<td>(r)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(g)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(p)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Filling</td>
<td>(f)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(e)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Shape</td>
<td>(s)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(o)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(d)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Domain dimensions

Constraint Systems Laboratory

1/23/2012

SARA 2011
Reformulation by Value Interchangeability

- **Filling**≡empty, **Color**≡red
- **Number** yields 3 domain partitions by neighborhood interchangeability (meta), replacing 6³ solutions by 3³ subproblems
- Enforcing \( N = \bigoplus N^\# \) replaces 3³ by 4 subproblems

\[
\begin{align*}
S &= \bigoplus S^\# \\
N &= \bigoplus N^\# \\
\end{align*}
\]

\[
\begin{align*}
V_1 & \quad V_2 & \quad V_3 \\
D_1 &= D_2 = D_3 \\
D_1 &\neq D_2 &\neq D_3
\end{align*}
\]
Reformulation Strategy for Set

\[ P_o: \text{Original MD-CSP} \]
\[ \text{One-dim constraints:} \quad \{ C_1, C_2, C_3, \ldots, C_n \} \]

Exploit approximate symmetries to enforce \( C_1 \)

\[ P_1: \text{A set of reformulated CSPs} \]
\[ \text{One-dim constraints:} \quad \{ C_2, C_3, \ldots, C_n \} \]

Exploit approximate symmetries to enforce \( C_2 \)

• Which dimension to choose first?
  ↷ For Set, heuristics based on data in ‘Domain Table:’
  Fewest subproblems first (infamously, Fail First Principle)
Selecting Domain Dimension

- Goal: Reduce branching factor
  - In example below, no card in domain has a shaded filling, thus, subproblems for $\text{Filling}=s$ and $F\neq$ do not exist

<table>
<thead>
<tr>
<th>Domain</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_4$</th>
<th>$c_5$</th>
<th>$c_6$</th>
<th>$c_7$</th>
<th>$c_8$</th>
<th>$c_9$</th>
<th>$\Sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Color</td>
<td>$r$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>$g$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$p$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Filling</td>
<td>$f$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>$e$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>$s$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Our reformulation algorithm for Set
  - Uses ‘Domain Table’ & ‘Summary of Domain Table’
  - Has 4 tests & 5 heuristics
Algorithms: Finding all Solutions

1. Brute-force search (BF)
   - 3-nested for-loops generate all combinations, then test for solutions
   - Contradicts 40+ years of CP research & experience 😞
   - Does not scale \((12^3 \sim d^n)\)

2. Backtrack search (Basic Solver)
   - Symmetry breaking (lexicographic ordering)
   - Both forward-checking (equality) & back-checking (All-diff constraints)

3. Reformulation-based algorithm
   - Uses 2 data structures: ‘Domain Table’ & ‘Summary of Domain Table’
   - Includes 5 selection heuristics
   - Open subproblems maintained in an agenda: room for heuristics (1Sol)
   - Empirical tests: randomly selected ‘hands’ of 3 to 81 cards, results averaged over of 1,000 runs
Results

- **#CC,#NV**: Reformulation dramatically reduces # of combinations tested
- **CPU time** reflects the cost of setting up the data structures for the CSP & search

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>#Cards</th>
<th>#Sol</th>
<th>#CC</th>
<th>#NV</th>
<th>Time [msec]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brute Force</td>
<td>12</td>
<td>2.77</td>
<td>1956.8</td>
<td>220</td>
<td>0</td>
</tr>
<tr>
<td>BT Search</td>
<td>12</td>
<td>2.77</td>
<td>1726.6</td>
<td>80.77</td>
<td>62.46</td>
</tr>
<tr>
<td>Reformulation</td>
<td></td>
<td></td>
<td>85.1</td>
<td>12.65</td>
<td>5.85</td>
</tr>
<tr>
<td>Brute Force</td>
<td>81</td>
<td>1080</td>
<td>758808</td>
<td>85320</td>
<td>0</td>
</tr>
<tr>
<td>BT Search</td>
<td>81</td>
<td>1080</td>
<td>553365</td>
<td>4401</td>
<td>101.04</td>
</tr>
<tr>
<td>Reformulation</td>
<td>81</td>
<td>1080</td>
<td>31158</td>
<td>2565</td>
<td>39.44</td>
</tr>
</tbody>
</table>

• #CC,#NV: Reformulation dramatically reduces # of combinations tested

• CPU time reflects the cost of setting up the data structures for the CSP & search
Online Game gameofset.unl.edu

- Game running online
- Interface explaining the reformulation still in development
- Advertzmt: minesweeper.unl.edu & sudoku.unl.edu (CP-based)
Conclusions

• Contributions
  – A systematic approach to reformulation and ‘conditional’ symmetries
  – Applicability to real-world problems highly promising
  – A new toy problem for AI research & education 😊

• Technical issues
  – Generalize heuristics for dimension selection and problem decomposition
  – Explore other types of interchangeability/symmetries
  – Extend definition of MD-CSP to allow unequal/all-diff domains

• Modeling lesson
  – CSP variables and values are often ‘objects’ with attributes
  – So far, we have integrated those attributes in the constraint definitions
  – Let’s rethink CSP modeling: Maybe multi-dimensional CSPs are more common than we thought they are..