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

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/181

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

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
• National Science Foundation under grants CCF-0747009, CNS-0855139, and RI-1117956
• Science Foundation Ireland under grant 05/IN/1886
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.

<table>
<thead>
<tr>
<th>dim/dom</th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>$v_4$</th>
<th>$v_5$</th>
<th>$v_6$</th>
<th>$v_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>dim₁</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>dim₂</td>
<td>r</td>
<td>r</td>
<td>g</td>
<td>p</td>
<td>g</td>
<td>r</td>
<td>p</td>
</tr>
<tr>
<td>dim₃</td>
<td>f</td>
<td>e</td>
<td>e</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td></td>
</tr>
</tbody>
</table>

Value in domain

dim₁

dim₂

dim₃
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 domain 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
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(=3⁴) 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>
<th>(c_{10})</th>
<th>(c_{11})</th>
<th>(c_{12})</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>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</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>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</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>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<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>
<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>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<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>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Domain dimensions
Reformulation by Value Interchangeability

- Interchangeability: $6^3$ vs $3^3$ solutions
- Enforcing $N=\oplus N\neq$: $3^3$ vs 4 solutions
Reformulation Strategy for Set

\( P_0 \): Original MD-CSP
- One-dim constraints: \( \{ C_1, C_2, C_3, \ldots, C_n \} \)

Exploit approximate symmetries to enforce \( C_1 \)

\( P_1 \): A set of reformulated CSPs
- One-dim constraints: \( \{ C_2, C_3, \ldots, C_n \} \)

Exploit approximate symmetries to enforce \( C_2 \)

\[ P(A_i, a) \]
- \( \mathcal{C} = \{ A_2, A_3, A_4 \} \)
- \( D_1 = D_2 = D_3 \)

\[ P(A_i, b) \]
- \( \mathcal{C} = \{ A_2, A_3, A_4 \} \)
- \( D_1 = D_2 = D_3 \)

\[ P(A_i, ≠) \]
- \( \mathcal{C} = \{ A_2, A_3, A_4 \} \)
- \( D_1 ≠ D_2 ≠ D_3 \)

\[ P(A_i, ≠) \]
- \( \mathcal{C} = \{ A_p, A_k \} \)
- \( D_1 ≠ D_2 ≠ D_3 \)

Enforcing mutually exclusive constraints of each dimension

- 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 Filling=\textit{s} and F\# 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>0 1 1 0 0 0 0 0 1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 0 0 1 0 0 0 0 0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 0 0 0 1 1 1 1 0 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Color</td>
<td>1 1 0 0 1 0 1 1 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 0 1 0 1 0 0 0 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 0 0 1 0 1 0 0 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(r)</td>
<td>(g)</td>
<td>(p)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filling</td>
<td>1 0 0 1 1 1 0 0 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 1 1 0 0 0 0 1 1 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(e)</td>
<td>(f)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shape</td>
<td>0 0 1 0 0 1 0 0 0 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 0 1 0 0 0 0 1 1 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 1 0 1 1 0 0 0 0 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(o)</td>
<td>(d)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>1726.6</td>
<td>80.77</td>
<td>62.46</td>
<td></td>
<td>62.46</td>
</tr>
<tr>
<td>Reformulation</td>
<td>85.1</td>
<td>12.65</td>
<td>5.85</td>
<td></td>
<td>5.85</td>
</tr>
<tr>
<td>Brute Force</td>
<td>81</td>
<td>1080</td>
<td>758808</td>
<td>85320</td>
<td>101.04</td>
</tr>
<tr>
<td>BT Search</td>
<td>553365</td>
<td>4401</td>
<td>101.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reformulation</td>
<td>31158</td>
<td>2565</td>
<td>39.44</td>
<td></td>
<td>39.44</td>
</tr>
</tbody>
</table>
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 domains that are not all equal

• Modeling lessons
  – Dimensional domain ANSWER: use tuple representation in Zinc, Essence
    • CSP variables and values are often ‘objects’ with attributes
    • So far, we have integrated those attributes in the constraint definitions
    • Multi-dimensional CSPs are likely common in practice
    • Let’s not miss out on the opportunity of exploiting them
  – Dynamic reformulation of model during search ANSWER: Not yet