Adapting DPOP to exploit partial symmetries

AAMAS ’10 – DCR workshop

Xavier Olive and Hiroshi Nakashima

Graduate School of Informatics
Kyōto university

May 11, 2010
Introduction

- Symmetry definition, detection, exploitation
- preprocessing method (DCR’09)
- dynamic exploitation method (DCR’10)
Introduction

Definitions

Detection of symmetries

Exploitation of symmetries

Conclusion
Introduction

Definitions
- Distributed constraint programming
- What is a symmetry?

Detection of symmetries

Exploitation of symmetries

Conclusion
Constraint programming problem

- finite set of variables $\mathcal{X}$
- finite set of domains $\mathcal{D}$
- finite set of constraints $\mathcal{C}$ on a scope of variables

Example: $n$-queens problem

- Place $n$ queens on a chessboard so that they do not attack each other.
Distributed constraint programming problem

- finite set of variables $\mathcal{X} = \{x_1, \cdots x_k\}$
- finite set of domains $\mathcal{D} = \{d_1, \cdots d_k\}$
- finite set of constraints $\mathcal{C}$, $c \in \mathcal{C}$ a subset of $d_1 \times \cdots \times d_k$
- finite set of non necessarily different agents $\mathcal{A} = \{a_1, \cdots a_k\}$
Example: SensorDCSP benchmark

- Each mobile \( (t_1) \) must be tracked by exactly 3 sensors \( (s_i) \).
- Additional constraint: visibility
Example: SensorDCSP benchmark

Related concepts
- natural distribution
- privacy
- local/global
Introduction

Definitions

Distributed constraint programming

What is a symmetry?

Detection of symmetries

Exploitation of symmetries

Conclusion

Adapting DPOP to exploit partial symmetries
In geometry, a function that preserves the structure
Symmetry on the equations

\[ x_1 + y_1 = z_1 \]

\[ z_2 = y_2 + x_2 \]
Symmetry on the constraint graph
Symmetry in CSP

- A mapping on the CSP that preserves its solutions
  \((\text{solution symmetry})\)

  or

- A permutation of the variables (values) that leaves the set of constraints globally unchanged
  \((\text{problem symmetry})\)
Example: symmetries in SensorDCSP

Adapting DPOP to exploit partial symmetries
Introduction

Definitions

Detection of symmetries

The symmetry $\times$ distribution issue
The distributed detection algorithm (DCR’09)

Exploitation of symmetries

Conclusion

Adapting DPOP to exploit partial symmetries
The symmetry $\times$ distribution issue

A symmetry leaves the structure of the whole problem unchanged, but no agent holds a definition of the global problem.

Where to start?
Global symmetries over the definition of the whole problem

L \subset G \subset P

Local symmetries involve only local constraints

Partial symmetries over a partial definition of the whole problem

Adapting DPOP to exploit partial symmetries
Theorem

If $\sigma$ is a partial symmetry
  for all agents owning a variable in $\sigma$ and
  for all their neighbour agents
then $\sigma$ is a global symmetry for the whole problem.

Consequence

If an agent agrees with the neighbours of its neighbours about its partial symmetries, those are global.
# Introduction

# Definitions

## Detection of symmetries

The symmetry $\times$ distribution issue

The distributed detection algorithm (DCR’09)

# Exploitation of symmetries

# Conclusion
The distributed detection algorithm

- preprocessing method to any resolution algorithm
- identifies global symmetries from partial symmetries
- requires a post-exploitation (e.g., reformulation) to be efficient
The distributed detection algorithm

- set a priority order on the agents
- agent $a_i$ detects a partial symmetry
  ... then send them to involved agents (lower priority)
- if all neighbours agree, we have a global symmetry
  ... and all involved agents have this knowledge.
Question

*If only one agent disagrees on a partial symmetry, can we still optimise anything?*

3 ways to exploit symmetries

- exploiting during search
- adding constraints
- reformulating
Introduction

Definitions

Detection of symmetries

**Exploitation of symmetries**

- The DPOP algorithm
- The SymDPOP algorithm
- SensorDCSP benchmark

Conclusion

Adapting DPOP to exploit partial symmetries
Definition: DFS tree

- Structure built from constraint graph
- Two variables falling in different branches are unconstrained
The DPOP algorithm

- **Constraints** (UTIL) are joined/projected and propagated from all leaves to the root.
- **Values** (VALUE) are calculated from parent values and propagated to the leaves.

![DPOP Diagram]

Adapting DPOP to exploit partial symmetries
**Symmetric DFS tree**

- **Invariant structure** through variable permutation.
- \( x_1 \leftrightarrow x_3, \ x_4 \leftrightarrow x_6, \ x_5 \leftrightarrow x_7 \)
Symmetric DFS tree

- Same UTIL messages
- We propose a propagation **merging nodes and messages**

Adapting DPOP to exploit partial symmetries
Introduction

Definitions

Detection of symmetries

Exploitation of symmetries

The DPOP algorithm

The SymDPOP algorithm

SensorDCSP benchmark

Conclusion
Distribution of data

- Agents lack information for finding global symmetry

... but are able to detect **partial symmetry**

![Diagram showing distribution of data with nodes and edges representing variables x0, x1, x2, x3, x4, x5, x6, x7.]

According to •

Adapting DPOP to exploit partial symmetries
Distribution of data

- Agents lack information for finding global symmetry
  ... but are able to detect **partial symmetry**

![Diagram showing partial symmetries](image)
Distribution of data

- Agents lack information for finding global symmetry
  ... but are able to detect **partial symmetry**

Adapting DPOP to exploit partial symmetries
Distribution of data

- Agents lack information for finding global symmetry
  
  ... but are able to detect **partial symmetry**

Adapting DPOP to exploit partial symmetries
Searching global symmetries: SymUTIL propagation

- Building a global symmetry from partial ones
- Process merged with UTIL propagation

\[ C_{4,1} = (C, x_1 \Leftrightarrow x_3, x_4 \Leftrightarrow x_6) \]
Searching global symmetries: SymUTIL propagation

- Building a global symmetry from partial ones
- Process merged with UTIL propagation

\[ C_{5,1} = (C, x_1 \Leftrightarrow x_3, x_5 \Leftrightarrow x_7) \]
Searching global symmetries: SymUTIL propagation

- Building a global symmetry from partial ones
- Process merged with UTIL propagation

$C_{1,0} = (C, x_1 \Leftrightarrow x_3, x_0 \Leftrightarrow x_0)$
Optimisation of VALUE propagation

- SymUTIL → $x_0$ (root): we found a global symmetry
- We can also optimise the VALUE propagation

$\mathcal{V}_0$ will not reach $x_3$
Optimisation of VALUE propagation

- SymUTIL $\rightarrow x_0$ (root): we found a global symmetry
- We can also optimise the VALUE propagation

$V_1$ will not reach $x_6, x_7$
A problem with no global symmetry

The \((\bullet,\bullet,\bullet)\)-symmetry is not a \((\bullet)\)-symmetry.
A problem with no global symmetry

\( (x_1, y_0) \) -symmetry is not a \( (x) \) -symmetry.

and \( \bullet \) analyze their partial symmetries and build a new one including \( \bullet \).

ends the UTIL propagation.

Adapting DPOP to exploit partial symmetries

and \( \bullet \) start propagating their partial symmetry.
A problem with no global symmetry

- analyses its SymUTILs and build a new one including .
A problem with no global symmetry

- changes SymUTIL to UTIL propagation.
A problem with no global symmetry

The \((x_0, \cdot, \cdot)\)–symmetry is not a \((\cdot, \cdot, \cdot)\)–symmetry.

\(\cdot\) and \(\cdot\) start propagating their partial symmetry. \(\cdot\) analyses its SymUTILs and builds a new one including \(\cdot\). \(\cdot\) changes SymUTIL to UTIL propagation. \(\cdot\) ends the UTIL propagation.

Adapting DPOP to exploit partial symmetries
Adapting DPOP to exploit partial symmetries

**Question**

If only one agent disagrees on a partial symmetry, can we still optimise anything?

**The answer is yes**

We propagate partial symmetries the same way we propagate constraints. (join/project)
Introduction

Definitions

Detection of symmetries

**Exploitation of symmetries**

- The DPOP algorithm
- The SymDPOP algorithm
- SensorDCSP benchmark

Conclusion
Execution for 25 sensors in a multithreaded process:

- SymDPOP: 30%
- DDA/DPOP: 50%
Adapting DPOP to exploit partial symmetries

Execution for 25 sensors in a MPI process (6 nodes):
- SymDPOP: 20%
- DDA/DPOP: worse
Execution for 25 sensors, 100 mobiles

- **number of messages**

<table>
<thead>
<tr>
<th></th>
<th>DPOP</th>
<th>DDA/DPOP</th>
<th>SymDPOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDA</td>
<td></td>
<td></td>
<td>1898</td>
</tr>
<tr>
<td>DFS</td>
<td>63648</td>
<td>30114</td>
<td>63648</td>
</tr>
<tr>
<td>UTIL/VALUE</td>
<td>11048</td>
<td>5298</td>
<td>5298</td>
</tr>
<tr>
<td>total</td>
<td>74696</td>
<td>37310</td>
<td>68946</td>
</tr>
</tbody>
</table>

- **SymDPOP: 10%**

- **DDA/DPOP: 50%**

- **volume of messages**

<table>
<thead>
<tr>
<th></th>
<th>DPOP</th>
<th>DDA/DPOP</th>
<th>SymDPOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDA</td>
<td></td>
<td></td>
<td>1742</td>
</tr>
<tr>
<td>DFS</td>
<td>2824</td>
<td>1889</td>
<td>2824</td>
</tr>
<tr>
<td>UTIL/VALUE</td>
<td>1977</td>
<td>1285</td>
<td>1406</td>
</tr>
<tr>
<td>total</td>
<td>4801</td>
<td>4916</td>
<td>4230</td>
</tr>
</tbody>
</table>

- **SymDPOP: 13%**

- **DDA/DPOP: worse**
Conclusion

- Symmetry exploitation improves time and communication performance

- SymDPOP treats symmetry as entities to be joined and projected

- SymDPOP yields better communication optimisation, even if the problem is not globally symmetric.