Modelling in Constraint Programming

Helmut Simonis

Cork Constraint Computation Centre
Computer Science Department
University College Cork
Ireland

CP-AI-OR 2009 Masterclass

Licence

This work is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.
Acknowledgments

The author is partially supported by Science Foundation Ireland (Grant Number 05/IN/I886). This material was developed as part of the ECLiPSe ELearning course: http://4c.ucc.ie/~hsimonis/ELeaarning/index.htm
Support from Cisco Systems and the Silicon Valley Community Foundation is gratefully acknowledged.

Objectives

- Overview of modelling in CP
- Based on example problems
- Introducing concepts as we go
- Showing the use of different global constraints
- Discussing search strategies
- Presentation of typical application domains
Outline

- Basic Modelling
- Choosing a Model
- Symmetry Breaking
- Customizing Search
- Complex Abstractions
- More Global Constraints
- A Hybrid Model

ECLiPSe

- Open sourced constraint programming language
- Development goes back to 1985
- ECRC, ICL, IC-Parc, PTL, Cisco
- http://www.eclipse-CLP.org/
- Specialities
  - Develop new solvers for specific domains
  - Integration with MIP
Self-study course in constraint programming
Supported by Cisco Systems and Silicon Valley Community Foundation
Multi-media format, video lectures, slides, handout etc
http://4c.ucc.ie/~hsimonis/ELearning/index.htm
Full course will be presented at ACP summer school 2009

Part I
Basic Modelling
What we want to introduce

- Basic Modelling
  - Variables
  - Constraints
  - Search
- Global Constraints
  - Powerful modelling abstractions
  - Non-trivial propagation
  - Different consistency levels
- Example: Sudoku puzzle

Problem Definition

Sudoku
Fill in numbers from 1 to 9 so that each row, column and block contain each number exactly once.
A variable for each cell, ranging from 1 to 9
A 9x9 matrix of variables describing the problem
Preassigned integers for the given hints
alldifferent constraints for each row, column and 3x3 block

Reminder: alldifferent

Argument: list of variables
Meaning: variables are pairwise different
Reasoning: Forward Checking (FC)
  - When variable is assigned to value, remove the value from all other variables
  - If a variable has only one possible value, then it is assigned
  - If a variable has no possible values, then the constraint fails
  - Constraint is checked whenever one of its variables is assigned
  - Equivalent to decomposition into binary disequality constraints
Declarations

:-module(sudoku).
:-export(top/0).
:-lib(ic).

top:-
    problem(Matrix),
    model(Matrix),
    writeln(Matrix).

Data

problem([[](4, _, 8, _, _, _, _, _, _),
    [](_, _, _, 1, 7, _, _, _, _),
    [](_, _, _, _, 8, _, _, 3, 2),
    [](_, _, 6, _, _, 8, 2, 5, _),
    [](_, 9, _, _, _, _, _, 8, _),
    [](_, 3, 7, 6, _, _, 9, _, _),
    [](2, 7, _, _, 5, _, _, _, _),
    [](_, _, _, _, 1, 4, _, _),
    [](_, _, _, _, _, 6, _, 4))].
model(Matrix):-
  Matrix[1..9,1..9] :: 1..9,
  (for(I,1,9),
    param(Matrix) do
      alldifferent(Matrix[I,1..9]),
      alldifferent(Matrix[1..9,I])
  ),
  (multifor([I,J],[1,1],[7,7],[3,3]),
    param(Matrix) do
      alldifferent(flatten(Matrix[I..I+2,J..J+2]))
  ),
  flatten_array(Matrix,List),
  labeling(List).

- Problem shown as matrix
- Each cell corresponds to a variable
- Instantiated: Shows integer value (large)
- Uninstantiated: Shows values in domain
- Problem shown as matrix
- Currently active constraint highlighted
- Values removed at this step shown in blue
- Values assigned at this step shown in red
Propagation Steps (Forward Checking)

After Setup (Forward Checking)

Helmut Simonis  Modelling in CP  19

Helmut Simonis  Modelling in CP  20
Bounds Consistency

Definition

A constraint achieves *bounds consistency*, if for the lower and upper bound of every variable, it is possible to find values for all other variables between their lower and upper bounds which satisfy the constraint.

Domain Consistency

Definition

A constraint achieves *domain consistency*, if for every variable and for every value in its domain, it is possible to find values in the domains of all other variables which satisfy the constraint.

- Also called *generalized arc consistency* (GAC)
- or *hyper arc consistency*
Should all constraints achieve domain consistency?

- Domain consistency is usually more expensive than bounds consistency
  - Overkill for simple problems
  - Nice to have choices
- For some constraints achieving domain consistency is NP-hard
  - We have to live with more restricted propagation

Improved Propagation in ECLiPSe

- `ic_global` library bounds consistent version
- `ic_global_gac` library domain consistent version
- Choose which version to use by using module annotation
- Choice can be passed as parameter
After Setup (Bounds Consistency)

Initial State (Domain Consistency)
Propagation Steps (Domain Consistency)

After Setup (Domain Consistency)
Comparison

<table>
<thead>
<tr>
<th>Forward Checking</th>
<th>Bounds Consistency</th>
<th>Domain Consistency</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 8 1 7 3 9 5 2</td>
<td>4 8 5 6 2 7 3 1</td>
<td>4 2 8 5 6 3 1 7</td>
</tr>
<tr>
<td>3 1 7 9 4 2 6 8</td>
<td>7 6 4 8 9 3 2 5</td>
<td>5 1 7 2 4 6 8 3</td>
</tr>
<tr>
<td>1 4 6 8 2 5 7 9</td>
<td>1 4 6 8 2 5 7 9</td>
<td>1 4 6 8 2 5 7 9</td>
</tr>
<tr>
<td>5 9 2 6 8 3 9 4</td>
<td>5 9 2 1 8 6 3 1</td>
<td>5 9 2 4 1 8 6 3</td>
</tr>
<tr>
<td>8 3 7 6 2 5 9 4 1</td>
<td>8 3 7 6 2 5 9 4 1</td>
<td>8 3 7 6 2 5 9 4 1</td>
</tr>
<tr>
<td>2 7 5 8 1 4 6</td>
<td>2 7 5 8 1 4 6</td>
<td>2 7 4 5 6 8 1 4</td>
</tr>
<tr>
<td>1 4 6 8 2 5 7 9</td>
<td>1 4 6 8 2 5 7 9</td>
<td>1 4 6 8 2 5 7 9</td>
</tr>
<tr>
<td>6 4 7 9 1 7 6 4</td>
<td>6 4 7 9 1 7 6 4</td>
<td>6 8 2 1 4 5 7 6</td>
</tr>
<tr>
<td>1 4 6 8 2 5 7 9</td>
<td>1 4 6 8 2 5 7 9</td>
<td>1 4 6 8 2 5 7 9</td>
</tr>
</tbody>
</table>

Typical?

- This does not always happen
- Sometimes, two methods produce same amount of propagation
- Possible to predict in certain special cases
- In general, tradeoff between speed and propagation
- Not always fastest to remove inconsistent values early
- But often required to find a solution at all
Simple search routine

- Enumerate variables in given order
- Try values starting from smallest one in domain
- Complete, chronological backtracking

Search Tree (Forward Checking)
Search Tree (Bounds Consistency)

Search Tree (Domain Consistency)
Observations

- Search tree much smaller for bounds/domain consistency
- Does not always happen like this
- Smaller tree = Less execution time
- Less reasoning = Less execution time
- Problem: Finding best balance
- For Sudoku: not good enough, should not require any search!

Solution

```
4 2 8 5 6 3 1 7 9
3 5 9 1 7 2 4 6 8
7 6 1 4 8 9 5 3 2
1 4 6 3 9 8 2 5 7
5 9 2 7 4 1 3 8 6
8 3 7 6 2 5 9 4 1
2 7 4 9 5 6 8 1 3
6 8 3 2 1 4 7 9 5
9 1 5 8 3 7 6 2 4
```
Summary

- Constraint programs consist of
  - Variables
  - Constraints
  - Search

- Keep problem structure in constraint model
- Exploit structure through global constraints
- Different consistency levels give more or less propagation
- One way to control cost/benefit compromise

More Information


H. Simonis.
Sudoku as a constraint problem.

I. Lynce and J. Ouaknine.
Sudoku as a SAT problem.
In *9th International Symposium on Artificial Intelligence and Mathematics*, January 2006.

H. Simonis.
Kakuro as a constraint problem.
Part II

Choosing the Model

What we want to introduce

- How to come up with a model for a problem
- Why choosing a good model is an art
- Channeling
- Projection
- Redundant constraints
Sports Scheduling

Tournament Planning

We plan a tournament with 8 teams, where every team plays every other team exactly once. The tournament is played on 7 days, each team playing on each day. The games are scheduled in 7 venues, and each team should play in each venue exactly once.

As part of the TV arrangements, some preassignments are done: We may either fix the game between two particular teams to a fixed day and venue, or only state that some team must play on a particular day at a given venue. The objective is to complete the schedule, so that all constraints are satisfied.

Example

<table>
<thead>
<tr>
<th></th>
<th>City 1</th>
<th>City 2</th>
<th>City 3</th>
<th>City 4</th>
<th>City 5</th>
<th>City 6</th>
<th>City 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td>7, 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 2</td>
<td>2</td>
<td>1, 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 3</td>
<td>7</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 4</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Day 5</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Day 6</td>
<td></td>
<td></td>
<td>5, 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 7</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>1, 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A More Abstract Formulation

Rooms Puzzle, (Thomas G. Room, 1955)

Place numbers 1 to 8 in cells so that each row and each column has each number exactly once, each cell contains either no numbers or two numbers (which must be different from each other), and each combination of two different numbers appears in exactly one cell.

Puzzle presented by R. Finkel
How to come up with a model

- What are the variables/what are their values?
- How can we express the constraints?
- Do we have these constraints in our system?
- Does this do good propagation?
- Backtrack to earlier step as required

Requirements

- There are 8 teams, seven days and seven locations
- Each team plays each other team exactly once
- Each team plays 7 games (redundant)
- Each team plays in each location exactly once
- Each team plays on each day exactly once
- A game consists of two (different) teams
- There are four games on each day (redundant)
- There are four games at each location (redundant)
- In any location there is at most one game at a time
Idea 1

- Matrix \( \text{Day} \times \text{Game} \ (7 \times 4) \)
- Each cell contains two variables, denoting teams
- Easy to say that team plays once on each day, \textit{alldifferent}
- Columns don’t have significance
- Model does not mention location, how to add this?
- How to express that each team plays each other once?

Idea 2, Change problem structure

- Matrix of \( \text{Day} \times \text{Location} \ (7 \times 7) \)
- Each cell contains two variables, each denoting a team
- How do we avoid symmetry inside cell?
- Need special value (0) to denote that there is no game
- In one cell, either both or none of the variables are 0
- Easy to say that each row and column contains each team exactly once
- Except for value 0, can not use \textit{alldifferent}
- Link between two variables in cell to state that game needs two different teams
- How to express that each (ordered) pair occurs exactly once?
Idea 3, Add location variables

- Model as in Idea 1, matrix \textit{Day} \times \textit{Game}
- Each cell contains two variables for teams and one for location
- Easy to state that games on one day are in different locations
- How to express condition that each team plays in each location once?
- Also, how to express that each team plays each other exactly once?

Idea 4, Use variables for pairs

- Matrix \textit{Day} \times \textit{Location}
- Each cell contains one variable ranging over (sorted) pairs of teams, and special value 0 (no game)
- Each pair value occurs once, except for 0
  - Special constraint \texttt{alldifferent0}
  - Or use \texttt{gcc}
- How to state that each team plays once per day?
- How to state that each team plays in each location?
Idea 5: If all else fails, use binary variables

- Binary variable stating that team \( i \) plays in location \( j \) at day \( k \)
- Three dimensional matrix
- Each team plays once on each day
- Each team plays once in each location
- Each game has two (different) teams, needs auxiliary variable
- Each pair of team meets once, needs auxiliary variables

Idea 6: An even bigger binary model

- Use four dimensions
- Team \( i \) meets team \( j \) in location \( k \) on day \( l \)
- \( 3136 = 8 \times 8 \times 7 \times 7 \) variables
- Constraints all linear
- Why use finite domain constraints?
Idea 7: A different mapping

- Each team plays each other exactly once, one variable for each combination (\(8 \times 7 / 2 = 28\) variables)
- Decide when and where this game is played, values range over combinations of days and locations (\(7 \times 7 = 49\) values)
- All variables must be different (no two games at same time and location)
- Each team plays 7 games, by construction
- How to express that each team plays once per day?
- How to express that each team plays in each location once?
### Numbering Values

<table>
<thead>
<tr>
<th></th>
<th>City 1</th>
<th>City 2</th>
<th>City 3</th>
<th>City 4</th>
<th>City 5</th>
<th>City 6</th>
<th>City 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Day 2</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>Day 3</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>Day 4</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
</tr>
<tr>
<td>Day 5</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
</tr>
<tr>
<td>Day 6</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>41</td>
<td>42</td>
</tr>
<tr>
<td>Day 7</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
</tr>
</tbody>
</table>

- Day 1 corresponds to values 1..7
- Four variables can take these values
- Day 2 corresponds to values 8..14, etc
- One constraint per day
- Exactly four of all variables take their value in the set ...
- Seven such constraints
Four games at each location

- City 1 corresponds to values 1, 8, 15, 22, 29, 36, 43
- Four variables can take these values
- City 2 corresponds to values 2, 9, 16, 23, 30, 37, 44
- One constraint per location
- Exactly four of all variables take their value in the set ...
- Seven such constraints over 28 variables each

Teams play once on a day (at a location)

- Select those variables which correspond to Team \( i \)
- Exactly one of those variables takes its value in the set 1..7
- Same for all other days
- Same for all other teams
- 56 Constraints over 7 variables each
- Similar for teams and locations, another 56 constraints
Are we there yet?

- 28 variables with 49 possible values
- 1 alldifferent
- 7 exactly constraints over all variables (Days)
- 7 exactly constraints over all variables (Locations)
- 56 exactly constraints over 7 variables each (Days)
- 56 exactly constraints over 7 variables each (Locations)
- Forgotten anything?
- Check the requirements

Do we satisfy the requirements?

- There are 8 teams, seven days and seven locations
- Each team plays each other team exactly once
- Each team plays 7 games (redundant)
- Each team plays in each location exactly once
- Each team plays on each day exactly once
- A game consists of two (different) teams
- There are four games on each day (redundant)
- There are four games at each location (redundant)
- In any location there is atmost one game at a time
What about the exactly constraint?

- ECLiPSe doesn’t provide this constraint
  - Other system might do, could switch system
- Implement it
  - Extend gcc to allow multiple values
  - Should be last resort
- Emulate constraint with others

Idea 8: Mapping games to days and locations

- For each game to be played, we have two variables
  - One ranges over the days
  - The other over the locations
- Easy to state that there are four games per day and location
- Easy to state that each team plays once per day and location
- How do we express that no two games are played at the same location and the same time?
  - If we had an alldifferent over pairs of variables...
  - Not in ECLiPSe
We have four games on each day

- Each row value is taken four times amongst the variables
- \( gcc([gcc(4,4,1),\ldots,gcc(4,4,7)],\text{Rows}) \)
- Similar for columns:
- \( gcc([gcc(4,4,1),\ldots,gcc(4,4,7)],\text{Cols}) \)

Reminder: \( gcc(\text{Pattern}, \text{Variables}) \)

- \( gcc \) Global Cardinality Constraint
- Pattern is list of terms \( gcc(\text{Low}, \text{High}, \text{Value}) \)
- The overall number of variables taking value \( \text{Value} \) is between \( \text{Low} \) and \( \text{High} \)
- Generalization of \text{alldifferent}
- Domain consistent version in ECLiPSe
Each team plays once per day

- For the seven variables which describe games of a team
- Each row value is taken exactly once amongst the variables
- Could use
  \[ \text{gcc}([\text{gcc}(1,1,1),\ldots,\text{gcc}(1,1,7)],\text{Vars}) \]
- But \text{alldifferent}(\text{Vars}) is more compact
- Similar for columns

---

How do the models differ?

<table>
<thead>
<tr>
<th>Idea</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( D \times G \times {f,s} \rightarrow T )</td>
</tr>
<tr>
<td>2</td>
<td>( D \times L \times {f,s} \rightarrow T \cup {0} )</td>
</tr>
</tbody>
</table>
| 3    | \( D \times G \times \{f,s\} \rightarrow T \)  
\( D \times G \rightarrow L \) |
| 4    | \( D \times L \rightarrow T \triangle T \cup \{0\} \) |
| 5    | \( T \times D \times L \rightarrow \{0,1\} \) |
| 6    | \( T \times T \times D \times L \rightarrow \{0,1\} \) |
| 7    | \( T \triangle T \rightarrow D \times L \) |
| 8    | \( T \triangle T \rightarrow D \)  
\( T \triangle T \rightarrow L \) |

D Days  
T Teams  
L Locations  
G Games
## Requirements Capture

<table>
<thead>
<tr>
<th>Idea</th>
<th>Requirement</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N</td>
<td>?</td>
<td>Y</td>
<td>?</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>?</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>?</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>?</td>
<td>Y</td>
<td>?</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>C</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>C</td>
<td>NL</td>
<td>L</td>
<td>L</td>
<td>NL</td>
<td>L</td>
<td>L</td>
<td>NL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>C</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>E</td>
<td>E</td>
<td>C</td>
<td>E</td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>
| 8    | C           | C | C | C | A | A | C | G | G | ?

### Comments on models

<table>
<thead>
<tr>
<th>Idea</th>
<th>Main point</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>missing locations, first second symmetry</td>
</tr>
<tr>
<td>2</td>
<td>spare value, first second symmetry</td>
</tr>
<tr>
<td>3</td>
<td>first second symmetry</td>
</tr>
<tr>
<td>4</td>
<td>spare value</td>
</tr>
<tr>
<td>5</td>
<td>0/1, non-linear constraints</td>
</tr>
<tr>
<td>6</td>
<td>0/1, large matrix</td>
</tr>
<tr>
<td>7</td>
<td>needs exactly constraint</td>
</tr>
<tr>
<td>8</td>
<td>needs alldifferent on tuples</td>
</tr>
</tbody>
</table>
Instead of expressing all constraints over one set of variables
Use multiple sets of variables (*viewpoints*)
Decide which constraint to express over which variables
Allows more freedom on how to express problem
Link the different variables with *channeling* constraints

In Our Case

- Combine ideas 7 and 8
- One set of variables ranging over pairs
- Another using two variables per game for day and location
- How to combine variables?
- Minimize loss of information
- Link pair variables to row and column variables
- Pair variable uses cell numbers 1-49 as values
- Row and column variables indicate on which day (row) and in which location (column) the game is played
- Pair value 23 = row 4, column 2
- element constraint to link the variables
- Two projections from $D \times L$ space onto $D$ and $L$
Mapping cells to rows and columns

<table>
<thead>
<tr>
<th>City 1</th>
<th>City 2</th>
<th>City 3</th>
<th>City 4</th>
<th>City 5</th>
<th>City 6</th>
<th>City 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Day 2</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>Day 3</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>Day 4</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>Day 5</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
</tr>
<tr>
<td>Day 6</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>41</td>
</tr>
<tr>
<td>Day 7</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
</tr>
</tbody>
</table>

`element(23, [1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 4], 4),`

`element(23, [1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6, 7], 2),`

- This is one common type, a projection
- Another common type is the inverse
  - Link a variable $A \rightarrow B$ to another $B \rightarrow A$
  - Typically used for bijective mappings
  - Built-in `inverse/2`
- Also used: Boolean channeling
  - Link variables $A \rightarrow B$ and $A \times B \rightarrow \{0, 1\}$
  - Built-in `bool_channeling/3`
Two sets of variables (Req 1, 2, 3, 6, by construction)

- Pair variables \( (T \triangle T \rightarrow D \times L) \)
  - \textit{alldifferent} (Req 9)

- Day and Location variables \( (T \triangle T \rightarrow D), (T \triangle T \rightarrow L) \)
  - \textit{gcc} (Req 4, 5)
  - \textit{alldifferent} (Req 7, 8)

- Channeling Constraints
  - \textit{element} projection from pairs onto rows and columns

- Search only on pair variables

Handling of hints (I)

<table>
<thead>
<tr>
<th></th>
<th>City 1</th>
<th>City 2</th>
<th>City 3</th>
<th>City 4</th>
<th>City 5</th>
<th>City 6</th>
<th>City 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td>7, 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 2</td>
<td>2</td>
<td>1, 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 3</td>
<td>7</td>
<td></td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 4</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Day 5</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Day 6</td>
<td></td>
<td></td>
<td>5, 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 7</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1, 3</td>
<td></td>
</tr>
</tbody>
</table>

- This value (17) can not be used by pairs not involving team 8
- One of the pairs involving team 8 must use this value (17)
Handling of hints (II)

<table>
<thead>
<tr>
<th>Day</th>
<th>City 1</th>
<th>City 2</th>
<th>City 3</th>
<th>City 4</th>
<th>City 5</th>
<th>City 6</th>
<th>City 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td>7, 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 2</td>
<td>2</td>
<td>1, 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 3</td>
<td>7</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 4</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Day 5</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Day 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5, 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 7</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1, 3</td>
</tr>
</tbody>
</table>

- The pair involving teams 5 and 7 must take value 5, fixes variable

Problem Data

\[
\text{hint}(1, 8, [2-[8], 5-[5, 7], 8-[2], 9-[1, 5], 15-[7], 17-[8], 26-[2], 27-[5], 28-[1], 29-[8], 34-[1], 39-[4, 5], 43-[4], 47-[1, 3])).
\]
Main Program

top(Problem,L):-
    hint(Problem,N,Hints),
    N1 is N-1,
    N2 is N//2,
    NrVars is N*N1//2,
    SizeDomain is N1*N1,
    length(L,NrVars),
    L :: 1..SizeDomain,
    create_pairs(N,Contains,Names),
    ic_global_gac:alldifferent(L),
    process_hints(L,Contains,Hints),
...

Main Program (continued)

project_row_cols(L,N1,Rows,Cols),
limit(Rows,N2,N1),
limit(Cols,N2,N1),
separate(Contains,Rows,N,SplitRows),
separate(Contains,Cols,N,SplitCols),
(foreach(K,SplitRows) do
    ic_global_gac:alldifferent(K)
),
(foreach(K,SplitCols) do
    ic_global_gac:alldifferent(K)
),
search(L,0,input_order,indomain,
    complete,[]).
Create Pairs and Names

create_pairs(N,Contains,Names):-
   (for(I,1,N-1),
    fromto(Names,A1,A,[]),
    fromto(Contains,B1,B,[]),
    param(N) do
      (for(J,I+1,N),
       fromto(A1,[Name|AA],AA,A),
       fromto(B1,[I-J|BB],BB,B),
       param(I) do
         concat_string([I,J],Name)
      )
   ).

Projecting Rows and Columns

project_row_cols(L,N,Rows,Cols):-
   generate_tables(N,RowTable,ColTable),
   (foreach(X,L),
    foreach(R,Rows),
    foreach(C,Cols),
    param(RowTable,ColTable) do
      element(X,RowTable,R),
      element(X,ColTable,C)
   ).
generate_tables(N,RowTable,ColTable):-
  (for(I,1,N),
   fromto(RowTable,A1,A,[]),
   fromto(ColTable,B1,B,[]),
   param(N) do
     (for(J,1,N),
      fromto(A1,[I|AA],AA,A),
      fromto(B1,[J|BB],BB,B),
      param(I) do
        true
     )
  )
).

separate(Contains,Rows,Values,SplitRows):-
  (for(Value,1,Values),
   foreach(SplitRow,SplitRows),
   param(Contains,Rows) do
     (foreach(A-B,Contains),foreach(V,Rows),
      fromto([],R,R1,SplitRow),
      param(Value) do
        (memberchk(Value,[A,B]) ->
         R1 = [V|R]
        ;
         R1 = R
        )
     )
  ).
Set up gcc constraint

\[
\text{limit}(L, \text{Bound}, \text{Values}) :\neg \\
\quad (\text{for}(I, 1, \text{Values}), \\
\quad \text{foreach}(\text{gcc}(	ext{Bound}, \text{Bound}, I), \text{Pattern}), \\
\quad \text{param}(\text{Bound}) \text{ do true} \\
\quad ), \\
\quad \text{gcc}(\text{Pattern}, L).
\]

Setting up hints

\[
\text{process\_hints}(L, \text{Contains}, \text{Hints}) :\neg \\
\quad (\text{foreach}(\text{Pos}-\text{Values}, \text{Hints}), \\
\quad \text{param}(L, \text{Contains}) \text{ do } \\
\quad \text{process\_hint}(\text{Pos}, \text{Values}, L, \text{Contains}) \\
\quad ).
\]

\[
\text{process\_hint}(\text{Pos}, [A,B], L, \text{Contains}) :\neg \% \text{ clause 1} \\
\quad !, \\
\quad \text{match\_hint}(A-B, \text{Contains}, L, X), \\
\quad X \#\neq \text{Pos}.
\]
Setting up hints

process_hint(Pos, [Value], L, Contains):— % clause 2
(foreach(X, L),
foreach(A-B, Contains),
fromto([], R, R1, Required),
param(Pos, Value) do
  (not_mentioned(A, B, Value) ->
   X \= Pos,
   R1 = R
  ;
   R1 = [X|R]
  ),
occurrences(Pos, Required, 1).

not_mentioned(A, B, V):—
  A \= V,
  B \= V.

match_hint(H, [H|_], [X|_], X):—
!.
match_hint(H, [__|T], [__|R], X):—
  match_hint(H, T, R, X).
Before Search

Values

Solution
Search Tree with input order

How to improve?

- Try different search strategy
- Use `first_fail` dynamic variable selection
Observation

- It does not work
- Search tree is slightly larger than before!
## Redundant Modelling

### Adding value index Channeling

#### Improving Handling of Hints

### Missing Propagation

<table>
<thead>
<tr>
<th>Day</th>
<th>City 1</th>
<th>City 2</th>
<th>City 3</th>
<th>City 4</th>
<th>City 5</th>
<th>City 6</th>
<th>City 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td>8</td>
<td></td>
<td></td>
<td>7, 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 2</td>
<td>2, 1, 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 3</td>
<td>7, 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 4</td>
<td>2, 5, 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 5</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 6</td>
<td>5, 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 7</td>
<td>4</td>
<td>1, 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```

<table>
<thead>
<tr>
<th>Day 1</th>
<th>City 1</th>
<th>City 2</th>
<th>City 3</th>
<th>City 4</th>
<th>City 5</th>
<th>City 6</th>
<th>City 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Day 2</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>Day 3</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>Day 4</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
</tr>
<tr>
<td>Day 5</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
</tr>
<tr>
<td>Day 6</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>41</td>
<td>42</td>
</tr>
<tr>
<td>Day 7</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
</tr>
</tbody>
</table>
```
Missing Propagation

<table>
<thead>
<tr>
<th>Day 1</th>
<th>City 1</th>
<th>City 2</th>
<th>City 3</th>
<th>City 4</th>
<th>City 5</th>
<th>City 6</th>
<th>City 7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 2</td>
<td>2</td>
<td>1, 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 3</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 4</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Day 5</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 6</td>
<td></td>
<td></td>
<td></td>
<td>5, 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 7</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>1, 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Day 1</th>
<th>City 1</th>
<th>City 2</th>
<th>City 3</th>
<th>City 4</th>
<th>City 5</th>
<th>City 6</th>
<th>City 7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 2</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>Day 3</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>Day 4</td>
<td>20</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
</tr>
<tr>
<td>Day 5</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
</tr>
<tr>
<td>Day 6</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>41</td>
<td>42</td>
</tr>
<tr>
<td>Day 7</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
</tr>
</tbody>
</table>

Helmut Simonis  Modelling in CP  101

Helmut Simonis  Modelling in CP  102
Missing Propagation

<table>
<thead>
<tr>
<th>Day 1</th>
<th>City 1</th>
<th>City 2</th>
<th>City 3</th>
<th>City 4</th>
<th>City 5</th>
<th>City 6</th>
<th>City 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 2</td>
<td>8</td>
<td>7, 5</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Day 3</td>
<td>2</td>
<td>1, 3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Day 4</td>
<td>7</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Day 5</td>
<td>8</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Day 6</td>
<td>4</td>
<td>5, 4</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Day 7</td>
<td>4</td>
<td>1, 3</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Day 1</th>
<th>City 1</th>
<th>City 2</th>
<th>City 3</th>
<th>City 4</th>
<th>City 5</th>
<th>City 6</th>
<th>City 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 2</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>Day 3</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>Day 4</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
</tr>
<tr>
<td>Day 5</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
</tr>
<tr>
<td>Day 6</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
</tr>
<tr>
<td>Day 7</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
</tr>
</tbody>
</table>
**Missing Propagation**

<table>
<thead>
<tr>
<th></th>
<th>City 1</th>
<th>City 2</th>
<th>City 3</th>
<th>City 4</th>
<th>City 5</th>
<th>City 6</th>
<th>City 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td>8</td>
<td></td>
<td></td>
<td>7,5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 2</td>
<td>2, 1,5</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 3</td>
<td>7</td>
<td></td>
<td></td>
<td>2, 5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 4</td>
<td>8</td>
<td></td>
<td></td>
<td>5, 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 5</td>
<td>4</td>
<td></td>
<td></td>
<td>1, 3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Why is this?**

- Constraints involved:
  - $gcc$ constraint on row: four variables can use values from this row
  - four occurrence constraints for hints: One of the variables must take this value
- No interaction between constraints, only between constraints and variables
- We do not detect that value 1 can not be used
- Eventual solution respects condition, model is correct
- We are concerned about propagation, not just correctness

**Remarks:**
- Constraints involved: gcc constraint on row: four variables can use values from this row
- four occurrence constraints for hints: One of the variables must take this value
- No interaction between constraints, only between constraints and variables
- We do not detect that value 1 can not be used
- Eventual solution respects condition, model is correct
- We are concerned about propagation, not just correctness
Adding redundant constraints

- Add constraints which do more propagation, but do not affect solutions
- Lead to smaller search tree, hopefully faster solution
- Introduction requires understanding of (lack of) propagation
- Visualization is key to detect missing propagation

First Attempt: Adding 0/1 viewpoint

- \( \text{Day} \times \text{Location} \) matrix of 0/1 variables
- Indicates if there is a game on this day at this location
- Row/column sums: 4 games in each row/column
- Hint given for cell: Game variable is 1
Channeling Constraint

- Link pair variables $P_i$ to 0/1 variables $Y_j$ as *value-index*
- $(\exists i \text{ s.t. } P_i = v) \iff Y_v = 1$
- Propagation:
  - $P_i = v \Rightarrow Y_v = 1$
  - $Y_v = 0 \Rightarrow \forall i: P_i \neq v$
  - $(\forall i: v \notin d(P_i)) \Rightarrow Y_v = 0$
  - $Y_v = 1 \Rightarrow \text{occurrence}(V, P_1...P_n, N), N \geq 1$

```
value_set_channeling(L,Hints):-
  dim(Matrix,[7,7]),
  Matrix[1..7,1..7] :: 0..1,
  flatten_array(Matrix,ValueSet),
  value_set_channel(L,ValueSet,1),
  (foreach(K-_,Hints),param(Matrix) do
    coor(K,I,J),
    subscript(Matrix,[I,J],1))
).```

Helmut Simonis  Modelling in CP  109

Added Program
Impact of Redundant Constraints

Before Search

Without

With value index channeling

Modelling in CP 111

Modelling in CP 112
Solution

Search Tree
Problem
Model
Search
Redundant Modelling

Still Missing Propagation

<table>
<thead>
<tr>
<th>City 1</th>
<th>City 2</th>
<th>City 3</th>
<th>City 4</th>
<th>City 5</th>
<th>City 6</th>
<th>City 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Day 2</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>Day 3</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>Day 4</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>Day 5</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
</tr>
<tr>
<td>Day 6</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>41</td>
</tr>
<tr>
<td>Day 7</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
</tr>
</tbody>
</table>

Game 12 can not be played on day 1 at locations 5 or 6

Helmut Simonis
Modelling in CP

115
Game 12 can not be played on day 1 at locations 5 or 6

Game 12 can not be played on day 1 at locations 5 or 6
Game 12 can not be played on day 1 at locations 5 or 6.
Our model does not deal well with hints

- Preset game is ok, leads to variable assignment
- Preset team is weak, adds new constraint
- As there is no interaction of this constraint with the other constraints, there is no initial domain restriction
- Model is correct, but lazy

Second Attempt: Improving the handling of hints

<table>
<thead>
<tr>
<th>Day</th>
<th>City 1</th>
<th>City 2</th>
<th>City 3</th>
<th>City 4</th>
<th>City 5</th>
<th>City 6</th>
<th>City 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td>7, 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 2</td>
<td>2</td>
<td>1, 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 3</td>
<td>7</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 4</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Day 5</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Day 6</td>
<td></td>
<td></td>
<td></td>
<td>5, 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 7</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

- This value can not be used by pairs not involving team 8
- One of the pairs involving team 8 must use this value
- These values can not be used by any pair involving team 8
Redundant Constraints

- Red value can not be used by pairs not involving team 8
  - disequalities
- One of the pairs involving team 8 must use red value
  - occurrences(gcc) constraint
- Yellow values can not be used by any pair involving team 8
  - disequalities

![City Map]

### Added Program

```
improved_hint(Pos,[Value],L,Contains):-
    (foreach(X,L),foreach(A-B,Contains),
    fromto([],R,R1,Required),
    param(Pos,Value) do
        (not_mentioned(A,B,Value) ->
            X #\= Pos,R1 = R
        ;
            R1 = [X|R]
        ),
    occurrences(Pos,Required,1),
    excluded_locations(Pos,Excluded),
    exclude_values(Required,Excluded).
```

Helmut Simonis  Modelling in CP  124
excluded_locations(Pos, Excluded) :-
    coor(Pos, X, Y),
    (for(I, 1, 7),
        fromto([], A, A1, E1),
        param(Y, Pos) do
            coor(K, I, Y),
            (Pos = K ->
                A1 = A
            ;
                A1 = [K|A]
            )
    ),
    ...

... (for(J, 1, 7),
    fromto(E1, A, A1, Excluded),
    param(X, Pos) do
        coor(K, X, J),
        (Pos = K ->
            A1 = A
        ;
            A1 = [K|A]
        )
    ).
Added Program

```prolog
exclude_values(Vars,Values):-
    (foreach(X,Vars),
        param(Values) do
            (foreach(Value,Values),
                param(X) do
                    X \= Value
            )
    ).
```

Before Search

![Before Search Diagram]
Impact of improved hint handling

With index set channeling

Improved Hints

Observation

- We don’t need the value index channeling
- It is subsumed by the improved hint treatment
- Always worthwhile to check if constraints are still required after modifying model
Conclusions

- Many ways of modelling even simple problems
- Selection of “best” model difficult
  - Depends on constraints available
  - Often needs experimentation
- How do we measure if one model is “better” than another?
  - Execution time?
  - Size of search tree?
  - Scalability?
- Definition of variables is key
- Explore choices by considering mapping operators

Channeling - Combining viewpoints
- Express some constraints in one, others in second viewpoint
- Channeling constraints to link the viewpoints
- Decide which model to use for search

Redundant Constraints - Improving constraint propagation
- Constraints are logically implied by other constraints
- Provide more propagation to reduce search space
More Information


Barbara Smith. Modelling.


What we want to introduce

- Using lex constraints to remove symmetries
- BIBD - Balanced Incomplete Block Designs
- Only one of many ways to deal with symmetry in problems
Problem Definition

BIBD (Balanced Incomplete Block Design)

A BIBD is defined as an arrangement of \( v \) distinct objects into \( b \) blocks such that each block contains exactly \( k \) distinct objects, each object occurs in exactly \( r \) different blocks, and every two distinct objects occur together in exactly \( \lambda \) blocks. A BIBD is therefore specified by its parameters \((v, b, r, k, \lambda)\).

Motivation: Test Planning

Consider a new release of some software with \( v \) new features. You want to regression test the software against combinations of the new features. Testing each subset of features is too expensive, so you want to run \( b \) tests, each using \( k \) features. Each feature should be used \( r \) times in the tests. Each pair of features should be tested together exactly \( \lambda \) times. How do you arrange the tests?
Another way of defining a BIBD is in terms of its incidence matrix, which is a binary matrix with \( v \) rows, \( b \) columns, \( r \) ones per row, \( k \) ones per column, and scalar product \( \lambda \) between any pair of distinct rows.

A binary \( v \times b \) matrix. Entry \( V_{ij} \) states if item \( i \) is in block \( j \).
- Sum constraints over rows, each sum equal \( r \)
- Sum constraints over columns, each sum equal \( k \)
- Scalar product between any pair of rows, the product value is \( \lambda \).
Top Level Program

:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).

top:-
    bibd(6,10,5,3,2,Matrix), writeln(Matrix).

bibd(V,B,R,K,L,Matrix):-
    model(V,B,R,K,L,Matrix), Set up model
    extract_array(row, Matrix, List), Get list
    search(List, 0, input_order, indomain, complete, []). Search

Constraint Model

model(V,B,R,K,L,Matrix, Method):-
    dim(Matrix, [V,B]), Define Binary Matrix
    Matrix[1..V, 1..B] :: 0..1,
    (for(I,1,V), param(Matrix,B,R) do
        sumlist(Matrix[I, 1..B], R)
    ), Row Sum = R
    (for(J,1,B), param(Matrix, V,K) do
        sumlist(Matrix[1..V,J], K)
    ), Column Sum = K
    (for(I,1,V-1), param(Matrix, V,B,L) do
        (for(I1,I+1,V), param(Matrix, I,B,L) do
            scalar_product(Matrix[I, 1..B], Matrix[I1, 1..B], L)
        )
    )
scalar_product(XVector, YVector, V):-
  collection_to_list(XVector, XList),
  collection_to_list(YVector, YList), \( \Rightarrow \) Get lists
  foreach(X, XList), \( \Rightarrow \) Iterate over lists
  foreach(Y, YList), \( \Rightarrow \) ...in parallel
  fromto(0, A, A1, Term) do \( \Rightarrow \) Build term
    A1 = A + X * Y \( \Rightarrow \) Construct term
  ),
  eval(Term) #\( \neq \) V. \( \Rightarrow \) State Constraint

Search Routine

- Static variable order
- First fail does not work for binary variables
- Enumerate variables by row
- Use utility predicate `extract_array/3`
- Assign with `indomain`, try value 0, then value 1
- Use simple `search` call
Basic Model - First Solution

Finding all Solutions - Search Tree 200 Nodes
Observation

- Surprise! There are many solutions
Search Tree 400 Nodes

Search Tree 500 Nodes
Search Tree 1000 Nodes

Search Tree 2000 Nodes
Problem

- There are too many solutions to collect in a reasonable time
- Most of these solutions are very similar
- If you take one solution and
  - exchange two rows
  - and/or exchange two columns
- ... you have another solution
- Can we avoid exploring them all?

Symmetry Breaking Techniques

- **Remove all symmetries**
  - Reduce the search tree as much as possible
  - May be hard to describe all symmetries
  - May be expensive to remove symmetric parts of tree
- **Remove some symmetries**
  - Search is not reduced as much
  - May be easier to find some symmetries to remove
  - Cost can be low
Symmetry Breaking Techniques

- Symmetry removal by forcing partial, initial assignment
  - Easy to understand
  - Rather weak, does not affect search
- Symmetry removal by stating constraints
  - Removing all symmetries may require exponential number of constraints
  - Can conflict with search strategies
- Symmetry removal by controlling search
  - At each node, decide if it needs to be explored
  - Can be expensive to check

Solution used here: Double Lex

- Partial symmetry removal by adding lexicographical ordering constraints
- Our problem has full row and column symmetries
- Any permutation of rows and/or columns leads to another solution
- Idea: Order rows lexicographically
- Rows must be different from each other, strict order on rows
- Columns might be identical, non strict order on columns
  - This can be improved in some cases
- Constraints only between adjacent rows(columns)
Added Constraints

\[
dim(\text{Matrix}, [V,B]), \\
(\text{for}(I,1,V-1), \\
\quad \text{param}(\text{Matrix},B) \text{ do} \\
\qquad I_1 \text{ is } I+1, \\
\qquad \text{lex}_\text{less}(\text{Matrix}[I_1,1..B], \text{Matrix}[I,1..B])) \\
), \Rightarrow \text{ Row lex constraints} \\
(\text{for}(J,1,B-1), \\
\quad \text{param}(\text{Matrix},V) \text{ do} \\
\qquad J_1 \text{ is } J+1, \\
\qquad \text{lex}_\text{leq}(\text{Matrix}[1..V,J_1], \text{Matrix}[1..V,J])) \\
), \Rightarrow \text{ Column lex constraints}
\]

Using two global constraints

- \text{lex}_\text{leq}(\text{List1}, \text{List2})
  - \text{List1} \text{ is lexicographical smaller than or equal to } \text{List2}
  - Achieves domain consistency

- \text{lex}_\text{less}(\text{List1}, \text{List2})
  - \text{List1} \text{ is lexicographical smaller than } \text{List2}
  - Achieves domain consistency
Example propagation \( \text{lex}_{\text{less}} \)

Before

\[
\begin{bmatrix}
2, \\
Y_1 \in \{0, 1, 2\}, \\
2,
\end{bmatrix}
\begin{bmatrix}
X_2 \in \{1, 3, 4\}, \\
1, \\
1,
\end{bmatrix}
\begin{bmatrix}
X_3 \in \{1, 2, 3\}, \\
Y_3 \in \{0, 1, 2, 3\}, \\
2,
\end{bmatrix}
\begin{bmatrix}
X_4 \in \{1, 2\}, \\
Y_4 \in \{0, 1\}, \\
1,
\end{bmatrix}
\begin{bmatrix}
X_5 \in \{3, 4\}, \\
Y_5 \in \{0, 1\}, \\
1,
\end{bmatrix}
\]

After

\[
\begin{bmatrix}
2, \\
Y_3 \in \{2, 3\}, \\
2,
\end{bmatrix}
\begin{bmatrix}
X_3 \in \{1, 2\}, \\
Y_3 \in \{2, 3\}, \\
1,
\end{bmatrix}
\begin{bmatrix}
X_4 \in \{1, 2\}, \\
Y_4 \in \{0, 1\}, \\
1,
\end{bmatrix}
\begin{bmatrix}
X_5 \in \{3, 4\}, \\
Y_5 \in \{0, 1\}, \\
1,
\end{bmatrix}
\]

Complete Search Tree with Double Lex
- Enormous reduction in search space
- We are solving a different problem!
- Not just good for finding all solutions, also for first solution!
- Value choice not optimal for finding first solution
- There is a lot of very shallow backtracking, can we avoid that?

**Effort for First Solution**

**Basic Model**

**With double Lex**
Alternative Value Order

:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).

top:-
    bibd(6,10,5,3,2,Matrix), writeln(Matrix).

bibd(V,B,R,K,L,Matrix):-
    model(V,B,R,K,L,Matrix),
    extract_array(row,Matrix,List),
    search(L,0,input_order,
        indomain_max," Start with 1 complete,[]").
Observation

- First solution is found more quickly
- Size of tree for all solutions unchanged
- Value order does not really affect search space when exploring all choices!

Effort for All Solutions

Assign 0, then 1

Assign 1, then 0
Row- or Column- wise Assignment?

- We did assign matrix by row, why?
- What happens if we assign variables by column?
Observation

- Good, but not as good as row order
- Value choice (0/1) or (1/0) does not really matter even for first solution
- Changing the variable selection does affect size of search space, even for all solutions

Effort for All Solutions

By Row  
By Column
Possible Explanations

- There are fewer rows than columns
- Strict lex constraints on rows, but not on columns
  - More impact of first row
- Needs better understanding

<table>
<thead>
<tr>
<th>v</th>
<th>b</th>
<th>r</th>
<th>k</th>
<th>λ</th>
<th>asym</th>
<th>lcx²</th>
<th>STAB</th>
<th>lcx² + SNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>24</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>36</td>
<td>5,987</td>
<td>344</td>
<td>311</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>46</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td>21</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>26</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>12,800</td>
<td>21</td>
<td>101</td>
</tr>
<tr>
<td>7</td>
<td>35</td>
<td>15</td>
<td>3</td>
<td>5</td>
<td>109</td>
<td>33,304</td>
<td>542</td>
<td>282</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>7</td>
<td>7</td>
<td>3</td>
<td>5</td>
<td>118</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>21</td>
<td>21</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>30</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>864</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>18</td>
<td>9</td>
<td>5</td>
<td>4</td>
<td>21</td>
<td>8,031</td>
<td>302</td>
<td>139</td>
</tr>
<tr>
<td>7</td>
<td>42</td>
<td>18</td>
<td>3</td>
<td>6</td>
<td>418</td>
<td>250,878</td>
<td>2,334</td>
<td>1,247</td>
</tr>
<tr>
<td>22</td>
<td>22</td>
<td>7</td>
<td>7</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>49</td>
<td>21</td>
<td>3</td>
<td>7</td>
<td>1,508</td>
<td>1,460,332</td>
<td>8,821</td>
<td>4,353</td>
</tr>
<tr>
<td>8</td>
<td>28</td>
<td>14</td>
<td>4</td>
<td>6</td>
<td>2,310</td>
<td>2,058,523</td>
<td>17,890</td>
<td>11,424</td>
</tr>
<tr>
<td>19</td>
<td>19</td>
<td>9</td>
<td>9</td>
<td>4</td>
<td>6</td>
<td>6,520</td>
<td>71</td>
<td>17</td>
</tr>
<tr>
<td>10</td>
<td>30</td>
<td>9</td>
<td>3</td>
<td>2</td>
<td>960</td>
<td>724,662</td>
<td>24,563</td>
<td>15,169</td>
</tr>
<tr>
<td>31</td>
<td>31</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>864</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>56</td>
<td>24</td>
<td>3</td>
<td>8</td>
<td>5,413</td>
<td>6,941,124</td>
<td>32,038</td>
<td>14,428</td>
</tr>
<tr>
<td>9</td>
<td>36</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>22,521</td>
<td>14,843,772</td>
<td>315,531</td>
<td>85,605</td>
</tr>
<tr>
<td>7</td>
<td>63</td>
<td>27</td>
<td>3</td>
<td>9</td>
<td>?</td>
<td>28,079,394</td>
<td>105,955</td>
<td>43,259</td>
</tr>
<tr>
<td>15</td>
<td>35</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>80</td>
<td>32,127,296</td>
<td>6,782</td>
<td>35,183</td>
</tr>
<tr>
<td>21</td>
<td>28</td>
<td>8</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>26</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>2461</td>
<td>3,664,243</td>
<td>83,337</td>
<td>31,323</td>
</tr>
<tr>
<td>11</td>
<td>22</td>
<td>10</td>
<td>5</td>
<td>4</td>
<td>4393</td>
<td>6,143,408</td>
<td>106,522</td>
<td>32,908</td>
</tr>
<tr>
<td>12</td>
<td>22</td>
<td>11</td>
<td>6</td>
<td>5</td>
<td>?</td>
<td>?</td>
<td>228,146</td>
<td>76,572</td>
</tr>
<tr>
<td>25</td>
<td>25</td>
<td>9</td>
<td>9</td>
<td>3</td>
<td>?</td>
<td>?</td>
<td>17,016</td>
<td>1,355</td>
</tr>
<tr>
<td>16</td>
<td>24</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>?</td>
<td>?</td>
<td>769,482</td>
<td>76,860</td>
</tr>
</tbody>
</table>
Scalability

- $\text{lex}^2$ good, but not good enough
- Still leaves too many symmetries to explore
- Better techniques in the literature
  - STAB, group theory based, Puget 2003.
  - SBNO, local search based domination check, Prestwich, 2008.

Do we need binary variables?

- The 0/1 model does very little propagation
- Consider a model with finite domain variables
- Each of $b$ blocks consists of $k$ variables ranging over $v$ values
- The values in a block must be alldifferent (ordered)
- Each value can occur $r$ times
- Scalar product more difficult
- Even better expressed with finite set variables
Conclusions

- Symmetry breaking can have huge impact on model
- Mainly works for pure problems
- Partial symmetry breaking with additional constraints
- Double lex for row/column symmetries
- Only one variant of many symmetry breaking techniques

More Information


What we want to introduce

- Problem decomposition
  - Decide which problem to solve
  - Not always required to solve complete problem in one go
- Modelling with bin packing
- Customized search routines can bring dramatic improvements
- Understanding what is happening important to find improvements

Problem Definition

Progressive Party

The problem is to timetable a party at a yacht club. Certain boats are to be designated hosts, and the crews of the remaining boats in turn visit the host boats for several successive half-hour periods. The crew of a host boat remains on board to act as hosts while the crew of a guest boat together visits several hosts. Every boat can only host a limited number of guests at a time (its capacity) and crew sizes are different. The party lasts for 6 time periods. A guest boat cannot not revisit a host and guest crews cannot meet more than once. The problem facing the rally organizer is that of minimizing the number of host boats.
A Further Decomposition

High Level Problem Decomposition

- Phase 1: Select minimal set of host boats
  - Manually
- Phase 2: Create plan to assign guest boats to hosts in multiple periods
  - Done as a constraint program

<table>
<thead>
<tr>
<th>Boat</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>6</td>
<td>8</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Crew</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Boat</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>8</td>
<td>12</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Crew</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Boat</th>
<th>29</th>
<th>30</th>
<th>31</th>
<th>32</th>
<th>33</th>
<th>34</th>
<th>35</th>
<th>36</th>
<th>37</th>
<th>38</th>
<th>39</th>
<th>40</th>
<th>41</th>
<th>42</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Crew</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
### Idea

- Decompose problem into multiple, simpler sub problems
- Solve each sub problem in turn
- Provides solution of complete problem
- Challenge: How to decompose so that good solutions are obtained?
- How to show optimality of solution?

### Selecting Host Boats

- Some additional side constraints
  - Some boats must be hosts
  - Some boats may not be hosts
- Reason on total or spare capacity
- No solution with 12 boats (with side constraints)
Solution to Phase 1

- Select boats 1 to 12 and 14 as hosts
- Many possible problem variants by selecting other host boats

Phase 2 Sub-problem

- Host boats and their capacity given
- Ignore host teams, only consider free capacity
- Assign guest teams to host boats
Model

- Assign guest boats to hosts for each time period
- Matrix (size \( NrGuests \times NrPeriods \)) of domain variables \( x_{ij} \)
- Variables range over possible hosts 1..\( NrHosts \)

Constraints

- Each guest boat visits a host boat at most once
- Two guest boats meet at most once
- All guest boats assigned to a host in a time period fit within spare capacity of host boat
Each guest visits a host at most once

- The variables for a guest and different time periods must be pairwise different
- `alldifferent` constraint on rows of matrix
- `alldifferent`\(\{x_{ij}|1 \leq j \leq NrPeriods\}\)

Two guests meet at most once

- The variables for two guests can have the same value for at most one time period
- Constraints on each pair of rows in matrix
- \(x_{i_1j} = x_{i_2j}, i_1 \neq i_2 \Rightarrow x_{i_1k} \neq x_{i_2k} 1 \leq k \leq NrPeriods, k \neq j\)
All guests assigned to a host in a time period fit within spare capacity of host boat

- Capacity constraint expressed as bin packing for each time period
- Each host boat is a bin with capacity from 0 to its unused capacity
- Each guest is an item to be assigned to a bin
- Size of item given by crew size of guest boat

Global constraint

\[ \text{bin\_packing} \text{(Assignment, Sizes, Capacity)} \]

- Items of different sizes are assigned to bins
- Assignment of item modelled with domain variable (first argument)
- Size of items fixed: integer values (second argument)
- Each bin may have a different capacity
- Capacity of each bin given as a domain variable (third argument)
Main Program

top:-
    top(10,6).

top(Problem,Size):-
    problem(Problem,Hosts,Guests),
    model(Hosts,Guests,Size,Matrix),
    writeln(Matrix).

Data

problem(10,
    [10,10,9,8,8,8,8,8,8,7,6,6,4],
    [7,6,5,5,5,4,4,4,4,4,4,4,4,4,3,
     3,2,2,2,2,2,2,2,2,2,2,2,2,2,2]).
Creating Variables

```prolog
model(Hosts, Guests, NrPeriods, Matrix):-
    length(Hosts, NrHosts),
    length(Guests, NrGuests),
    dim(Matrix, [NrGuests, NrPeriods]),
    Matrix[1..NrGuests,1.. NrPeriods] :: 1..NrHosts,
    ...  
```

Setting up `alldifferent` constraints

```prolog
...  
(for(I, 1, NrGuests),
    param(Matrix, NrPeriods) do
    ic:alldifferent(Matrix[I,1..NrPeriods])
),
...```

Helmut Simonis  Modelling in CP  196
Setting up bin_packing constraints

... 
(for(J,1,NrPeriods),
  param(Matrix,NrGuests, Guests, Hosts) do
    make_bins(Hosts,Bins),
    bin_packing(Matrix[1..NrGuests,J],
                Guests,Bins)
  ),
...

Each pair of guests meet atmost once

... 
(for(I,1,NrGuests-1),
  param(Matrix,NrGuests,NrPeriods) do
    (for(I1,I+1,NrGuests),
     param(Matrix,NrPeriods,I) do
      card_leq(Matrix[I,1..NrPeriods],
                Matrix[I1,1..NrPeriods],1)
     )
  ),
...
Call search

... extract_array(col, Matrix, List), assign(List).

Make Bin variables

make_bins(HostCapacity, Bins) :-
    (foreach(Cap, HostCapacity),
     foreach(B, Bins) do
        B :: 0..Cap
    ).
Each pair of guests meet atmost once

card_leq(Vector1,Vector2,Card):-
collection_to_list(Vector1,List1),
collection_to_list(Vector2,List2),
(foreach(X,List1),
 foreach(Y,List2),
 fromto(0,A,A+B,Term) do
   #=(X,Y,B)
),
 eval(Term) #=< Card.

assign(List):-
    search(List,0,input_order,indomain,
       complete,[]).
Naive Search (Compact view)

Naive Search (Zoomed)
Observations

- Not too many wrong choices
- But very deep backtracking required to discover failure
- Most effort wasted in “dead” parts of search tree

First Fail strategy

assign(List):-
    search(List,0,first_fail,indomain,
    complete,[]).
First Fail Search

Helmut Simonis  Modelling in CP  207

First Fail Search (Zoomed)

Helmut Simonis  Modelling in CP  208
Observations

- Assignment not done in row or column mode
- Tree consists of straight parts without backtracking
- ... and nearly fully explored parts

Idea

- Assign variables by time period
- Within one time period, use `first_fail` selection
- Solves bin packing packing for each period completely
- Clearer impact of disequality constraints
- Serial composition of search procedures
Layered Search

assign(Matrix, NrPeriods, NrGuests):-
  (for(J,1,NrPeriods),
   param(Matrix, NrGuests) do
     search(Matrix[1..NrGuests,J], 0,
       first_fail, indomain, complete,[])
  ).
Layered Solution (Zoomed)

Observations

- Deep backtracking for last time period
- No backtracking to earlier time periods required
- Small amount of backtracking at other time periods
Idea

- Use credit based search
- But not for complete search tree
- Loose too much useful work
- Backtrack independently for each time period
- Hope to correct wrong choices without deep backtracking

Credit Based Search

- Explore top of tree completely, based on credit
- Start with fixed amount of credit
- Each node consumes one credit unit
- Split remaining credit amongst children
- When credit runs out, start bounded backtrack search
- Each branch can use only $K$ backtracks
- If this limit is exceeded, jump to unexplored top of tree
assign(Matrix,NrPeriods,NrGuests):-
  (for(J,1,NrPeriods),
    param(Matrix,NrGuests) do
      NSq is NrGuests*NrGuests,
      search(Matrix[1..NrGuests,J],0,
        first_fail,indomain,
        credit(NSq,10),[])
  ).
Layered with Credit Search (Zoomed)

Observations

- Improved search
- Need more sample problems to really understand impact
Idea

- Randomize value selection
- Remove bias picking bins in same order
- Allows to add restart
- When spending too much time without finding solution
- Restart search from beginning
- Randomization will explore other initial assignments
- Do not get caught in “dead” part of search tree

assign(Matrix, NrPeriods, NrGuests):-
    repeat,
    (for(J, 1, NrPeriods),
        param(Matrix, NrGuests) do
            NSq is NrGuests * NrGuests,
            once(search(Matrix[1..NrGuests, J], 0, first_fail, indomain_random,
                credit(NSq, 10), []))
    ),
    !.
Randomized Search

Observations

- Avoids deep backtracking in last time periods
- Perhaps by mixing values more evenly
- Impose fewer disequality constraints for last periods
- Easier to find solution
- Should allow to find solutions with more time periods
Changing time periods

<table>
<thead>
<tr>
<th>Problem</th>
<th>Size</th>
<th>Naive</th>
<th>FF</th>
<th>Layered</th>
<th>Credit</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
<td>0.812</td>
<td>1.453</td>
<td>1.515</td>
<td>0.828</td>
<td>1.922</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>14.813</td>
<td>2.047</td>
<td>2.093</td>
<td>1.219</td>
<td>2.469</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>79.109</td>
<td>3.688</td>
<td>50.250</td>
<td>3.234</td>
<td>3.672</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>141.609</td>
<td>55.156</td>
<td>6.328</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10.281</td>
</tr>
</tbody>
</table>

Observations

- Randomized method is strongest for this problem
- Not always fastest for smaller problem sizes
- Restart required for size 9 problems
- Same model, very different results due to search
- Very similar results for other problem instances
Idea: There is no real effect of including later time periods in constraint model

Only current time period matters

Decomposition: Set up model for one period at a time

<table>
<thead>
<tr>
<th>Old</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bin packing</td>
<td>Bin packing</td>
</tr>
<tr>
<td>Alldifferent</td>
<td>Domain restrictions</td>
</tr>
<tr>
<td>Meet at most once</td>
<td>Disequalities between guest boats</td>
</tr>
</tbody>
</table>
- Guest boats = Nodes
- Host boats = Colors
- Disequality constraints = Edges in graph
Visualization (Time period 3)

Visualization (Time period 4)
Visualization (Time period 5)

Visualization (Time period 6)
Solving the Graph Coloring Problem

- Use disequality constraints
- Weak propagation
- Extract alldifferent constraints
- Edge clique cover problem
- Choice of consistency method
- Use somedifferent global constraint
- Heavy
- Interaction with bin packing constraint

<table>
<thead>
<tr>
<th>Nr</th>
<th>Size</th>
<th>Solved</th>
<th>Min</th>
<th>Max</th>
<th>Avg</th>
<th>Solved</th>
<th>Min</th>
<th>Max</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>100</td>
<td>0.187</td>
<td>0.343</td>
<td>0.226</td>
<td>100</td>
<td>0.33</td>
<td>0.38</td>
<td>0.35</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>100</td>
<td>0.218</td>
<td>0.515</td>
<td>0.271</td>
<td>100</td>
<td>0.39</td>
<td>0.49</td>
<td>0.44</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>100</td>
<td>0.250</td>
<td>2.469</td>
<td>0.382</td>
<td>100</td>
<td>0.50</td>
<td>0.72</td>
<td>0.57</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>100</td>
<td>0.266</td>
<td>9.906</td>
<td>1.253</td>
<td>100</td>
<td>0.74</td>
<td>1.46</td>
<td>1.01</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>100</td>
<td>0.375</td>
<td>136.828</td>
<td>23.314</td>
<td>100</td>
<td>1.47</td>
<td>41.72</td>
<td>4.68</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>100</td>
<td>0.218</td>
<td>2.375</td>
<td>0.624</td>
<td>100</td>
<td>0.37</td>
<td>0.52</td>
<td>0.43</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>100</td>
<td>0.266</td>
<td>3.453</td>
<td>1.117</td>
<td>100</td>
<td>0.47</td>
<td>1.64</td>
<td>0.73</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>100</td>
<td>0.297</td>
<td>15.421</td>
<td>2.348</td>
<td>100</td>
<td>0.75</td>
<td>7.16</td>
<td>2.69</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>100</td>
<td>0.469</td>
<td>107.187</td>
<td>20.719</td>
<td>99</td>
<td>4.41</td>
<td>162.96</td>
<td>33.54</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>100</td>
<td>0.219</td>
<td>3.266</td>
<td>0.551</td>
<td>100</td>
<td>0.37</td>
<td>0.56</td>
<td>0.43</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>100</td>
<td>0.250</td>
<td>3.734</td>
<td>0.889</td>
<td>100</td>
<td>0.49</td>
<td>1.45</td>
<td>0.74</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>100</td>
<td>0.296</td>
<td>21.360</td>
<td>2.005</td>
<td>100</td>
<td>0.84</td>
<td>11.64</td>
<td>2.85</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>100</td>
<td>1.078</td>
<td>173.266</td>
<td>34.774</td>
<td>96</td>
<td>4.41</td>
<td>164.44</td>
<td>40.10</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>100</td>
<td>0.219</td>
<td>9.922</td>
<td>2.443</td>
<td>100</td>
<td>0.39</td>
<td>0.72</td>
<td>0.47</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>100</td>
<td>0.360</td>
<td>25.297</td>
<td>3.531</td>
<td>100</td>
<td>0.55</td>
<td>2.33</td>
<td>0.87</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>100</td>
<td>0.438</td>
<td>53.547</td>
<td>8.848</td>
<td>100</td>
<td>1.23</td>
<td>11.38</td>
<td>3.68</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>63</td>
<td>3.062</td>
<td>494.109</td>
<td>206.324</td>
<td>94</td>
<td>8.35</td>
<td>166.90</td>
<td>59.55</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>100</td>
<td>0.203</td>
<td>7.922</td>
<td>1.498</td>
<td>100</td>
<td>0.53</td>
<td>5.29</td>
<td>1.87</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>100</td>
<td>0.266</td>
<td>28.000</td>
<td>5.889</td>
<td>100</td>
<td>1.77</td>
<td>132.82</td>
<td>29.72</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>100</td>
<td>0.219</td>
<td>15.219</td>
<td>2.147</td>
<td>100</td>
<td>0.58</td>
<td>31.84</td>
<td>2.74</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>100</td>
<td>0.407</td>
<td>64.312</td>
<td>11.328</td>
<td>88</td>
<td>3.24</td>
<td>152.37</td>
<td>56.92</td>
</tr>
</tbody>
</table>


What we want to introduce

- Complex abstractions
- Global constraints about objects described by multiple variables
- Problem: Rectangle packing

Overview

- SICStus Prolog 4.02
- Not competitive in ECLiPSe
- Evaluate different search strategies
- Problem: Packing all squares from $1 \times 1$ to $n \times n$ in smallest rectangle
- Difficulty: Aspect ratio of enclosing rectangle not known
Motivation

- Related to large-scale, real-life problems
  - IC floor planning, PCB design, architectural design
- Mix of feasible and infeasible subproblems
- Amount of slack changes with subproblem
- New results for open problems
  - Packing in Rectangle (N=26, 27)
  - Packing in Square (N=26, 27, 29, 30, 31, 35)
Search for candidate enclosing rectangle
Area must be larger than sum of items to be placed
Search in order of increasing area
and increasing “squareness”
Check each candidate for (in)feasibility until first solution is found
Observation: Only limited number of candidates explored
**Basic Model**

- 2D Objects are not overlapping
- Each item described by tuple \((X, Y, W, H)\)
- Global reasoning much stronger than considering only two objects at a time
- Easy to add side constraints (distance, orientation)
- More general constraint `geost`
Cumulative resource constraint
- Objects are tasks with start $S$, duration $D$ and resource use $R$
- Can the tasks be scheduled so that at each time point resource use does not exceed $L$
- Central to CP success in scheduling

Search Strategies
- naive
- x then y
- disjunctive
- semantic disjunctive
- dual
- interval
- split
- xy interval
**Naive**

- Place items in order of decreasing size
- Fix $X$ and $Y$ value for each item
- Depth-first search to explore search space
- Problem: Large number of alternatives considered

**X then Y**

- Fix $X$ value for all items, before assigning $Y$ values
- Intuition: Once all $X$ values are fixed, the $Y$ values are very constrained
- Ideal: Search tree $N$, not $2 \times N$ levels deep
- Problem: May lead to deep thrashing if propagation too weak
Disjunctive

Y₁ ≥ Y₂ + H₂ (above)
X₂ ≥ X₁ + W₁ (left)

W₂
H₂
X₂, Y₂
Y₂ ≥ Y₁ + H₁ (below)

X₁ ≥ X₂ + W₂ (right)

Key idea: Fixing intervals, not values
Fixing variables to values is too restrictive
Select “area” in which item is placed
Allows items to shift slightly
Restrict domain to intervals
Only at end fix actual values

Interval Based Strategies
Forcing Obligatory Parts

Small Interval  Number of intervals: large

Large Interval  Number of intervals: small

Obligatory parts

Variants

(X) Interval
- Split all $X$ variables into intervals
- Then fix $X$ values
- Then treat all $Y$ variables the same way

Split
- Split $X$ variables into intervals
- Split $Y$ variables into intervals
- Then fix values

XY Interval
- For each item, split $X$ and $Y$ variables into intervals
- Then fix values
Problem
Search Strategies
Results

Strategies Comparison

<table>
<thead>
<tr>
<th>Problem Size</th>
<th>Korf</th>
<th>BlueBlocker</th>
<th>naive</th>
<th>naive gaps</th>
<th>xtheny</th>
<th>disj</th>
<th>semantic</th>
<th>dual</th>
<th>interval</th>
<th>split</th>
<th>xy</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>17</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>18</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>19</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>21</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>22</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>23</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>24</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>25</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>26</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>27</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Impact of Interval Length

Rectangle Placement Overview

<table>
<thead>
<tr>
<th>N</th>
<th>Surface</th>
<th>K</th>
<th>Width</th>
<th>Height</th>
<th>Area</th>
<th>Loss</th>
<th>Time</th>
<th>Clautiaux</th>
<th>Korf</th>
<th>BlueBlocker</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>2109</td>
<td>14</td>
<td>31</td>
<td>69</td>
<td>2139</td>
<td>1.42</td>
<td>00:01</td>
<td>31:33</td>
<td>1:08</td>
<td>1:29</td>
</tr>
<tr>
<td>19</td>
<td>2470</td>
<td>12</td>
<td>47</td>
<td>53</td>
<td>2491</td>
<td>0.85</td>
<td>00:01</td>
<td>72:53:18</td>
<td>8:15</td>
<td>4:11</td>
</tr>
<tr>
<td>20</td>
<td>2870</td>
<td>14</td>
<td>34</td>
<td>85</td>
<td>2890</td>
<td>0.70</td>
<td>00:02</td>
<td>-</td>
<td>13:32</td>
<td>15:03</td>
</tr>
<tr>
<td>21</td>
<td>3311</td>
<td>19</td>
<td>38</td>
<td>88</td>
<td>3344</td>
<td>1.00</td>
<td>00:07</td>
<td>-</td>
<td>1:35:08</td>
<td>1:32:01</td>
</tr>
<tr>
<td>22</td>
<td>3795</td>
<td>15</td>
<td>39</td>
<td>98</td>
<td>3822</td>
<td>0.71</td>
<td>00:51</td>
<td>-</td>
<td>6:46:15</td>
<td>4:51:23</td>
</tr>
<tr>
<td>23</td>
<td>4324</td>
<td>19</td>
<td>64</td>
<td>68</td>
<td>4352</td>
<td>0.65</td>
<td>03:58</td>
<td>-</td>
<td>36:54:50</td>
<td>29:03:49</td>
</tr>
<tr>
<td>24</td>
<td>4900</td>
<td>18</td>
<td>56</td>
<td>88</td>
<td>4928</td>
<td>0.57</td>
<td>05:56</td>
<td>-</td>
<td>213:33:00</td>
<td>146:38:48</td>
</tr>
<tr>
<td>25</td>
<td>5525</td>
<td>17</td>
<td>43</td>
<td>129</td>
<td>5547</td>
<td>0.40</td>
<td>40:38</td>
<td>-</td>
<td>-</td>
<td>see paper</td>
</tr>
<tr>
<td>26</td>
<td>6201</td>
<td>21</td>
<td>70</td>
<td>89</td>
<td>6230</td>
<td>0.47</td>
<td>03:41:43</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>27</td>
<td>6930</td>
<td>21</td>
<td>47</td>
<td>148</td>
<td>6956</td>
<td>0.38</td>
<td>11:30:02</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
### Optimal Solution (N=27)

| 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

### Packing In Square

- Pack all squares $1 \times 1$ to $N \times N$ into smallest square
- Fewer candidates to check, start with lower bound
  
  \[
  \lceil \sqrt{\sum_{i=1}^{N} i^2} \rceil
  \]
- Often, lower bound is reached $\implies$ optimal
- More slack in optimal solution than for rectangle packing
- Improved symmetry breaking possible
Optimal Solution for Packing in Square

<table>
<thead>
<tr>
<th>Problem Size</th>
<th>26</th>
<th>27</th>
<th>29</th>
<th>30</th>
<th>31</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal Solution</td>
<td>80</td>
<td>84</td>
<td>93</td>
<td>98</td>
<td>103</td>
<td>123</td>
</tr>
<tr>
<td>$T_{opt}$</td>
<td>12:26</td>
<td>00:04</td>
<td>11:06</td>
<td>2:07</td>
<td>00:18</td>
<td>1:10:07</td>
</tr>
<tr>
<td>$T_{proof}$</td>
<td>1:25:22</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Optimal Solution (N=35)
Rectangles may or may not be rotated
- Introduces new degree of freedom
- Does not change constraint model
- Rotation handled as part of search
- Surprising: impact quite small
Dominoes, Optimal Solution (N=22)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Search Strategies</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple items of same size</td>
<td>- Introduces additional symmetries</td>
<td>- Removing symmetry by lexicographic constraints</td>
</tr>
<tr>
<td></td>
<td>- Constrain all $k$ items of same size</td>
<td>- $(x_1, y_1) &lt;<em>{lex} (x_2, y_2) &lt;</em>{lex} \cdots &lt;_{lex} (x_k, y_k)$</td>
</tr>
<tr>
<td></td>
<td>- Works for square or rectangle items</td>
<td></td>
</tr>
</tbody>
</table>
• Rectangle packing problems can be solved by off-the-shelf constraint technology
• Choice of search strategy of prime importance
• Impossible to evaluate model ignoring search
• Out-performs previous best solution by factor 1000

More Information

R. E. Korf.
Optimal rectangle packing: Initial results.

R. E. Korf.
Optimal rectangle packing: New results.

M. D. Moffitt and M. E. Pollack.
Optimal rectangle packing: A meta-CSP approach.
A. Aggoun and N. Beldiceanu.
Extending CHIP in order to solve complex scheduling problems.

N. Beldiceanu and E. Contejean.
Introducing global constraints in CHIP.

N. Beldiceanu and M. Carlsson.
Sweep as a generic pruning technique applied to the non-overlapping rectangles constraint.

N. Beldiceanu, M. Carlsson, and E. Poder.
New filtering for the cumulative constraint in the context of non-overlapping.
A generic geometrical constraint kernel in space and time for handling polymorphic -dimensional objects.

H. Simonis, B. O’Sullivan.
Search Strategies for Rectangle Packing.

Part VI
More Global Constraints
What we want to introduce

- Car sequencing problem
- gcc global cardinality constraint
- sequence constraint
- Search does not always have to be based on original problem variables
- Can be useful to consider additional variables which allow more clever search

Problem Definition

Car Sequencing

We have to schedule a number of cars for production on an assembly line. Each car is of a certain type, and we know how many cars of each type we have to produce. Car types differ in the options they require, i.e. sun-roof, air conditioning. For each option, we have capacity limits on the assembly line, expressed as \( k \) cars out of \( n \) consecutive cars on the line may have some option. Find an assignment which produces the correct number of cars of each type, while satisfying the capacity constraints.
Example (DSV88)

- 100 cars
- 18 types
- 5 options
  - Option 1: 1 out of 2
  - Option 2: 2 out of 3
  - Option 3: 1 out of 3
  - Option 4: 2 out of 5
  - Option 5: 1 out of 5

<table>
<thead>
<tr>
<th>Type</th>
<th>Cars Required</th>
<th>Option 1</th>
<th>Option 2</th>
<th>Option 3</th>
<th>Option 4</th>
<th>Option 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Improved Search Strategy

Assign start time (sequence number) to each car
- 100 variables, each with 100 values
- Handling of car types implicit
- Symmetry breaking for cars of same type (inequalities)?
- Capacity constraints?

Assign car type to each slot on assembly line
- 100 variables, 18 values
- How to control number of cars of each type?
- How to express capacity constraints?
100 variables ranging over car types
- \textit{gcc} constraint to control number of items with same type
- \(5 \times 100\) 0/1 variables indicating use of option for each slot
- \textit{element} constraints to map car types to options used
- \textit{sequence} constraints to enforce limits on each option

\textbf{Reminder:} \texttt{gcc(Pattern, Variables)}

- \texttt{gcc} \textit{global cardinality constraint}
- \texttt{Pattern} is list of terms \texttt{gcc(Low, High, Value)}
- The overall number of variables taking value \texttt{Value} is between \texttt{Low} and \texttt{High}
- Generalization of \texttt{alldifferent}
Reminder: \( \text{element}(X, \text{List}, Y) \)

- List is a list of integers
- The \( X^{th} \) element of List is \( Y \)
- The index starts from 1
- Typical uses:
  - Projection
  - Cost

\( \text{sequence}_{-\text{total}}(\text{Min}, \text{Max}, \text{Low}, \text{High}, K, \text{Vars}) \)

- Variables \( \text{Vars} \) have 0/1 domain
- Between \( \text{Min} \) and \( \text{Max} \) variables have value 1
- For every sub-sequence of length \( K \), between \( \text{Low} \) and \( \text{High} \) variables have value 1
sequence_total  Example

\[\{X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}\} :: 0..1,\]
\[\text{sequence_total}(2, 3, 1, 2, 3,\]
\[\{X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}\}),\]
\[X_1 = 0, X_4 = 0, X_7 = 0, X_{10} = 0\]
Mathematical Equivalent

\[ \text{Vars} = [x_1, x_2, \ldots, x_N] \]
\[ \text{Min} \leq \sum_{1 \leq i \leq N} x_i \leq \text{Max} \]
\[ 1 \leq s \leq N - k + 1 : \quad \text{Low} \leq \sum_{s \leq j \leq s+k-1} x_j \leq \text{High} \]

- Pruning very different when using finite domain inequalities
- Currently no domain consistent implementation of \text{sequence\_total}
- Weaker version \text{sequence} (no global counters) domain consistent
- Currently using decomposition:
  - \text{sequence\_total} = \text{sequence} + \text{gcc} + \text{more}
Main Program

:-module(car).
:-export(top/0).
:-lib(ic).
:-lib(ic_global_gac).

top:-
    problem(Problem),
    model(Problem,L),
    writeln(L).

Structure Definitions

:-local struct(problem(cars,
    models,
    required,
    using_options,
    value_order)).

:-local struct(option(k,
    n,
    index_set,
    total_use)).
Model (Part 1)

model(problem{cars:NrCars,  
    models:NrModels,  
    required:Required,  
    using_options:List,  
    value_order:Ordered},L):-

    length(L,NrCars),  
    L :: 1..NrModels,  
    (foreach(Cnt,Required),  
        count(J,1,_),  
        foreach(gcc(Cnt,Cnt,J),Card) do  
            true  
        ),  
    gcc(Card,L),  
    ...

Model (Part 2)

    (foreach(option{k:K,  
        n:N,  
        index_set:IndexSet,  
        total_use:Total},List),  
        param(L,NrCars) do  
            (foreach(X,L),  
                foreach(B,Binary),  
                param(IndexSet) do  
                    element(X,IndexSet,B)  
                ),  
            sequence_total(Total,Total,0,K,N,Binary)  
        ),  
    search(L,0,input_order,ordered(Ordered),
Data

\[
\text{problem(100,18,}
\begin{align*}
&[5,3,7,1,10,2,11,5,4,6,12,1,1,5,9,5,12,1], \\
&\text{option(1,2,}[1,2,3,5,6,7,8,14],} \\
&\quad [1,1,1,0,1,1,1,0,0,0,0,1,0,0,0,0],48), \\
&\text{option(2,3,}[1,2,3,4,5,9,10,11,15],} \\
&\quad [1,1,1,1,0,0,0,1,1,1,0,0,1,0,0,0],57), \\
&\text{option(1,3,}[3,4,8,11,12,13,18],} \\
&\quad [0,0,1,1,0,0,0,1,1,1,0,0,0,0,1],28), \\
&\text{option(2,5,}[2,4,7,10,13,17],} \\
&\quad [0,1,0,1,0,0,1,0,1,0,0,0,0,1,0],34), \\
&\text{option(1,5,}[1,6,9,12,16],} \\
&\quad [1,0,0,0,1,0,1,0,0,0,1,0,0,1,0,0],17)].
\end{align*}
\]

Data Generation

- Data not really stored as facts
- Generated from text data files in different format
- Benchmark set from CSPLIB
  (http://www.csplib.org)
Assignment Step 4
Another Example (PR97)

- 100 cars
- 22 types
- 5 options
  - Option 1: 1 out of 2
  - Option 2: 2 out of 3
  - Option 3: 1 out of 3
  - Option 4: 2 out of 5
  - Option 5: 1 out of 5

Second Example: Car Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Cars Required</th>
<th>Option 1</th>
<th>Option 2</th>
<th>Option 3</th>
<th>Option 4</th>
<th>Option 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>8</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Search (Stopped After 1000 Nodes)

Observation

- This does not look good
- Typical *thrashing* behaviour
- We made a wrong choice at some point
- ... but did not detect it
- Many additional choices are made before failure is detected
- We have to explore the complete tree under the wrong choice
- This is far too expensive
Change of Search Strategy

- Do not label car slot variables
- Decide instead if slot should use an option or not
- This restricts the car models which can be placed in this slot
- Start with the most restricted option
- When all options are assigned, the car type is fixed
- Potential problem: We now have 500 instead of 100 decision variables
- Naive searchspace $2^{500} = 3.2 \times 10^{150}$ instead of $2^{100} = 1.7 \times 10^{134}$

Second Modification

- Instead of assigning values left to right
- Start assigning in middle of board
- And alternate around middle until you reach edges
- Idea: Slots at edges are less constrained, i.e. easier to assign
- Save those slots until the end
Observations

- Important to start in middle
- Making hard choices first
- Concentrate on difficult to satisfy sub-problem
- Number of choices is much smaller than number of variables
- Some assignments lead to a lot of propagation

Conclusions

- Introduced global constraint `sequence`
- Reuse `gcc` and `element`
- Search on auxiliary variables can work well
- Raw search space measures are unreliable
- Modelling idea
  - Decide what to make in a given time slot
  - ... and not when to schedule some given activity
Mehmet Dincbas, Helmut Simonis, and Pascal Van Hentenryck.
Solving the car-sequencing problem in constraint logic programming.

Jean-Charles Regin and Jean-Francois Puget.
A filtering algorithm for global sequencing constraints.

Christine Solnon, Van Dat Cung, Alain Nguyen, and Christian Artigues.

Willem Jan van Hoeve, Gilles Pesant, Louis-Martin Rousseau, and Ashish Sabharwal.
Revisiting the sequence constraint.
Michael J. Maher, Nina Narodytska, Claude-Guy Quimper, and Toby Walsh.
Flow-based propagators for the sequence and related global constraints.
What we want to introduce

- Hybridisation by decomposition
- Combination of MIP and FD solver
- Best current solution to routing and wavelength assignment problem

Problem Definition

Routing and Wavelength Assignment (Demand Acceptance)

In an optical network, traffic demands between nodes are assigned to a route through the network and a specific wavelength. The route (called *lightpath*) must be a simple path from source to destination. Demands which are routed over the same link must be allocated to different wavelengths, but wavelengths may be reused for demands which do not meet. The objective is to find a combined routing and wavelength assignment which maximizes the number of accepted demands.
Example Network (NSF, 5 wavelengths)

Lightpath from node 5 to node 13 (5 ⇒ 13)
Conflict with demand 1 ⇒ 12: Use different frequencies

Conflict with demand 1 ⇒ 12: Use different path
Conflict with demand 1 ⇒ 12: Reject demand

Solution Approaches

- Greedy heuristic
- Optimization algorithm for complete problem
- Decomposition into two problems
  - Route maximal number of demands
  - Assign wavelengths
Step 1: Route Maximal Number of Demands

- Ignore wavelengths
- Capacity constraints on all links
- Solve as MIP problem
- Source aggregation
- Find DAG to supply (all) demands with shared source
- Maximize number of accepted demands

Notation

- $y_{sd}$, integer number of accepted demands from $s$ to $d$
- $z_{se}$, integer capacity used on edge $e$ to satisfy demands sourced in $s$
- $C$, number of available wavelengths, edge capacity
- $P_{sd}$, requested number of demands from $s$ to $d$
- $T_s$, total number of requested demands sourced from $s$
- $D_s$, nodes which have a requested demand sourced in $s$
Optimal cost is upper bound for full problem
LP Relaxation is also upper bound for full problem
No 0/1 variables in model
Source aggregation has massive impact on efficiency
  - Much better than treating each demand on its own
  - Reason 1: Reduced number of variables
  - Reason 2: Avoids symmetries due to multiple demands between nodes
Finding Accepted Demands

- Solution to MIP does not tell how demands are routed
- Program required to convert source “tree” into sets of paths
- Conversion not deterministic, may allow different solutions
- Solution may contain loops, these need to be removed

Step 2: Assign Wavelengths

- For each accepted demand, find frequency
- All demands routed over a link compete for frequencies
- Graph coloring problem
- Graph given as sets of cliques
- Solve with finite domains
- If solution found, then optimal for complete problem
Model (Step 2)

- $X_d$ finite domain variable 1..$C$ for each accepted demand
- One `alldifferent` constraint for each edge
- Many `alldifferent` constraints are at capacity
- Possible to improve model

What Happens If No Solution Found

- Problem infeasible
  - Remove some demand and try again until solution found
  - Possibly sub-optimal solution of high quality
  - Different solution to MIP problem may lead to optimal solution
- No solution found within time limit
  - Try harder!
  - Improve reasoning and/or search technique
  - Special techniques to show infeasibility
Solution Approach

MIP Resource Model

Extract Accepted Demands

FD Graph Coloring

Remove Demand

Infeasible

Yes

Provide Explanation

No

Solution

Demand Matrix (100 Demands)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Color | Distance
---|---
1 | 1
2 | 2
3 | 3
4 | 4
≥ 5 | 5

Helmut Simonis Modelling in CP 333

Helmut Simonis Modelling in CP 334
Source Model Solution

Source Node 1

Accepted Demands (86 Demands)
## Observations

- Accepted demands do not always use shortest path
- Tendency to reject demands with larger minimal distance
- These use more resources
- Not compensated in objective function
- Not fair
Graph Coloring Problem

Resource Requirements

Graph Coloring Problem

Problem
Model
Worked Example
Results

Graph Coloring Problem

Problem
Model
Worked Example
Results
Graph Coloring Solution

<table>
<thead>
<tr>
<th>Color</th>
<th>Wavelength</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Observation

- All demands could be assigned to frequencies
- Optimal solution to complete problem
Explaining Infeasibility

Ad-hoc: Find pattern which show infeasibility
- Find large cliques
- If clique is larger than number of colors, problem is infeasible
- This is simple for graphs given

General explanation techniques
- Active research area
Find minimal subset of constraints which is infeasible
- Conflict set
- Works when overall problem fails without search
- Requires some trick to be applied here
Accepted Demands (86 Demands)

Does it scale?

- Fixed network structure
  - nsf 14 nodes, 42 edges
  - eon 20 nodes, 78 edges
  - mci 19 nodes, 64 edges
  - brezil 27 nodes, 140 edges
- Random network structure
  - Sizes from 30 to 100 nodes
  - Edge density 0.25
  - 500 demands, 30 wavelengths
Compared to MIP Model for Complete Problem

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>brezil</td>
<td>500</td>
<td>15</td>
<td>98</td>
<td>483.84</td>
<td>0.92</td>
<td>1.34</td>
<td>483.86</td>
<td>1.218.40</td>
<td>14103.84</td>
<td></td>
</tr>
<tr>
<td>brezil</td>
<td>600</td>
<td>20</td>
<td>100</td>
<td>590.96</td>
<td>1.00</td>
<td>1.34</td>
<td>590.96</td>
<td>6076.81</td>
<td>87767.95</td>
<td></td>
</tr>
<tr>
<td>brezil</td>
<td>700</td>
<td>25</td>
<td>98</td>
<td>695.48</td>
<td>0.81</td>
<td>1.80</td>
<td>695.48</td>
<td>13623.15</td>
<td>78463.89</td>
<td></td>
</tr>
<tr>
<td>brezil</td>
<td>800</td>
<td>25</td>
<td>99</td>
<td>781.37</td>
<td>1.44</td>
<td>11.47</td>
<td>781.39</td>
<td>75676.81</td>
<td>15456.50</td>
<td></td>
</tr>
<tr>
<td>eon</td>
<td>500</td>
<td>20</td>
<td>100</td>
<td>471.56</td>
<td>0.65</td>
<td>0.77</td>
<td>471.56</td>
<td>352.21</td>
<td>585.45</td>
<td></td>
</tr>
<tr>
<td>eon</td>
<td>600</td>
<td>25</td>
<td>100</td>
<td>574.80</td>
<td>0.82</td>
<td>1.13</td>
<td>574.80</td>
<td>1411.67</td>
<td>2877.88</td>
<td></td>
</tr>
<tr>
<td>eon</td>
<td>700</td>
<td>30</td>
<td>100</td>
<td>677.35</td>
<td>1.05</td>
<td>1.81</td>
<td>677.35</td>
<td>1727.52</td>
<td>3568.13</td>
<td></td>
</tr>
<tr>
<td>eon</td>
<td>800</td>
<td>35</td>
<td>100</td>
<td>779.17</td>
<td>1.28</td>
<td>1.94</td>
<td>779.17</td>
<td>2485.64</td>
<td>4116.11</td>
<td></td>
</tr>
<tr>
<td>mci</td>
<td>500</td>
<td>25</td>
<td>100</td>
<td>486.38</td>
<td>0.80</td>
<td>2.28</td>
<td>486.38</td>
<td>1023.16</td>
<td>1664.31</td>
<td></td>
</tr>
<tr>
<td>mci</td>
<td>600</td>
<td>30</td>
<td>100</td>
<td>585.18</td>
<td>1.27</td>
<td>29.81</td>
<td>585.18</td>
<td>1621.30</td>
<td>2895.88</td>
<td></td>
</tr>
<tr>
<td>mci</td>
<td>700</td>
<td>35</td>
<td>100</td>
<td>684.00</td>
<td>1.30</td>
<td>3.53</td>
<td>684.00</td>
<td>1987.23</td>
<td>3428.41</td>
<td></td>
</tr>
<tr>
<td>mci</td>
<td>800</td>
<td>40</td>
<td>100</td>
<td>782.86</td>
<td>1.68</td>
<td>5.21</td>
<td>782.86</td>
<td>2316.88</td>
<td>4402.44</td>
<td></td>
</tr>
<tr>
<td>nsf</td>
<td>500</td>
<td>25</td>
<td>100</td>
<td>495.20</td>
<td>0.50</td>
<td>0.60</td>
<td>495.20</td>
<td>82.85</td>
<td>173.19</td>
<td></td>
</tr>
<tr>
<td>nsf</td>
<td>600</td>
<td>30</td>
<td>100</td>
<td>588.63</td>
<td>0.66</td>
<td>0.98</td>
<td>588.63</td>
<td>155.90</td>
<td>373.63</td>
<td></td>
</tr>
<tr>
<td>nsf</td>
<td>700</td>
<td>35</td>
<td>100</td>
<td>678.44</td>
<td>1.35</td>
<td>1.35</td>
<td>678.44</td>
<td>205.82</td>
<td>586.61</td>
<td></td>
</tr>
<tr>
<td>nsf</td>
<td>800</td>
<td>40</td>
<td>100</td>
<td>727.15</td>
<td>0.95</td>
<td>1.56</td>
<td>727.15</td>
<td>173.53</td>
<td>410.97</td>
<td></td>
</tr>
</tbody>
</table>

Increasing Number of Demands

<table>
<thead>
<tr>
<th>Network</th>
<th>Dem.</th>
<th>( \lambda )</th>
<th>Opt.</th>
<th>Avg LP</th>
<th>Avg MIP</th>
<th>Avg FD</th>
<th>Max Gap</th>
<th>Avg MIP Time</th>
<th>Max MIP Time</th>
<th>Avg FD Time</th>
<th>Max FD Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>eon</td>
<td>800</td>
<td>30</td>
<td>100</td>
<td>741.78</td>
<td>741.78</td>
<td>741.78</td>
<td>0.00</td>
<td>0.15</td>
<td>0.17</td>
<td>0.83</td>
<td>1.61</td>
</tr>
<tr>
<td>eon</td>
<td>900</td>
<td>40</td>
<td>100</td>
<td>880.59</td>
<td>880.59</td>
<td>880.59</td>
<td>0.00</td>
<td>0.14</td>
<td>0.16</td>
<td>1.18</td>
<td>2.17</td>
</tr>
<tr>
<td>eon</td>
<td>1000</td>
<td>40</td>
<td>100</td>
<td>950.36</td>
<td>950.36</td>
<td>950.36</td>
<td>0.00</td>
<td>0.15</td>
<td>0.17</td>
<td>1.37</td>
<td>3.42</td>
</tr>
<tr>
<td>eon</td>
<td>1100</td>
<td>50</td>
<td>100</td>
<td>1082.61</td>
<td>1082.61</td>
<td>1082.61</td>
<td>0.00</td>
<td>0.14</td>
<td>0.16</td>
<td>1.71</td>
<td>2.83</td>
</tr>
<tr>
<td>eon</td>
<td>1200</td>
<td>50</td>
<td>100</td>
<td>1156.38</td>
<td>1156.38</td>
<td>1156.38</td>
<td>0.00</td>
<td>0.15</td>
<td>0.17</td>
<td>2.07</td>
<td>3.92</td>
</tr>
<tr>
<td>eon</td>
<td>1300</td>
<td>50</td>
<td>100</td>
<td>1219.82</td>
<td>1219.82</td>
<td>1219.82</td>
<td>0.00</td>
<td>0.16</td>
<td>0.18</td>
<td>2.22</td>
<td>5.24</td>
</tr>
<tr>
<td>eon</td>
<td>1400</td>
<td>60</td>
<td>100</td>
<td>1361.47</td>
<td>1361.47</td>
<td>1361.47</td>
<td>0.00</td>
<td>0.15</td>
<td>0.16</td>
<td>2.92</td>
<td>4.94</td>
</tr>
<tr>
<td>eon</td>
<td>1500</td>
<td>60</td>
<td>99</td>
<td>1428.78</td>
<td>1428.78</td>
<td>1428.78</td>
<td>1.00</td>
<td>0.15</td>
<td>0.17</td>
<td>4.22</td>
<td>106.97</td>
</tr>
<tr>
<td>eon</td>
<td>1600</td>
<td>70</td>
<td>100</td>
<td>1565.90</td>
<td>1565.90</td>
<td>1565.90</td>
<td>0.00</td>
<td>0.15</td>
<td>0.16</td>
<td>3.89</td>
<td>8.48</td>
</tr>
<tr>
<td>eon</td>
<td>1700</td>
<td>70</td>
<td>100</td>
<td>1637.47</td>
<td>1637.47</td>
<td>1637.47</td>
<td>0.00</td>
<td>0.16</td>
<td>0.17</td>
<td>4.58</td>
<td>13.59</td>
</tr>
<tr>
<td>eon</td>
<td>1800</td>
<td>80</td>
<td>100</td>
<td>1769.86</td>
<td>1769.86</td>
<td>1769.86</td>
<td>0.00</td>
<td>0.15</td>
<td>0.16</td>
<td>5.19</td>
<td>8.81</td>
</tr>
<tr>
<td>eon</td>
<td>1900</td>
<td>80</td>
<td>99</td>
<td>1844.46</td>
<td>1844.46</td>
<td>1844.46</td>
<td>1.00</td>
<td>0.15</td>
<td>0.17</td>
<td>7.23</td>
<td>163.41</td>
</tr>
<tr>
<td>eon</td>
<td>2000</td>
<td>90</td>
<td>100</td>
<td>1972.66</td>
<td>1972.66</td>
<td>1972.66</td>
<td>0.00</td>
<td>0.15</td>
<td>0.17</td>
<td>6.34</td>
<td>9.61</td>
</tr>
</tbody>
</table>
Random Networks (Edge Density 0.25, 100 Runs Each)

<table>
<thead>
<tr>
<th>Network</th>
<th>Dem.</th>
<th>λ</th>
<th>Opt.</th>
<th>Avg LP</th>
<th>Avg MIP</th>
<th>Avg FD</th>
<th>Avg MIP Time</th>
<th>Max MIP Time</th>
<th>Avg FD Time</th>
<th>Max FD Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>r30</td>
<td>500</td>
<td>30</td>
<td>100</td>
<td>391.82</td>
<td>391.82</td>
<td>391.82</td>
<td>0.45</td>
<td>0.55</td>
<td>0.12</td>
<td>0.16</td>
</tr>
<tr>
<td>r40</td>
<td>500</td>
<td>30</td>
<td>100</td>
<td>424.58</td>
<td>424.58</td>
<td>424.58</td>
<td>1.07</td>
<td>1.23</td>
<td>0.14</td>
<td>0.17</td>
</tr>
<tr>
<td>r50</td>
<td>500</td>
<td>30</td>
<td>100</td>
<td>437.69</td>
<td>437.69</td>
<td>437.69</td>
<td>2.13</td>
<td>2.38</td>
<td>0.09</td>
<td>0.13</td>
</tr>
<tr>
<td>r60</td>
<td>500</td>
<td>30</td>
<td>100</td>
<td>447.21</td>
<td>447.21</td>
<td>447.21</td>
<td>3.92</td>
<td>4.34</td>
<td>0.08</td>
<td>0.16</td>
</tr>
<tr>
<td>r70</td>
<td>500</td>
<td>30</td>
<td>100</td>
<td>453.41</td>
<td>453.41</td>
<td>453.41</td>
<td>6.78</td>
<td>7.50</td>
<td>0.10</td>
<td>0.17</td>
</tr>
<tr>
<td>r80</td>
<td>500</td>
<td>30</td>
<td>100</td>
<td>457.65</td>
<td>457.65</td>
<td>457.65</td>
<td>10.75</td>
<td>11.95</td>
<td>0.10</td>
<td>0.17</td>
</tr>
<tr>
<td>r90</td>
<td>500</td>
<td>30</td>
<td>100</td>
<td>464.69</td>
<td>464.69</td>
<td>464.69</td>
<td>16.08</td>
<td>17.45</td>
<td>0.08</td>
<td>0.22</td>
</tr>
<tr>
<td>r100</td>
<td>500</td>
<td>30</td>
<td>100</td>
<td>466.67</td>
<td>466.67</td>
<td>466.67</td>
<td>22.74</td>
<td>25.22</td>
<td>0.09</td>
<td>0.25</td>
</tr>
</tbody>
</table>

**Observations**

- MIP and LP relaxation of phase 1 are very good bounds
- Solved to optimality in most cases
- Simple decomposition quite effective
- Good solution even if initial graph coloring infeasible
- Special structure of graph coloring helps FD model
Conclusions

- Combination of MIP and FD solver in problem decomposition
- Each doing what they do best
  - MIP: optimal solution, select items to include
  - FD: find feasible solution, explain infeasibility
- Hybrid model produces very high quality results
- Proven optimality in over 99.85% of problems tested
- Near optimal solutions by relaxation
- Much faster than monolithic MIP solution

More Information


Brigitte Jaumard, Christophe Meyer, and Babacar Thiongane.
ILP formulations for the routing and wavelength assignment problem: Symmetric systems.

Brigitte Jaumard, Christophe Meyer, and Babacar Thiongane.
Comparison of ILP formulations for the RWA problem.

Ulrich Junker.
Quickxplain: Conflict detection for arbitrary constraint propagation algorithms.
In *IJCAI’01 Workshop on Modelling and Solving problems with constraints (CONS-1)*, Seattle, WA, USA, August 2001.
Helmut Simonis.
Constraint applications in networks.

Helmut Simonis.
A Hybrid Constraint Model for the Routing and Wavelength Assignment Problem.
submitted for publication.
http://4c.ucc.ie/~hsimonis/rwa.pdf
Shown examples of modelling techniques with constraints

Basic steps
- Choose variables
- Choose constraints
- Choose search
- Evaluate and change as required

What did I miss?
- All of scheduling
- All of configuration
- Pattern based constraints
  - regular, context-free
  - Use in time tabling
- Table based constraints
Search is Part of Modelling

- Choice of search strategy is important
  - Although big improvements in automated search
  - Most search variants can be built from small number of orthogonal concepts
    - Variable choice
    - Value choice
    - Form of search tree

Do we need another black box?

- SAT, MIP are largely black boxes
  - If it works, very good
  - If it doesn’t work, you are in trouble
- With power comes responsibility
Visualization is Important

- Understand what is happening
- Understand what is not happening
- Without need to know why
- Nice separation of concerns

Which constraints do we need?

- How many global constraints can one learn?
- Size of code base required
- Compared to benefits earned
Modelling as Art

- Many possible ways to model problems
- Creative step
- Big performance difference
- Surprises happen, intuition is not everything

Modelling as Craft

- Modelling can be learned
- Practice makes perfect!
- Consider different mappings of concepts encountered
- Still quite system specific
  - Which constraints are available
  - How much propagation is done
  - How fast is implementation
How do we know if one model is “better” than another?
- Length of description?
- Execution time?
- Size of search tree?
- Scalability?