Chapter 9: Choosing the Model (Sports Scheduling)

Helmut Simonis

Cork Constraint Computation Centre
Computer Science Department
University College Cork
Ireland

ECLiPSe ELearning

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.
Outline

1. Problem
2. Model
3. Program
4. Search
5. Redundant Modelling

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
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></td>
<td>1, 5</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></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>1, 3</td>
<td></td>
</tr>
</tbody>
</table>
Solution

<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>6, 8</td>
<td></td>
<td>1, 2</td>
<td></td>
<td>5, 7</td>
<td></td>
<td>3, 4</td>
</tr>
<tr>
<td>Day 2</td>
<td>2, 3</td>
<td>1, 5</td>
<td></td>
<td>4, 8</td>
<td>6, 7</td>
<td></td>
<td>5, 6</td>
</tr>
<tr>
<td>Day 3</td>
<td>1, 7</td>
<td>2, 4</td>
<td>3, 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 4</td>
<td>4, 7</td>
<td></td>
<td></td>
<td>2, 6</td>
<td>3, 5</td>
<td>1, 8</td>
<td></td>
</tr>
<tr>
<td>Day 5</td>
<td>5, 8</td>
<td></td>
<td></td>
<td>3, 6</td>
<td>1, 4</td>
<td>2, 7</td>
<td></td>
</tr>
<tr>
<td>Day 6</td>
<td>3, 7</td>
<td>1, 6</td>
<td>4, 5</td>
<td></td>
<td></td>
<td>2, 8</td>
<td></td>
</tr>
<tr>
<td>Day 7</td>
<td>4, 6</td>
<td>2, 5</td>
<td>7, 8</td>
<td>1, 3</td>
<td></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 atmost 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, \texttt{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 \texttt{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 $Day \times Game$
- 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 $Day \times Location$
- 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 $\text{alldifferent0}$
  - Or use $gcc$
- 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 $l$
- $3136 = 8*8*7*7$ 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*7/2=28 variables)
- Decide when and where this game is played, values range over combinations of days and locations (7*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?

Expand Idea 7 into Full Model
## 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
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

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?

1. There are 8 teams, seven days and seven locations
2. Each team plays each other team exactly once
3. Each team plays 7 games (redundant)
4. Each team plays in each location exactly once
5. Each team plays on each day exactly once
6. A game consists of two (different) teams
7. There are four games on each day (redundant)
8. There are four games at each location (redundant)
9. 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 an 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
  - \( gcc([gcc(4,4,1),...,gcc(4,4,7)], \text{Rows}) \)
- Similar for columns:
  - \( gcc([gcc(4,4,1),...,gcc(4,4,7)], \text{Cols}) \)

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

<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>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>N</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>C</td>
</tr>
<tr>
<td>5</td>
<td>C</td>
</tr>
<tr>
<td>6</td>
<td>C</td>
</tr>
<tr>
<td>7</td>
<td>C</td>
</tr>
<tr>
<td>8</td>
<td>C</td>
</tr>
</tbody>
</table>

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>
Viewpoints and Channeling

- 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>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>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>

\[ \text{element(23, [1,1,1,1,1,1,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,4,4,4,4,4,4,5,5,5,5,5,5,6,6,6,6,6,6,6,6,6,7,7,7,7,7,7], 4)}, \]
\[ \text{element(23, [1,2,3,4,5,6,7,1,2,3,4,5,6,7,1,2,3,4,5,6,7,1,2,3,4,5,6,7,1,2,3,4,5,6,7,1,2,3,4,5,6,7,1,2,3,4,5,6,7], 2)}, \]

Channeling Constraints

- 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`
Selected Model

- Two sets of variables (Req 1, 2, 3, 6, by construction)
- Pair variables \(T \triangle T \rightarrow D \times L\)
  - alldifferent (Req 9)
- Day and Location variables \((T \triangle T \rightarrow D), (T \triangle T \rightarrow L)\)
  - gcc (Req 4, 5)
  - alldifferent (Req 7, 8)
- Channeling Constraints
  - element projection from pairs onto rows and columns
- Search only on pair variables

Handling of hints (I)

<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></td>
<td>1, 5</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>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 4</td>
<td></td>
<td>8</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>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></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>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></td>
<td></td>
<td></td>
<td>1, 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Problem Data

```
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])).
```
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),
    ...

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):-
geominate_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)
  ).
Generating Projection Tables

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
         )
   ).

Extract row variables

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

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

Setting up hints

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

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

Vars

Solution
Search Tree with input order

- Try different search strategy
- Use `first_fail` dynamic variable selection
Search Tree with first fail

Observation

- It does not work
- Search tree is slightly larger than before!
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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 1</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 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>
</tr>
<tr>
<td>Day 4</td>
<td></td>
<td>2</td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 5</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 6</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>
</tr>
</tbody>
</table>

<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<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>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>Day 3</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>Day 4</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
</tr>
<tr>
<td>Day 5</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>41</td>
</tr>
<tr>
<td>Day 6</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
</tr>
<tr>
<td>Day 7</td>
<td>48</td>
<td>49</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Missing Propagation

#### Adding value index Channeling

Improving 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>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<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>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<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>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<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>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<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>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<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>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<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>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Missed 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>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>2</td>
<td>2</td>
<td>1, 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

---

### Adding value index

### Channeling

### Improving Handling of Hints

### Redundant Modelling

---

#### Choosing the Model

- Adding value index
- Channeling
- Improving Handling of Hints
- Redundant Modelling

---

### Problem

- Model
- Program
- Search

---

#### Helmut Simonis

Choosing the Model 63
Missing Propagation

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
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:
  - $P_i = v \Rightarrow Y_v = 1$
  - $Y_v = 0 \Rightarrow \forall i : P_i \neq v$
  - $(\forall i : v \notin d(P_i)) \Rightarrow Y_v = 0$
  - $Y_v = 1 \Rightarrow \text{occurrence}(V, P_1...P_n, N), N \geq 1$

Added Program

```prolog
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),
    (for(I,1,7),param(Matrix) do
        sumlist(Matrix[I,1..7],4),
        sumlist(Matrix[1..7,I],4)
    ),
    (foreach(K-,Hints),param(Matrix) do
        coor(K,I,J),
        subscript(Matrix,[I,J],1)
    ).
```

Helmut Simonis  Choosing the Model  69
Impact of Redundant Constraints

Without

With value index channeling
Helmut Simonis  Choosing the Model  75

Still Missing Propagation

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

Helmut Simonis  Choosing the Model  76

Still Missing Propagation

Game 12 can not be played on day 1 at locations 5 or 6
Still 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></td>
<td>7, 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 2</td>
<td>2, 1, 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 3</td>
<td>7</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 4</td>
<td></td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 5</td>
<td>8</td>
<td></td>
<td></td>
<td>5, 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day 6</td>
<td></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>

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

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></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></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></td>
<td></td>
<td>5, 4</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>

- 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

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

Added Program

```prolog
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).  
```
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]
    )
  ),
).

...
exlude_values(Vars,Values):-
   (foreach(X,Vars),
    param(Values) do
      (foreach(Value,Values),
        param(X) do
          X #\= Value
      )
    ).
Impact of improved hint handling

- 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


1. Is there potential for symmetry breaking in this problem?
2. Develop one of the alternative models considered into a full program and compare its performance.
3. Consider the use of channeling for the N-queens problem, for the Sudoku puzzle. What could you use as a second viewpoint, and how to link the views together?