Chapter 20: Working with Implications

Helmut Simonis

Cork Constraint Computation Centre
Computer Science Department
University College Cork
Ireland

ECLiPSe ELearning
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. Program
3. Improvements
What we want to introduce

- Solving a placement problem without specialized constraints
- Decomposition into pattern generation and set partitioning
- Using implications to propagate information
Outline

1 Problem

2 Program

3 Improvements
Shikaku

The puzzle is played in a given recti-linear grid. Some grid cells contain numbers. The task is to partition the grid area into rectangular rooms satisfying the conditions:

1. Each room contains exactly one number.
2. The area of the room is equal to the number in it.
Solution Approaches

- The right way
- The wrong way
Using one `geost` constraint (Carlsson, Beldiceanu et al.)

- Each room is an object with fixed surface
- Shapes defined by $Width \times Height = Surface$
- Each room has $x, y$ position and size $w, h$
- Each room must contain cell with hint
- Rooms must fit into given space
- Rooms do not overlap
Using \textit{regular} (Lagerkvist, Pesant)

0/1 Variables $x_{ijk}$ cell $(i,j)$ belongs to room $k$

Each cell must belong to exactly one room (equality)

Use regular expression to describe possible shapes

Disjunction of possible placements

One \textit{regular} constraint for each room

Possible to combine regular expressions for multiple rooms
The Right Way (III)

SMT?
The Wrong Way (Naive)

- Each room $k$ has position $x_k$, $y_k$ and size $w_k, h_k$
- Room $k$ has fixed surface $S_k = w_k * h_k$
- Room $k$ contains fixed hint at $U_k$, $V_k$
  - $x_k \leq U_k < x_k + w_k$
  - $y_k \leq V_k < y_k + h_k$
- Any rooms $k$ and $l$ do not overlap
Expressing Non-Overlap with Disjunctive

\[ Y_1 \geq Y_2 + H_2 \quad \text{above} \]
\[ X_2 \geq X_1 + W_1 \quad \text{left} \]
\[ W_2 \]
\[ H_2 \]
\[ X_2, Y_2 \]
\[ Y_2 \geq Y_1 + H_1 \quad \text{below} \]
\[ X_1 \geq X_2 + W_2 \quad \text{right} \]
Problems

- Weak propagation
- *Width* and *Height* not in sync
- Estimate for room surface much too low
- Disjunctive does not propagate much
Idea

- Problem decomposition
  - Phase 1: Generate possible placement pattern for each room
  - Phase 2: Pick exactly one pattern for each room
Phase 1: Pattern Generation

- Solve with finite domains
- For each room, solve
  - Each room $k$ has position $x_k, y_k$ and size $w_k, h_k$
  - Room $k$ has fixed surface $S_k = w_k \times h_k$
  - Room $k$ contains fixed hint at $U_k, V_k$
    - $x_k \leq U_k < x_k + w_k$
    - $y_k \leq V_k < y_k + h_k$
  - All other hints are outside the room
    - Expressed as negation of inside
- Find all solutions
Phase 1 Results

- Possible placements for each room $1 \leq p \leq P_k$
- Position $x_{kp}$, $y_{kp}$, size $w_{kp}, h_{kp}$
- Predicate $q(k, p, i, j)$ pattern $p$ of room $k$ contains cell $(i, j)$
Phase 2: Set Partitioning

- 0/1 integer variable $z_{kp}$ if pattern $p$ is used for room $k$
- Select one pattern per room
- Cover every cell with exactly one pattern
- Constrain all variables which belong to pattern which contain cell
Model Phase 2

solve

\[
\forall k \in K : \sum_{1 \leq p \leq P_k} z_{kp} = 1
\]

\[
\forall (i, j) \in \text{Rect} : \sum_{\{k, p | q(k, p, i, j)\}} z_{kp} = 1
\]

\[
z_{kp} \in \{0, 1\}
\]
Model Phase 2

solve

\[ z_{kp} \in \{0, 1\} \]

\[ \forall k \in K : \sum_{1 \leq p \leq P_k} z_{kp} = 1 \]

\[ \forall (i, j) \in \text{Rect} : \sum_{\{k, p \mid q(k, p, i, j)\}} z_{kp} = 1 \]

The first set of equations is subsumed by the second
Intuition Behind Constraints

- If only one pattern remains possible for a room, this must be chosen
- If a pattern is selected for a room, no other pattern can be selected
- Every cell must be covered by a pattern
- If a pattern covering some cell is selected, no other pattern covering the same cell may be selected
Outline

1. Problem
2. Program
3. Improvements
:-module(pure).
:-export(top/0).
:-use_module(structures).
:-use_module(data).
:-use_module(utility).
:-lib(ic).

top:-
  data(Set,Nr,Grade,X,Y,Matrix),
  solve(Set,Nr,Grade,X,Y,Matrix).
:-module(structures).

:-export struct(hint(i,j,n,d)).
:-export struct(rectangle(x,y,w,h, n, % size of rectangle k, % hint nr, links alternative rectangles cnt, % id of rectangle var % 0/1 variable, design used )).
:-export struct(overlap(point,cnt,k,var)).
:-module(data).
:-export(data/6).
data('nikoli-web' ,0 ,easy,10 ,10 ,
    [[
      [[x ,8 ,4 ,x ,x ,x ,x ,4 ,x ,x ]],
      [[x ,x ,x ,x ,x ,x ,x ,6 ,x ,x ]],
      [[x ,x ,x ,3 ,3 ,x ,x ,x ,x ,x ]],
      [[x ,x ,x ,x ,x ,x ,2 ,x ,x ,x ]],
      [[5 ,4 ,x ,x ,x ,x ,2 ,x ,x ,x ]],
      [[x ,x ,x ,9 ,x ,x ,x ,x ,6 ,7 ]],
      [[x ,x ,x ,8 ,x ,x ,x ,x ,x ,x ]],
      [[x ,x ,x ,x ,x ,4 ,5 ,x ,x ,x ]],
      [[x ,x ,4 ,x ,x ,x ,x ,x ,x ,x ]],
      [[x ,x ,5 ,x ,x ,x ,x ,5 ,6 ,x ]]
    ]).

Helmut Simonis
Shikaku
solve(Set, Nr, Grade, X, Y, Matrix):-
    writeln(solving(Set, Nr, Grade)),
    (multifor([I, J], [1, 1], [X, Y]),
     fromto([], A, A1, Hints),
     fromto(0, B, B1, _Types),
     param(Matrix) do
         hint_cell(Matrix, I, J, A, A1, B, B1)
    ),
    create_rectangles(Hints, [], [],
        Rectangles, X, Y),
    numbering(Rectangles),
    extract_vars(Rectangles, var of rectangle, List),
    List :: 0..1,
    ...

...
create_overlap(Rectangles,Overlap),
group_by(point of overlap,Overlap,Grouped),
(foreach(_-Group,Grouped) do
    extract_vars(Group,var of overlap,List),
    sum(List) #= 1
),
search(List,0,input_order,indomain,complete,[]),
writeln(List).
Extracting Hints

\[
\text{hint\_cell}(\text{Matrix}, I, J, \\
A, [\text{hint}\{i: I, j: J, n: N, d: D1\}|A], \\
D, D1) :- \\
\text{subscript}(\text{Matrix}, [I, J], N), \\
\text{integer}(N), \\
!, \\
D1 \text{ is } D + 1.
\]

\[
\text{hint\_cell}(_\text{Matrix}, _I, _J, A, A, B, B).
\]
create_rectangles([],_,Rectangles,Rectangles,_Width,_Height).
create_rectangles([Hint|Hints],Old,RIn,ROut,Width,Height):-
    findall(Rectangle,
        rectangle(Hint,Hints,Old,Width,Height,Rectangle),
        Rectangles),
    append(Rectangles,RIn,R1),
    create_rectangles(Hints,[Hint|Old],R1,ROut,Width,Height).
rectangle(hint{i:I,j:J,n:N,d:K},Hints,Old, Width,Height,  
  rectangle{x:X,y:Y,w:W,h:H,n:N,k:K}):-  
  X :: 1..Width,  
  Y :: 1..Height,  
  W :: 1..N,  
  H :: 1..N,  
  W*H #= N,  
  X+W-1 #=< Width,  
  Y+H-1 #=< Height,  
  inside(X,Y,W,H,I,J),  
  outsides(X,Y,W,H,Hints),  
  outsides(X,Y,W,H,Old),  
  search([X,Y,W,H],0,input_order,indomain,  
          complete,[]).
inside(X,Y,W,H,I,J):-
   I #>= X,
   J #>= Y,
   I #< X+W,
   J #< Y+H.

outsides(X,Y,W,H,L):-
   (foreach(hint{i:I,j:J},L),
    param(X,Y,W,H) do
     outside(X,Y,W,H,I,J)
   )
).

outside(X,Y,W,H,I,J):-
   I #< X or J #< Y or I #>=X+W or J #>= Y+H.
Creating Overlap Structures

create_overlap(Rectangles,Overlap):-
  (foreach(Rect,Rectangles),
   fromto([],A,A1,Overlap) do
    create_overlap1(Rect,A,A1)
  ).

create_overlap1(rectangle{x:X,y:Y,w:W,h:H,}
    cnt:Cnt,k:K,var:Var},In,Out):-
  (multifor([[I,J],[X,Y],[X+W-1,Y+H-1]]),
   fromto(In,A,
     [overlap{point:point(I,J),
       cnt:Cnt,k:K,var:Var}|A],Out),
   param(Cnt,Var,K) do
    true
  ).
extract_vars(Structures, Arg, List):-
    (foreach(Struct, Structures),
     param(Arg),
     foreach(X, List) do
       arg(Arg, Struct, X)
    ).

numbering(Rectangles):-
    (foreach(rectangle{cnt:C}, Rectangles),
     count(C, 1, _) do
       true
    ).
Result After Constraint Setup

The diagram shows a shikaku puzzle, where the goal is to divide the grid into rectangles and squares such that each rectangle has a number that matches the area of the rectangle.

The numbers in the grid indicate the size of the rectangles. For example, a cell with the number 9 indicates a 3x3 rectangle, and a cell with the number 2 indicates a 1x2 rectangle.
Observation

- Only small part of problem filled in
- Missing information
- Some cells must belong to a room regardless of pattern used
Missing Information, Example
Adding View

- Variables which indicate which room some cell belongs to
- 0/1 integer variables $w_{ijk}$, whether cell $(i, j)$ belongs to room $k$
- Most $w_{ijk}$ entries are zero, as room $k$ does not cover cell $(i, j)$
- Each cell must belong to a room
  - $\forall (i, j) : \sum_{k \in K} w_{ijk} = 1$
Channeling Constraints

- A cell belongs to a room, if one of the pattern covering the cell is selected
- Linear constraint
  \[ \forall(i, j, k) : \sum_{p|q(k, p, i, j)} z_{kp} = w_{ijk} \]
- Implications
  \[ \forall(k, p, i, j) \text{ s.t. } q(k, p, i, j) : z_{kp} \Rightarrow w_{i, j, k} \]
- Propagation is equivalent
<table>
<thead>
<tr>
<th>Problem</th>
<th>Program</th>
<th>Improvements</th>
<th>Result After Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>36</td>
<td>2</td>
<td>35</td>
</tr>
<tr>
<td>6</td>
<td>34</td>
<td>2</td>
<td>33</td>
</tr>
<tr>
<td>6</td>
<td>33</td>
<td>2</td>
<td>32</td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>4</td>
<td>31</td>
</tr>
<tr>
<td>4</td>
<td>31</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>29</td>
<td>6</td>
<td>28</td>
</tr>
<tr>
<td>2</td>
<td>28</td>
<td>2</td>
<td>27</td>
</tr>
<tr>
<td>2</td>
<td>27</td>
<td>2</td>
<td>26</td>
</tr>
<tr>
<td>2</td>
<td>26</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>25</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>1</td>
<td>24</td>
<td>1</td>
<td>23</td>
</tr>
<tr>
<td>1</td>
<td>23</td>
<td>1</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>22</td>
<td>1</td>
<td>21</td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>1</td>
<td>19</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>18</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>1</td>
<td>17</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Helmut Simonis  Shikaku
Not What We Expected

- Some cells are marked
- Others are still missing
- Cells marked can only be covered by one room
- Why is this happening?
A, B, C ∈ \{0, 1\}

A + B + C = 1 ∧ A + B = 1 \Rightarrow C = 0

This does not happen!

Constraints only see variables, not other constraints

We would need global constraint on sets of equations for this
Implications Too Weak

- We can add redundant constraints
- If a cell belongs to a room, no pattern for other rooms using that cell can be selected
  - $\forall (i, j, k) : \sum_{\{k', p' | q(k', p', i, j) \land k \neq k'\}} z_{k'p'} + w_{ijk} = 1$
  - $\forall (i, j, k, k', p') \text{ s.t. } q(k', p', i, j) : w_{ijk} \Rightarrow \neg z_{k'p'}$
- If a cell belongs to a room, no pattern for this room which does not cover the cell can be selected
  - $\forall (i, j, k) : \sum_{\{p | \neg q(k, p, i, j)\}} z_{kp} + w_{ijk} = 1$
  - $\forall (i, j, k, p) \text{ s.t. } \neg q(k, p, i, j) : w_{ijk} \Rightarrow \neg z_{kp}$
Result After Adding Redundant Constraints
More to be done
Look through example problems
Find cases where problem is not solved by propagation
Try and find redundant constraints
Example
Reasoning on Space Required
Space required for rooms 76, 94, 102, 107
Another view: finite domain variables for room assignment

Variables $v_{ij}$ state that cell $(i, j)$ belongs to room $v_{ij}$

Connection to $w_{ijk}$ variables via

\[ \text{bool\_channel}(v_{ij}, [w_{ij1}, w_{ij2}, ..., w_{ijk}[K]]) \]

constraints

Each value must occur specified time

gcc constraint with fixed counts
Improvement

- **gcc** reasons on all hints together
- Just adding $\sum_{(i,j)} w_{ijk} = S_k$ does not do this
Deduction by gcc (partial)
### Unsolved Part

<p>| | | | | | | | | | | | | | | | |</p>
<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>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>86</td>
<td>86</td>
<td>95</td>
<td>99</td>
<td>99</td>
<td>108</td>
<td>112</td>
<td>117</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>86</td>
<td>86</td>
<td>95</td>
<td>104</td>
<td>100</td>
<td>104</td>
<td>4</td>
<td>108</td>
<td>113</td>
<td>117</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>86</td>
<td>86</td>
<td>96</td>
<td>104</td>
<td>100</td>
<td>104</td>
<td>6</td>
<td>109</td>
<td>113</td>
<td>117</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>92</td>
<td>7</td>
<td>100</td>
<td>6</td>
<td>104</td>
<td>100</td>
<td>109</td>
<td>113</td>
<td>117</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>87</td>
<td>92</td>
<td>4</td>
<td>100</td>
<td>6</td>
<td>105</td>
<td>100</td>
<td>109</td>
<td>113</td>
<td>117</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>87</td>
<td>92</td>
<td>4</td>
<td>100</td>
<td>6</td>
<td>105</td>
<td>100</td>
<td>109</td>
<td>118</td>
<td>118</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>87</td>
<td>93</td>
<td>4</td>
<td>105</td>
<td>100</td>
<td>105</td>
<td>100</td>
<td>109</td>
<td>118</td>
<td>118</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>88</td>
<td>93</td>
<td>4</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>123</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Helmut Simonis**  **Shikaku**
What to do?

- Pattern are pair-wise compatible with each other
- There is enough space
- But there is only one solution
Looking more closely

<p>| | | | | | | | | | | |</p>
<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>6</td>
<td>86</td>
<td>86</td>
<td>95</td>
<td>99</td>
<td>99</td>
<td>108</td>
<td>112</td>
<td>117</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>86</td>
<td>9</td>
<td>95</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>108</td>
<td>113</td>
<td>117</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>86</td>
<td>9</td>
<td>95</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>109</td>
<td>113</td>
<td>117</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>92</td>
<td>96</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>109</td>
<td>113</td>
<td>117</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>87</td>
<td>92</td>
<td>96</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>109</td>
<td>113</td>
<td>117</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>87</td>
<td>92</td>
<td>96</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>109</td>
<td>118</td>
<td>118</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>87</td>
<td>93</td>
<td>96</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>109</td>
<td>118</td>
<td>118</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>88</td>
<td>93</td>
<td>96</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>123</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

Helmut Simonis
Shikaku
Looking more closely

<p>| | | | | | | | | | | | | | |</p>
<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>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>86</td>
<td>86</td>
<td>95</td>
<td>99</td>
<td>99</td>
<td>108</td>
<td>112</td>
<td>117</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>86</td>
<td>86</td>
<td>95</td>
<td>104</td>
<td>104</td>
<td>108</td>
<td>113</td>
<td>117</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>86</td>
<td>86</td>
<td>96</td>
<td>104</td>
<td>104</td>
<td>109</td>
<td>113</td>
<td>117</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>92</td>
<td>96</td>
<td>100</td>
<td>104</td>
<td>100</td>
<td>109</td>
<td>113</td>
<td>117</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>87</td>
<td>92</td>
<td>96</td>
<td>100</td>
<td>106</td>
<td>100</td>
<td>109</td>
<td>113</td>
<td>117</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>87</td>
<td>92</td>
<td>96</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>109</td>
<td>118</td>
<td>118</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>87</td>
<td>93</td>
<td>96</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>109</td>
<td>118</td>
<td>118</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>88</td>
<td>93</td>
<td>96</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>123</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Helmut Simonis  Shikaku
Looking more closely
Looking more closely
As Equations

- For one cell we have
  \[ a_{104} + c_{100} = 1 \]
- For the other cell we have
  \[ a_{104} + c_{100} + a_{105} = 1 \]
  - Implies \( a_{105} = 0 \)
  - Similar, \( a_{104} = 0 \)
  - Therefore \( c_{100} = 1 \)
All done? No, One Problem Still Open
<table>
<thead>
<tr>
<th>Set</th>
<th>Nr</th>
<th>Grade</th>
<th>X</th>
<th>Y</th>
<th>K</th>
<th>Fix</th>
<th>Alt</th>
<th>Cov</th>
<th>Ini</th>
<th>Set</th>
<th>Red</th>
<th>GCC</th>
<th>Sub</th>
<th>SAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>nikoli-web</td>
<td>0</td>
<td>easy</td>
<td>10</td>
<td>10</td>
<td>20</td>
<td>3</td>
<td>80</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nikoli-web</td>
<td>1</td>
<td>easy</td>
<td>10</td>
<td>10</td>
<td>14</td>
<td>2</td>
<td>81</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nikoli-web</td>
<td>2</td>
<td>easy</td>
<td>10</td>
<td>10</td>
<td>16</td>
<td>1</td>
<td>83</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nikoli-web</td>
<td>3</td>
<td>easy</td>
<td>10</td>
<td>10</td>
<td>28</td>
<td>3</td>
<td>96</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nikoli-web</td>
<td>4</td>
<td>easy</td>
<td>10</td>
<td>18</td>
<td>44</td>
<td>15</td>
<td>78</td>
<td>78</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nikoli-web</td>
<td>5</td>
<td>medium</td>
<td>10</td>
<td>18</td>
<td>38</td>
<td>3</td>
<td>115</td>
<td>21</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nikoli-web</td>
<td>6</td>
<td>medium</td>
<td>10</td>
<td>18</td>
<td>36</td>
<td>1</td>
<td>210</td>
<td>3</td>
<td>153</td>
<td>50</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nikoli-web</td>
<td>7</td>
<td>medium</td>
<td>14</td>
<td>24</td>
<td>68</td>
<td>4</td>
<td>293</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nikoli-web</td>
<td>8</td>
<td>hard</td>
<td>14</td>
<td>24</td>
<td>70</td>
<td>8</td>
<td>380</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nikoli-web</td>
<td>9</td>
<td>hard</td>
<td>20</td>
<td>36</td>
<td>128</td>
<td>4</td>
<td>872</td>
<td>8</td>
<td>196</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nikoli-web</td>
<td>10</td>
<td>hard</td>
<td>20</td>
<td>36</td>
<td>120</td>
<td>1</td>
<td>986</td>
<td>1</td>
<td>774</td>
<td>627</td>
<td>36</td>
<td>36</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>giant</td>
<td>0</td>
<td>easy</td>
<td>7</td>
<td>7</td>
<td>12</td>
<td>2</td>
<td>46</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>giant</td>
<td>1</td>
<td>-</td>
<td>35</td>
<td>21</td>
<td>149</td>
<td>0</td>
<td>1002</td>
<td>2</td>
<td>995</td>
<td>345</td>
<td>29</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>giant</td>
<td>2</td>
<td>-</td>
<td>35</td>
<td>21</td>
<td>189</td>
<td>0</td>
<td>888</td>
<td>5</td>
<td>855</td>
<td>324</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>giant</td>
<td>3</td>
<td>-</td>
<td>35</td>
<td>21</td>
<td>209</td>
<td>5</td>
<td>1023</td>
<td>6</td>
<td>830</td>
<td>583</td>
<td>256</td>
<td>256</td>
<td>133</td>
<td>0</td>
</tr>
<tr>
<td>giant</td>
<td>4</td>
<td>-</td>
<td>35</td>
<td>21</td>
<td>190</td>
<td>2</td>
<td>849</td>
<td>8</td>
<td>495</td>
<td>70</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>giant</td>
<td>5</td>
<td>-</td>
<td>35</td>
<td>21</td>
<td>137</td>
<td>1</td>
<td>990</td>
<td>3</td>
<td>929</td>
<td>657</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>giant</td>
<td>6</td>
<td>-</td>
<td>35</td>
<td>21</td>
<td>160</td>
<td>2</td>
<td>900</td>
<td>5</td>
<td>323</td>
<td>113</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>giant</td>
<td>7</td>
<td>-</td>
<td>35</td>
<td>21</td>
<td>135</td>
<td>4</td>
<td>1111</td>
<td>4</td>
<td>984</td>
<td>866</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>giant</td>
<td>8</td>
<td>-</td>
<td>35</td>
<td>21</td>
<td>140</td>
<td>1</td>
<td>979</td>
<td>3</td>
<td>943</td>
<td>646</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Results (II)

<table>
<thead>
<tr>
<th>Set</th>
<th>Nr</th>
<th>Grade</th>
<th>X</th>
<th>Y</th>
<th>K</th>
<th>Fix</th>
<th>Alt</th>
<th>Cov</th>
<th>Ini</th>
<th>Set</th>
<th>Red</th>
<th>GCC</th>
<th>Sub</th>
<th>SAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>giant</td>
<td>1701</td>
<td>hard</td>
<td>45</td>
<td>31</td>
<td>255</td>
<td>10</td>
<td>1731</td>
<td>25</td>
<td>617</td>
<td>178</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>giant</td>
<td>1702</td>
<td>hard</td>
<td>45</td>
<td>31</td>
<td>206</td>
<td>3</td>
<td>1727</td>
<td>11</td>
<td>1497</td>
<td>277</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>giant</td>
<td>1801</td>
<td>hard</td>
<td>45</td>
<td>31</td>
<td>189</td>
<td>12</td>
<td>1468</td>
<td>35</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>giant</td>
<td>1802</td>
<td>hard</td>
<td>45</td>
<td>31</td>
<td>276</td>
<td>11</td>
<td>1517</td>
<td>34</td>
<td>769</td>
<td>217</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>giant</td>
<td>1901</td>
<td>hard</td>
<td>45</td>
<td>31</td>
<td>188</td>
<td>8</td>
<td>1443</td>
<td>36</td>
<td>148</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>giant</td>
<td>1902</td>
<td>hard</td>
<td>45</td>
<td>31</td>
<td>213</td>
<td>4</td>
<td>1813</td>
<td>11</td>
<td>1449</td>
<td>1018</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>giant</td>
<td>2001</td>
<td>hard</td>
<td>45</td>
<td>31</td>
<td>221</td>
<td>16</td>
<td>1217</td>
<td>92</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>giant</td>
<td>2002</td>
<td>hard</td>
<td>45</td>
<td>31</td>
<td>278</td>
<td>6</td>
<td>1617</td>
<td>20</td>
<td>1259</td>
<td>758</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Why Model the Wrong Way?

- We want to mimic human reasoning
- Human don’t have a built-in geost constraint
- Most humans use rules to describe solution process
  - If a cell can only be covered by one room, then it must be assigned to the room
- Both views (decide between pattern and assign cell to room) are used by humans
- At some point people
  - Either invent special global constraints
  - Or use SAC
Conclusions

- Shikaku - interesting little puzzle
- Solved by decomposition and reasoning on implications
- Could be solved by strong global constraint
- We use incremental process adding constraints as required
- Close to human solving method
- One open problem left, same for geost (M. Carlsson)