Chapter 20: Working with Implications

Helmut Simonis

Cork Constraint Computation Centre
Computer Science Department
University College Cork
Ireland

ECLiPSe ELearning Overview

Licence

Outline

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

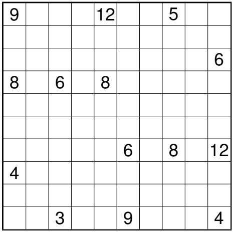
1. Problem
2. Program
3. Improvements

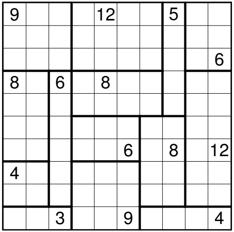
Problem Definition

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

The Right Way (I)

- 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

The Right Way (II)

- Using `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 `regular` constraint for each room
- Possible to combine regular expressions for multiple rooms

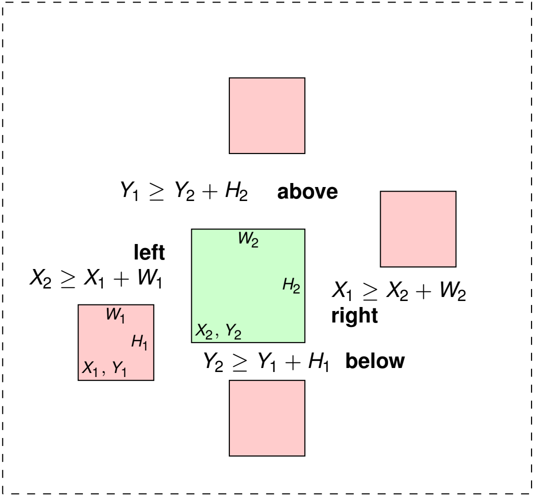
The Right Way (III)

- 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



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 $S_k = w_k * h_k$
  - 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

$$z_{kp} \in \{0, 1\}$$

$$\forall k \in K : \sum_{1 \le p \le P_k} z_{kp} = 1$$

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

Model Phase 2

solve

$$z_{kp} \in \{0, 1\}$$

$$\forall k \in K : \sum_{1 \le p \le P_k} z_{kp} = 1$$

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

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

Top Level

```
:-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).
```

Structure Definitions

```
:-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)).
```

Sample Data

```
:-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 )
   )).
```

Main Program (I)

```
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,
    ...
```

Main Program (II)

```
...
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

```
hint_cell(Matrix,I,J,
          A,[hint{i:I,j:J,n:N,d:D1}|A],
          D,D1):-
    subscript(Matrix,[I,J],N),
    integer(N),
    !,
    D1 is D+1.
hint_cell(_Matrix,_I,_J,A,A,B,B).
```

Creating Pattern

```
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).
```

Phase 1 FD Model

```
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,[]).
```

Phase 1 FD Model Utilities

```
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.
```

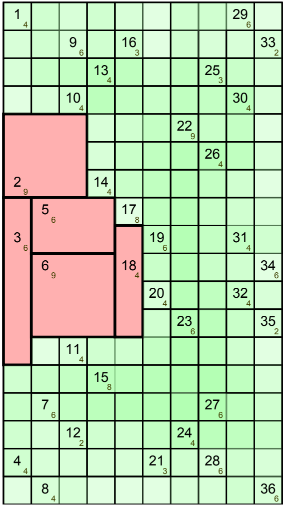
Creating Overlap Structures

```
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
    ).
```

Utility

```
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
     ).
```

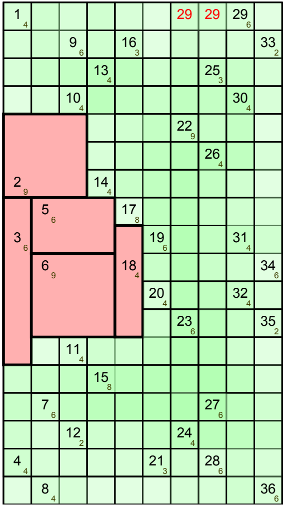
Result After Constraint Setup



Observation

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

Missing Information, Example



Outline

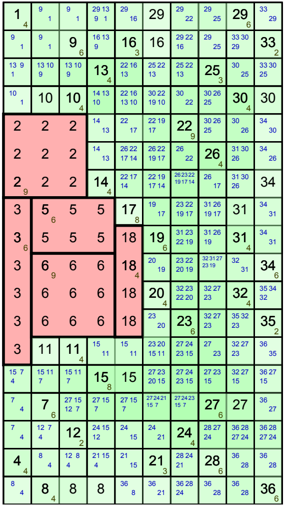
Adding View

- 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
  - $\forall (i,j): \quad \sum_{k \in K} w_{ijk} = 1$

Channeling Constraints

- A cell belongs to a room, if one of the pattern covering the cell is selected
- Linear constraint
  - $\forall(i,j,k): \quad \sum_{\{p|q(k,p,i,j)\}} z_{kp} = w_{ijk}$
- Implications
  - $\forall(k,p,i,j) \text{ s.t. } q(k,p,i,j): \quad z_{kp} \Rightarrow w_{i,j,k}$
- Propagation is equivalent

Result After Improvement



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?

Missing Reasoning: Example

- $A, B, C \in \{0, 1\}$
- $A + B + C = 1 \wedge A + B = 1 \Rightarrow C = 0$
- 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
  - $\forall(i,j,k): \quad \sum_{\{k',p' \mid q(k',p',i,j) \wedge k \neq k'\}} z_{k'p'} + w_{ijk} = 1$
  - $\forall(i,j,k,k',p')$ s.t. $q(k',p',i,j): \quad w_{ijk} \Rightarrow \neg z_{k'p'}$
- If a cell belongs to a room, no pattern for this room which does not cover the cell can be selected
  - $\forall(i,j,k): \quad \sum_{\{p \mid \neg q(k,p,i,j)\}} z_{kp} + w_{ijk} = 1$
  - $\forall(i,j,k,p)$ s.t. $\neg q(k,p,i,j): \quad w_{ijk} \Rightarrow \neg z_{kp}$

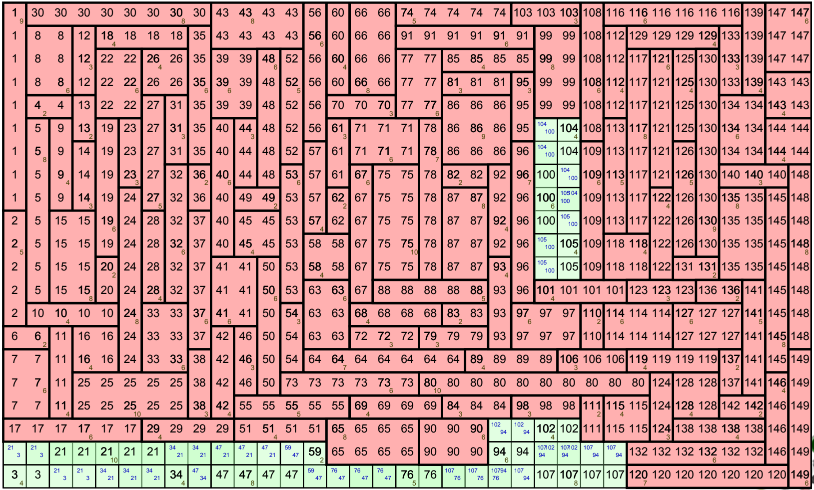
Result After Adding Redundant Constraints

1_{4}	9	9	9	29	29	29	29	29_{6}	29
1	9	9_{6}	9	16_{3}	16	16	25	33	33_{2}
1	10	10	13_{4}	13	13	13	25_{3}	30	30
1	10	10_{4}	14	22	22	22	25	30_{4}	30
2	2	2	14	22	22	22_{9}	26	26	34
2	2	2	14	22	22	22	26_{4}	26	34
2_{9}	2	2	14_{4}	17	17	17	17	31	34
3	5_{6}	5	5	17_{8}	17	17	17	31	34
3_{6}	5	5	5	18	19_{6}	19	19	31_{4}	34
3	6_{9}	6	6	18_{4}	19	19	19	31	34_{6}
3	6	6	6	18	20_{4}	32	32	32_{4}	32
3	6	6	6	18	20	23_{6}	23	23	35_{2}
3	11	11_{4}	11	11	20	23	23	23	35
7	7	7	15_{8}	15	20	24	27	27	27
7	7_{6}	7	15	15	21	24	27_{6}	27	27
4	4	12_{2}	15	15	21	24_{4}	28	28	28
4_{4}	4	12	15	15	21_{3}	24	28_{6}	28	28
8	8_{4}	8	8	36	36	36	36	36	36_{6}

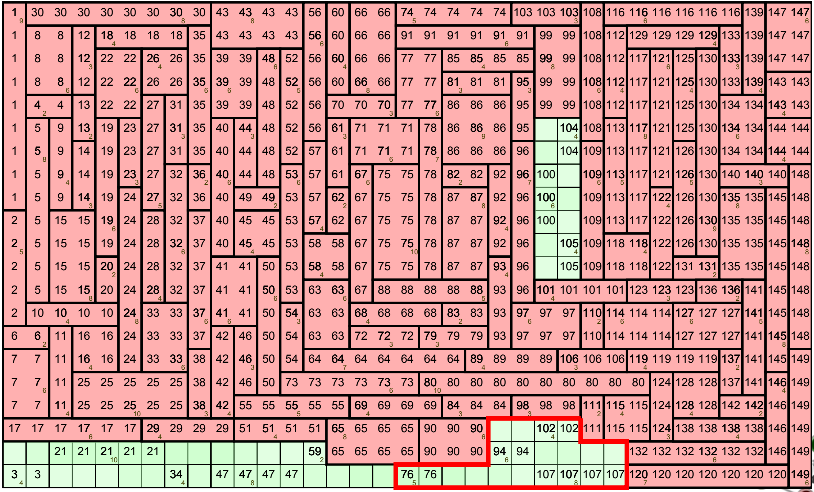
Problem Solved?

- More to be done
- Look through example problems
- Find cases where problem is not solved by propagation
- Try and find redundant constraints

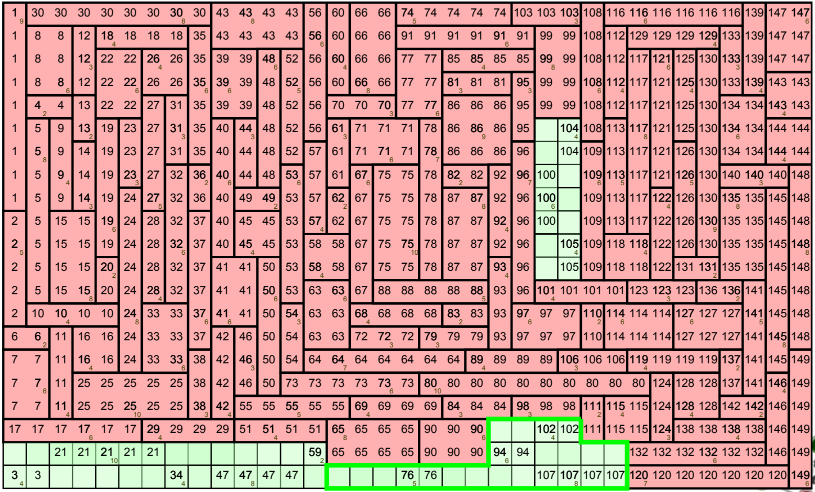
Example



Reasoning on Space Required



Space required for rooms 76, 94, 102, 107



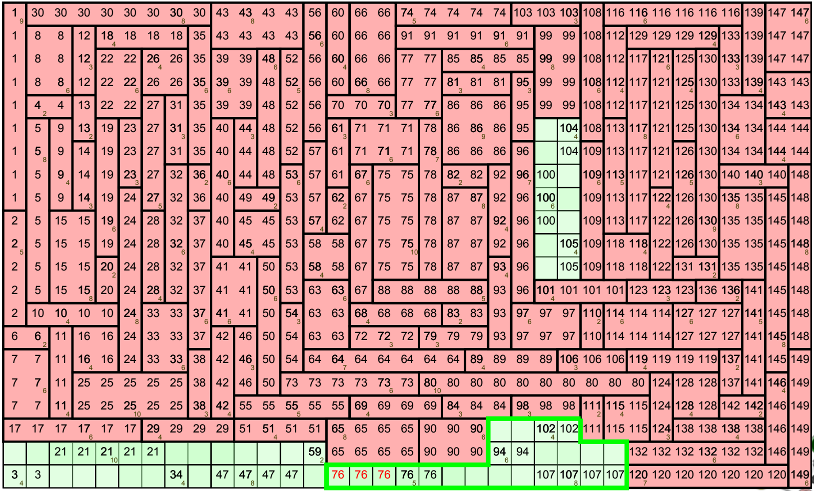
How to model this?

- 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 `bool_channel`(v_{ij},[w_{ij1},w_{ij2},...,$w_{ij|K|}$]) constraints
- Each value must occur specified time
- `gcc` constraint with fixed counts

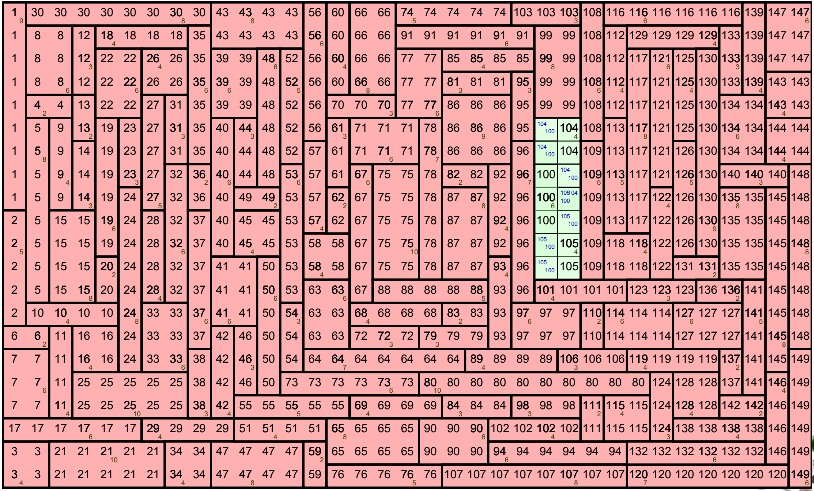
Improvement

- `gcc` reasons on all hints together
- Just adding $\sum_{(i,j)} w_{ijk} = S_k$ does not do this

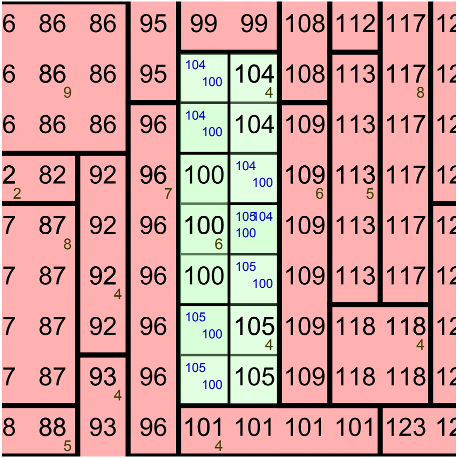
Deduction by gcc (partial)



Result



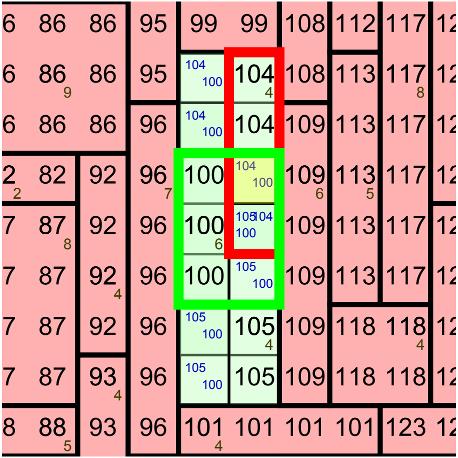
Unsolved Part



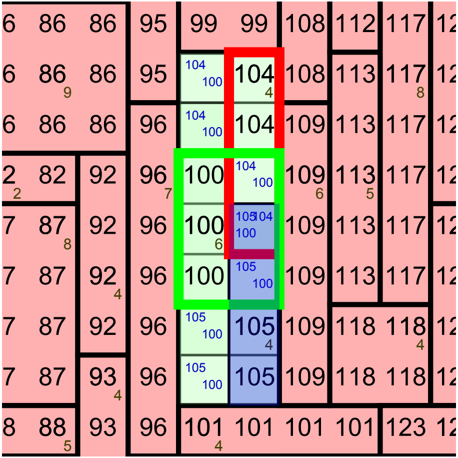
What to do?

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

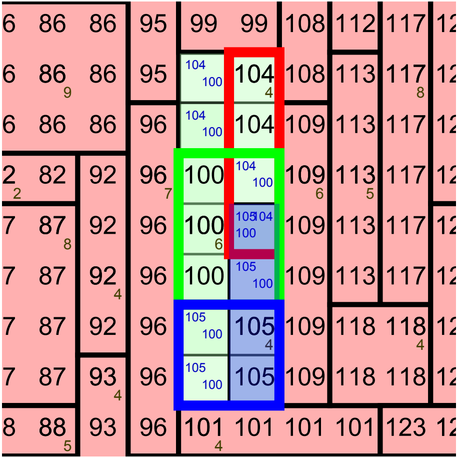
Looking more closely



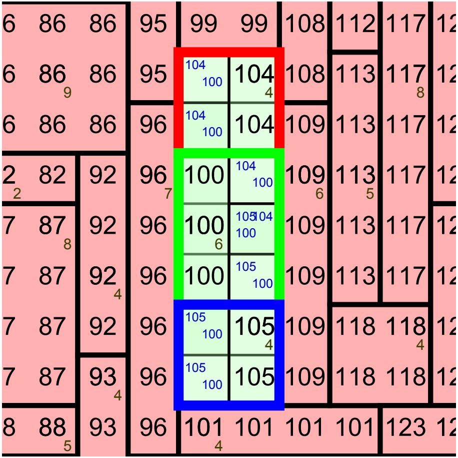
Looking more closely



Looking more closely



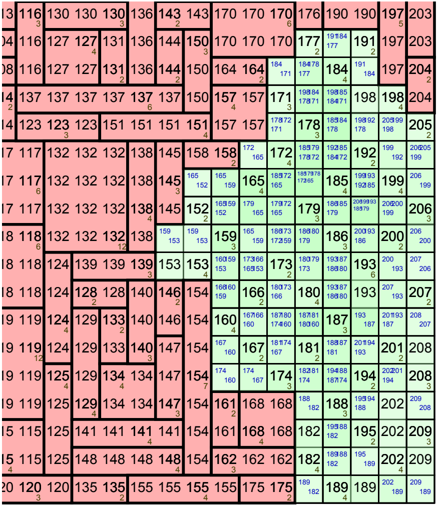
Looking more closely



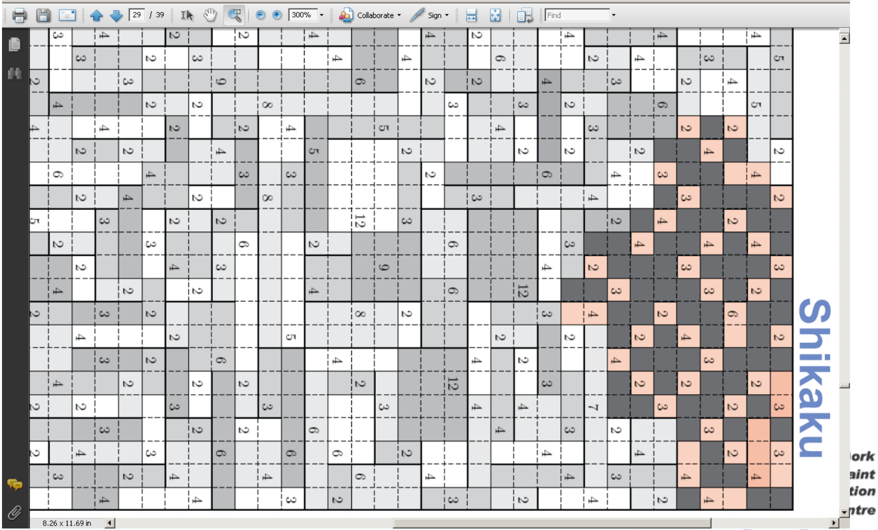
As Equations

- 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



Also Open with `geost` (Image: N. Beldiceanu)



Results (I)

Set	Nr	Grade	X	Y	K	Fix	Alt	Cov	Ini	Set	Red	GCC	Sub	SAC
nikoli-web	0	easy	10	10	20	3	80	6	0	0	0	0	0	0
nikoli-web	1	easy	10	10	14	2	81	5	0	0	0	0	0	0
nikoli-web	2	easy	10	10	16	1	83	2	0	0	0	0	0	0
nikoli-web	3	easy	10	10	28	3	96	12	0	0	0	0	0	0
nikoli-web	4	easy	10	18	44	15	78	78	0	0	0	0	0	0
nikoli-web	5	medium	10	18	38	3	115	21	0	0	0	0	0	0
nikoli-web	6	medium	10	18	36	1	210	3	153	50	0	0	0	0
nikoli-web	7	medium	14	24	68	4	293	16	0	0	0	0	0	0
nikoli-web	8	hard	14	24	70	8	380	17	0	0	0	0	0	0
nikoli-web	9	hard	20	36	128	4	872	8	196	0	0	0	0	0
nikoli-web	10	hard	20	36	120	1	986	1	774	627	36	36	0	0
giant	0	easy	7	7	12	2	46	2	0	0	0	0	0	0
giant	1	-	35	21	149	0	1002	2	995	345	29	7	0	0
giant	2	-	35	21	189	0	888	5	855	324	0	0	0	0
giant	3	-	35	21	209	5	1023	6	830	583	256	256	133	0
giant	4	-	35	21	190	2	849	8	495	70	0	0	0	0
giant	5	-	35	21	137	1	990	3	929	657	0	0	0	0
giant	6	-	35	21	160	2	900	5	323	113	0	0	0	0
giant	7	-	35	21	135	4	1111	4	984	866	7	7	0	0
giant	8	-	35	21	140	1	979	3	943	646	0	0	0	0

Results (II)

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

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)