Using the Global Constraint Seeker for Learning Structured Constraint Models: A First Attempt

Nicolas Beldiceanu and Helmut Simonis

TASC Team (INRIA/CNRS)
Mines des Nantes
France
Cork Constraint Computation Centre
Computer Science Department
University College Cork
Ireland

AICS 2011

Points to Remember

- Learning constraint models from positive and negative examples
- Start with vector of values
- Group into regular pattern
- Find constraint pattern that apply on group elements
- Using *Constraint Seeker* for *Global Constraint Catalog*
- Works for highly structured problems

Outline

Learning Constraint Models

- Constraint models can be hard to write
- Can we generate them automatically?
- User gives example solutions and non-solutions
- System suggests constraints
- User accepts/rejects constraints and/or gives more samples

Constraint Acquisition

- Active research area over last ten years
- Version space learning from AI
- Does not scale for non-binary constraints

Outline

Global Constraint Catalog

- Large collection of global constraints from literature
- Developed over years by SICS and Ecole des Mines
- 354 global constraints described on 2800 pages
- Formal description of constraints available in Prolog
- 270 constraints have executable specification

Constraint Seeker

- CP 2011 paper by Beldiceanu and Simonis
- How to find a constraint in catalog from examples
- Describe what the constraint should do (ground instances)
- System finds ranked list of potential candidate constraints
- On-line tool at `http://seeker.mines-nantes.fr/`

Learning Process

- Start with flat sample
- Group variables in systematic way
- Generate instances of constraints
- Find potential constraint pattern
- Rank by relevance
- Remove implied pattern by dominance checker

Variable Grouping

matrix partition (m_1, m_2, s_1, s_2) treat data as matrix $n = m_1 \times m_2$ and create $s_1 \times s_2$ blocks

diagonal extract main diagonals of $m \times m$ matrix

modulo partition

block partition

sliding window generator

triangular difference table

Generate Instances

- Combine Groups
  - individually
  - as pairs
  - as matrix
- Add arguments
  - as pattern
  - through functional dependency
  - avoid guessing

Relevance Check

- Constraint Program
  - For each group, a variable describes which constraint is used
- Bi-criteria optimization
  - Compactness
  - Ranking

Compactness

- How compact is the selection of constraints
- Ideally, only one constraint used for all groups
- Or, regular pattern with short period
- Or, pattern with few changes

Ranking

- How likely is this constraint for these arguments
- Defined in detail for Constraint Seeker
- Multi-criteria
  - Argument structure (functional dependency, crispness)
  - Solution density (approximation)
  - Importance of constraint

Dominance Check

- Certain constraint pattern are dominated by others
- Weaker than full implication, syntactic check only
- Implications between constraints
- Properties of constraints
  - Contractible (alldifferent)
  - Extensible (atleast)
- New meta-data in constraint catalog

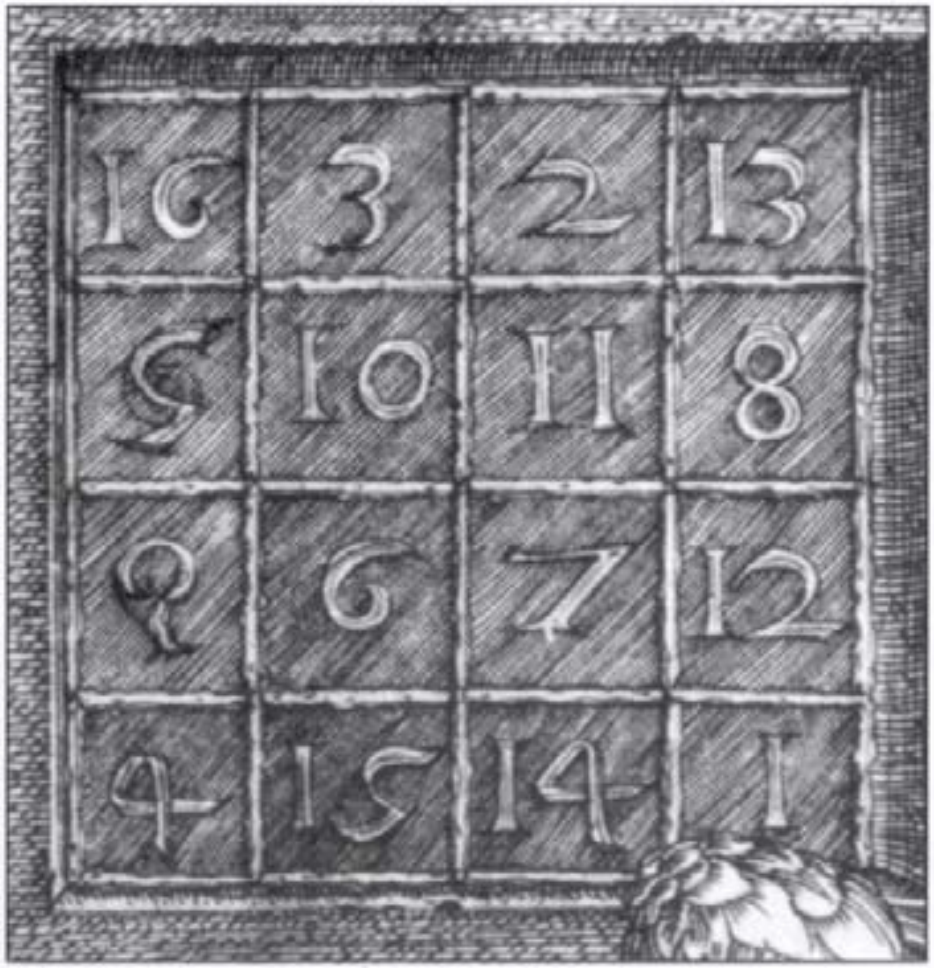
Outline

Magic Square of order n

- Take all numbers from 1 to n^2
- Arrange in $n \times n$ matrix
- All rows, columns and main diagonals must have the same sum

Famous Magic Square (Albrecht Duerer)

Input 16, 3, 2, 13, 5, 10, 11, 8, 9, 6, 7, 12, 4, 15, 14, 1



Generated Constraint Pattern (1)

Generator	matrix(16,1,16,1)
Partition	original sequence of values
Constraint(s)	$1\times$`alldifferent_consecutive_values` $1\times$`symmetric_alldifferent`, extra parameter $[1, 2, \ldots, 16]$

What are these constraints?

alldifferent elements are pairwise different from each other

alldifferent_consecutive_values n elements are alldifferent and range from a to $a + n - 1$

symmetric_alldifferent elements are alldifferent and $x_i = j \implies x_j = i$

Generated Constraint Pattern (2)

Generator	matrix(4,4,1,4)			
Partition	16_1	3_2	2_3	13_4
	5_5	10_6	11_7	8_8
	9_9	6_{10}	7_{11}	12_{12}
	4_{13}	15_{14}	14_{15}	1_{16}
Constraint(s)	$4\times$`sum_ctr`, extra parameters $=$, 34			

Generated Constraint Pattern (3)

Generator	matrix(4,4,4,1)			
Partition	16_1 5_5 9_9 4_{13}	3_2 10_6 6_{10} 15_{14}	2_3 11_7 7_{11} 14_{15}	13_4 8_8 12_{12} 1_{16}
Constraint(s)	$4\times$`sum_ctr`, extra parameters $=$, 34			

Generated Constraint Pattern (4)

Generator	matrix(8,2,4,1)
Partition	16_1 3_2 2_3 13_4 5_5 10_6 11_7 8_8 9_9 6_{10} 7_{11} 12_{12} 4_{13} 15_{14} 14_{15} 1_{16}
Constraint(s)	$4\times$`sum_ctr`, extra parameters $=$, 34

Generated Constraint Pattern (5)

Generator	matrix(2,8,2,2)
Partition	16_1 3_2 2_3 13_4 5_5 10_6 11_7 8_8 9_9 6_{10} 7_{11} 12_{12} 4_{13} 15_{14} 14_{15} 1_{16}
Constraint(s)	$4\times$`sum_ctr`, extra parameters $=$, 34

Generated Constraint Pattern (6)

Generator	diagonal
Partition	16_1 3_2 2_3 13_4 5_5 10_6 11_7 8_8 9_9 6_{10} 7_{11} 12_{12} 4_{13} 15_{14} 14_{15} 1_{16}
Constraint(s)	$2\times$`sum_ctr`, extra parameters $=$, 34 $2\times$`strictly_decreasing`

Solutions to Generated Model

13	3	2	16
8	10	11	5
12	6	7	9
1	15	14	4

13	2	3	16
8	11	10	5
12	7	6	9
1	14	15	4

16	2	5	11
3	13	10	8
9	7	4	14
6	12	15	1

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

16	2	3	13
5	11	10	8
9	7	6	12
4	14	15	1

Can we learn basic model from random, positive samples?

- Select random subset of all solutions to 4x4 magic squares
- See how many constraint pattern are suggested
- Four constraint pattern required for basic magic square model
- Converges quite rapidly

Number of Pattern Found in 100 Runs, for Different Sample Sizes

Size	1	2	3	4	5	6	7	8	9	10	11
1	-	-	-	28	26	19	15	2	5	2 -	3
2	-	-	-	69	25	3	2	-	-	-	1
3	-	-	-	87	12	1	-	-	-	-	-
4	-	-	-	93	6	1	-	-	-	-	-
5	-	-	-	95	5	-	-	-	-	-	-
6	-	-	-	99	1	-	-	-	-	-	-
7	-	-	-	98	2	-	-	-	-	-	-
8	-	-	-	100	-	-	-	-	-	-	-
9	-	-	-	100	-	-	-	-	-	-	-

Balanced Incomplete Block Designs (v, b, r, k, λ)

- Consists of v distinct items and b blocks
- Each block contains k distinct objects
- Each item occurs in exactly r distinct blocks
- Two distinct items occur together in exactly λ blocks

Sample (7,7,3,3,1) Design

0	0	0	0	1	1	1
0	0	1	1	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	1

Constraint Pattern Found

Partition	Constraints
matrix(7,7,7,1)	all pairs: 21×`scalar_product` 7×`sum_ctr`, extra parameters =, 3 matrix: 1×`lex_chain_less` all pairs: 21×`lex_less`
matrix(7,7,1,7)	all pairs: 21×`scalar_product` 7×`sum_ctr`, extra parameters =, 3 matrix: 1×`lex_chain_less` all pairs: 21×`lex_less`
diagonal	2×`no_peak`

Orthogonal Latin Squares

Latin Square of order n A $n \times n$ matrix containing values 1 to n, such that each row and column contains each number from 1 to n exactly once

Orthogonal Latin Squares Two Latin Squares (a_{ij}) and (b_{ij}) are orthogonal, if the pairs $< a_{ij}, b_{ij} >$ are pairwise different

Example Orthogonal Latin Squares

0	2	1	3	4	6	5
6	1	4	2	5	3	0
1	0	6	5	3	4	2
2	5	0	4	6	1	3
5	3	2	6	1	0	4
3	4	5	1	0	2	6
4	6	3	0	2	5	1

0	6	5	2	1	4	3
5	0	4	3	2	1	6
2	1	3	5	0	6	4
1	4	2	0	6	3	5
6	3	0	1	4	5	2
4	5	1	6	3	2	0
3	2	6	4	5	0	1

data given as vector

0, 2, 1, 3, 4, 6, 5, 6, 1, 4, 2, 5, 3, 0, 1, 0, 6, 5, 3, 4, 2, 2, 5, 0, 4,
6, 1, 3, 5, 3, 2, 6, 1, 0, 4, 3, 4, 5, 1, 0, 2, 6, 4, 6, 3, 0, 2, 5, 1,
0, 6, 5, 2, 1, 4, 3, 5, 0, 4, 3, 2, 1, 6, 2, 1, 3, 5, 0, 6, 4, 1, 4, 2, 0,
6, 3, 5, 6, 3, 0, 1, 4, 5, 2, 4, 5, 1, 6, 3, 2, 0, 3, 2, 6, 4, 5, 0, 1

Constraint Pattern Found

Partition	Constraints
matrix(14,7,7,1)	14×`alldifferent_consecutive_values`
matrix(14,7,1,7)	14×`alldifferent_consecutive_values`
matrix(2,49,2,1)	1×`lex_alldifferent`
matrix(7,14,7,7)	2×`sum_ctr` with extra parameters $=$, 147
matrix(7,14,7,1) matrix(14,7,7,7) matrix(49,2,7,1) matrix(7,14,7,2) matrix(14,7,2,7) matrix(14,7,1,7)	1×`lex_alldifferent`

Points to Remember

- Learning constraint models from positive and negative examples
- Start with vector of values
- Partition into regular pattern
- Find constraints that apply on partition elements
- Using *Constraint Seeker* for *Global Constraint Catalog*
- Works for highly structured problems