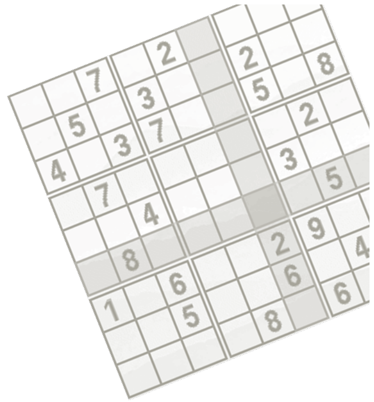


Constraint Programming

Roman Barták

Department of Theoretical Computer Science and Mathematical Logic

Constraint Modeling

- Exploiting the principles of constraint satisfaction, but **programming them ad-hoc** for a given problem.
  - flexibility (complete customisation to a given problem)
  - speed (for a given problem)
  - expensive in terms of initial development and maintenance

- Exploiting an **existing constraint solver**.
  - usually integrated to a host language as a library
  - contains core constraint satisfaction algorithms
  - the user can focus on problem modelling
  - It is hard to modify low-level implementation (domains,…)
  - sometimes possible to implement own constraints
  - frequently possible to implement own search strategies

A typical structure of constraint models:

```
declare_variables( Variables),
post_constraints( Variables),
labeling( Variables ).
```

Definition of variables and their domains

Definition of constraints

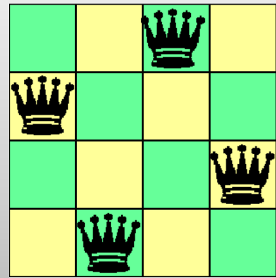
Declarative model

Control part
- **exploration of space of assignments**
- **assigning values to variables**
- **looking for one, all, or optimal solution**

N-queens

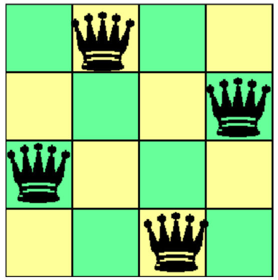
Propose a constraint model for solving the **N-queens problem** (place N queens to a chessboard of size NxN such that there is no conflict).

```
Variables: X_1,…,X_n, Y_1,…,Y_n
Domain: 1,…,N
Constraints:
   all_different({X_1,…,X_n}),
   all_different({Y_1,…,Y_n}),
   ∀i<j: |X_i - X_j| \= |Y_i - Y_j|
```



Solutions (for 4 queens) in the form (X_i,Y_i)

```
[(1,2),(2,4),(3,1),(4,3)]
[(1,3),(2,1),(3,4),(4,2)]
[(1,2),(2,4),(4,3),(3,1)]
[(1,3),(2,1),(4,2),(3,4)]
[(1,2),(3,1),(2,4),(4,3)]
[(1,3),(3,4),(2,1),(4,2)]
[(1,2),(3,1),(4,3),(2,4)]
[(1,3),(3,4),(4,2),(2,1)]
…
```



Where is the problem?

- Different assignments describe the same solution!
- There are only two different solutions (very „similar“ solutions).
- The search space is non-necessarily large.

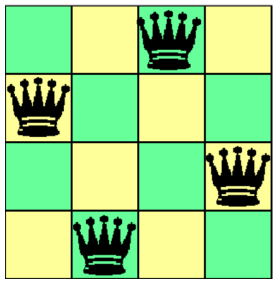
N-queens: a better model

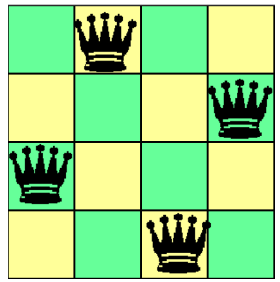
Pre-assign queens to columns, use only variables for rows

```
Variables: X1,…,Xn
Domain: 1,…,N
Constraints:
   all_different({X1,…,Xn}),
   ∀i<j: |Xi – Xj| \= j-i
```

Solutions (for 4 queens) in the form (X_i,Y_i)

```
[2,4,1,3]
[3,1,4,2]
```





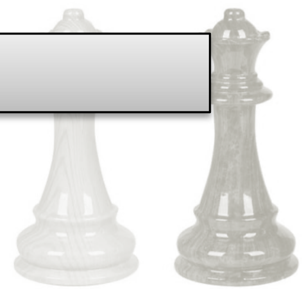
Model properties:

- fewer variables (= smaller state space)
- fewer constraints (= faster propagation)

Remove symmetrical solutions:

```
X1 =< ceiling(N/2)
```

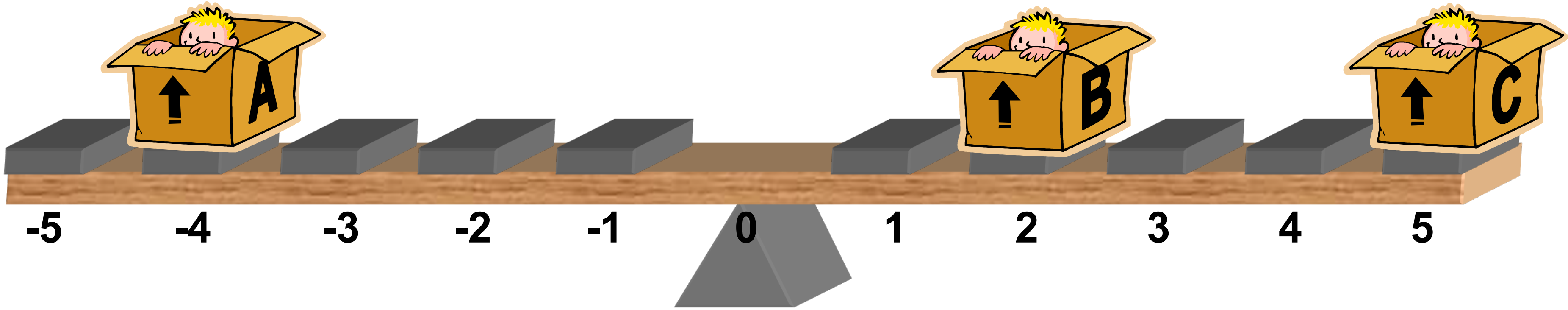
a so-called **symmetry breaking constraint**



Seesaw problem

The problem:

Adam (36 kg), Boris (32 kg) and Cecil (16 kg) want to sit on a seesaw with the length 10 foots such that the minimal distances between them are more than 2 foots and the seesaw is balanced.



A constraint model:

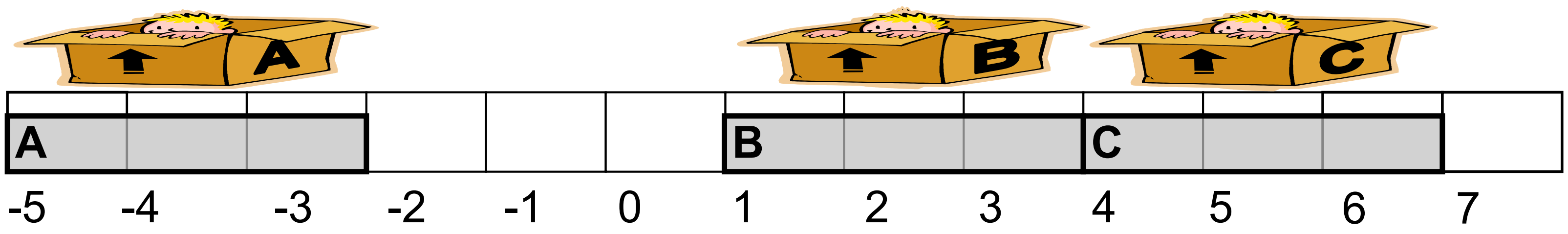
`A,B,C in -5..5`	position
`36*A+32*B+16*C = 0`	equilibrium state
`|A-B|>2, |A-C|>2, |B-C|>2`	minimal distances

Seesaw problem: a different perspective

```
A,B,C in -5..5,
A =< 0,
36*A+32*B+16*C = 0,
abs(A-B)>2,
abs(A-C)>2,
abs(B-C)>2
```

```
A in -4..0
B in -1..5
C in -5..5
```

- A set of similar constraints typically indicates a structured sub-problem that can be represented using a **global constraint.**



- We can use a global constraint describing **allocation of activities to exclusive resource**.

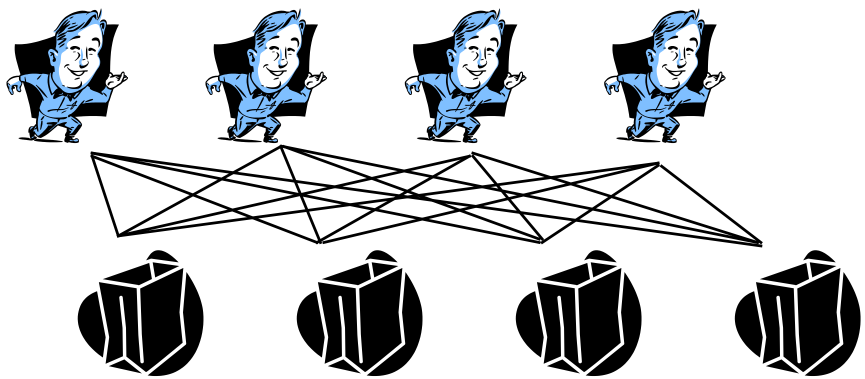
```
A,B,C in -5..5,
A =< 0,
36*A+32*B+16*C = 0,
cumulative([task(A,3,_,1,1),task(B,3,_,1,2),
            task(C,3,_,1,3)],[limit(1)]),
```

task(start,duration,end,capacity,id)

```
A in -4..0
B in -1..5
C in (-5..-3) \/ (-1..5)
```

The problem:

There are 4 workers and 4 products and a table describing the efficiency of producing the product by a given worker. The task is assign workers to products (one to one) in such a way that the total efficiency is at least 19.



	P1	P2	P3	P4
W1	7	1	3	4
W2	8	2	5	1
W3	4	3	7	2
W4	3	1	6	3

A constraint model:

`W1,W2,W3,W4 in 1..4` a product per worker

`all_different([W1,W2,W3,W4])` different products

$T_{1,W1}+T_{2,W2}+T_{3,W3}+T_{4,W4} \geq 19$ total efficiency

Why do we **assign products to workers?**

Cannot we do it in an opposite way, that is, to **assign a worker to a product?**

Of course, we can **swap the role of values and variables!**

- This new model is called a **dual model.**

```
:-use_module(library(clpfd)).

assignment_dual(Products):-
    Products = [P1,P2,P3,P4],

    domain(Products,1,4),
    all_different(Products),
    element(P1,[7,8,4,3],EP1),
    element(P2,[1,2,3,1],EP2),
    element(P3,[3,5,7,6],EP3),
    element(P4,[4,1,2,3],EP4),
    EP1+EP2+EP3+EP4 #>= 19,

    labeling([ff],Products).
```

element(X,List,Y) ⇔ $List_X = Y$

```
P1 in 1..2
P2 in 1..4
P3 in 2..4
P4 in 1..4
```

Number of choice points	
Primal model	**15**
Dual model	**11**

Which model is better?

- In this particular case, the dual model propagates earlier (thus it is assumed to be better).

We can combine both primal and dual model in a single model to get better domain pruning.

```
:-use_module(library(clpfd)).

assignment_combined(Workers):-
    Workers= [W1,W2,W3,W4],
    domain(Workers,1,4),
    all_different(Workers),
    element(W1,[7,1,3,4],EW1),
    element(W2,[8,2,5,1],EW2),
    element(W3,[4,3,7,2],EW3),
    element(W4,[3,1,6,3],EW4),
    EW1+EW2+EW3+EW4 #>= 19,

    Products = [P1,P2,P3,P4],
    domain(Products,1,4),
    all_different(Products),
    element(P1,[7,8,4,3],EP1),
    element(P2,[1,2,3,1],EP2),
    element(P3,[3,5,7,6],EP3),
    element(P4,[4,1,2,3],EP4),
    EP1+EP2+EP3+EP4 #>= 19,

    assignment(Workers,Products),

    labeling([ff],Workers).
```

a primal model

```
W1 in (1..2)\/{4}
W2 in 1..4
W3 in 2..4
W4 in 2..4
```

a dual model (redundant)

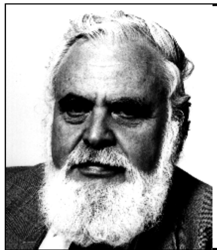
```
P1 in 1..2
P2 in 1..4
P3 in 2..4
P4 in 1..4
```

a channelling constraint

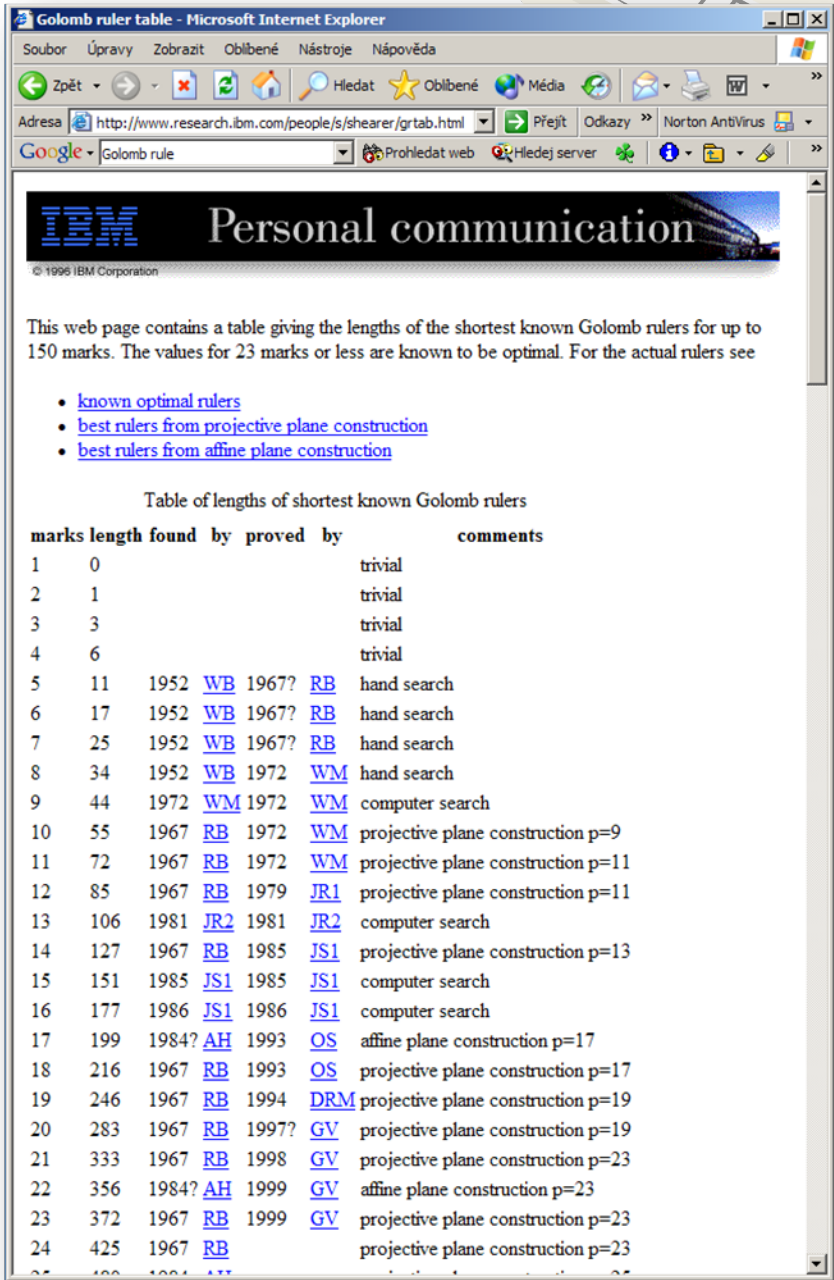
labelling one model is enough

- A **ruler with M marks** such that **distances** between any two marks are **different**.
- The **shortest ruler** is the optimal ruler.

- **Hard** for M≥16, no exact algorithm for M ≥ 24!
- Applied in **radioastronomy**.



Solomon W. Golomb
Professor
University of Southern California
http://csi.usc.edu/faculty/golomb.html



This web page contains a table giving the lengths of the shortest known Golomb rulers for up to 150 marks. The values for 23 marks or less are known to be optimal. For the actual rulers see

- known optimal rulers
- best rulers from projective plane construction
- best rulers from affine plane construction

Table of lengths of shortest known Golomb rulers

marks	length	found	by	proved	by	comments
1	0					trivial
2	1					trivial
3	3					trivial
4	6					trivial
5	11	1952	WB	1967?	RB	hand search
6	17	1952	WB	1967?	RB	hand search
7	25	1952	WB	1967?	RB	hand search
8	34	1952	WB	1972	WM	hand search
9	44	1972	WM	1972	WM	computer search
10	55	1967	RB	1972	WM	projective plane construction p=9
11	72	1967	RB	1972	WM	projective plane construction p=11
12	85	1967	RB	1979	JR1	projective plane construction p=11
13	106	1981	JR2	1981	JR2	computer search
14	127	1967	RB	1985	JS1	projective plane construction p=13
15	151	1985	JS1	1985	JS1	computer search
16	177	1986	JS1	1986	JS1	computer search
17	199	1984?	AH	1993	OS	affine plane construction p=17
18	216	1967	RB	1993	OS	projective plane construction p=17
19	246	1967	RB	1994	DRM	projective plane construction p=19
20	283	1967	RB	1997?	GV	projective plane construction p=19
21	333	1967	RB	1998	GV	projective plane construction p=23
22	356	1984?	AH	1999	GV	affine plane construction p=23
23	372	1967	RB	1999	GV	projective plane construction p=23
24	425	1967	RB			projective plane construction p=23

Golomb ruler – a model

A base model:

Variables $X_1, \ldots, X_M$ with the domain 0..M*M

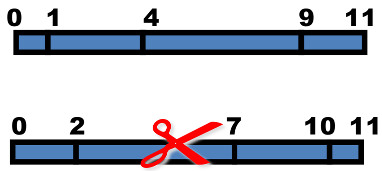
$X_1 = 0$ *ruler start*

$X_1 < X_2 < \ldots < X_M$ *no permutations of variables*

$\forall i<j \; D_{i,j} = X_j - X_i$ *difference variables*

all_different($\{D_{1,2}, D_{1,3}, \ldots D_{1,M}, D_{2,3}, \ldots D_{M,M-1}\}$)

Model extensions:



$D_{1,2} < D_{M-1,M}$ *symmetry breaking*

better bounds (**implied constraints**) for $D_{i,j}$

$$D_{i,j} = D_{i,i+1} + D_{i+1,i+2} + \ldots + D_{j-1,j}$$

so **$D_{i,j} \geq \Sigma_{j-i} = (j-i)*(j-i+1)/2$** *lower bound*

$$X_M = X_M - X_1 = D_{1,M} = D_{1,2} + D_{2,3} + \ldots D_{i-1,i} + D_{i,j} + D_{j,j+1} + \ldots + D_{M-1,M}$$

$$D_{i,j} = X_M - (D_{1,2} + \ldots D_{i-1,i} + D_{j,j+1} + \ldots + D_{M-1,M})$$

so **$D_{i,j} \leq X_M - (M-1-j+i)*(M-j+i)/2$** *upper bound*

Golomb ruler - some results

- **What is the effect of different constraint models?**

size	base model	base model + symmetry	base model + symmetry + implied constraints
7	220	80	30
8	1 462	611	190
9	13 690	5 438	1 001
10	120 363	49 971	7 011
11	2 480 216	985 237	170 495

time in milliseconds on Mobile Pentium 4-M 1.70 GHz, 768 MB RAM

- **What is the effect of different search strategies?**

size	fail first			leftmost first		
	enum	*step*	*bisect*	*enum*	*step*	*bisect*
7	40	60	40	30	30	30
8	390	370	350	220	190	200
9	2 664	2 384	2 113	1 182	1 001	921
10	20 870	17 545	14 982	8 782	7 011	6 430
11	1 004 515	906 323	779 851	209 251	170 495	159 559

time in milliseconds on Mobile Pentium 4-M 1.70 GHz, 768 MB RAM

Constraint satisfaction is a technology for **declarative solving combinatorial (optimization) problems**.

Constraint modeling

- describing problems as constraint satisfaction problems (variables, domains, constraints)

Constraint satisfaction

- local search techniques
- combination of depth-first search with inference (constraint propagation/consistency techniques)
- ad-hoc algorithms encoded in global constraints
- soft constraints to express preferences

It is easy to model problems in terms of a CSP

… but it is complicated to design solvable models.

Department of Theoretical Computer Science and Mathematical Logic
bartak@ktiml.mff.cuni.cz