

# Introduction to Constraint Satisfaction

**Roman Barták**

Charles University, Prague (CZ)

Problem formulation, historical overview, applications

- **Logic-based puzzle**, whose goal is to enter digits 1-9 in cells of 9×9 table in such a way, that no digit appears twice or more in every row, column, and 3×3 sub-grid.

9	6	3	1	7	4	2	5	8
1	7	8	3	2	5	6	4	9
2	5	4	6	8	9	7	3	1
8	2	1	4	3	7	5	9	6
4	9	6	8	5	2	3	1	7
7	3	5	9	6	1	8	2	4
5	8	9	7	1	3	4	6	2
3	1	7	2	4	6	9	8	5
6	4	2	5	9	8	1	7	3

## A bit of history

**1979:** first published in New York under the name „Number Place“

**1986:** became popular in Japan

**Sudoku** – from Japanese "Sudji wa dokushin ni kagiru"  
"the numbers must be single" or "the numbers must occur once"

**2005:** became popular in the western world

recognised pu... arrived in a m... the late... puzzle helps prevent the... of Alzheimer's and offers... some comfort for short-term illness... through its therapeutic thought... either symmetrical or... symmetrical. People may argue... some approach has advantages... another, but differences in... each type of Sudoku are... "carrés magiques" or... "Sudoku with no letter, we believe that Sudoku... better when they are... as there is greater... to create challenging and... Unlike... Sudoku puzzles... rendering arr... layout obsolete

**SUDOKU**

Complete the grid so that every row, column and every three-by-three box contains the digits 1 to 9. Solve the puzzle using logic and reasoning alone, no maths involved.

Difficulty

9				8		7	3	
	7				2			
	4			1				
			7	9				
8	5	4				9	6	
	9		6		8			
1		3		7	4	2		
	5	2					1	8

More Sudoku at: [sudoku solver.com](http://sudoku solver.com)

## How to find out which digit to fill in?

x	x	6		①	3			
3	9	x					①	
2	1	8				4		

- Use information that each digit appears exactly once in each row and column.

## What if this is not enough?

- Look at columns  
or combine information  
from rows and columns

		6	x	1	3			
3	9		x	x	2		1	
②	1	8	x	x	x	4		
8	7		②					
			8	6	1			
					7		4	9
		3				7		8
	4						2	5
			9	②		3		

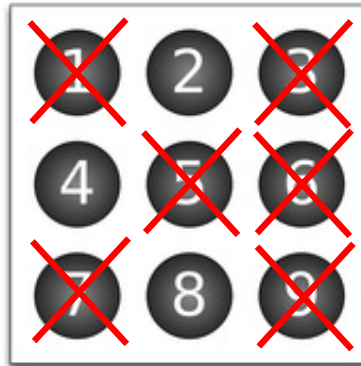
		6		1	3	<sup>2</sup>	×	<sup>2</sup>
3	9				②	×	1	×
②	1	8				4	×	×
8	7		2					
			8	6	1			
					7		4	9
		3				7		8
	4						②	5
			9	2		3		

- If neither rows and columns provide enough information, we can note allowed digits in each cell.

- The position of a digit can be inferred from positions of other digits and restrictions of Sudoku that each digit appears one in a column (row, sub-grid)

	5	6		1	3			
3	9				2		1	
2	1	8				4		
8	⑦		2			6		1
			8	6	1			
					7		4	9
		3				7	9	8
	4					1	2	⑤
			9	2		3	6	4

5	3			7			
6			1	9	5		
	9	8					6
8				6			3
4			8		3		1
7				2			6
	6					2	8
			4	1	9		5
				8			7
							9

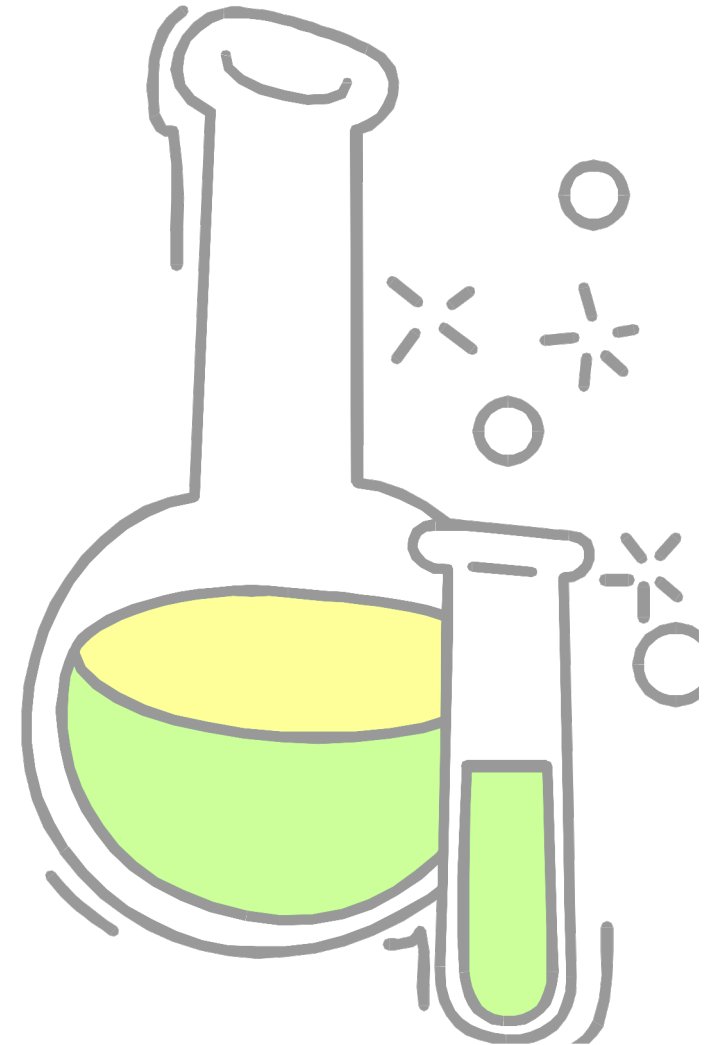


We can see every cell as a **variable** with possible values from **domain**  $\{1, \dots, 9\}$ .

There is a binary inequality **constraint** between all pairs of variables in every row, column, and sub-grid.

Such formulation of the problem is called a **constraint satisfaction problem**.

- 1. Introduction:** problem formulation, historical overview, applications.
- 2. Search approaches:** local search (hill climbing, min-conflicts), depth-first search (backtracking, backjumping, backmarking).
- 3. Local consistency techniques:** arc consistency and algorithms to achieve it (AC-3, AC-4).
- 4. Higher-level consistency techniques:** path-consistency, k-consistency, global constraints.
- 5. Integration** of consistency with search, value/variable ordering heuristics. **Optimization** problems. Problem **modelling**.

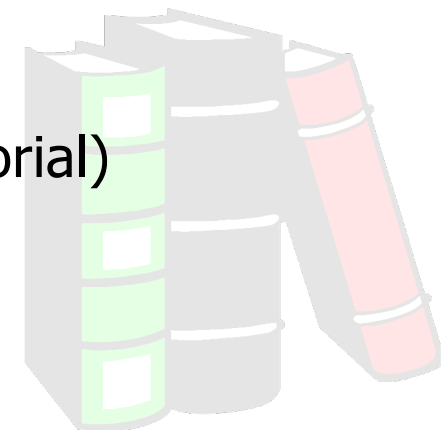


## Books

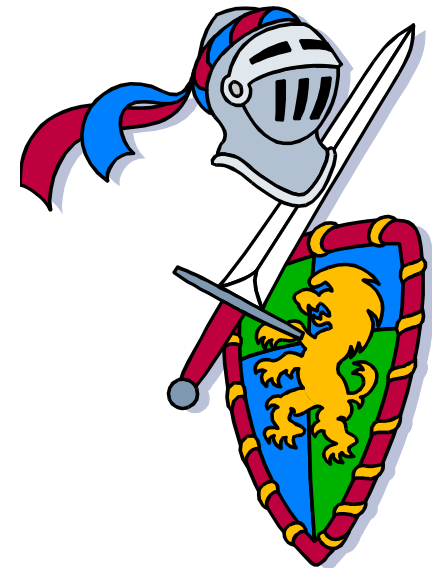
- P. Van Hentenryck: **Constraint Satisfaction in Logic Programming**, MIT Press, 1989
- E. Tsang: **Foundations of Constraint Satisfaction**, Academic Press, 1993
- K. Marriott, P.J. Stuckey: **Programming with Constraints: An Introduction**, MIT Press, 1998
- R. Dechter: **Constraint Processing**, Morgan Kaufmann, 2003
- **Handbook of Constraint Programming**, Elsevier, 2006

## On-line resources

- **Charles University Course Web** (transparencies)  
<http://ktiml.mff.cuni.cz/~bartak/podminky/>
- **On-line Guide to Constraint Programming** (tutorial)  
<http://ktiml.mff.cuni.cz/~bartak/constraints/>



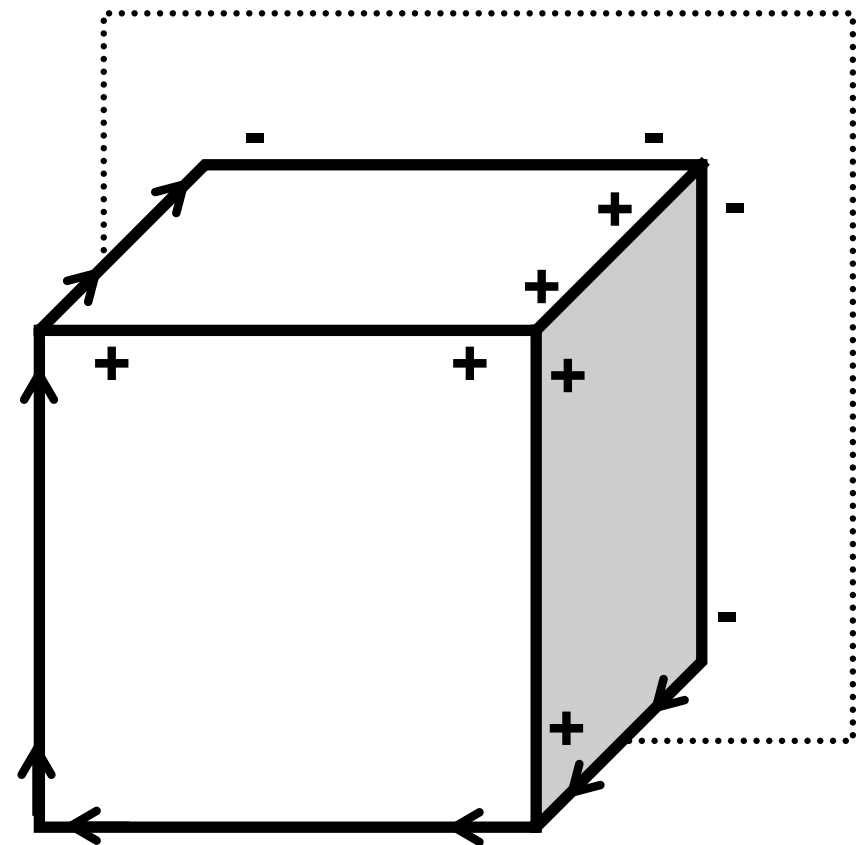
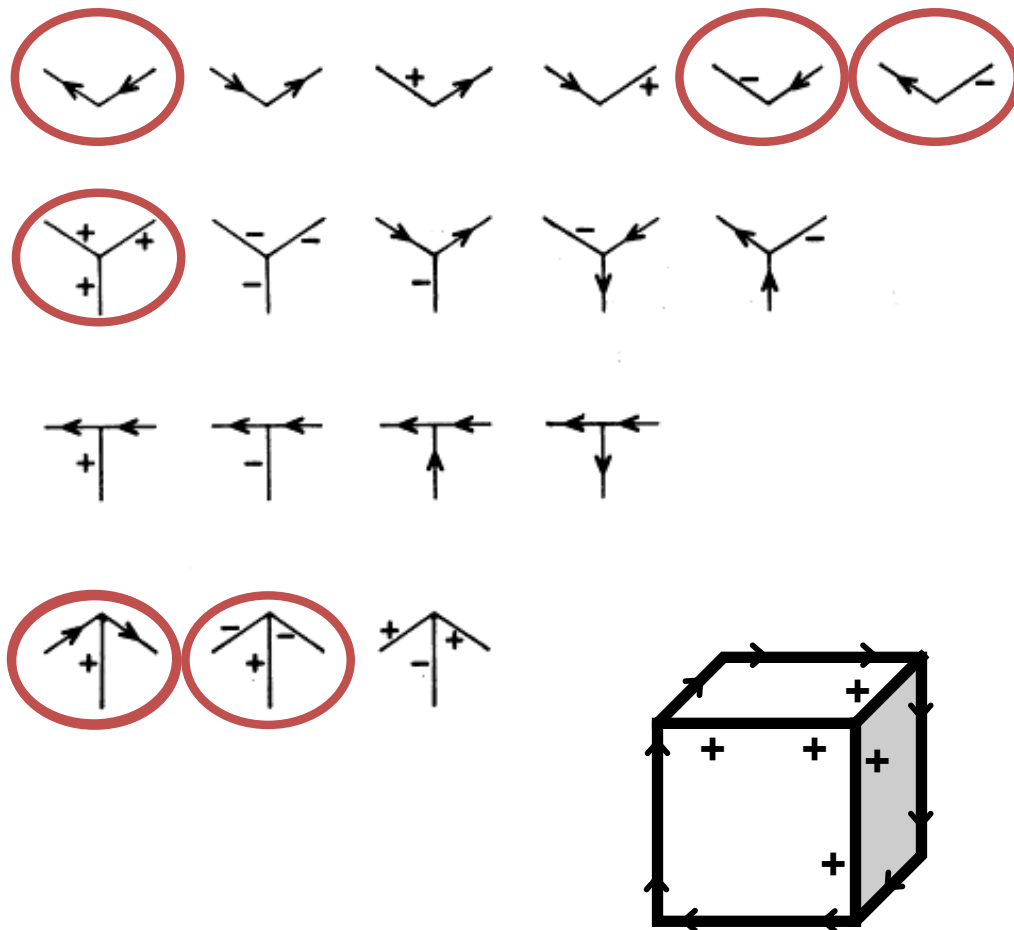
- **Artificial Intelligence**
  - Scene labelling (Waltz 1975)
  - How to help the search algorithm?
- **Interactive Graphics**
  - Sketchpad (Sutherland 1963)
  - ThingLab (Borning 1981)
- **Logic Programming**
  - unification → constraint solving  
(Gallaire 1985, Jaffar, Lassez 1987)
- **Operations Research and Discrete Mathematics**
  - NP-hard combinatorial problems





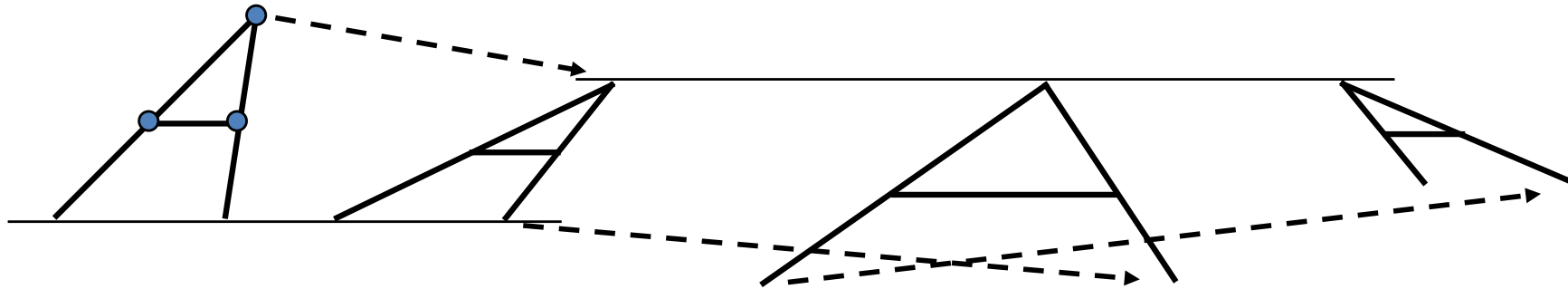
## inferring 3D meaning of lines in a 2D drawing

- convex (+), concave (-) and border ( $\leftarrow$ ) edges
- we are looking for a physically feasible interpretation



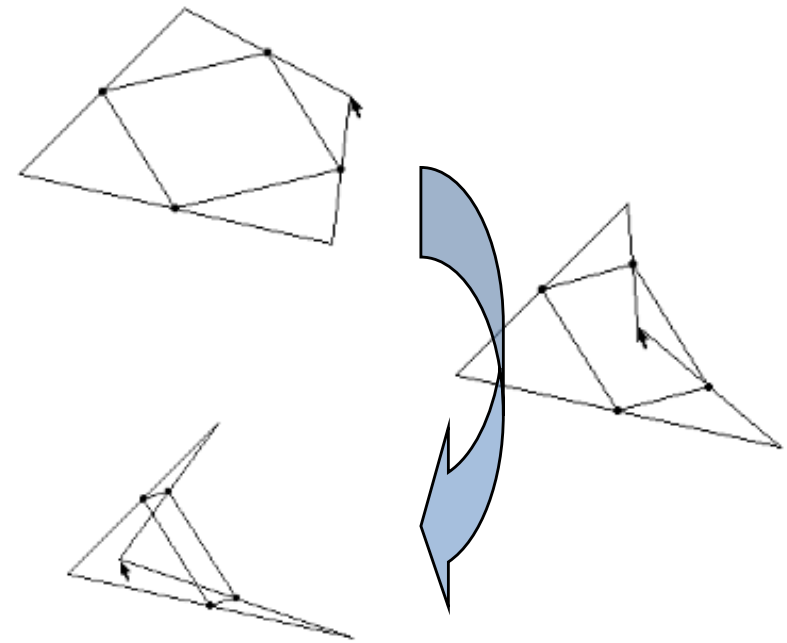
## manipulating graphical objects described via constraints

<http://ktiml.mff.cuni.cz/~bartak/diploma/downloads.html>



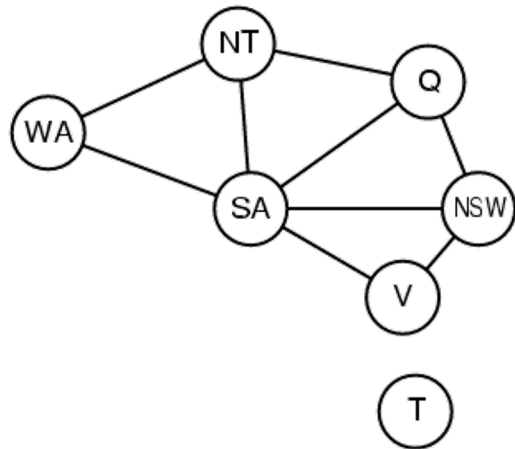
ThingLab Browser

Printer	-----	insert	TextLabel
<b>QTheorem</b>	<b>picture</b>	delete	TextThing
Quadrilateral	structure	constrain	Thermometer
Rectangle	values	merge	<b>ThingLabLine</b>
Resistor	-----	<b>move</b>	ThingSlider
SmallFixedBridge		edit	Times
TextConnector		-----	Triangle
TextDemo			TwoLeadedObject



<http://www.cs.washington.edu/research/constraints/>

Assign colours (red, blue, green) to states, such that neighbours have different colours.



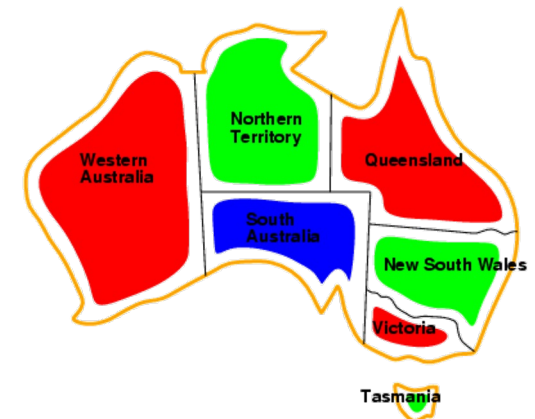
## CSP Model

- variables: {WA, NT, Q, NSW, V, SA, T}
- domains: {r, b, g}
- constraints: WA  $\neq$  NT, WA  $\neq$  SA etc.

Can be described as a **constraint network** (nodes=variables, edges=constraints)

## Solution

WA = r, NT = g, Q = r, NSW = g,  
V = r, SA = b, T = g



Assign digits 0,...,9 to letters S,E,N,D,M,O,R,Y in such a way that:

- SEND + MORE = MONEY
- different letters are assigned to different digits
- S and M are different from 0

### Model 1:

$$\begin{aligned}
 &E, N, D, O, R, Y \text{ in } 0..9, \quad S, M \text{ in } 1..9 \\
 &\qquad\qquad\qquad 1000*S + 100*E + 10*N + D \\
 + &\qquad\qquad\qquad 1000*M + 100*O + 10*R + E \\
 = &10000*M + 1000*O + 100*N + 10*E + Y
 \end{aligned}$$

### Model 2:

using „carry“ 0-1 variables

$$\begin{aligned}
 &E, N, D, O, R, Y \text{ in } 0..9, \quad S, M \text{ in } 1..9, \quad P1, P2, P3 \text{ in } 0..1 \\
 &D+E = 10*P1+Y \\
 &P1+N+R = 10*P2+E \\
 &P2+E+O = 10*P3+N \\
 &P3+S+M = 10*M + O
 \end{aligned}$$

all\_different(S,E,N,D,M,O,R,Y)

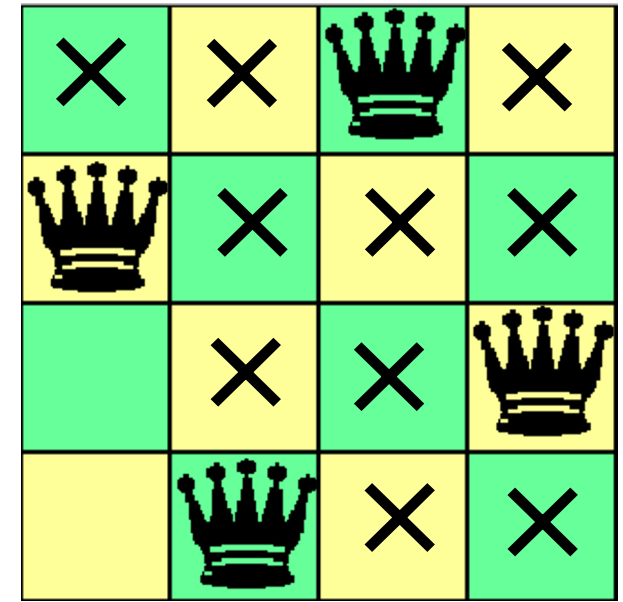
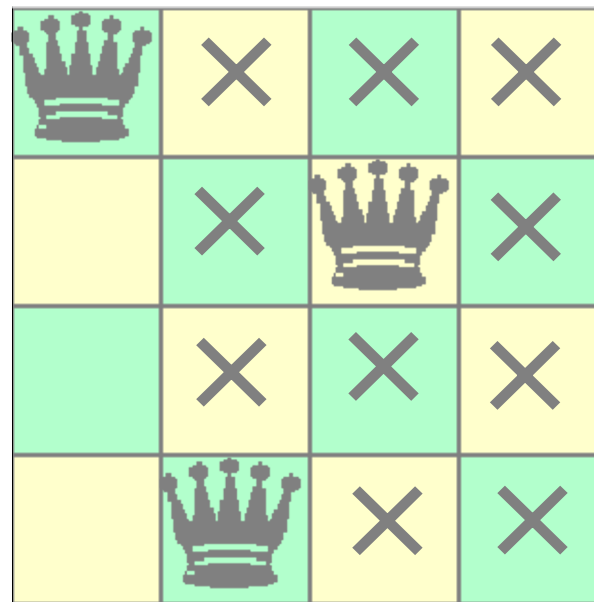
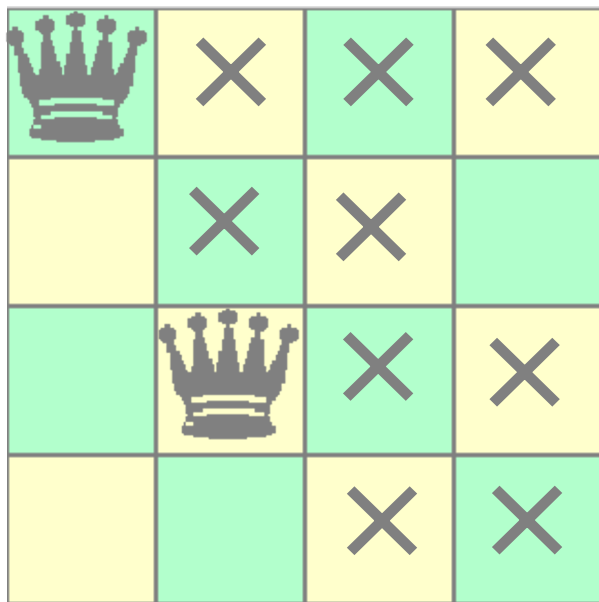
allocate  $N$  queens to a chess board of size  $N \times N$  in a such way that no two queens attack each other

the core decision: each queen is located in its own column

**variables:**  $N$  variables  $r(i)$  with the domain  $\{1, \dots, N\}$

**constraints:** no two queens attack each other

$$\forall i \neq j \quad r(i) \neq r(j) \wedge |i-j| \neq |r(i)-r(j)|$$



## Bioinformatics

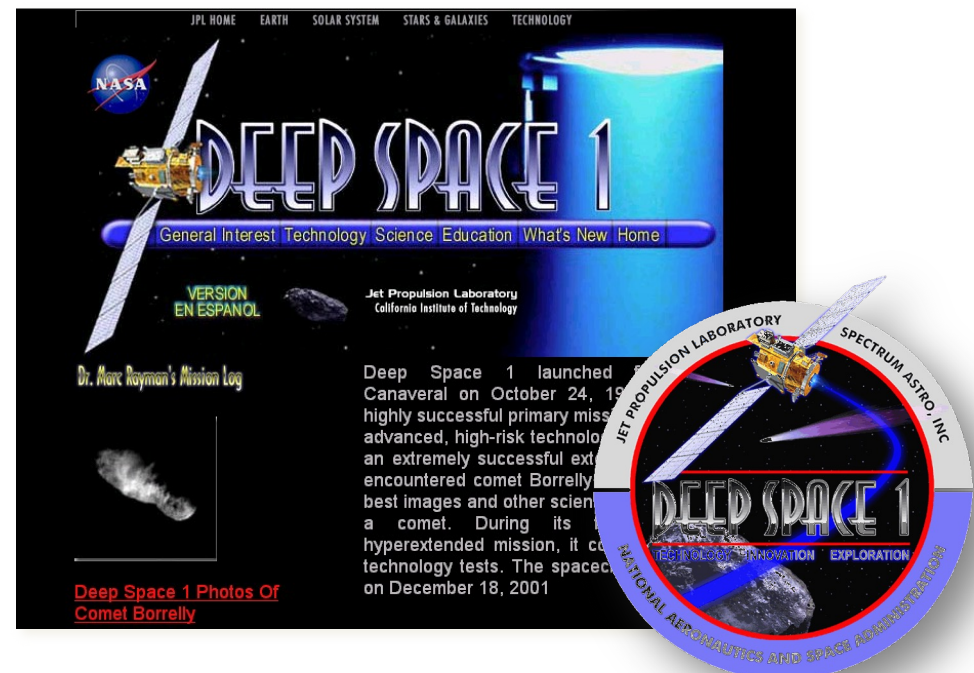
- DNA sequencing (Celera Genomics)
- deciding the 3D structure of proteins from the sequence of amino acids



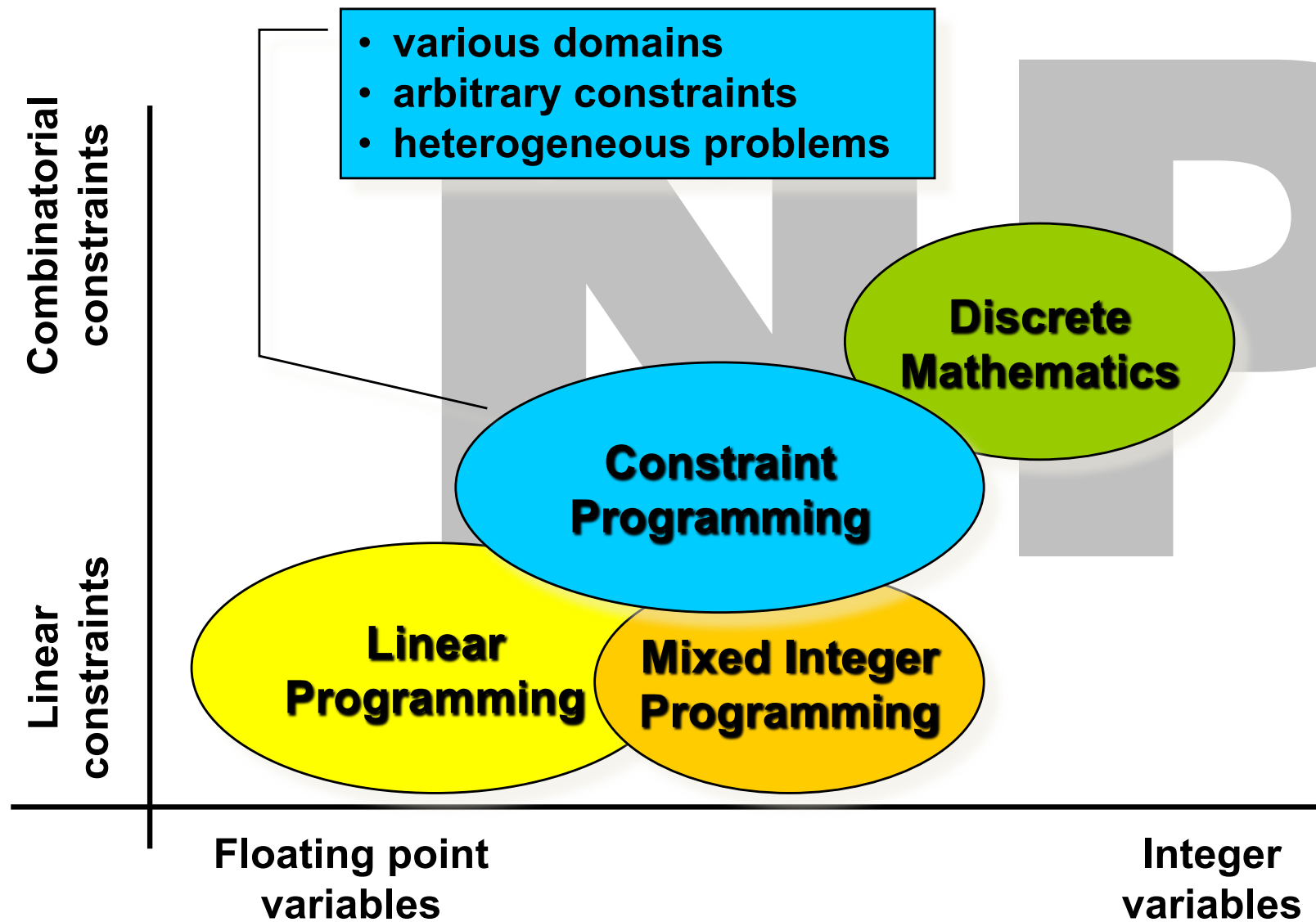
The image shows a screenshot of the Celera Discovery website. The website header includes the Celera logo and navigation links for Therapeutic Discovery, Corporate Info, Investor, Press, Careers, and Online Business. The main content area features a 'Therapeutic Discovery' section with a sub-header 'Discovery' and a paragraph: 'At Celera Genomics, we integrate advanced technologies toward discovery and development of new therapeutics. Leveraging our proteomic, bioinformatic, and genomic capabilities, we seek to identify and validate drug targets and diagnostic marker candidates in the quest for new medicines.' Below this is a 'CELERAUpdate' section with several news items dated from October 8 to November 7, 2002. At the bottom, there is a 'CELERAEvents' section and a 'CELERA DISCOVERY SYSTEM' logo with the tagline 'The essential tool for the life science researcher' and a link to 'www.celeradiscoverysystem.com'. A 3D protein structure, rendered in pink, yellow, and blue, is overlaid on the right side of the website screenshot.

## Planning and Scheduling

- automated planning of spacecraft activities (Deep Space 1)
- manufacturing scheduling



The image shows a screenshot of the Deep Space 1 mission website. The header includes navigation links for JPL HOME, EARTH, SOLAR SYSTEM, STARS & GALAXIES, and TECHNOLOGY. The main content area features the title 'DEEP SPACE 1' in large, stylized letters, with a sub-header 'General Interest Technology Science Education What's New Home'. Below this is a 'VERSION EN ESPANOL' link and a 'Jet Propulsion Laboratory California Institute of Technology' logo. The main text describes the mission: 'Deep Space 1 launched from Canaveral on October 24, 1998... highly successful primary mission... advanced, high-risk technology... an extremely successful extended mission... encountered comet Borrelly... best images and other scientific data... a comet. During its hyperextended mission, it completed... technology tests. The spacecraft was de-orbited on December 18, 2001.' A circular logo for 'DEEP SPACE 1' is overlaid on the bottom right, featuring the mission name and 'JET PROPOSITION LABORATORY' and 'SPECTRUM ASTRO, INC.' around the perimeter. The logo also includes the text 'DEEP SPACE 1' and 'JET PROPOSITION LABORATORY' and 'SPECTRUM ASTRO, INC.' around the perimeter.



## Constraint Satisfaction Problem (CSP) consists of:

- a finite set of **variables**
  - describe attributes of the solution  
for example a location of a queen in the chess board
- **domains** – finite sets of possible values for variables
  - describe options that we need to decide  
for example, rows for queens
  - sometimes, there is a common super domain for all the variables  
and individual variables' domains are defined via unary constraints
- a finite set of **constraints**
  - constraint is a relation over a subset of variables  
for example  $\text{locationA} \neq \text{locationB}$
  - constraint can be defined in extension (a set of compatible value tuples) or using a formula (see above)



**A feasible solution** of a constraint satisfaction problem is a complete consistent assignment of values to variables.

- **complete** = each variable has assigned a value
- **consistent** = all constraints are satisfied

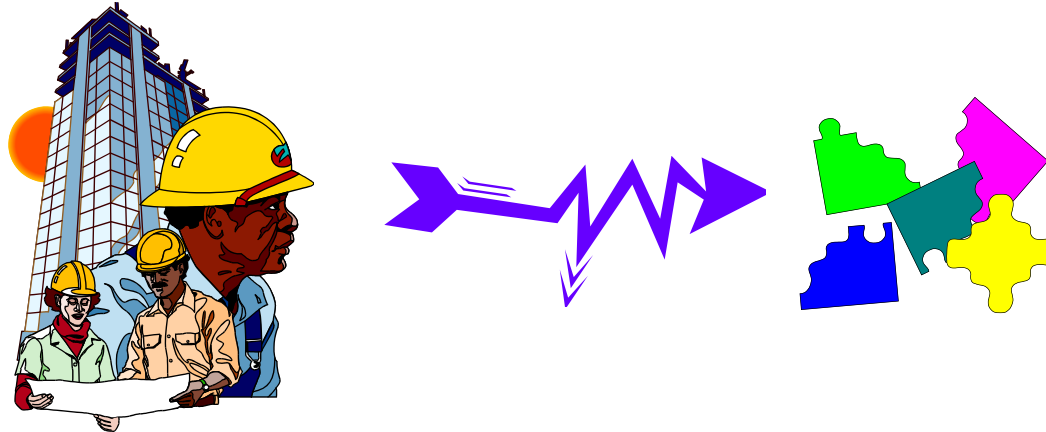
Sometimes we may look for **all** the feasible solutions or for the number of feasible solutions.

**An optimal solution** of a constraint satisfaction problem is a feasible solution that minimizes/maximizes a value of some objective function.

- **objective function** = a function mapping feasible solutions to real numbers

## Problem Modelling

How to describe a problem as a constraint satisfaction problem?



## Solving Techniques

How to find values for the variables satisfying all the constraints?



## Representation of constraints:

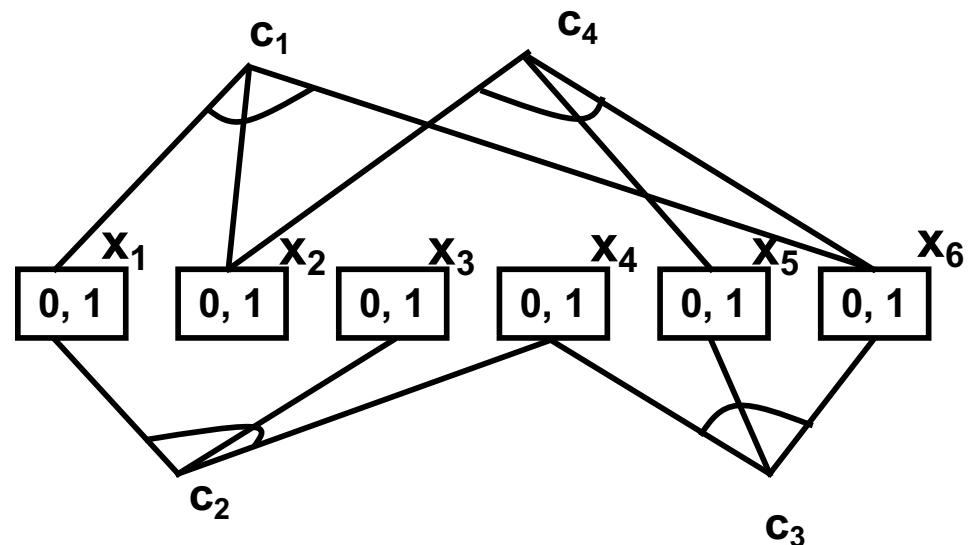
- intentional (algebraic/logic formulae)
- in extension (a set of compatible value tuples, 0-1 matrix)

## Representation of a CSP as a (hyper)graph

- nodes = variables
- (hyper)edges = constraints

## Example:

- variables  $x_1, \dots, x_6$   
with domain  $\{0,1\}$
- $C_1: x_1 + x_2 + x_6 = 1$
- $C_2: x_1 - x_3 + x_4 = 1$
- $C_3: x_4 + x_5 - x_6 > 0$
- $C_4: x_2 + x_5 - x_6 = 0$



The world is not binary ...

but it can be transformed to a binary one!

## **Binary CSP**

CSP + all the constraints are binary

Note: unary constraints can be easily encoded in the domain of a variable

## **Equivalence of CSPs**

Two constraint satisfaction problems are equivalent if they have the same sets of solutions.

## **Extended Equivalence of CSPs**

Problem solutions can be syntactically transformed between the problems.

**Can any CSP be transformed to an (extended) equivalent binary CSP?**

## Swapping variables and constraints.

- k- ary constraint  $c$  is converted to a **dual variable**  $v_c$  with the domain consisting of compatible tuples
- for each pair of constraints  $c$  a  $c'$  sharing some variables there is a **binary constraint** between  $v_c$  a  $v_{c'}$  restricting the dual variables to tuples in which the original shared variables take the same value

### Example:

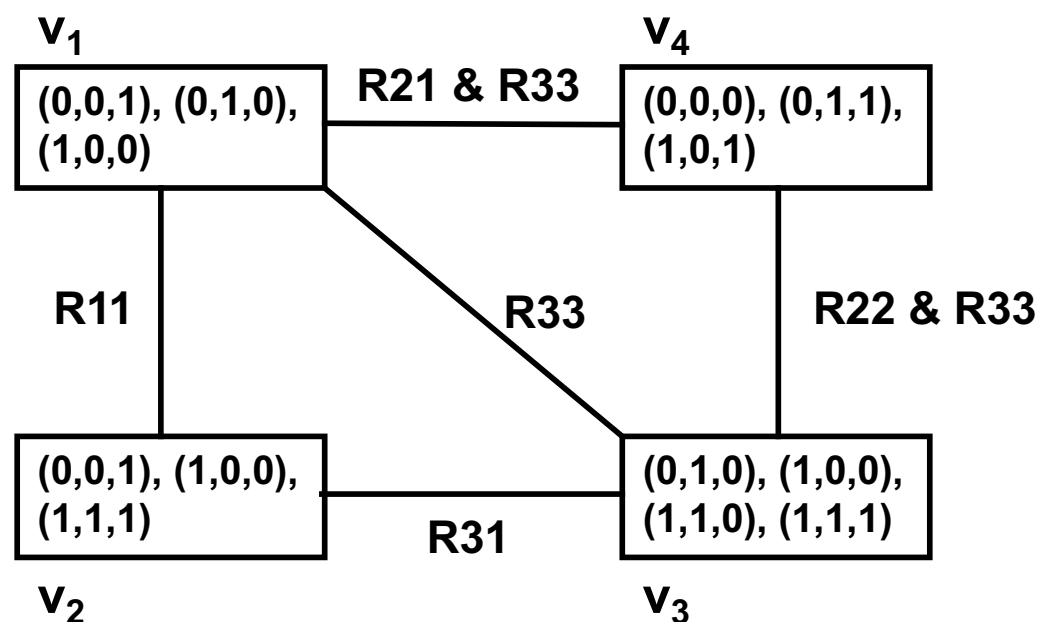
– variables  $x_1, \dots, x_6$   
with domain  $\{0,1\}$

–  $C_1: x_1 + x_2 + x_6 = 1$

–  $C_2: x_1 - x_3 + x_4 = 1$

–  $C_3: x_4 + x_5 - x_6 > 0$

–  $C_4: x_2 + x_5 - x_6 = 0$

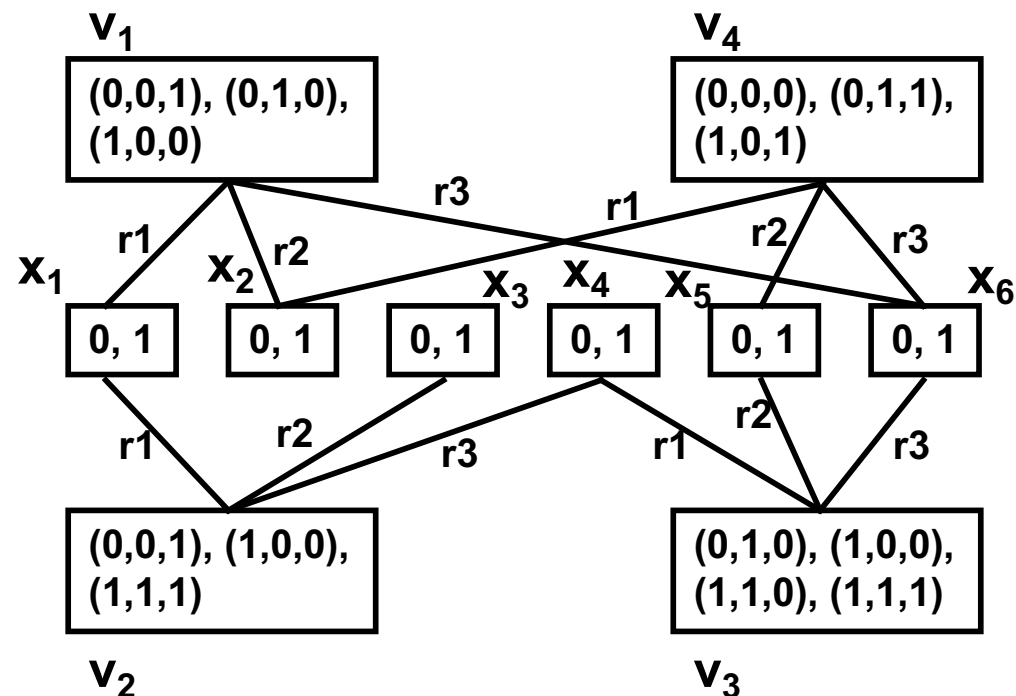


## New dual variables for (non-binary) constraints.

- k-ary constraint  $c$  is translated to a **dual variable**  $v_c$  with the domain consisting of compatible tuples
- for each variable  $x$  in the constraint  $c$  there is a constraint between  $x$  and  $v_c$  restricting tuples of dual variable to be compatible with  $x$

### Example:

- variables  $x_1, \dots, x_6$  with domain  $\{0,1\}$
- $C_1: x_1 + x_2 + x_6 = 1$
- $C_2: x_1 - x_3 + x_4 = 1$
- $C_3: x_4 + x_5 - x_6 > 0$
- $C_4: x_2 + x_5 - x_6 = 0$



## **Why do we do binarisation?**

- a unified form of a CSP
- many solving approaches are formulated for binary CSPs
- tradition (historical reasons)

## **Which encoding is better?**

- hard to say ;-)
- dual encoding:  
better propagation but constraints in extension
- hidden variable encoding:  
keeps original variables but weaker propagation

## **Binary vs non-binary constraints**

- more complex propagation algorithms for non-binary constraints
- exploiting semantics of constraints for more efficient and stronger domain filtering

The goal: **find a complete and consistent instantiation of variables**

Two **core solving approaches**:

- **exploring complete but possibly inconsistent assignments**

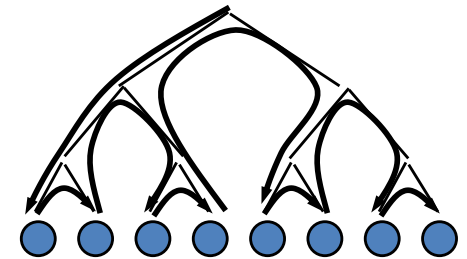
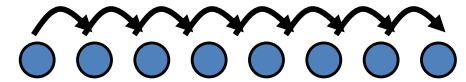
until a consistent assignment is found

- generate and test, local search

- **extending a partial consistent assignment**

until a complete assignment is reached

- backtracking and its extensions



We can explore assignments in two ways:

- **systematically** (explore all possible assignments systematically)

- a complete method, but could be too slow

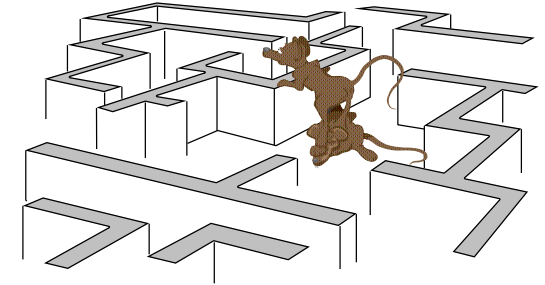
- **non-systematically** (some assignments can be skipped)

- an incomplete method, but can found solution much faster

## Note:

We will use constraints in a *passive way*, just to verify whether the given assignment (even partial) satisfies the constraint.





## Work plan:

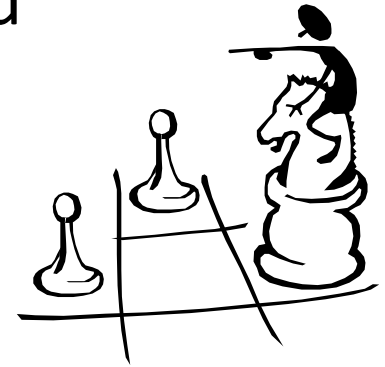
- start simple (with a trivial algorithm)
- find weaknesses of the algorithm
- repair the weaknesses to get better algorithms

## In particular:

- start with **generate and test** method
- **improve the generator**
  - local search methods (HC, RW, TS, GSAT, GENET, SA)
- **merge the generator with the tester**
  - backtracking methods
  - improvements of chronological backtracking
    - backjumping, dynamic backtracking, backmarking

Probably the most general problem solving method

- 1) generate a candidate for solution
- 2) test if the candidate is really a solution



How to apply GT to CSP?

- 1) assign values to all variables
- 2) test whether all the constraints are satisfied

**GT explores complete but inconsistent assignments** until a (complete) consistent assignment is found.

```
procedure GT(X:variables, C:constraints)
```

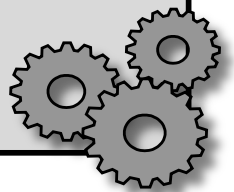
```
  V ← construct a first complete assignment of X
```

```
  while V does not satisfy all the constraints C do
```

```
    V ← construct systematically a complete assignment next to V
```

```
  end while
```

```
  return V
```



The greatest weakness of GT is **exploring too many “visibly” wrong assignments.**

**Example:**

$X::\{1,2\}, Y::\{1,2\}, Z::\{1,2\}$

$X = Y, X \neq Z, Y > Z$



X	1	1	1	1	2	2	2
Y	1	1	2	2	1	1	2
Z	1	2	1	2	1	2	1



## How to improve GT?

### – **smart generator**

- the next assignment improves over the current assignment
- the core idea of local search techniques

### – **merged generate and test stages** (earlier detection of clash)

- constraints are tested as soon as all involved variables are instantiated
- backtracking



© 2024 Roman Barták

Charles University, Faculty of Mathematics and Physics

[bartak@ktiml.mff.cuni.cz](mailto:bartak@ktiml.mff.cuni.cz)