Constraint Programming
– Introduction –

Christophe Lecoutre

CRIL-CNRS UMR 8188
Universite d'Artois
Lens, France

January 2021

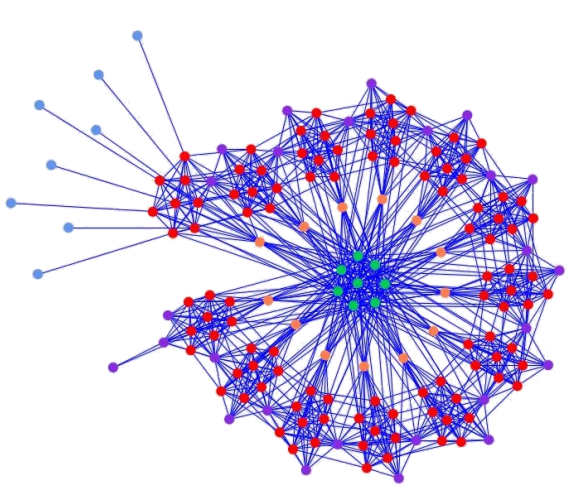
Outline

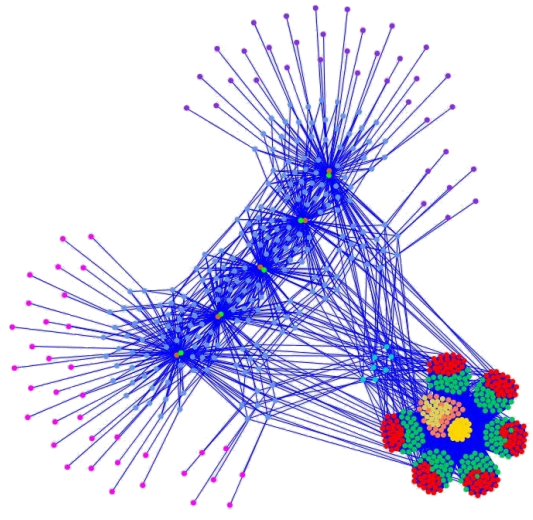
Constraint Programming

is about

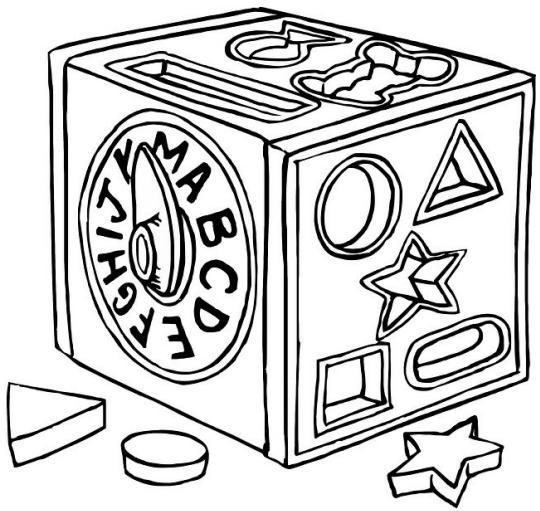
modeling and solving

Problems under Constraints





Certainly, the first Problem you Solved!



This is a matching problem !

Remember of your first Riddles!

Which sequence of successive four integer numbers sum up to 14?

We need four integer variables:

- $x_1 \in \mathbb{N}$, $x_2 \in \mathbb{N}$, $x_3 \in \mathbb{N}$, $x_4 \in \mathbb{N}$

We need some equality constraints:

- $x_1 + 1 = x_2$
- $x_2 + 1 = x_3$
- $x_3 + 1 = x_4$
- $x_1 + x_2 + x_3 + x_4 = 14$

Quite often, you face Optimization Problems!

A witch can elaborate two kinds of magic potions:

- the former for inspiring love
- the latter for restoring youth

Theses potions are composed of dribble of toads (U_t), wings of dragons (U_d) and powder of spiders (U_s), with the following proportions:

- 1 potion for love requires 3 U_t, 1 U_d and 1 U_s
- 1 potion for youth requires 2 U_t, 3 U_d and 2 U_s

Quite Often, Optimization Problems!

The witch possesses 800 U_t, 700 U_d and 400 U_s. Selling magic potions brings in:

- 4 crowns for each love potion
- 5 crowns for each youth potion

Considering her available ingredients, how many love potions (x) and youth potions (y) should be prepared by the witch in order to optimize her benefit?

$$\text{Maximize : } 4x + 5y$$

$$\text{such that: } \begin{cases} 3x + 2y \leq 800 \\ x + 3y \leq 700 \\ x + 2y \leq 400 \end{cases}$$

Your first PyCSP^3 Model

```
from pycsp3 import *

# x is the number of magic love potions
x = Var(range(400))

# y is the number of magic youth potions
y = Var(range(400))

satisfy(
     3*x + 2*y <= 800,
     x +  3*y <= 700,
     x + 2*y <= 400
)

maximize(
     4*x + 5*y
)
```

You use Constraint Programming every day!

- or your grand-parents!

Sudoku

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

You use Constraint Programming every day!

Sudoku

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

Variables

Domains

Fill the empty cells with values in 1..9

Constraints

While considering that:

- each row must contain different numbers
- each column must contain different numbers
- each bloc (3×3 square) must contain different numbers

You use Constraint Programming every day!

Sudoku

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

Solution: assignment of a value to each variable such that no constraint is violated

- A Sudoku puzzle is a very simple (instance of a) Constraint Satisfaction Problem

You use Constraint Programming every day!

Sudoku

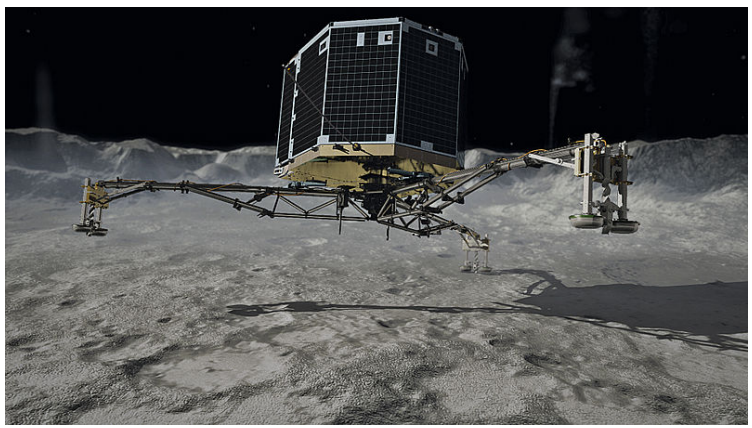
123 456 789	2	123 456 789	5	123 456 789	1	123 456 789	9	123 456 789
8	123 456 789	123 456 789	2	123 456 789	3	123 456 789	123 456 789	6
123 456 789	3	123 456 789	123 456 789	6	123 456 789	123 456 789	7	123 456 789
123 456 789	123 456 789	1	123 456 789	123 456 789	123 456 789	6	123 456 789	123 456 789
5	4	123 456 789	123 456 789	123 456 789	123 456 789	123 456 789	1	9
123 456 789	123 456 789	2	123 456 789	123 456 789	123 456 789	7	123 456 789	123 456 789
123 456 789	9	123 456 789	123 456 789	3	123 456 789	123 456 789	8	123 456 789
2	123 456 789	123 456 789	8	123 456 789	4	123 456 789	123 456 789	7
123 456 789	1	123 456 789	9	123 456 789	7	123 456 789	6	123 456 789

Solving the puzzle with CP means:

- reasoning with constraints in order to prune values in variable domains
- assigning variables in order to construct solutions

CP Success Stories: CP Has Landed on the Comet

On June 13th 2015, the robot-lab Philae woke up on the comet 67P/Churyumov-Gerasimenko to resume a series of experiments.



Some plans executed by Philae are modelled and solved using constraint programming technology. Several dedicated algorithms were developed by the Operations Research & Constraints group of the LAAS-CNRS lab.

CP Success Stories: Pythagorean Triples Problem

Problem: Can the set $\mathbb{N} = \{1, 2, \ldots, \}$ of natural numbers be divided into two parts, such that no part contains a triple (a, b, c) with $a^2 + b^2 = c^2$?

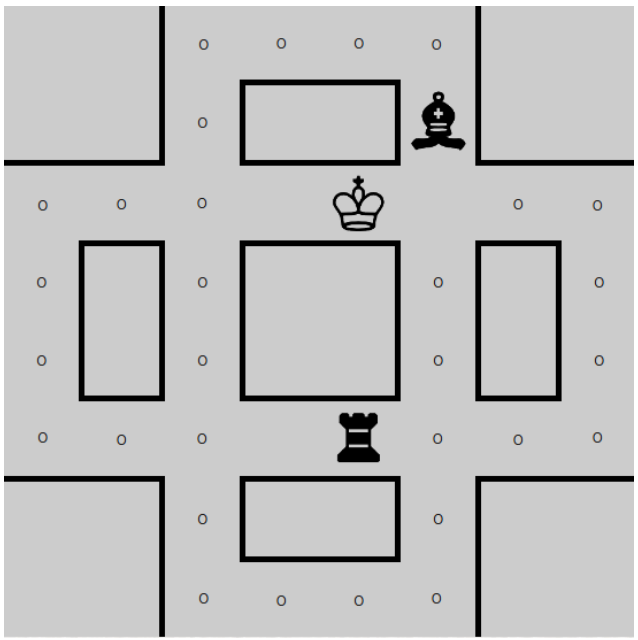
A prize for the solution was offered by Ronald Graham over two decades ago.

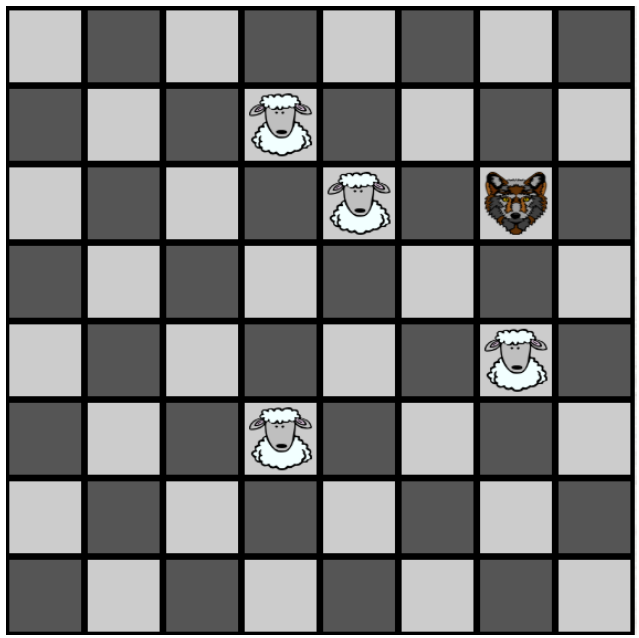
This problem has been solved by a hybrid SAT approach, employing both look-ahead and CDCL solvers.

This is the largest computed-aided Mathematics proof (200TB) ever presented in the literature.

CP Success Stories: General Game Playing (GGP)

General game playing (GGP) is the design of artificial intelligence programs to be able to play more than one game successfully.





CP Success Stories: General Game Playing (GGP)

Woodstock, a constraint-based general game player developped at CRIL, is based on two key ideas:

- constraint-based search propagation (the MAC component), and
- Bandit-based stochastic sampling (the UCB component).

WoodStock won the 2016 International General Game Playing Competition (IGGPC'16)

CP Success Stories: Music in CP

Constraint programming techniques applied to contemporary music composition

Two illustrative applications (with the courtesy of Charlotte Truchet, LINA).

1. A problem on harmony (Georges Bloch). Given a fixed chord, we have the following constraints:
   - minimize EstradaDistance(chord[i], FixedChord)
   - minimize GeorgesDistance(VF(chord[i]), VF(chord[i+1]))
2. Asynchronous rhythms (Mauro Lanza). Each pattern is played repetitively on one voice.
   - Variables: rhythmical patterns of fixed durations $d_1, \ldots, d_n$.
   - Constraints: no simultaneous onsets between two voices, for a given duration D.

Play

Classical CP Applications: Flow Shop Scheduling

Given n machines and m jobs, knowing that:

- each job requires exactly n operations to be executed
- the ith operation of a job must be executed on the ith machine
- no machine can perform more than one operation simultaneously
- for each operation of each job, execution time is specified

Find the best schedule, i.e. the one with the shortest possible total job execution makespan.

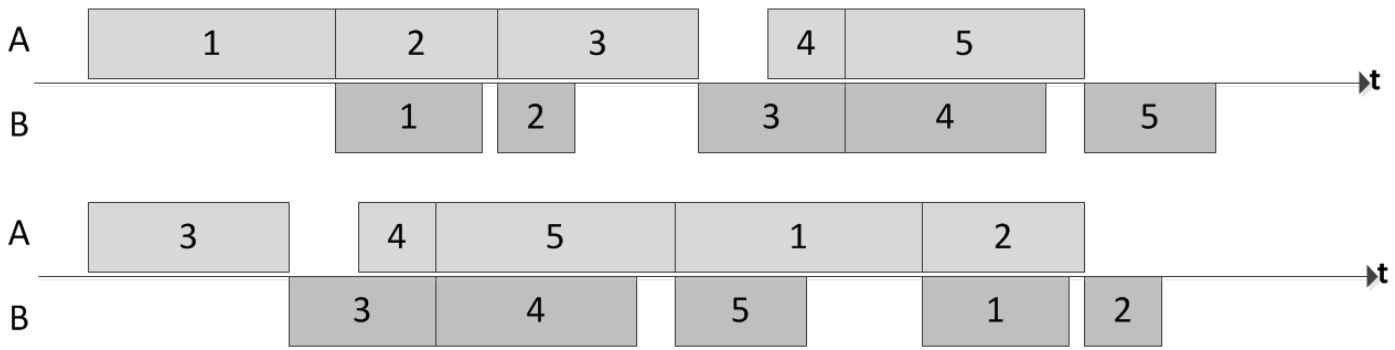


Figure: Example of (no-wait) flow-shop scheduling with 5 jobs on 2 machines A and B. A comparison of total makespan is given for two different job sequences.

Classical CP Applications: Car Sequencing

Given an assembly line for producing cars with various options, knowing that:

- different stations install different options
- stations can handle at most a certain percentage of the passing cars
- cars requiring a certain option must not be bunched together

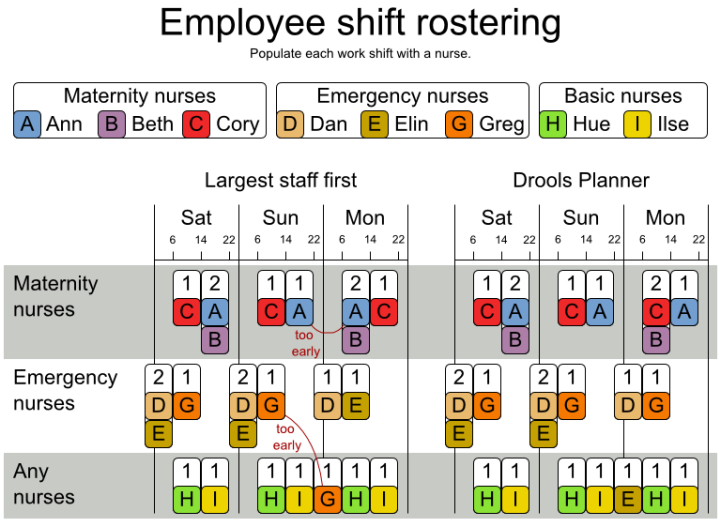
Find a production order respecting the capacity constraints of the stations.

Classical CP Applications: Nurse Rostering

Given hospital shifts to be taken by nurses, knowing that:

- nurses have various qualifications
- the hospital must respect some working/resting patterns
- each hospital service has some needs

Find an optimal assignment of nurses to shifts.

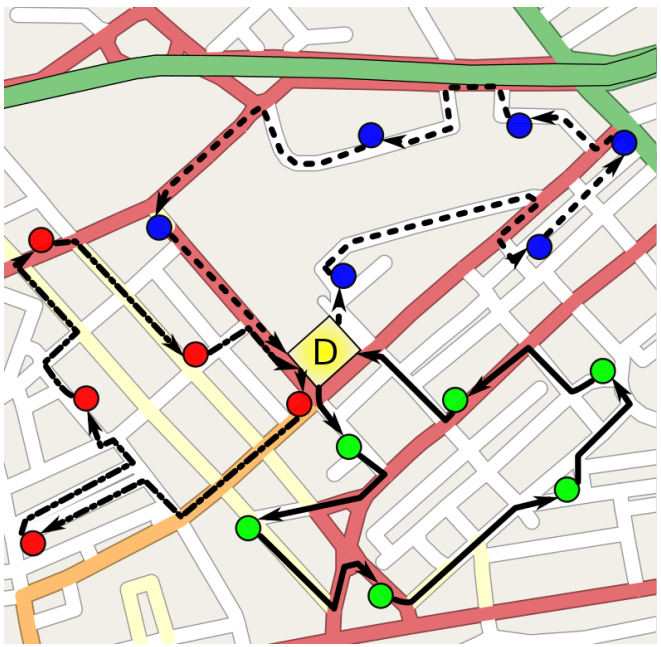


Classical CP Applications: Vehicle Routing

Given orders made by some costumers, knowing that:

- a fleet of vehicles is available
- a depot stores products

Find the best route of each delivery vehicle.



Classical CP Applications: Container Ship Loading

Given containers to be loaded on some ships, knowing that:

- some cranes are available
- there are specific requests on containers to be loaded

Find the fastest loading schedule

No CP used here!

Classical CP Applications

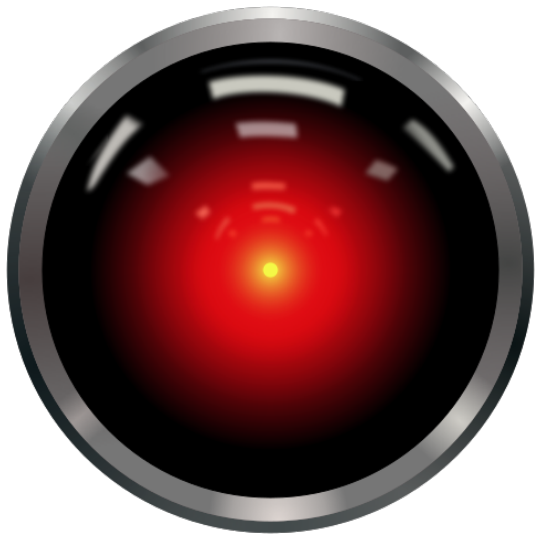
Many other application domains:

- Configuration
- Bio-informatics
- Planning
- Airport Scheduling
- Model Checking
- Data Mining
- ...

Which Roots?

CP has roots in:

- Artificial Intelligence (AI)
- Operations Research (OR)



Which Target?

Problems subject to constraints and/or objective functions.

Actually, optimization admits many frameworks in the litterature:

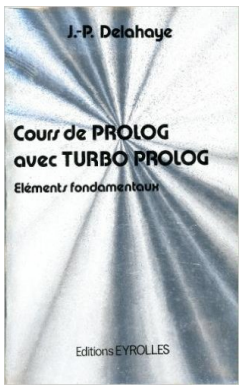
- Mathematical Programming
  - Linear Programming
  - Integer Linear Programming
  - Nonlinear programming
  - ...
- Meta-heuristics
  - Trajectory-based
  - Population-based
- **Constraint Programming**

Which Motto?

"Art lives from constraints and dies from freedom"

Which History?

- Started as Constraint Logic Programming (1987)
  - Introduction of constraints in logic programming
  - First implementation in Prolog III, CLP, CHIP



Then moved into its own paradigm (Constraint Programming)

- Creation of languages/libraries
- Creation of autonomous solvers
- Separation from the logic world

Constraint Programming

Constraint Programming (CP) is a general framework proposing simple, general and efficient algorithmic solutions to problems under constraints.

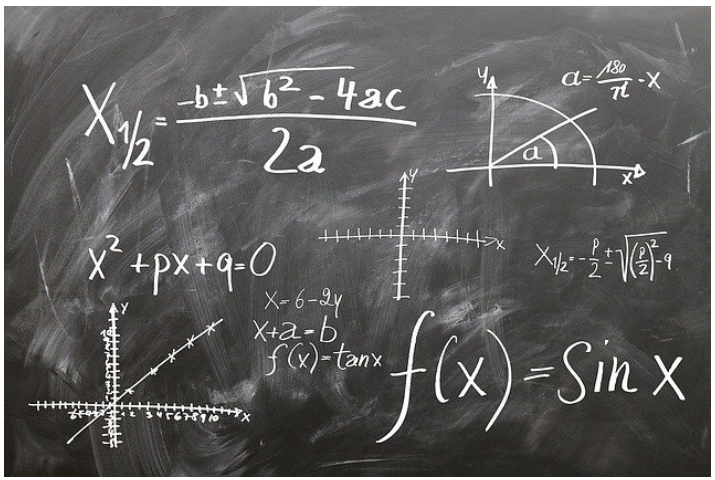
CP is attractive because there is a clear separation between:

- on the one hand, its formalism that makes easy the representation of many problems
- on the other hand, a large pool of algorithms and heuristics to find solutions

Modeling

There are then two main stages with CP:

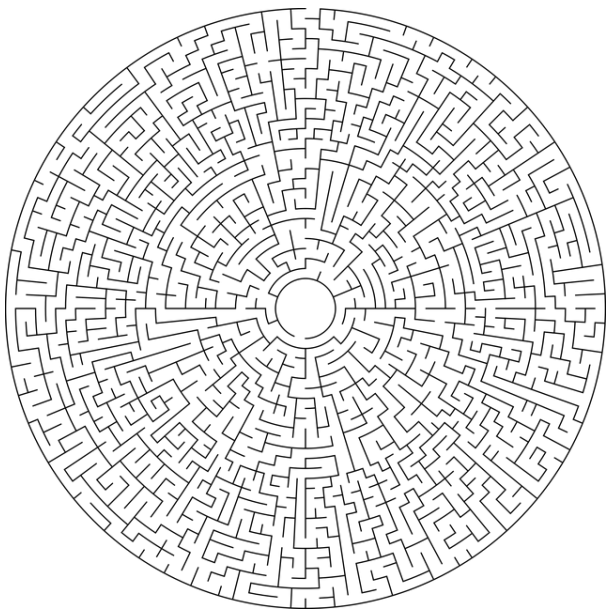
1. In a first stage, called modeling, the problem must be formally represented by means of variables, constraints, and possibly objective functions.



Ideally, this stage is purely declarative, but in practice a limited process of programming may be required (e.g., with a logic or object language).

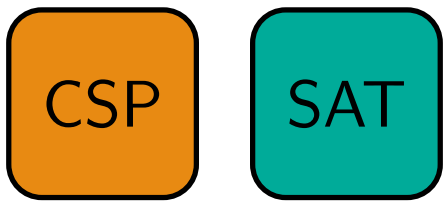
Solving

2 In a second stage, called solving, the problem modelled by the user must be tackled by a software tool in order to automatically obtain one solution, all solutions, an optimal solution, ...



Constraint Satisfaction

The **Constraint Satisfaction Problem** (CSP) resides at the core of constraint programming. An *instance* of this problem is represented by a **constraint network** (CN).

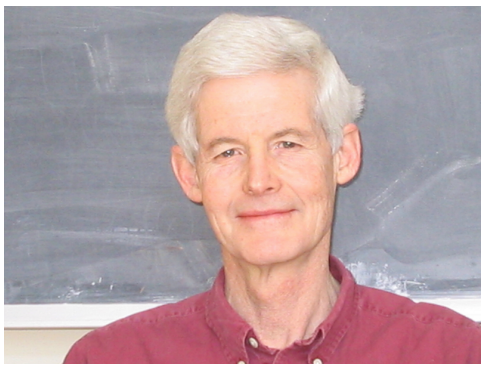


Note that SAT is closely related to CSP:

- variables are Boolean
- constraints are clauses (disjunctions of variables and their negations)

Untractability

No polynomial algorithm is known for both decision problems CSP and SAT.



Stephen Cook showed that SAT is a NP-complete problem

Remark.

In practice, people try to find efficient algorithms for a large range of problems by exploiting their structures. This is a motivating challenge.

Bibliography

- Foundations of constraint satisfaction (Tsang, 1993)
- Principles of Constraint Programming (Apt, 2003)
- Constraint Processing (Dechter, 2003)
- Constraint-based local search (van Hentenryck & Michel, 2005)
- Handbook of Constraint Programming (Rossi *et al.*, 2006)
- Constraint Propagation (Bessiere, 2006)
- Constraint Networks (Lecoutre, 2009)
- Global constraints: a survey (Régin, 2011)
- Integrated Methods for Optimization (Hooker, 2012)

Variable

Definition

A **variable** (with name) x is an unknown entity that must be given a value from a set called the (current) domain of x and denoted by $dom(x)$.

Remark.

The domain of a variable x usually evolves over time but it is always included in a set called initial domain of x and denoted by $dom^0(x)$.

Example.

The variables x and y have current domains:

$$dom(x) = \{2, 3, 4\} \text{ and } dom(y) = \{1, 3, 4\}.$$

Their initial domains are:

$$dom^0(x) = dom^0(y) = \{1, 2, 3, 4, 5\}.$$

Cartesian Product

Definition (Tuple)

A tuple τ is a sequence of values, between parentheses and with comma used as a separator. A tuple containing r values is called a r-tuple. The *ith* value of a r-tuple, with $1 \leq i \leq r$, is denoted by $\tau[i]$.

Definition

Let $S_1, S_2, \ldots, S_r$ be a sequence of r sets. The Cartesian product

$$\Pi_{i=1}^{i=r} S_i = S_1 \times S_2 \times \cdots \times S_r$$

is the set

$$\{(a_1, a_2, \ldots, a_r) \mid a_1 \in S_1, a_2 \in S_2, \ldots, a_r \in S_r\}.$$

Each element of $\Pi_{i=1}^{i=r} S_i$ corresponds to a r-tuple.

Example

Example.

Let x, y and z be three variables such that $dom(x) = dom(y) = \{a, b\}$ and $dom(z) = \{a, c\}$. Then, we have:

$$dom(x) \times dom(y) \times dom(z) = \left\{ \begin{array}{c} (a, a, a), \\ (a, a, c), \\ (a, b, a), \\ (a, b, c), \\ (b, a, a), \\ (b, a, c), \\ (b, b, a), \\ (b, b, c) \end{array} \right\}.$$

Relation

A relation is simply a subset of a Cartesian product.

Definition (Relation)

A relation R defined on a sequence of r sets $S_1, S_2, \ldots, S_r$ is a subset of the Cartesian product $\Pi_{i=1}^{i=r} S_i$, i.e., we have: $R \subseteq \Pi_{i=1}^{i=r} S_i$.

Example.

A relation R defined on $dom(x)$, $dom(y)$, $dom(z)$ is for example:

$$R = \left\{ \begin{array}{c} (a,a,c), \\ (b,a,a), \\ (b,a,c), \\ (b,b,c) \end{array} \right\} \subset \left\{ \begin{array}{c} (a,a,a), \\ (a,a,c), \\ (a,b,a), \\ (a,b,c), \\ (b,a,a), \\ (b,a,c), \\ (b,b,a), \\ (b,b,c) \end{array} \right\}.$$

Constraint

Definition

A **constraint** (with name) c is defined over a (totally ordered) set of variables, called **scope** of c and denoted by $scp(c)$, by a relation $rel(c)$ that describes the set of tuples allowed by c for the variables of its scope.

The **arity** of a constraint c is the number of variables involved in c, i.e, $|scp(c)|$. We say that c is:

- unary iff its arity is 1,
- binary iff its arity is 2,
- ternary iff its arity is 3,
- ...
- non-binary iff its arity is strictly greater than 2.

Representing Constraints

Formally, a constraint is defined a mathematical relation. In practice there are three different ways of representing a constraint:

- in intension, by using a Boolean formula (predicate),
- in extension, by listing tuples,
- implicitly by referring to a so-called global constraint.

Intensio(nal) Constraints

Definition (Intension Constraint)

A constraint c is intensional, or defined in intension, iff it is described by a Boolean formula (predicate) that represents a function that is defined from $\Pi_{x \in scp(c)} dom^0(x)$ to $\mathbb{B}$.

Example.

A binary intensional constraint:

$$v \leq w + 2$$

A ternary intensional constraint:

$$x \neq y \wedge x \neq z \wedge y \neq z$$

Remark.

Intension constraints are typically used for representing arithmetic constraints.

Extension(al) Constraints

Definition (Extension Constraint)

A constraint c is extensional (or defined in extension) iff it is explicitly described,

- either positively by listing the tuples allowed by c,
- or negatively by listing the tuples disallowed by c.

Example.

If $dom(x) \times dom(y) \times dom(z) = \{0, 1, 2\}^3$, then the ternary constraint from the previous slide can be defined positively as:

$$\langle x, y, z \rangle \in \{(0, 1, 2), (0, 2, 1), (1, 0, 2), (1, 2, 0), (2, 1, 0), (2, 0, 1)\}$$

Remark.

Extension Constraints are also called Table Constraints.

Global Constraints

Definition (Global Constraint)

A global constraint is a constraint pattern that captures a precise relational semantics and that can be applied over an arbitrary number of variables.

For example, the semantics of `allDifferent` is that all variables must take different values.

Example.

Our previous ternary constraint can be defined by:

$$\texttt{allDifferent}(x, y, z)$$

Remark.

Many global constraints will be introduced later in this course.

CSP Instance

Definition

An instance P of the Constraint Satisfaction Problem (CSP), also called Constraint Network (CN), is composed of:

- a finite set of variables, denoted by $vars(P)$,
- a finite set of constraints, denoted by $ctrs(P)$, such that $\forall c \in ctrs(P), scp(c) \subseteq vars(P)$.

Remark.

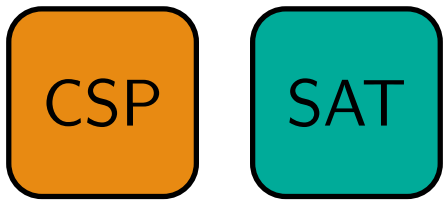
A pair (x, a) with $x \in vars(P)$ and $a \in dom(x)$ is sometimes called a **literal of** P (or even, by language abuse, a value of P).

Exercice.

1. Propose a definition for the concept of solution (of a specified CSP instance).
2. Find the smallest and the greatest solution (according to the lexicographic order) of the following CSP instance P:
   - $vars(P) = \{w, x, y, z\}$ with
     - $dom(w) = \{1, 2, 3\}$
     - $dom(x) = \{1, 2, 3\}$
     - $dom(y) = \{1, 2, 3, 4\}$
     - $dom(z) = \{1, 2, 3, 4\}$
   - $ctrs(P) = \{$
     $w = x,$
     $x \leq y + 1,$
     $y > z,$
     $\langle x, z \rangle \in \{(1, 2), (2, 1), (2, 4), (3, 3)\}$
     $\}$
3. Draw :
   - the constraint graph of P

CSP versus SAT

The problem CSP involves determining whether a given constraint network has a solution or not. CSP "generalizes" SAT.



Example.

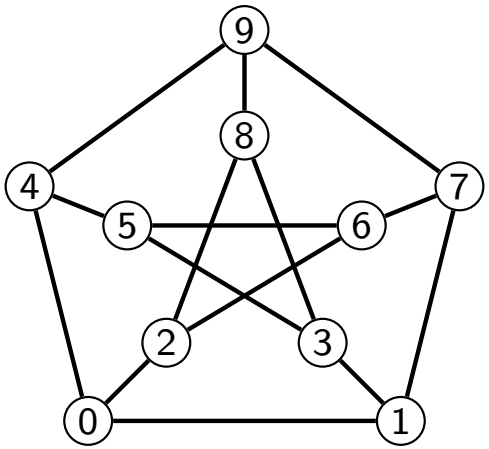
For the following CNF formula:

$$(x \vee y \vee \neg z) \wedge (\neg w \vee \neg x) \wedge (w \vee \neg y \vee z)$$

we obtain a constraint network P such that:

- $vars(P) = \{w, x, y, z\}$ s.t. $\forall v \in vars(P), dom(v) = \mathbb{B}$
- $ctrs(P) = \{x \vee y \vee \neg z, \neg w \vee \neg x, w \vee \neg y \vee z\}$

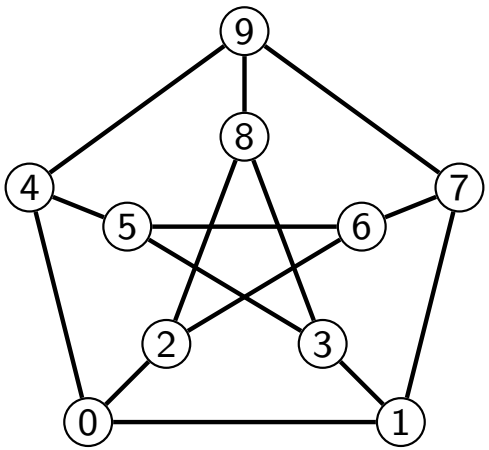
Another Illustration: Graph Coloring Problem



We have to assign:

- at each node of a graph Variables
- a color taken in $\{R, G, B\}$ Domains
- such that no adjacent nodes have the same color Constraints

Graph Coloring Problem



Data (input)

- a graph $G = (V, E)$
- a set of colors $\{R, G, B\}$

The **CSP instance** P:

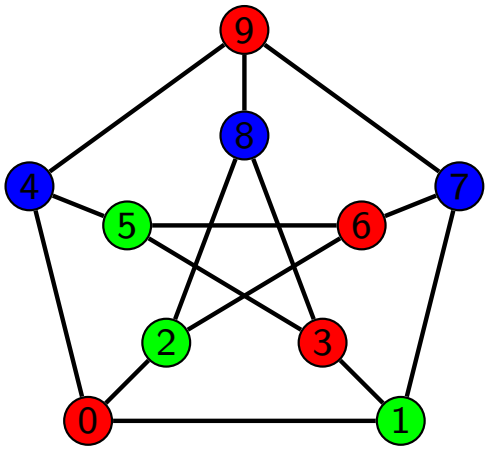
- $vars(P) = \{x_0, x_1, \ldots, x_9\}$ with $\forall i \in 0..9, dom(x_i) = \{R, G, B\}$
- $ctrs(P) = \{x_i \neq x_j : 0 \leq i < j \leq 9 \wedge (i, j) \in E\}$

Remark.

Each variable represents a node, and each constraint is binary.

Graph Coloring Problem

A solution



Remark.

Variables have symbolic domains here. However, most of the solvers handle integer variables instead.

Exercice: write the PyCSP^3 model.

What about Optimization?

Definition

An objective function o for a CN P associates a numerical value, called objective value, with each complete instantiation of the variables of P.
We have o:

$$\Pi_{x \in vars(P)} dom(x) \rightarrow \mathbb{R}$$

An objective function allows us:

- to distinguish between solutions,
- to identify the best ones, under the spectrum of either minimization or maximization.

Remark.

In this course, we shall mainly focus on mono-objective optimization.

COP Instance

Definition

An instance P of the Constraint Optimisation Problem (COP) is composed of:

- a finite set of variables, denoted by $vars(P)$,
- a finite set of constraints, denoted by $ctrs(P)$, such that $\forall c \in ctrs(P), scp(c) \subseteq vars(P)$,
- an objective function denoted by $obj(P)$, to be minimized or maximized.

Remark.

A COP instance can be seen as an underlying CSP instance together with an objective function.

Solution to a CSP/COP Instance

Definition

A solution of a CSP instance P is the assignment of a value to each variable of P such that all constraints of P are satisfied. The set of solutions of P is denoted by $sols(P)$.

Definition

- A solution of a COP instance is a solution of the underlying CSP.
- For minimization (resp., maximization), an optimal solution is a solution whose objective value is less than or equal to (resp., greater than or equal) the objective value of any other solution.

Finding an optimal solution may be very "complicated"

Modeling with (Global) Constraints

We illustrate the very popular constraints:

- `allDifferent`
- `sum`
- `extension` (or `table`)

Constraint `allDifferent`

Syntax.

`allDifferent`(X) with X the scope of the constraint.

Semantics.

An instantiation of X satisfies the constraint iff all assigned values are different.

Important.

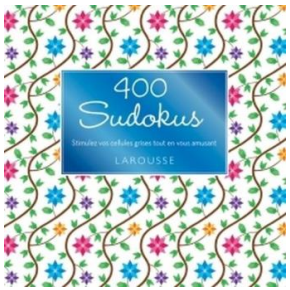
This constraint is useful for many problems.

Example.

Let the constraint `allDifferent`(w, x, y, z):

- $\{w = 1, x = 2, y = 2, z = 3\}$ does not satisfy the constraint
- $\{w = 1, x = 2, y = 4, z = 3\}$ satisfies the constraint

A Sudoku Puzzle



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

Modeling Sudoku

Example.

We can define P such that:

- $vars(P) = \{$
  $x_{11}, x_{12}, \ldots, x_{19},$
  $x_{21}, x_{22}, \ldots, x_{29},$
  $\ldots$
  $x_{91}, x_{92}, \ldots, x_{99}$
  $\}$
  $with\ dom(x_{ij}) = \{1, \ldots, 9\}, \forall i, j \in 1..9$

- $ctrs(P) = \{$
  $\texttt{allDifferent}(x_{11}, x_{12}, \ldots, x_{19}),$
  $\texttt{allDifferent}(x_{21}, x_{22}, \ldots, x_{29}),$
  $\ldots$
  $\}$

Remark.

For each clue, add a unary constraint

A Solution to the Sudoku Puzzle

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

Puzzle

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

Solution

Constraint sum

Syntax.

A constraint sum is written as:

$$\sum_{i=1}^{r} c_i x_i \text{ <op> } L$$

where:

- $c_i \in \mathbb{Z}, \forall i \in 1..r$
- <op> $\in \{<, \leq, \geq, >, =, \neq, \in, \notin\}$
- L an integer ($\in \mathbb{Z}$), an interval or a variable

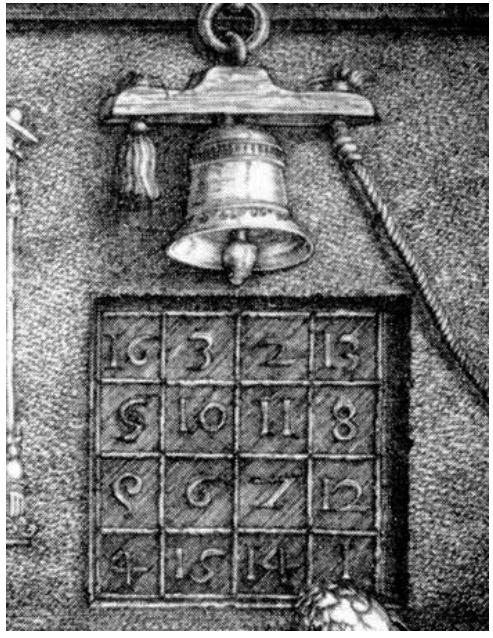
Semantics.

Immediate

Example.

Constraint $c_{wxyz} : w + 2x + 4y + 5z \geq 42$

Magic Squares



Melencolia (1514) from German Painter Albrecht Dürer

2	9	4
7	5	3
6	1	8

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

2	15	22	3	23
16	13	20	12	4
19	17	1	21	7
10	11	14	24	6
18	9	8	5	25

Modeling Magic Squares

Example.

For a magic square of order 3, we can define P such that:

- $vars(P) = \{$
  $x_{11}, x_{12}, x_{13},$
  $x_{21}, x_{22}, x_{23},$
  x_{31}, x_{32}, x_{33}
  $\}$
  $with\ dom(x_{ij}) = \{1, \ldots, 9\}, \forall i, j \in 1..9$

- $ctrs(P) = \{$
  $\sum_{j=1}^{3} x_{1j} = 15$
  $\sum_{j=1}^{3} x_{2j} = 15$
  $\ldots$
  $\}$

Remark.

There is a constraint `sum` for each main diagonal.

Constraint `extension` (or `table`)

Syntax.

This constraint can be written $X \in T$ with :

- $X = \langle x_1, x_2, \ldots, x_r \rangle$, the scope of the constraint
- T, a set of r-tuples

For a negative table constraint, replace $\in$ by $\notin$.

Remark.

Useful/mandatory constraint when no semantics is known.

Generating Crossword Grids

MOTS CROISÉS PAR JEAN-FRANÇOIS DEMAY

GRILLE N° 48

I II III IV V VI VII VIII IX X

1
2
3
4
5
6
7
8
9
10

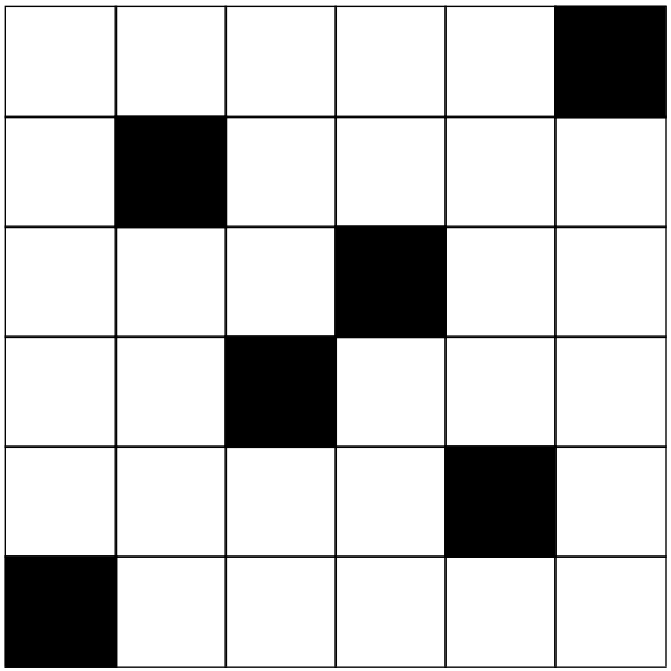
HORIZONTAL:
1. Dans quelques semaines, ce sera une bonne occasion de se rendre dans une boîte interdite aux mineurs. **2.** Boursouflé chez les grosses têtes. Phosphore. **3.** Défunts depuis pou. Petite montagne. **4.** Devient huile quand elle est grosse. Méfiance quand elle dort ! **5.** Espion burlesque. Diminutif de lady. **6.** Paresseux. Un jeune qui porte des sabots. **7.** Grabuge. Safari pour John Wayne. **8.** Conçais des chanières. **9.** Roulée. Alcoolisant le rouge. **10.** Comme une situation à France Télécom.

VERTICAL:
I. Ainsi sont les tombes à la Toussaint. **II.** Suicidaire grec. Conjonction. Vieille note. **III.** Paltoquet. **IV** Connu. **V.** Vieille langue. Coopératives agricoles en Israël. **VI.** Écrits sur un cours. Animés. **VII.** Homme de cour chez les Ottomans. Rendit tout doux. **VIII.** Prénom révolutionnaire. Sucrerie populaire. **IX.** Coordonne. Vouent un culte. **X.** Un politicien qui s'habille en noir.

Solution de la grille n° 47 :

1. Bling-bling
2. Oulan-Bator
3. Aïeul. Rêva
4. Trot. AS. ET
5. Pénibles
6. En. Encre
7. Ôrâmes. HEC
8. LED. Kwaï
9. Lee. Entôle
10. Exxon Mobil

I. Boat-people
II. Luirent. Ex
III. Iléon. Alex
IV. Nautisme
V. GNL. Éden
VI. BB. Alès. Nm
VII. Larsen. KTO
VIII. Ite. Schwob
IX. Nove. Reali
X. Gratte-ciel



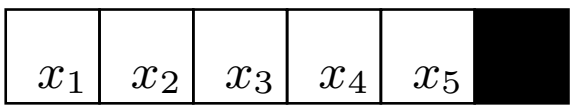
Modeling Crosswords

For a specified grid, we can define P such that:

- $vars(P)$ contains a variable x with $dom(x) = \{a, \ldots, z\}$ for each white cell,
- $ctrs(P)$ contains a constraint `extension` for each (maximal) sequence of white cells.

Example.

For the row:



we can generate the constraint:

$$\langle x_1, x_2, x_3, x_4, x_5 \rangle \in \{abbes, abdos, \ldots, zozos\}$$

Remark.

A word such as "abbes" is a shortcut for the tuple (a, b, b, e, s).

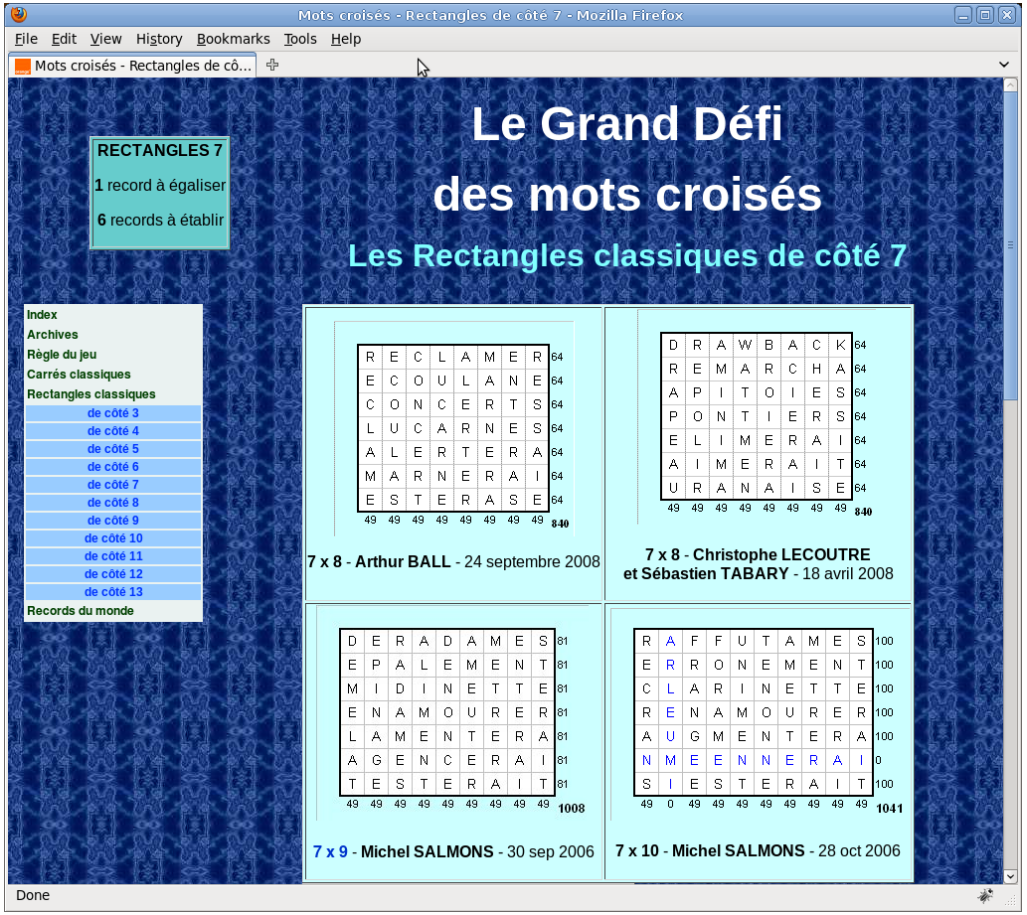
A solution to the Crossword Instance

A	L	O	H	A	■
X	■	R	I	C	H
I	C	E	■	H	A
O	R	■	W	E	I
M	A	M	A	■	F
■	G	E	N	O	A

Good results obtained with the solver AbsCon (Ace):

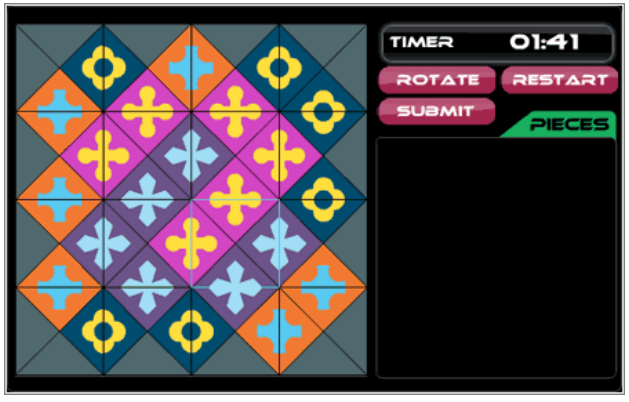
- generic heuristics
- generic filtering (STR)
- restarts

With Sébastien



A Last Problem before Leaving: Eternity II

Toy Puzzle: 16 pieces



Commercial Version: 256 pieces – 2 millions dollars to win



Eternity II

With Christian

Attempt with the solver AbsCon (Ace):

- two different models
- generic heuristic
- restarts

Best result: -29 (480-29 = 451 correct links) obtained in a few minutes (but after 2/3 weeks for reflection/modeling/development)

Epilog to Eternity II

Apt, K.R. 2003.
Principles of Constraint Programming.
Cambridge University Press.

Bessiere, C. 2006.
Constraint Propagation.
Chap. 3, pages 29–83 of: Handbook of Constraint Programming.
Elsevier.

Dechter, R. 2003.
Constraint processing.
Morgan Kaufmann.

Hooker, J.N. 2012.
Integrated Methods for Optimization.
Springer.

Lecoutre, C. 2009.
Constraint networks: techniques and algorithms.
ISTE/Wiley.

Régin, J.-C. 2011.
Global Constraints: a survey.
Chap. 2, pages 63–134 of: Hybrid Optimization.
Springer.

Rossi, F., van Beek, P., & Walsh, T. (eds). 2006.
Handbook of Constraint Programming.
Elsevier.

Tsang, E. 1993.
Foundations of constraint satisfaction.
Academic Press.

van Hentenryck, P., & Michel, L. 2005.
Constraint-based local search.
MIT Press.