Notes on Constraint Programming

Hadrien Cambazard

16/10/2016

1 Introductive Example

A toy problem of timetabling :
— Four meetings A, B, C, D of one hour each have to take place in the same room between 8h and 13h
— A and C must end at 10h and 11h respectively at the latest
— C must take place before B
— B must take place before D with exactly one or two hours of break inbetween

A problem is modelled by **variables**, **domains** and **constraints** :

$$\begin{array}{ll} (1.1) & \textsc{AllDifferent}(x_A, x_B, x_C, x_D) \\ (1.2) & x_B + d + 1 = x_D, \\ (1.3) & x_C < x_B, \\ (1.4) & x_A \in \{1,2\}, x_C \in \{1,2,3\}, x_B, x_D \in \{1,2,3,4,5\}, d \in \{1,2\} \end{array} \tag{1}$$

2 Definitions and fundamentals

2.1 Constraint network, Solution

Definition n°1 - Constraint Network
A *Constraint Network* $\mathcal{P}$ is a triplet $(\mathcal{X}, \mathcal{D}, \mathcal{C})$, where :
— $\mathcal{X}$ is a set of *variables* $\{x_1, \ldots, x_n\}$
— $\mathcal{D}$ is a *domain* on $\mathcal{X}$, that is, a set $\{\mathcal{D}(x_1), \ldots, \mathcal{D}(x_n)\}$
  — where $\mathcal{D}(x_i) \subset \mathbb{Z}$ is the *finite* set of values that x_i can take
— $\mathcal{C}$ is a set of constraints $\{c_1, \ldots, c_m\}$ defining possible relations between variables

Definition n°2 - Constraint
A *Constraint* c is a pair $(\mathcal{X}(c), \mathcal{R}(c))$ where :
— $\mathcal{X}(c)$ is a sequence of variables. The length of $\mathcal{X}(c) = (x_{i_1}, \ldots, x_{i_k})$ is called the *arity* of c
— $\mathcal{R}(c)$ is a relation of arity k over $\mathbb{Z}$, that is, a subset of $\mathbb{Z}^k$ (a list of feasible **tuples**)

EXERCISE n°1: A constraint network for magic square

A magic square of order n is an arrangement of the integers 1 to n^2 in a square, such that the rows, columns, and diagonals all sum to the same value. A square remains "essentially similar" if it is rotated or transposed, or flipped so that the order of rows is reversed. Thus there exists 8 different magic squares sharing one **standard** form.

A square is in standard form if the following two conditions apply :
— the element at position [1,1] (top left corner) is the smallest of the four corner elements ; and
— the element at position [1,2] (top and second from left cell) is smaller than the element in [2,1].
Give a constraint network to model a magic square of order n.

Definition n°3 - Solution
Given a constraint network $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$. An *instantiation* σ on a set $\mathcal{Y} = \{x_1, \ldots, x_k\}$ of variables is a mapping from variables to values :
— σ is said *valid* iff $\forall x_i \in \mathcal{Y},\ \sigma(x_i) \in \mathcal{D}(x_i)$
— σ *violates* a constraint c iff $\mathcal{X}(c) \subseteq \mathcal{Y}$ and $\sigma(\mathcal{X}(c)) \notin \mathcal{R}(c)$
— σ is said *consistent* iff it is valid and it does not violate any constraint in $\mathcal{C}$
— A *solution* to $\mathcal{P}$ is a consistent instantiation of $\mathcal{X}$

EXERCISE n°2: Solutions to constraint networks - Micro-structure versus constraint graph

— Give a solution to the following constraint network (figure 1) :

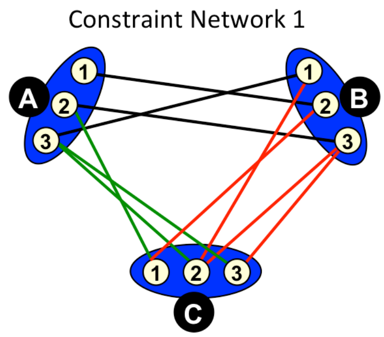

FIGURE 1 – A constraint network represented by its micro-structure.

— Given a binary constraint network and its micro-structure, what is a solution from a graph point of view ?

EXERCISE n°3: Binary and N-ary networks

— Define a **binary extensional** network equivalent to model (1) of the introduction.
— Define the **n-ary extensional constraint** associated to constraint (1.2) of model (1).

The Constraint Satisfaction Problem (CSP) is to find a solution to a given constraint network.

2.2 Local consistencies as properties

Definition n°4 - Arc Consistency

Let $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a constraint network,
— A *valid* tuple σ of the constraint c ($\sigma \in \mathcal{R}(c)$) is called a *support* of c
  — i.e., A solution σ of the constraint network $(\mathcal{X}(c), \mathcal{D}, \{c\})$
— A value $v \in \mathcal{D}(x)$ is *consistent* with c iff it belongs to a support of c
— A domain $\mathcal{D}$ is *Arc Consistent* iff $\forall c \in \mathcal{C},\ \forall x \in \mathcal{X}(c),\ \forall v \in \mathcal{D}(x),\ v$ is consistent with c

Iteratively removing non-consistent values of the constraints converges toward a unique fix-point : **The largest Arc Consistent subdomain** of $\mathcal{P}$ (*AC* closure of $\mathcal{P}$).

EXERCISE n°4: AC closure

What is the AC closure of the following constraint network :
$\mathcal{X} = \{x, y, z\}$, $\mathcal{D} = \{\mathcal{D}(x) = \{1, 2, 3, 4\}, \mathcal{D}(y) = \{2, 3, 4\}, \mathcal{D}(z) = \{2, 3\}\}$,
$\mathcal{C} = \{c_1 : \text{ALLDIFFERENT}(x, y, z), c_2 : x + 2y - z \leq 4\}$,
Give a support for $(x, 1)$ in c_1. Give a support of $(x, 1)$ in c_2 that is not consistent with c_1.

EXERCISE n°5: AC closure

Figure 2 shows two contraint networks. Typically network 1 correspond to :
— $\mathcal{X} = \{x_A, x_B, x_C\}$, $\mathcal{C} = \{c_1, c_2, c_3\}$
— $\mathcal{D} = \{\mathcal{D}(x_A) = \mathcal{D}(x_B) = \mathcal{D}(x_C) = \{1, 2, 3\}\}$
— $\mathcal{X}(c_1) = \{x_A, x_B\}, \mathcal{R}(c_1) = \{(1, 2), (2, 3), (3, 1)\}$
— $\mathcal{X}(c_2) = \{x_A, x_C\}, \mathcal{R}(c_2) = \{(2, 1), (3, 2), (3, 3)\}$

— $\mathcal{X}(c_3) = \{x_B, x_C\}, \mathcal{R}(c_3) = \{(1,2),(2,1),(3,2),(3,3)\}$

Questions :

— Give a solution to each constraint network of figure 2.

— What is the AC closure of the two constraint networks ?

— What values are **globally** inconsistent ?

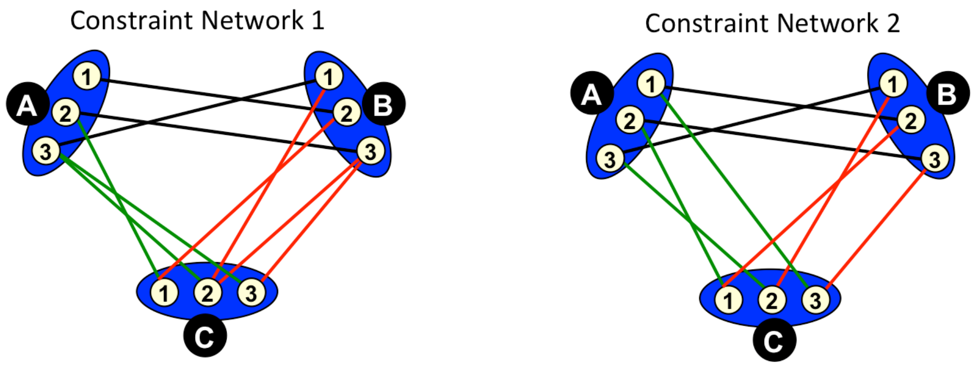

FIGURE 2 – Two constraint networks (courtesy to Romuald Debruyne).

Definition n°5 - Bound Consistency and Range Consistency

Let $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ be constraint network

— A tuple σ (on Y) is *bounds valid* iff $\forall x_i \in \mathcal{Y},\ \underline{x_i} \leq \sigma(x_i) \leq \overline{x_i}$

— A *bounds valid* tuple σ of the constraint c (i.e $\in \mathcal{R}(c)$) is a *bounds support* of c

— i.e., A solution σ of the constraint network $(\mathcal{X}(c), \mathcal{B}, \{c\})$ where $\forall x,\ \mathcal{B}(x) = [\underline{x}, \ldots, \overline{x}]$

— A value $v \in \mathcal{D}(x)$ is *bounds consistent* with c iff it belongs to a bounds support of c

— A domain $\mathcal{D}$ is *Bounds Consistent* iff $\forall c \in \mathcal{C},\ \forall x \in \mathcal{X}(c),\ \underline{x}$ and $\overline{x}$ are bounds consistent with c

— A domain $\mathcal{D}$ is *Range Consistent* iff $\forall c \in \mathcal{C},\ \forall x \in \mathcal{X}(c),\ \forall v \in \mathcal{D}(x),\ v$ is bounds consistent with c

---

EXERCISE n°6: decomposition in differences, BC and GAC (from [12])

Consider the network with variables $x_1, \ldots, x_6$, domains $D(x_1) = D(x_2) = \{1,2\}, D(x_3) = D(x_4) = \{2,3,5,6\}, D(x_5) = \{5\}, D(x_6) = [3, \ldots, 7]$ and a constraint ALLDIFFERENT$(x_1, \ldots, x_6)$. Give the domains of the variables after applying Bound-Consistency (BC) and Arc-Consistency (AC). Give also the domains after applying Arc-consistency on a constraint network where the ALLDIFFERENT$(x_1, \ldots, x_6)$ is replaced by a clique of differences $x_i \neq x_j,\ \forall i < j \leq 6$.

AC, BC, RC are properties of the domains.

Definition n°6 - Singleton Arc Consistency

A network $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ is Singleton Arc Consistent (SAC) if and only if for all $x_i \in \mathcal{X}$, for all $v_i \in D(x_i)$, the subproblem $P|_{x_i = v_i}$ is not arc inconsistent.

2.3 Local consistencies as algorithms

3 Intensional and Global constraints

3.1 Some common constraints : linear, element, channeling

3.1.1 Linear inequation

For sake of simplicity we restrict the constraint to all a_i and b_i in $\mathbb{N}^*$, all $\underline{x_i} \geq 0$ and $c \in \mathbb{N}$.

$$\sum_{i=1}^{n_1-1} a_i x_i - \sum_{i=n_1}^{n} b_i x_i \leq c \qquad | \qquad \sum_{i=1}^{n_1-1} a_i x_i - \sum_{i=n_1}^{n} b_i x_i \geq c \tag{2}$$

EXERCISE n°7:

- Give the AC closure for $D(x_1) = D(x_2) = \{0, 1, 2, 3, 4\}$, $D(x_3) = \{2, 3, 4\}$, $3x_1 - 2x_2 + 4x_3 \leq 7$.
- Give a GAC algorithm for the $\leq$ linear inequality (constraint (2)).

3.1.2 Linear equation

For sake of simplicity we restrict the constraint to all a_i in $\mathbb{N}^*$ and all $\underline{x_i} \geq 0$ and $c \in \mathbb{N}$..

$$\sum_{i=1}^{n} a_i x_i = c \tag{3}$$

EXERCISE n°8:

Can you give a polynomial GAC algorithm for the linear equality constraint (3) ? What of BC ? What filtering algorithm do you suggest ?

EXERCISE n°9:

Give the AC closure for $D(x_1) = \{0, 1, 2\}$, $D(x_2) = \{0, 1\}$, $D(x_3) = \{0, 1\}$, $2x_1 + 3x_2 + 4x_3 = 7$. What would be the result of your previous filtering algorithm (Exo (7)) on this example ?

3.1.3 Element

$$\textsc{Element}(y, t = [a_1, \ldots, a_n], x) \qquad | \qquad \textsc{Element}\mathrm{V}(y, t = [z_1, \ldots, z_n], x) \tag{4}$$

EXERCISE n°10:

Assuming that ELEMENT is enforcing AC, compare the two following CP models :

Model 1 : $D(x_1) = D(x_2) = D(x_3) = \{0, 1\}$ $D(y) = [0, 100]$, $\mathcal{C} = \{10x_1 + 3x_2 + 5x_3 = y, x_1 + x_2 + x_3 = 1\}$

Model 2 : $D(x) = \{0, 1, 2\}$, $D(y) = [0, 100]$, $\mathcal{C} = \{\textsc{Element}(y, [10, 3, 5], x)\}$

3.1.4 Counting occurrences

$$\textsc{AtLeast}(y, [x_1, \ldots, x_m], a) \quad | \quad \textsc{AtMost}(y, [x_1, \ldots, x_m], a) \quad | \quad \textsc{Count}(y, [x_1, \ldots, x_m], a) \quad (5)$$

EXERCISE n°11:

We must assign n clients to at most m depots that deliver goods to the clients. Each client must be served by one single depot. A transportation cost c_{ij} is paid if client i is delivered by depot j. A depot can serve at most w_j clients. The problem is to decide which depot serves each client to minimize the total transportation cost.

The constraint among counts the number of variables using values in a given set :

$$\textsc{Among}(y, [x_1, \ldots, x_m], [a_1, \ldots, a_n]) \quad (6)$$

3.1.5 Usefull constraints for redundant modelling

$$\textsc{BoolChanneling} : x_i = j \Leftrightarrow b_{ij} = 1 \qquad \textsc{Inverse} : x_i = j \Leftrightarrow y_j = i \quad (7)$$

3.2 Assignment and counting

3.2.1 Alldifferent (from [13, 8, 10])

$$\textsc{Alldifferent}(x_1, \ldots, x_n) \quad (8)$$

Hall's Marriage Theorem [6] : *If a group of men and women marry only if they have been introduced to each other previously, then a complete set of marriages is possible if and only if every subset of men has collectively been introduced to at least as many women, and vice versa.*

Bound Consistency

Definition n°7 - Hall interval

Let $x_1, x_2, \ldots, x_n$ be variables with respective finite domains $D(x_1), D(x_2), \ldots, D(x_n)$. Given an interval I of values, define $K_I = \{x_i | D(x_i) \subseteq I\}$. I is a Hall interval if $|I| = |K_I|$.

Theorem n°1

$\textsc{alldifferent}(x_1, \ldots, x_n)$ is BC if and only if $|D(x_i)| \geq 1$ $(i = 1, \ldots, n)$ and :

1. for each interval I : $|K_I| \leq |I|$,
2. for each **Hall interval** I : $\{\underline{x_i}, \overline{x_i}\} \cap I = \emptyset$ for all $x_i \notin K_I$.

EXERCISE n°12: Bound consistency and Hall Intervals

$x_1 \in [3, 6]$, $x_2 \in [3, 4]$, $x_3 \in [2, 5]$, $x_4 \in [2, 4]$, $x_5 \in [3, 4]$, $x_6 \in [1, 6]$, $\textsc{Alldifferent}(x_1, \ldots, x_6)$.

- Give all Hall intervals and the state of the domains after enforcing BC.
- What filtering would you get if you decompose $\textsc{Alldifferent}$ into a clique of binary constraints (each achieving arc consistency) $x_i \neq x_j$, $\forall (i, j) \in [1, 6] \times [1, 6], i \neq j$?

EXERCISE n°13: Golomb rulers

The problem is to place n marks on a ruler so that the distance between each pair of marks is different and the length of the ruler is minimized. The golomb ruler is said to be of order n. Give a CP model for that problem. (the smallest open ruler is $n = 28$)

Arc Consistency

Definition n°8 - Tight set
Let x_1, x_2, ..., x_n be variables with respective finite domains $D(x_1), D(x_2), \dots, D(x_n)$. $K \subseteq \{x_1, ..., x_n\}$ is a tight set if $|K| = |D_K|$ ($D_K = \cup_{x_i \in K} D(x_i)$).

Theorem n°2
$\textsc{alldifferent}(x_1, \dots, x_n)$ is GAC if and only if $|D(x_i)| \geq 1$ ($i = 1, \dots, n$) and $D(x_i) \cap D_K = \emptyset$ for each **Tight set** $K \subseteq \{x_1, \dots, x_n\}$ and each $x_i \notin K_I$.

Theorem n°3 (from [10])
Let G be **the value graph** of a sequence of variables $X = \{x_1, x_2, \dots, x_n\}$ with respective finite domains $D(x_1), D(x_2), \dots, D(x_n)$. The constraint $\textsc{alldifferent}(x_1, \dots, x_n)$ is GAC if and only if every edge in G belongs to a matching in G covering X.

Theorem n°4 (from [2, 10])
Let G be a graph and M a maximum-size matching in G. An edge belongs to a maximum-size matching in G if and only if it either belongs to M, or to an even M-alternating elementary chain starting at an M-free vertex, or to an even M-alternating elementary cycle.

Note : an elementary chain is referred to as a path by some authors and an elementary cycle as a circuit.

Algorithm 1 GAC algorithm for $\textsc{Alldifferent}(X = \{x_1, \dots, x_n\})$

build the value graph G = (X, D(X), E)
compute maximum matching M in G
if ($|M| < |X|$) **then return** false
Define G_M by orienting G (edges in M are oriented from X to D(X), other edges in the opposite direction)
mark all arcs in G_M that are not in M as **unused**
compute SCCs in G_M and mark all arcs in a SCC as **used**
perform breadth-first in G_M search starting from M-free vertices, and mark all traversed arcs as **used**
for all arcs (x_i, d) in G_M marked as **unused do**
    remove d from $D(x_i)$
    if $D(x_i) = \emptyset$ **then return** false

3.2.2 Global Cardinality Constraint (from [11, 8])

For ease of simplicity we assume here that $| \cup_{x_i \in X} D(x_i)| = m$

$$\mathrm{GCC}(X = [x_1, \dots, x_n], [l_1, \dots, l_m], [u_1, \dots, u_m]) \tag{9}$$

EXERCISE n°14: GAC on GCC

$x_1 \in \{2\}, x_2 \in \{1, 2\}, x_3 \in \{2, 3\}, x_4 \in \{2, 3\}, x_5 \in \{1, 2, 3, 4\}, x_6 \in \{3, 4\}$,
$\mathrm{GCC}(X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$,[0,1,1,2],[3,2,1,3]).

— Give the domains after enforcing AC in the previous constraint network
— Give the state of the domains after propagation of a decomposition of the GCC into AtMost/AtLeast constraints

Theorem n°5 (from [11])
Let G be **the value network** of a sequence of variables $X = \{x_1, x_2, \dots, x_n\}$ with respective finite

domains $D(x_1), D(x_2), \ldots, D(x_n)$. The constraint $\mathrm{GCC}(X = [x_1, \ldots, x_n], [l_1, \ldots, l_m], [u_1, \ldots, u_m])$ is GAC if and only every arc in G belongs to a flow of value $|X|$ in the value network.

Theorem n°6 (from [1, 11])

Let f be a maximum flow in graph G, G_f its associated residual graph, and e an arc in G. There exists a maximum flow $f^{'}$ such that $f^{'}(e) > 0$ if and only if $f(e) > 0$ or e belongs to a circuit of G_f.

EXERCISE n°15: Balanced Academic curriculum (BACP)

The BACP is to design a balanced academic curriculum by assigning periods to courses in a way that the academic load of each period is balanced, i.e., as similar as possible.

An academic curriculum is defined by a set of courses $C = \{c_1, \ldots, c_n\}$ that have to be assigned in P periods. Some courses are required to others so that $R = \{(i, j)|c_j \text{ requires } c_i\}$. Each course c_i is associated to a number of credits or units s_i that represent the academic effort required to follow it. A minimum (resp maximum) α_1 (resp. β_1) number of academic credits per period is required (resp. allowed). A minimum (resp. maximum) α_2 (resp. β_2) number of courses is required (allowed). The goal is to minimise the maximum academic load for all periods.

The extended GCC (occurrences are now variables) :

$$\mathrm{GCC}(X = [x_1, \ldots, x_n], [o_1, \ldots, o_m]) \tag{10}$$

3.2.3 NValue (from [3, 7])

$$\textsc{NValue}(X = [x_1, \ldots, x_n], y) \mid \textsc{AtMostNvalue}([x_1, \ldots, x_n], y) \mid \textsc{AtLeastNvalue}([x_1, \ldots, x_n], y) \tag{11}$$

Enforcing GAC on NValue or even AtMostNvalue is NP-Hard.

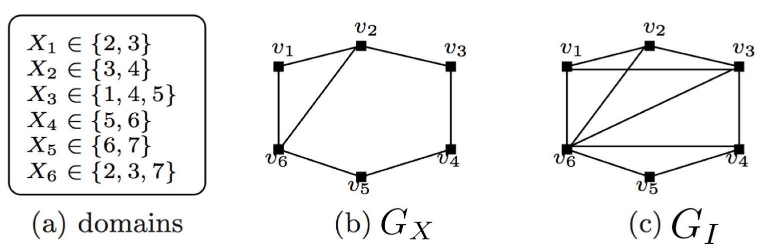

FIGURE 3 – Domains, intersection graph, interval graph (picture from [3]).

$\alpha(G)$ denotes the cardinality of a maximum independent set of the graph G.

Theorem n°7

AtMostNvalue($[x_1, \ldots, x_n], y$) is BC on y iff $|D(x_i)| \geq 1$ ($i = 1, \ldots, n$), $\alpha(G_I) \leq \overline{y}$ and $\alpha(G_I) \leq \underline{y}$.

EXERCISE n°16:

Consider the following constraint network : AtMostNvalue($[x_1, \ldots, x_6], y$) with $D(x_1) = [1, 6]$, $D(x_2) = \{2, 4\}$, $D(x_3) = \{1, 2\}$, $D(x_4) = [1, 2, 3]$, $D(x_5) = \{4, 5\}$, $D(x_6) = \{4, 5\}$, $D(y) = \{1, 2\}$. Give the interval graph, an independent set, enforce BC on y and suggest some filtering based on the graph viewpoint.

EXERCISE n°17: Guards

How many guards do you need to control the park of figure 4 ? A guard located at a cross-road can check

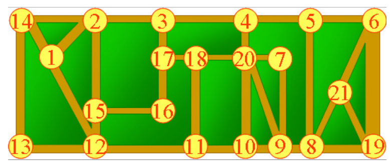

FIGURE 4 – The park and the possible observation points. Picture from [5]

all alleys intersecting that cross-road. We are looking for the minimum number of guards to ensure that all alleys are beeing watched. Some position might require to build a small watch-tower (when the alley is very long). The towers are of two different heights and the height required in position i is denoted $h_i \in \{0, 1, 2\}$ ($h_i = 0$ means that no towers is required in i, $h_i = 1$ is a medium tower, $h_i = 2$ is a big one) . We can have at most 3 towers of medium height and at most 2 big towers.

Question 1. Give a linear model for the problem.

Question 2. Give a CP model (discuss alternatives way to model the problem).

EXERCISE n°18:

Given a $n \times n$ chessboard, a **dominating set of queens** is a set of queens attacking all the cells of the board. The problem is to find a dominating set of queens of minimum cardinality. Give a CP model.

3.3 Scheduling, packing

3.3.1 BinPacking

$$\text{BINPACKING}([x_1, \ldots, x_n], [l_1, \ldots, l_m], [w_1, \ldots, w_n]) \tag{12}$$

EXERCISE n°19: Warehouse location

We must assign n clients to at most m depots that deliver goods to the clients. A client i requires a quantity d_i of goods. A transportation cost c_{ij} is paid if client i is delivered by depot j. A depot can serve at most w_j clients can deliver at most C_j units of goods. A fixed cost f is paid for each opened depot. The problem is to decide which depots to open and which depot serves each client so as to minimize the transportation cost. Give a constraint model for this "warehouse location" problem.

EXERCISE n°20:

Improve your previous model of the BACP.

3.3.2 Disjunctive (from [14])

$$\textsc{Disjunctive}([s_1,\dots,s_n],[e_1,\dots,e_n],[p_1,\dots,p_n]) \quad (13)$$

A disjunctive or Unary ressource : **A set of non-interruptible tasks T (activities) which must not overlap in time**. A task (activity) is described by three variables : (s_i, e_i, p_i).
- s_i is the starting time of the task
- e_i is the ending time
- p_i is the processing time

The convention is to have $s_i + p_i = e_i$ so that the task is not executed at time e_i but runs in $[s_i, e_i - 1]$.

- the earliest possible starting time : $est_i = \underline{s_i}$
- the latest possible completion time : $lct_i = \overline{e_i}$

For ease of simplicity we assume the processing time to be constant but the filtering can be applied using $\underline{p_i}$ if p_i is variable. **Earliest starting time**, **latest completion time** and **processing time** are also defined for sets of tasks Ω :

$$est_\Omega = min_{j\in\Omega}(est_j) \qquad | \qquad lct_\Omega = max_{j\in\Omega}(lct_j) \qquad | \qquad p_\Omega = \sum_{j\in\Omega} p_j$$

The filtering relies on estimations of the **earliest completion time** / **latest starting time** of a set Ω :

$$ect_\Omega = max_{\Omega'\subseteq\Omega}(est_{\Omega'} + p_{\Omega'}) \qquad | \qquad lst_\Omega = min_{\Omega'\subseteq\Omega}(lct_{\Omega'} - p_{\Omega'})$$

We focus on the update of $D(s_i)$ but all filtering rules given are symetric and can be used to update $D(e_i)$ (keep also in mind that BC is enforced on $s_i + p_i = e_i$)

1. **Compulsory parts** : Given a task i, if $lct_i - p_i < est_i + p_i$ then the interval $C_i = [lct_i - p_i, est_i + p_i[$ is compulsory. No other tasks can begin or end in the compulsory part of i.
2. **Overload checking** is a necessary condition for the Disjunctive to be satisfiable :

$$\forall \Omega \subseteq T, \qquad est_\Omega + p_\Omega \leq lct_\Omega$$

3. **Edge finding** is a filtering rule. Consider a set $\Omega \subset T$ and a task $i \notin \Omega$:

$$ect_{\Omega\cup\{i\}} + p_{\Omega\cup\{i\}} > lct_\Omega \qquad \Longrightarrow \qquad \underline{s_i} \geq ect_\Omega$$

4. **Not-Last** (resp. Not-First) is a filtering rule to detect that some task can not be scheduled last (resp. first) in a given set. The task i can not be scheduled after Ω ($i \notin \Omega$) if $est_\Omega + p_\Omega > lct_i - p_i$:

$$est_\Omega + p_\Omega > lct_i - p_i \qquad \Longrightarrow \qquad \overline{e_i} \leq max_{j\in\Omega}(\overline{e_j} - p_j)$$

5. **Detectable precedence** : a precedence $j << i$ between two tasks is discovered from the bounds :

$$est_i + p_i > lct_j - p_j \qquad \Longrightarrow \qquad j << i$$

The propagation rule is based on **all** the detected predecessors of i so that

$$\underline{s_i} \geq (ect_{\{j|j<<i\}})$$

EXERCISE n°21:

	E1	E2	E3
P1	5	33	5
P2	23	23	34
P3	5	55	37

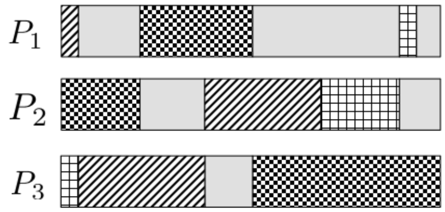

FIGURE 5 – Example of a schedule with $n = 3$ patients and $m = 3$ exams. Patient P1 does E3 immediatly, waits before E2 and waits again before E1.

n patients must each do m different examinations in a hospital. Each examination requires a specific medical team and has a duration that depends on the patient : d_{ij} is the duration of exam j for patient i. Examinations can be done in any order for a patient but each patient can not do two examinations at the same time. The goal is to close the service as early as possible. Figure 5 shows an example of data-set with 3 patients, 3 exams and the correspdonding durations. It also shows an example of solution. Give a CP model.

EXERCISE n°22: TSP-TW

Give a CP model for the TSP-TW. A nurse must visit and serve n patients (and come back to the hospital). Each patient must be served in a given time window $[a_i, b_i]$ and has a service time (care time) of c_i (the start of the service must be in the time-window). The nurse can eventually arrive at the patient's place before a_i but must wait a_i to start the service. The distance (resp. the time) between two patients i and j is denoted d_{ij} (resp t_{ij}). We are looking for the shortest tour (in distance) starting from the hospital (indexed by $i = 0$), visiting all patients and coming back to the hospital. Give a CP model.

3.3.3 Cumulative

We extend the task with a extra attribute, the height : h_i.

$$\textsc{Cumulative}([s_1, \ldots, s_n], [e_1, \ldots, e_n], [p_1, \ldots, p_n], [h_1, \ldots, h_n], C) \tag{14}$$

EXERCISE n°23:

m people must attend n classes (each class has a duration p_j of 30, 60, 90 or 120 minutes) between 8h00 and 17h00. k rooms can be used (they are all big enough to accomodate any meeting). Each participant i must follow 4 given and pre-defined classes : i_1, i_2, i_3, i_4. A lunch break has to take place between 12h and 13h. The problem is to design the planning ending as early as possible so we can all go to the beach.

---

4 Search techniques, heuristics, restart, randomization, LDS

5 Exercices on modelling

EXERCISE n°24: n-queens

Given a $n \times n$ chessboard, the problem is to place n-queens so that they don't attack each other. Give and discuss CP models.

EXERCISE n°25: Magic Series

A magic sequence of length n is a sequence of integers $x_0, \ldots x_{n-1}$ between 0 and $n-1$, such that for all i in 0 to $n-1$, the number i occurs exactly x_i times in the sequence. Example : [1,2,1,0]. Give a magic sequence for $n = 5$. Give a CP model to find magic sequences for a given n.

EXERCISE n°26: Redundant model for the TSP-TW

Extend your previous model for the TSP-TW with a redundant *viewpoint* that will strenghten the propagation of the lower bound.

EXERCISE n°27: Sport Scheduling

Consider n teams (n odd), $n/2$ periods and $n-1$ weeks.
- Every team must play against every other team
- A team plays exactly one game per week
- A team can play at most twice in the same period

	W1	W2	W3	W4	W5	W6	W7
P1	1 vs 2	1 vs 3	5 vs 8	4 vs 7	4 vs 8	2 vs 6	3 vs 5
P2	3 vs 4	2 vs 8	1 vs 4	6 vs 8	2 vs 5	1 vs 7	6 vs 7
P3	5 vs 6	4 vs 6	2 vs 7	1 vs 5	3 vs 7	3 vs 8	1 vs 8
P4	7 vs 8	5 vs 7	3 vs 6	2 vs 3	1 vs 6	4 vs 5	2 vs 4

Table 1 – Example of a solution of a sport scheduling problem for $n = 8$ teams.

EXERCISE n°28: Magic Squares

An order n magic square is a n by n matrix containing the numbers from 1 to n^2, such that each row, column and the two main diagonals equal the same sum. Give a CP model for this problem (pay attention to symetries).

EXERCISE n°29: Social golfer

The problem is to design m groups of n golfers over p weeks, such that each golfer plays in each week and no golfer plays in the same group as any other golfer twice. Give a CP model (pay attention to symetries).

Références

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows : Theory, Algorithms, and Applications.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[2] Claude Berge. Two Theorems in Graph Theory. *Proceedings of the National Academy of Sciences of the United States of America*, 43(9) :842–844, 1957.

[3] Christian Bessière, Emmanuel Hebrard, Brahim Hnich, Zeynep Kiziltan, and Toby Walsh. Filtering algorithms for the nvalue constraint. *Constraints*, 11(4) :271–293, 2006.

[4] Frédéric Boussemart, Fred Hemery, Christophe Lecoutre, and Lakhdar Sais. Boosting systematic search by weighting constraints. In Ramon López de Mántaras and Lorenza Saitta, editors, *ECAI*, pages 146–150. IOS Press, 2004.

[5] Romuald Debruyne. *Réseaux de contraintes à domaines discrets*. École des Mines de Nantes, 2002-2003.

[6] P. Hall. On representatives of subsets. *Journal of the London Mathematical Society*, s1-10(1) :26–30, 1935.

[7] Nina Narodytska. *Reformulation of Global Constraints*. PhD thesis, University of New South Wales, 2011.

[8] Claude-Guy Quimper. *Efficient Propagators for Global Constraints*. PhD thesis, University of Waterloo, 2006.

[9] Philippe Refalo. Impact-based search strategies for constraint programming. In Mark Wallace, editor, *CP*, volume 3258 of *Lecture Notes in Computer Science*, pages 557–571. Springer, 2004.

[10] Jean-Charles Régin. A filtering algorithm for constraints of difference in csps. In Barbara Hayes-Roth and Richard E. Korf, editors, *AAAI*, pages 362–367. AAAI Press / The MIT Press, 1994.

[11] Jean-Charles Régin. Generalized arc consistency for global cardinality constraint. In William J. Clancey and Daniel S. Weld, editors, *AAAI/IAAI, Vol. 1*, pages 209–215. AAAI Press / The MIT Press, 1996.

[12] Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006.

[13] Willem Jan van Hoeve. The alldifferent constraint : A survey. *CoRR*, cs.PL/0105015, 2001.

[14] Petr Vilím. *Global Constraints in Scheduling*. PhD thesis, Charles University in Prague, Faculty of Mathematics and Physics, Department of Theoretical Computer Science and Mathematical Logic, KTIML MFF, Universita Karlova, Malostranské náměstí 2/25, 118 00 Praha 1, Czech Republic, August 2007.