Problem Solving
Constraint Programming

Marie Pelleau
marie.pelleau@univ-cotedazur.fr

Greedy Algorithm

Principle

- At each step, a choice is made, the one that seems the best at that moment
- Builds a solution step by step
  - without revisiting previous decisions
  - by making at each step the choice that seems the best
  - hoping to achieve a global optimal result
- Greedy approach
  - depending on the problem, no guarantee of optimality (greedy heuristic)
  - low cost (compared to exhaustive enumeration)
  - intuitive choice

Local Search

Principle

- Start from an initial solution
- At each step, modify the solution
  - trying to improve the value of the objective function
  - hoping to achieve the global optimum
- Local approach
  - depending on the problem, no guarantee of optimality (heuristic)
  - low cost

Remark

When there are no objective function *Constraint Based Local Search*

Constraint Programming

- Tree Search
  - find a solution
  - find all solutions
  - find an optimal solution
  - prove the non-existence of a solution
- Complete Approach
  - guarantees optimality
  - more costly

Send More Money

Description

$$
\begin{array}{r}
S\,E\,N\,D \\
+\;\;M\,O\,R\,E \\
\hline
M\,O\,N\,E\,Y
\end{array}
$$

Contraintes possibles

$$
\begin{array}{ll}
 & s*1000 + e*100 + n*10 + d \\
C_1 : & + \; m*1000 + o*100 + r*10 + e \\
 & = m*10000 + o*1000 + n*100 + e*10 + y
\end{array}
$$

$C_2 : s \neq e$	$C_3 : s \neq n$	$C_4 : s \neq d$	$C_5 : s \neq m$	$C_6 : s \neq o$
$C_7 : s \neq r$	$C_8 : s \neq y$	$C_9 : e \neq n$	$C_{10} : e \neq d$	$C_{11} : e \neq m$
$C_{12} : e \neq o$	...	$C_{27} : o \neq r$	$C_{28} : o \neq y$	$C_{29} : r \neq y$

Send More Money

Description

$$
\begin{array}{r}
r_4\,r_3\,r_2\,r_1\; \\
S\,E\,N\,D \\
+\;\;M\,O\,R\,E \\
\hline
M\,O\,N\,E\,Y
\end{array}
$$

Contraintes possibles

$$
\begin{array}{lrl}
C_1 : & d + e = y + 10 * r_1 & r_1 \in \{0,1\} \\
C_2 : & r_1 + n + r = e + 10 * r_2 & r_2 \in \{0,1\} \\
C_3 : & r_2 + e + o = n + 10 * r_3 & r_3 \in \{0,1\} \\
C_4 : & r_3 + s + m = o + 10 * r_4 & r_4 \in \{0,1\} \\
C_5 : & r_4 = m &
\end{array}
$$

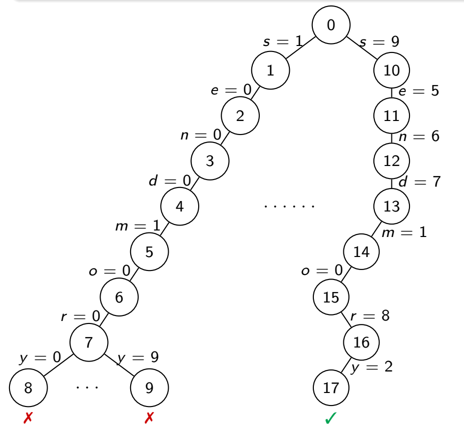
$C_6 : s \neq e$	$C_7 : s \neq n$	$C_8 : s \neq d$	$C_9 : s \neq m$	$C_{10} : s \neq o$
$C_{11} : s \neq r$	$C_{12} : s \neq y$	$C_{13} : e \neq n$	$C_{14} : e \neq d$	$C_{15} : e \neq m$
$C_{16} : e \neq o$	...	$C_{31} : o \neq r$	$C_{32} : o \neq y$	$C_{33} : r \neq y$

How to solve a CSP?

Generate and Test

Naive method

Generate all possible assignments and check if they correspond to solutions



Remark

To find the only solution, generates:

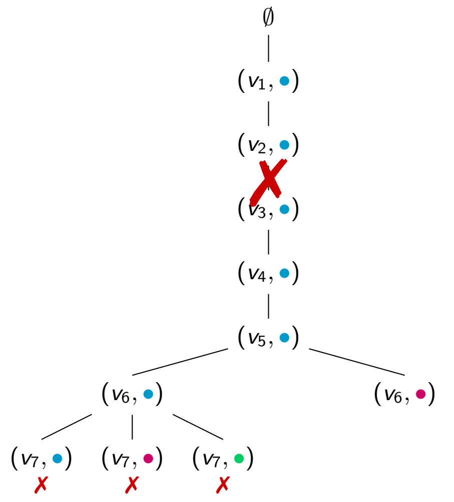
- $9^2 * 10^6 = 81\ 000\ 000$ leaves with the first model
- $2^4 * 9^2 * 10^6 = 1\ 296\ 000\ 000$ with the second

Can we do better?

Generate and Test

Coloriage de carte

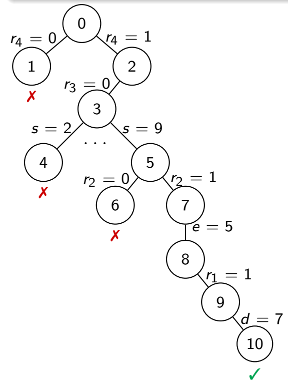
- $\mathcal{V} = \{v_1, \ldots, v_7\}$
- $D_1 = \cdots = D_7$
  $= \{\bullet, \bullet, \bullet\}$
- $C_1 : v_1 \neq v_2$
  $C_2 : v_1 \neq v_3$
  $C_3 : v_2 \neq v_3$
  $C_4 : v_2 \neq v_4$
  $C_5 : v_3 \neq v_4$
  $C_6 : v_3 \neq v_5$
  $C_7 : v_3 \neq v_6$
  $C_8 : v_4 \neq v_5$
  $C_9 : v_5 \neq v_6$
  $C_{10} : v_6 \neq v_7$



Forward Checking

As soon as a variable is assigned, we try to filter the values for the other variables

- Replace the variable with its value in all constraints
- Filtering can be applied if a constraint has only one remaining variable

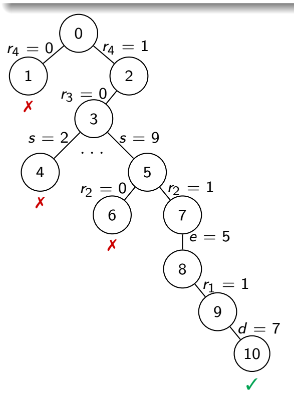


- $\{C_4 : r_3 + s + m = o + 10 * r_4,$
  $C_5 : r_4 = m\}$
  - $r_4 = 0$
    $\Rightarrow r_3 + s + m = o \wedge m = 0$ ✗
  - $r_4 = 1$
    $\Rightarrow r_3 + s + m = o + 10 \wedge m = 1$
    $\Rightarrow r_3 + s + 1 = o + 10$

Forward Checking

As soon as a variable is assigned, we try to filter the values for the other variables

- Replace the variable with its value in all constraints
- Filtering can be applied if a constraint has only one remaining variable



Remark

To find the only solution, generates:

- 483 840 leaves with the first model
- 57 with the second

Why wait for an assignment?

Method with Filtering

The **2 key steps** of constraint programming!

Propagation

Removes inconsistent values from the domains, meaning values that cannot be part of a solution.

Exploration

Assigns a value to a variable.

Propagation

Consistency for a constraint

Different types of consistency:

- Generalized arc consistency [Mackworth, 1977b]
- Path consistency [Montanari, 1974]
- Bound consistency [van Hentenryck et al., 1995]
- ...

All of these rely on the notion of support

Solving method | Method with Filtering

Propagation

Consistency for a constraint

Definition (Support)

Let $v_1, \dots, v_n$ be variables with finite discrete domains $D_1, \dots, D_n$, and C be a constraint. The value $x_i \in D_i$ has a **support** if and only if $\forall j \in [1, n], j \neq i, \exists x_j \in D_j$ such that $C(x_1, \dots, x_n)$ is true

Example

$C : r_4 = m$ with $D_{r_4} = [0, 1]$ and $D_m = [1, 9]$

- 1 for r_4 has a support: 1 for m because $C(1, 1)$ is true
- 0 for r_4 does not have a support: $\forall x_m \in D_m, C(0, x_m)$ is false

Constraint Programming 11 / 41

Solving method | Method with Filtering

Propagation

Consistency for a constraint

Definition (Support)

Let $v_1, \dots, v_n$ be variables with finite discrete domains $D_1, \dots, D_n$, and C be a constraint. The value $x_i \in D_i$ has a **support** if and only if $\forall j \in [1, n], j \neq i, \exists x_j \in D_j$ such that $C(x_1, \dots, x_n)$ is true

Example

$C : v_1 \neq v_2$ with $D_1 = D_2 = \{{\color{blue}\bullet}, {\color{red}\bullet}, {\color{green}\bullet}\}$

- ${\color{blue}\bullet}$ for v_1 has a support: ${\color{red}\bullet}$ for v_2 because $C({\color{blue}\bullet}, {\color{red}\bullet})$ is true
- ${\color{red}\bullet}$ for v_1 has a support: ${\color{blue}\bullet}$ for v_2 because $C({\color{red}\bullet}, {\color{blue}\bullet})$ is true
- ${\color{green}\bullet}$ for v_1 has a support: ${\color{blue}\bullet}$ for v_2 because $C({\color{green}\bullet}, {\color{blue}\bullet})$ is true

Constraint Programming 11 / 41

Solving method | Method with Filtering

Consistencies

Definition (Bound consistency)

Let $v_1, \dots, v_n$ be variables with finite discrete domains $D_1, \dots, D_n$, and C be a constraint. The domains are said to be **bound-consistent** (BC) for C if and only if $\forall i \in [1, n], D_i = [a_i, b_i]$, where a_i and b_i have a support.

Example

Consider two variables v_1, v_2 with domains $D_1 = D_2 = [-1, 4]$ and the constraint $v_1 = 2v_2$. The bound-consistent domains for this constraint are $D_1 = [0, 4]$ and $D_2 = [0, 2]$

Consistencies

Definition (Generalized Arc Consistency)

Let $v_1, \dots, v_n$ be variables with finite discrete domains $D_1, \dots, D_n$, and C be a constraint. The domains are said to be **generalized arc-consistent** (GAC) for C if and only if $\forall i \in [1, n], \forall x \in D_i, x$ has a support.

Example

Let v_1, v_2 be two variables with domains $D_1 = D_2 = [-1, 4]$ and the constraint $v_1 = 2v_2$. The arc-consistent domains for this constraint are $D_1 = \{0, 2, 4\}$ and $D_2 = \{0, 1, 2\}$

Arc Consistency

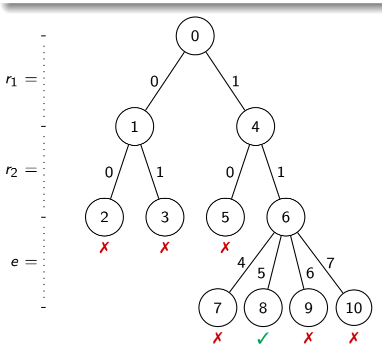
Several implementations

- AC1 and AC3 [Mackworth, 1977a]
- AC4 [Mohr and Henderson, 1986]
- AC5 [van Hentenryck et al., 1992]
- AC6 [Bessière, 1994]
- AC7 [Bessière et al., 1999]
- AC2001 [Bessière and Régin, 2001]
- AC3.2 and AC3.3 [Lecoutre et al., 2003]

Maintaining Generalized Arc Consistency

Two phases alternate:

- Propagation, using generalized arc consistency
- Exploration, making a choice



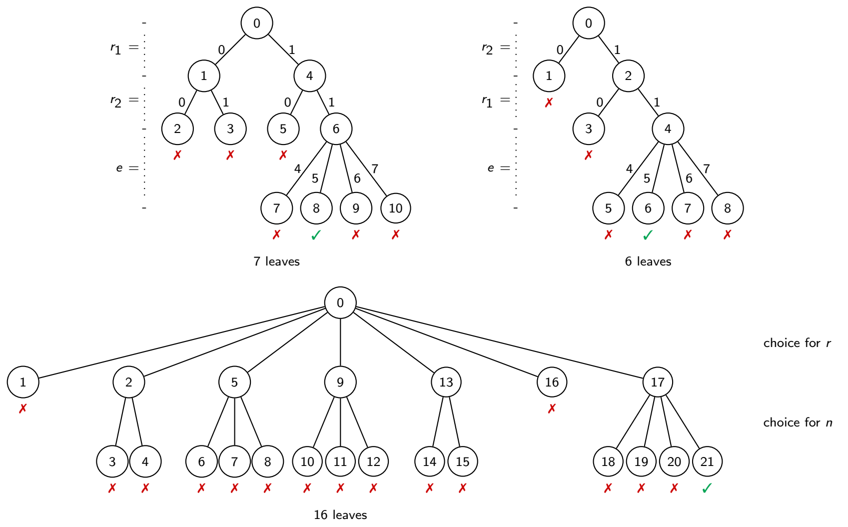
Remark

To find the only solution, generates:

- 6 leaves with the first model
- 7 with the second

Search Strategy

The order of the variables matters



Search Strategy

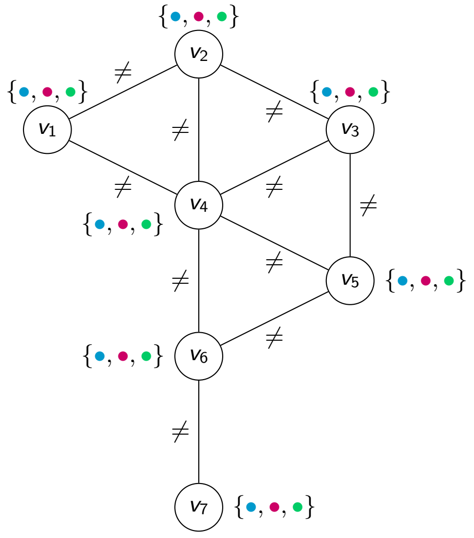
Choose a variable

- Having the smallest domain (dom), First-fail [Haralick and Elliott, 1979]
  "To succeed, try first where you are most likely to fail":
- Appearing in the greatest number of constraints (deg)
- dom + deg [Brélaz, 1979]
- dom/deg [Bessière and Régin, 1996]
- dom/$w$deg [Boussemart et al., 2004]
- ...

Search Strategy

Coloriage de carte

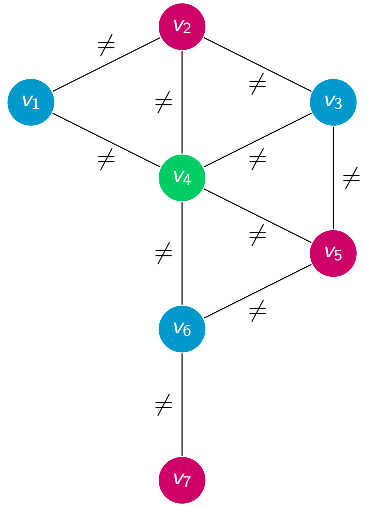
- $\mathcal{V} = \{v_1, \ldots, v_7\}$
- $D_1 = \cdots = D_7 = \{\bullet, \bullet, \bullet\}$
- $C_1 : v_1 \neq v_2$
  $C_2 : v_1 \neq v_4$
  $C_3 : v_2 \neq v_3$
  $C_4 : v_2 \neq v_4$
  $C_5 : v_3 \neq v_4$
  $C_6 : v_3 \neq v_5$
  $C_7 : v_4 \neq v_5$
  $C_8 : v_4 \neq v_6$
  $C_9 : v_5 \neq v_6$
  $C_{10} : v_6 \neq v_7$

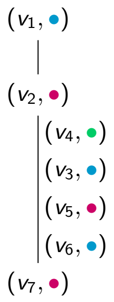


Search Strategy - dom

Coloriage de carte

- $\mathcal{V} = \{v_1, \ldots, v_7\}$
- $D_1 = \cdots = D_7 = \{\bullet, \bullet, \bullet\}$
- $C_1 : v_1 \neq v_2$
  $C_2 : v_1 \neq v_4$
  $C_3 : v_2 \neq v_3$
  $C_4 : v_2 \neq v_4$
  $C_5 : v_3 \neq v_4$
  $C_6 : v_3 \neq v_5$
  $C_7 : v_4 \neq v_5$
  $C_8 : v_4 \neq v_6$
  $C_9 : v_5 \neq v_6$
  $C_{10} : v_6 \neq v_7$

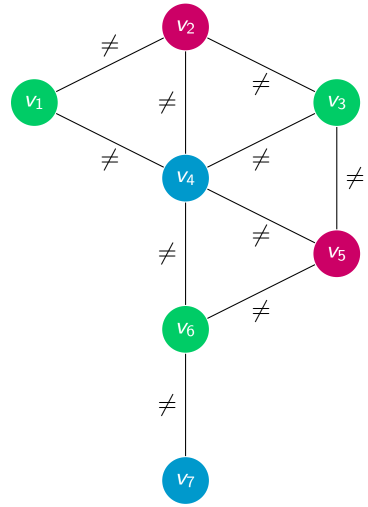


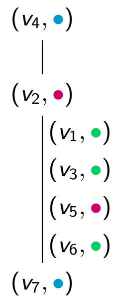


Search Strategy - deg

Coloriage de carte

- $\mathcal{V} = \{v_1, \ldots, v_7\}$
- $D_1 = \cdots = D_7 = \{\bullet, \bullet, \bullet\}$
- $C_1 : v_1 \neq v_2$
  $C_2 : v_1 \neq v_4$
  $C_3 : v_2 \neq v_3$
  $C_4 : v_2 \neq v_4$
  $C_5 : v_3 \neq v_4$
  $C_6 : v_3 \neq v_5$
  $C_7 : v_4 \neq v_5$
  $C_8 : v_4 \neq v_6$
  $C_9 : v_5 \neq v_6$
  $C_{10} : v_6 \neq v_7$



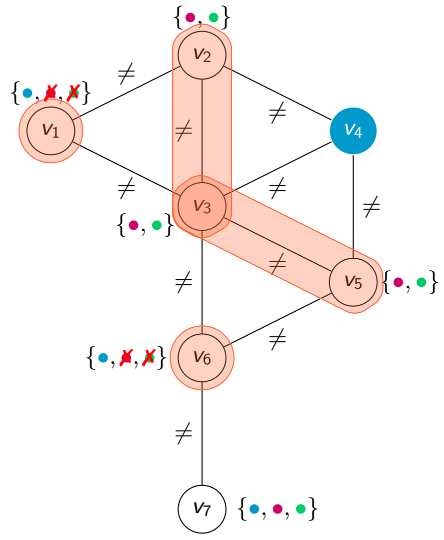


Does it work all the time?

Limites

Coloriage de carte

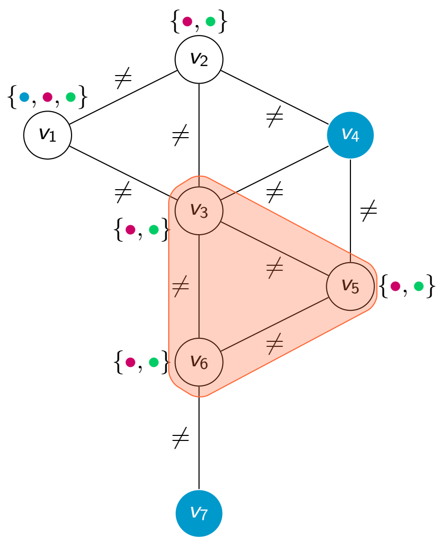
- $\mathcal{V} = \{v_1, \ldots, v_7\}$
- $D_1 = \cdots = D_7$ $= \{\bullet, \bullet, \bullet\}$
- $C_1 : v_1 \neq v_2$
  $C_2 : v_1 \neq v_3$
  $C_3 : v_2 \neq v_3$
  $C_4 : v_2 \neq v_4$
  $C_5 : v_3 \neq v_4$
  $C_6 : v_3 \neq v_5$
  $C_7 : v_3 \neq v_6$
  $C_8 : v_4 \neq v_5$
  $C_9 : v_5 \neq v_6$
  $C_{10} : v_6 \neq v_7$



Limites

Coloriage de carte

- $\mathcal{V} = \{v_1, \ldots, v_7\}$
- $D_1 = \cdots = D_7$ $= \{\bullet, \bullet, \bullet\}$
- $C_1 : v_1 \neq v_2$
  $C_2 : v_1 \neq v_3$
  $C_3 : v_2 \neq v_3$
  $C_4 : v_2 \neq v_4$
  $C_5 : v_3 \neq v_4$
  $C_6 : v_3 \neq v_5$
  $C_7 : v_3 \neq v_6$
  $C_8 : v_4 \neq v_5$
  $C_9 : v_5 \neq v_6$
  $C_{10} : v_6 \neq v_7$



Global Constraints

Allows representing a set of constraints

- Facilitates modeling
- Dedicated algorithm to remove inconsistent values from domains

Constraint Catalog [Beldiceanu et al., 2010]

The most well-known

- alldifferent
- cycle
- global_cardinality
- nvalue
- element

alldifferent Constraint

First presented in [Lauriere, 1978]
Returns **true** if all variables are pairwise different

Example

The difference constraints in the send + more = money problem can be rewritten as
$\text{alldifferent}(s, e, n, d, m, o, r, y)$

alldifferent Constraint

Map Coloring

- $\mathcal{V} = \{v_1, \ldots, v_7\}$
- $D_1 = \cdots = D_7 = \{\bullet, \bullet, \bullet\}$
- $C_1 : v_1 \neq v_2$
  $C_2 : v_1 \neq v_3$
  $C_3 : v_2 \neq v_3$
  $C_4 : v_2 \neq v_4$
  $C_5 : v_3 \neq v_4$
  $C_6 : v_3 \neq v_5$
  $C_7 : v_3 \neq v_6$
  $C_8 : v_4 \neq v_5$
  $C_9 : v_5 \neq v_6$
  $C_{10} : v_6 \neq v_7$

- $\mathcal{V} = \{v_1, \ldots, v_7\}$
- $D_1 = \cdots = D_7 = \{\bullet, \bullet, \bullet\}$
- $C_1 : \text{alldifferent}(v_1, v_2, v_3)$
  $C_2 : \text{alldifferent}(v_2, v_3, v_4)$
  $C_3 : \text{alldifferent}(v_3, v_4, v_5)$
  $C_4 : \text{alldifferent}(v_3, v_5, v_6)$
  $C_5 : v_6 \neq v_7$

alldifferent Constraint

- Not just syntactic sugar
  - Arc-consistency
    - Developed independently by [Costa, 1994] and [Régin, 1994]
    - Based on graph theory
  - Bound-consistency
    - Developed by [Puget, 1998] and later improved by [Mehlhorn and Thiel, 2000] and [Lopez-Ortiz et al., 2003]
    - Based on the concept of Hall's interval

Value Graph

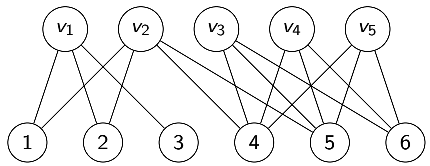
Definition (Value Graph)

From the variables and domains of a CSP, we can create a bipartite graph, called the **value graph**

- The vertices correspond to the variables and the values
- An edge connects a variable v_i and a value x if $x \in D_i$

Example

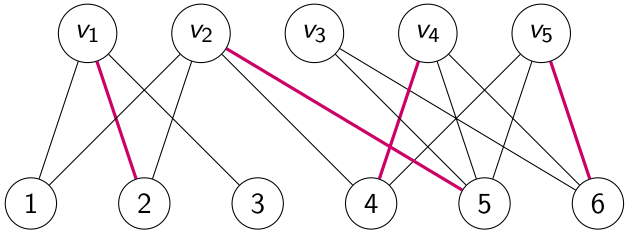
- $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$
- $D_1 = \{1, 2, 3\}$, $D_2 = \{1, 2, 4, 5\}$, et $D_3 = D_4 = D_5 = \{4, 5, 6\}$



Graph theory

Definition (Matching)

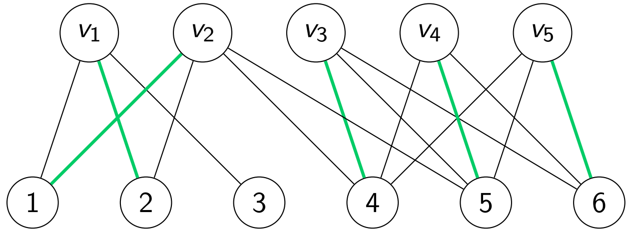
Given a graph $G = (V, E)$, a subset M of the edges E is called a **matching** if and only if no two edges share a vertex,



Graph theory

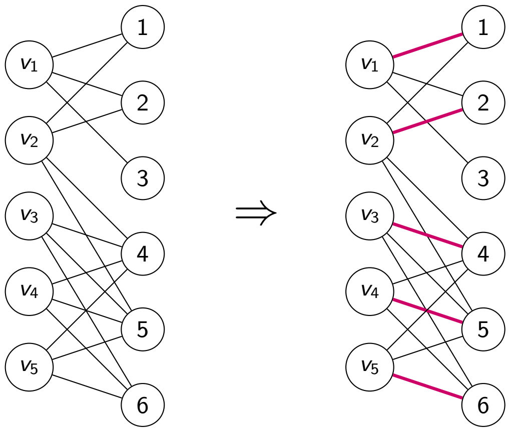
Definition (Maximal Matching)

A matching is said to be **maximal** if it contains the maximum number of edges possible.



The Hopcroft-Karp algorithm [Hopcroft and Karp, 1973] allows for calculating the maximal matching in a bipartite graph

Hopcroft-Karp algorithm



Strongly Connected Component

Definition (Directed Graph)

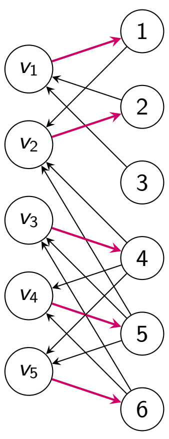
A **directed** graph $G = (V, E)$ is a graph where the edges have a direction, and they are called **arcs**

Definition (Strongly Connected Component)

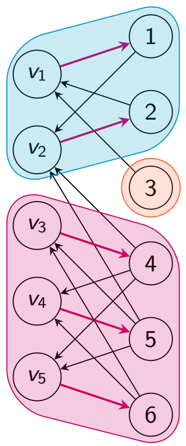
Given a **directed** graph $G = (V, E)$, a **strongly connected component** is a maximal set of vertices such that for each vertex in the set, there exists a path to every other vertex in the set

Tarjan's algorithm [Tarjan, 1972] efficiently computes the strongly connected components in a graph

Tarjan algorithm



$\Rightarrow$

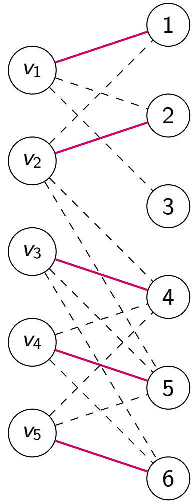


alldifferent: propagation for arc-consistency

Exemple

- $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$
- $D_1 = \{1, 2, 3\}$, $D_2 = \{1, 2, 4, 5\}$, et $D_3 = D_4 = D_5 = \{4, 5, 6\}$

- We find a maximal matching $\Rightarrow$ **a solution**

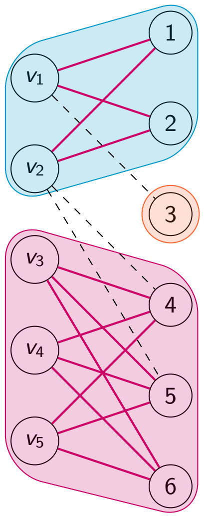


alldifferent: propagation for arc-consistency

Exemple

- $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$
- $D_1 = \{1, 2, 3\}$, $D_2 = \{1, 2, 4, 5\}$, et $D_3 = D_4 = D_5 = \{4, 5, 6\}$

- We find a maximal matching $\Rightarrow$ **a solution**
- We search for strongly connected components $\Rightarrow$ **permutations**

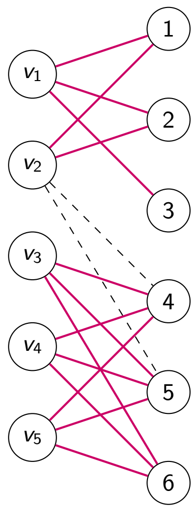


alldifferent: propagation for arc-consistency

Exemple

- $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$
- $D_1 = \{1, 2, 3\}$, $D_2 = \{1, 2, 4, 5\}$, et $D_3 = D_4 = D_5 = \{4, 5, 6\}$

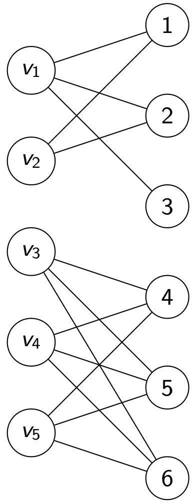
- We find a maximal matching $\Rightarrow$ **a solution**
- We search for strongly connected components $\Rightarrow$ **permutations**
- We add the isolated values to the initial domains



alldifferent: propagation for arc-consistency

Exemple

- $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$
- $D_1 = \{1, 2, 3\}$, $D_2 = \{1, 2\}$, et $D_3 = D_4 = D_5 = \{4, 5, 6\}$



- We find a maximal matching $\Rightarrow$ **a solution**
- We search for strongly connected components $\Rightarrow$ **permutations**
- We add the isolated values to the initial domains

Hall's Interval

Definition

Let $(v_1, \ldots, v_n)$ be variables with finite discrete domains $(D_1, \ldots, D_n)$. Given an interval I, we define $K_I = \{v_i \mid D_i \subseteq I\}$. I is a **Hall's interval** if $|I| = |K_I|$.

Example

Consider the following problem:

- $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$
- $D_1 = [1, 3]$, $D_2 = [1, 5]$, et $D_3 = D_4 = D_5 = [4, 6]$
- $I = [4, 6]$ is a Hall's interval because $K_I = \{v_3, v_4, v_5\}$ and we have $|I| = |K_I|$
- $I = [1, 3]$ is not a Hall's interval because $K_I = \{v_1\}$ and $|I| \neq |K_I|$

alldifferent: propagation for bound-consistency

- For each lower bound a and upper bound b of the domains, we check if $I = [a, b]$ is a Hall's interval
- If I is a Hall's interval, we can remove the values in I from the domains of variables in $\mathcal{V} \setminus K_I$

Exemple

Consider the following problem:

- $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$
- $D_1 = [1, 3]$, $D_2 = [1, 3]$, et $D_3 = D_4 = D_5 = [4, 6]$
- $I = [1, 6]$ is not a Hall's interval
- $I = [1, 5]$ is not a Hall's interval
- $I = [1, 3]$ is not a Hall's interval
- $I = [4, 5]$ is not a Hall's interval
- $I = [4, 6]$ is a Hall's interval $\Rightarrow$ **we remove the values 4, 5, 6 from the domains of variables not in K_I**

global_cardinality Constraint

First presented in [Oplobedu et al., 1989]

$$\text{global\_cardinality}(\underbrace{\{v_1, \ldots, v_n\}}_{\text{Variables}}, \underbrace{\{x_1, \ldots, p\}}_{\text{Values}}, \underbrace{\{nb_1, \ldots, nb_p\}}_{\text{Occurrences}})$$

Returns **true** if among the variables $\{v_1, \ldots, v_n\}$, there are nb_i variables having the value x_i

Exemple

$\text{global\_cardinality}(\{v_1, v_2, v_3, v_4, v_5, v_6\}, \{0, 1\}, \{2, 4\})$

In some cases, we can express an alldifferent using a global_cardinality

$\text{alldifferent}(v_1, v_2, v_3) = \text{global\_cardinality}(\{v_1, v_2, v_3\}, \{\bullet, \bullet, \bullet\}, \{1, 1, 1\})$

global_cardinality Constraint

- Arc-consistency
  - Developed by [Régin, 1996]
  - Based on a flow algorithm
- Bound-consistency
  - Developed by [Quimper et al., 2003]
  - Based on the concept of Hall's interval
  - Developed by [Katriel and Thiel, 2003]
  - Based on convexity to improve the efficiency of the flow algorithm

Sports Schedule

Description

- n teams, $n-1$ weeks, and $n/2$ periods
- each pair of teams plays exactly once
- each team plays one match every week
- each team plays at most 2 times in the period

Example (Possible solution)

	S1	S2	S 3	S4	S5	S6	S7
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

Magic Sequence

Description

A magic sequence of length n is a sequence of integers $v_0, \ldots, v_{n-1}$, where each integer $i \in \{0, \ldots, n-1\}$ appears exactly v_i times in the sequence

Magic Sequence ($n = 10$)

	0	1	2	3	4	5	6	7	8	9
v_i	6	2	1	0	0	0	1	0	0	0

Langford Sequence

Description

A Langford sequence is a sequence of integers $v_1, \ldots, v_{k \times n}$, where each integer $i \in \{1, \ldots, n\}$ appears exactly k times, and the two successive occurrences of i are separated by a distance of i

We consider here only the case for $k = 2$

Langford Sequence ($n = 7$)

4

7 3 6 2 5 3 2 4 7 6 5 1 4 1

7 1

Alice and Bob are Going to Work

Description

- Alice goes to work by car (30 to 40 minutes) or by bus (at least 60 min)
- Bob goes by bike (40 or 50 min) or by motorbike (20 to 30 min)
- This morning:
  - Alice left her house between 7:10 AM and 7:20 AM
  - Bob arrived at work between 8:00 AM and 8:10 AM
  - Alice arrived 10 to 20 minutes after Bob left

1. Model this problem
2. Is the story consistent?
3. When did Bob leave? Is it possible that he took his bike?
4. Is the story consistent if we add that:
   - Alice's car is broken down
   - Alice and Bob met on the way

Binairo – 2018 Exam

Description

A Belgian game, based on a square grid with only the digits 0 and 1. On each row and each column:

- there are as many 0's as 1's
- there cannot be more than 2 identical digits next to each other

No two rows or columns can be identical.

Example Grid

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

Beldiceanu, N., Carlsson, M., and Rampon, J.-X. (2010).
Global constraint catalog, 2nd edition.
Technical Report T2010:07, The Swedish Institute of Computer Science.

Bessière, C. (1994).
Arc-consistency and arc-consistency again.
Artificial Intelligence, 65(1):179–190.

Bessière, C., Freuder, E. C., and Régin, J.-C. (1999).
Using constraint metaknowledge to reduce arc consistency computation.
Artificial Intelligence, 107(1):125–148.

Bessière, C. and Régin, J.-C. (1996).
Mac and combined heuristics: Two reasons to forsake fc (and cbj?) on hard problems.
In *Proceedings of the Second International Conference on Principles and Practice of Constraint Programming*, volume 1118 of *Lecture Notes in Computer Science*. Springer.

Bessière, C. and Régin, J.-C. (2001).
Refining the basic constraint propagation algorithm.
In *Proceedings of the 17th International Joint Conference on Artificial intelligence (IJCAI'01)*, pages 309–315. Morgan Kaufmann.

Boussemart, F., Hemery, F., Lecoutre, C., and Sais, L. (2004).
Boosting systematic search by weighting constraints.
In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, (ECAI'2004)*, pages 146–150. IOS Press.

Brélaz, D. (1979).
New methods to color the vertices of a graph.
Communications of the ACM, 22(4):251–256.

Costa, M.-C. (1994).
Persistency in maximum cardinality bipartite matchings.
Operations Research Letters, 15(3):143–149.

Haralick, R. M. and Elliott, G. L. (1979).

Increasing tree search efficiency for constraint satisfaction problems.
In *Proceedings of the 6th International Joint Conference on Artificial intelligence (IJCAI'79)*, pages 356–364. Morgan Kaufmann Publishers Inc.

Hopcroft, J. E. and Karp, R. M. (1973).
An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs.
SIAM Journal on Computing, 2(4):225–231.

Katriel, I. and Thiel, S. (2003).
Fast bound consistency for the global cardinality constraint.
In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP'03)*, volume 2833 of *Lecture Notes in Computer Science*, pages 437–451. Springer Berlin / Heidelberg.

Lauriere, J.-L. (1978).
A language and a program for stating and solving combinatorial problems.
Artificial Intelligence, 10(1):29 – 127.

Lecoutre, C., Boussemart, F., and Hemery, F. (2003).
Exploiting multidirectionality in coarse-grained arc consistency algorithms.
In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP'03)*, volume 2833 of *Lecture Notes in Computer Science*, pages 480–494. Springer.

Lopez-Ortiz, A., Quimper, C.-G., Tromp, J., and Beek, P. V. (2003).
A fast and simple algorithm for bounds consistency of the all different constraint.
In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 245–250.

Mackworth, A. K. (1977a).
Consistency in networks of relations.
Artificial Intelligence, 8(1):99–118.

Mackworth, A. K. (1977b).
On reading sketch maps.
In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 598–606.

Mehlhorn, K. and Thiel, S. (2000).

Faster algorithms for bound-consistency of the sortedness and the alldifferent constraint.
In *Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming (CP '00)*, volume 1894 of *Lecture Notes in Computer Science*, pages 306–319. Springer.

Mohr, R. and Henderson, T. C. (1986).
Arc and path consistency revisited.
Artificial Intelligence, 28(2):225–233.

Montanari, U. (1974).
Networks of constraints: Fundamental properties and applications to picture processing.
Information Science, 7(2):95–132.

Oplobedu, A., Marcovitch, J., and Tourbier, Y. (1989).
Charme: Un langage industriel de programmation par contraintes, illustré par une application chez renault.
In *Proceedings of the Ninth International Workshop on Expert Systems and their Applications: General Conference*, pages 155–70.

Puget, J.-F. (1998).
A fast algorithm for the bound consistency of alldiff constraints.
In *Proceedings of the 15th National/10th Conference on Artificial Intelligence/Innovative applications of artificial intelligence (AAAI '98/IAAI '98)*, pages 359–366. American Association for Artificial Intelligence.

Quimper, C.-G., van Beek, P., López-Ortiz, A., Golynski, A., and Sadjad, S. (2003).
An efficient bounds consistency algorithm for the global cardinality constraint.
In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP'03)*, volume 2833 of *Lecture Notes in Computer Science*, pages 600–614. Springer Berlin / Heidelberg.

Régin, J.-C. (1994).
A filtering algorithm for constraints of difference in csps.
In *Proceedings of the 12th National Conference on Artificial Intelligence (Vol. 1)*, pages 362–367.

Régin, J.-C. (1996).
Generalized arc consistency for global cardinality constraint.
In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-96)*, pages 209–215.

Tarjan, R. (1972).
Depth-first search and linear graph algorithms.
SIAM Journal on Computing, 1(2):146–160.

van Hentenryck, P., Deville, Y., and Teng, C.-M. (1992).
A generic arc-consistency algorithm and its specializations.
Artificial Intelligence, 57.

van Hentenryck, P., Saraswat, V. A., and Deville, Y. (1995).
Design, implementation, and evaluation of the constraint language cc(fd).
In *Selected Papers from Constraint Programming: Basics and Trends*, pages 293–316. Springer-Verlag.