

Programmation par contraintes

Laurent Beaudou

On se trouve où ?

- **Un problème, une solution** : la solution est-elle une solution du problème ?
 - simulation, vérification

On se trouve où ?

- **Un problème, une solution** : la solution est-elle une solution du problème ?
 - simulation, vérification

- **Un problème** : trouver la meilleure solution du problème
 - programmation linéaire

On se trouve où ?

- **Un problème, une solution** : la solution est-elle une solution du problème ?
 - simulation, vérification
- **Un problème** : trouver une solution du problème
- **Un problème** : trouver la meilleure solution du problème
 - programmation linéaire

On se trouve où ?

- **Un problème, une solution** : la solution est-elle une solution du problème ?
 - simulation, vérification
- **Un problème** : trouver une solution du problème
 - **programmation par contrainte (CSP)**
- **Un problème** : trouver la meilleure solution du problème
 - programmation linéaire

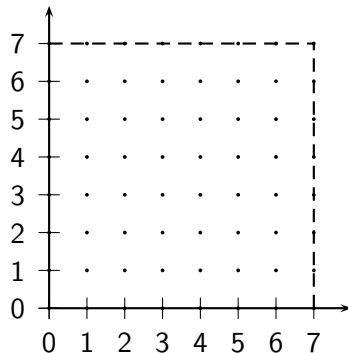
Un premier exemple

Domaine des variables

- $x, y \in \{0, 1 \dots 7\}$

Contraintes

- $x - y = 6$
- $x + y = 5$
- $x > y$



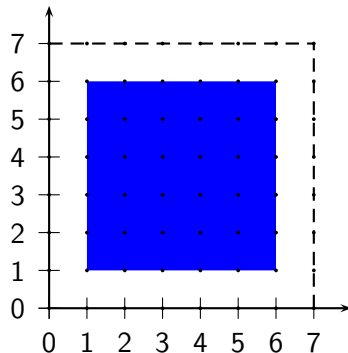
Un premier exemple

Domaine des variables

- $x, y \in \{0, 1 \dots 7\}$

Contraintes

- $x \neq y = 6$
- $x + y = 5$
- $x > y$



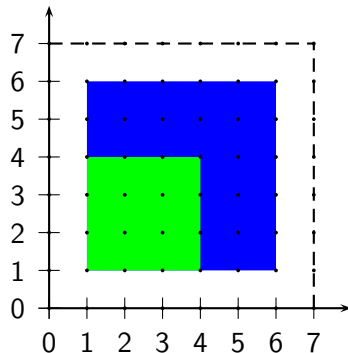
Un premier exemple

Domaine des variables

- $x, y \in \{0, 1 \dots 7\}$

Contraintes

- $x \quad y = 6$
- $x + y = 5$
- $x > y$



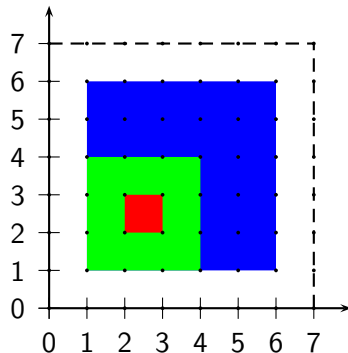
Un premier exemple

Domaine des variables

- $x, y \in \{0, 1 \dots 7\}$

Contraintes

- $x \neq y = 6$
- $x + y = 5$
- $x > y$



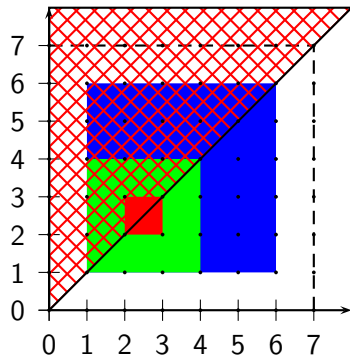
Un premier exemple

Domaine des variables

- $x, y \in \{0, 1 \dots 7\}$

Contraintes

- $x \leq y = 6$
- $x + y = 5$
- $x > y$



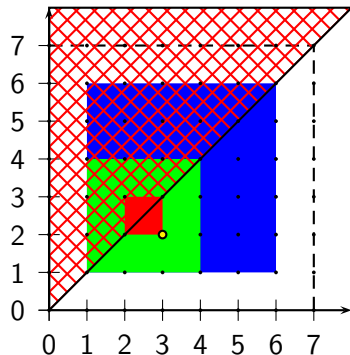
Un premier exemple

Domaine des variables

- $x, y \in \{0, 1 \dots 7\}$

Contraintes

- $x \leq y = 6$
- $x + y = 5$
- $x > y$



Programmation par contraintes : plan

- 1 Programmation par contrainte
- 2 Modélisation
- 3 Résolution

Programmation par contraintes : plan

1 Programmation par contrainte

2 Modélisation

3 Résolution

La programmation par contraintes

Objectif

- Trouver une **solution réalisable**
- Qui respecte des **contraintes** faciles à vérifier
 - Formules mathématiques simples, tableaux de valeurs possibles
- Avec des ensembles de valeurs possibles pour les **variables de décision**

Domaines

Définition

$X_1, X_2 \dots X_n$ les variables de décision

Domaine

La variable X_i doit prendre ses valeurs dans le domaine D_i

- discret
- fini ou infini

Domaines

Exercice

Sont-ce des domaines ?

- $\{1, 2, 3, 4\}$
- L'ensemble des entiers naturels.
- L'ensemble des réels
- Les entiers naturels impairs
- L'intervalle $[-1, 1]$
- Les points à coordonnées entières du plan

Domaines

Exercice

Sont-ce des domaines ?

- $\{1, 2, 3, 4\}$ ✓
- L'ensemble des entiers naturels. ✓
- L'ensemble des réels ✗
- Les entiers naturels impairs ✓
- L'intervalle $[-1, 1]$ ✗
- Les points à coordonnées entières du plan ✓

Contraintes

C'est quoi ?

Variables } Contraintes
Domaines }

Une contrainte restreint les valeurs que l'on peut affecter simultanément à des variables

Contraintes

C'est quoi ?

Variables } Contraintes
Domaines }

Une contrainte restreint les valeurs que l'on peut affecter simultanément à des variables

Exemples :

- $2x + 3y = 12$
- $x \neq 3y$
- $3x^2 = 3$
- (ABC) forme un triangle isocèle
- $A \cup B \subseteq C$
- x, y, z distincts deux à deux

Contraintes

Déclaration d'une contrainte

2 types de déclaration :

- **extension** : on énumère les valeurs admises
 - $(x = 1 \text{ et } y = 2) \text{ ou } (x = 2 \text{ et } y = 4) \text{ ou } (x = 3 \text{ et } y = 0)$
- **intension** : on utilise les signes mathématiques connus
 - $x < y$

Contraintes

Exemples supplémentaires

- Contraintes logiques
 - Si $x = 4$ alors $y = 5$
 - $x = y$ ou $x = 2y$
- Contraintes globales
 - Toutes les variables sont différentes
- Méta-contraintes
 - La valeur 5 est utilisée exactement 3 fois

Problème de satisfaction de contraintes

(X, D, C) tel que

$$\begin{cases} X = \{X_1, X_2, \dots, X_n\} & \text{l'ensemble des variables} \\ D = \{D_1, D_2, \dots, D_n\} & \text{où } D_i \text{ est le domaine de } X_i \\ C = \{C_1, C_2, \dots, C_m\} & \text{l'ensemble des contraintes} \end{cases}$$

Exemple

- $X = \{a, b, c, d\}$
- $D(a) = D(b) = D(c) = D(d) = \{0, 1\}$
- $C = \{a \neq b, c \neq d, a + c < b\}$

Solution d'un CSP

Une **affectation** est le fait d'instancier des variables.

$$A = \{(X_1, V_1), (X_3, V_3), (X_4, V_4)\}$$

associe la valeur V_1 de D_1 à la variable X_1 ...

Une affectation est :

- **partielle** ou **totale**
- **constistante** ou **inconsistante**

Une **solution** est une affectation totale consistante.

Solution d'un CSP

Problème

- $X = \{a, b, c, d\}$
- $D(a) = D(b) = D(c) = D(d) = \{0, 1\}$
- $C = \{a \neq b, c \neq d, a + c < b\}$

- $A_1 = \{(a, 1), (b, 0), (c, 0), (d, 0)\}$
- $A_2 = \{(a, 0), (b, 0)\}$
- $A_3 = \{(a, 1), (b, 0)\}$
- $A_4 = \{(a, 0), (b, 1), (c, 0), (d, 1)\}$

Solution d'un CSP

Problème

- $X = \{a, b, c, d\}$
- $D(a) = D(b) = D(c) = D(d) = \{0, 1\}$
- $C = \{a \neq b, c \neq d, a + c < b\}$

- $A_1 = \{(a, 1), (b, 0), (c, 0), (d, 0)\}$
- $A_2 = \{(a, 0), (b, 0)\}$
- $A_3 = \{(a, 1), (b, 0)\}$
- $A_4 = \{(a, 0), (b, 1), (c, 0), (d, 1)\}$

- totale, inconsistante
- partielle, inconsistante
- partielle, consistante
- totale, consistante

A_4 est donc une solution

Le jeu du : CSP, pas CSP !

- Connaissant les pièces de mon porte-monnaie, puis-je rendre exactement 8 euros et 57 centimes ?
- Connaissant les pièces de mon porte-monnaie, quelle est la meilleure façon de rendre 8 euros et 57 centimes ?
- Est-il possible de colorier une carte avec 5 couleurs sans frontière monochrome ?
- Sur un cercle, trouver un point à distance 2 d'une droite.

Le jeu du : CSP, pas CSP !

- Connaissant les pièces de mon porte-monnaie, puis-je rendre exactement 8 euros et 57 centimes ? ✓
- Connaissant les pièces de mon porte-monnaie, quelle est la meilleure façon de rendre 8 euros et 57 centimes ? ✗
- Est-il possible de colorier une carte avec 5 couleurs sans frontière monochrome ? ✓
- Sur un cercle, trouver un point à distance 2 d'une droite. ✗

Programmation par contraintes : plan

1 Programmation par contrainte

2 Modélisation

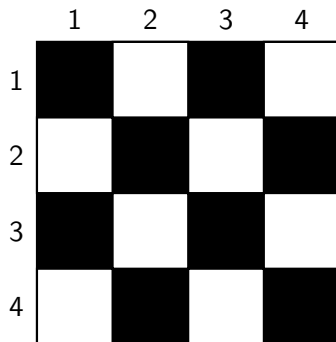
3 Résolution

Un constat

Un problème est rarement donné sous la forme d'un CSP

Le problème des reines

Jeu : placer 4 reines sur un échiquier 4x4 sans qu'elles ne se menacent

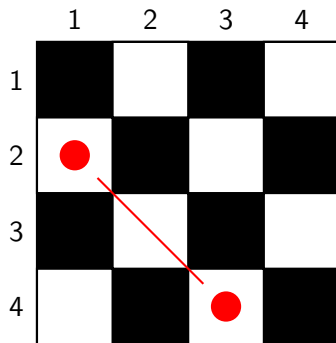


Le problème des reines

Jeu : placer 4 reines sur un échiquier 4 4

Une seule dame

- par ligne
- par colonne
- par diagonale

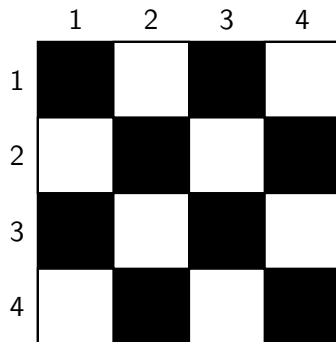


Le problème des reines

Jeu : placer 4 reines sur un échiquier 4 4

Modélisation

- **variables** L_1, L_2, L_3, L_4 et C_1, C_2, C_3, C_4
- **domaines** $L_i \in \{1, 2, 3, 4\}$ et $C_i \in \{1, 2, 3, 4\}$
- **contraintes** lignes, colonnes et diagonales différentes.



Modélisation

Modélisation

- **variables** L_1, L_2, L_3, L_4 et C_1, C_2, C_3, C_4
- **domaines** $L_i \in \{1, 2, 3, 4\}$ et $C_i \in \{1, 2, 3, 4\}$
- **contraintes**
 - $\forall i \neq j, L_i \neq L_j$
 $C_i \neq C_j$
 - $\forall i \neq j, L_i + C_i \neq L_j + C_j$
 $L_i \neq L_j \vee C_i \neq C_j$

	1	2	3	4
1				
2				
3				
4				

Modélisation

Attention

Il n'y a pas qu'une modélisation possible !

Modélisation

- **variables** L_1, L_2, L_3, L_4 et C_1, C_2, C_3, C_4
- **domaines** $L_i \in \{1, 2, 3, 4\}$ et $C_i \in \{1, 2, 3, 4\}$
- **contraintes**
 - $\forall i \neq j, L_i \neq L_j$
 $C_i \neq C_j$
 - $\forall i \neq j, L_i + C_i \neq L_j + C_j$
 $L_i \neq L_j \vee C_i \neq C_j$

	1	2	3	4
1	Black	White	Black	White
2	White	Black	White	Black
3	Black	White	Black	White
4	White	Black	White	Black

Modélisation

Attention

Il n'y a pas qu'une modélisation possible !

Modélisation

- **variables** L_1, L_2, L_3, L_4
- **domaines** $L_i \in \{1, 2, 3, 4\}$
- **contraintes**
 - $\forall i \neq j, L_i \neq L_j$
 - $\forall i \neq j, L_i + i \neq L_j + j$

	1	2	3	4
1				
2				
3				
4				


Exercice : le retour de monnaie

On s'intéresse à un distributeur automatique de boissons. L'utilisateur insère des pièces de monnaie pour un total de T centimes d'Euros, puis il sélectionne une boisson, dont le prix est de P centimes d'Euros (T et P étant des multiples de 10). Il s'agit alors de calculer la monnaie à rendre, sachant que le distributeur a en réserve E_2 pièces de 2€, E_1 pièces de 1€, C_{50} pièces de 50 centimes, C_{20} pièces de 20 centimes et C_{10} pièces de 10 centimes.

- Modélisez ce problème sous la forme d'un CSP.

Correction : le retour de monnaie

<handout :0>



Correction : le retour de monnaie

<handout :0>

- **Variables** : X_2 , X_1 , X_{50} , X_{20} , X_{10} , nombres de pièces à rendre

Correction : le retour de monnaie

<handout :0>

- **Variables** : $X_2, X_1, X_{50}, X_{20}, X_{10}$, nombres de pièces à rendre
- **Domaines** :
 - $D_{X_2} = \{0, 1, \dots, E_2\}$

Correction : le retour de monnaie

<handout :0>

- **Variables** : $X_2, X_1, X_{50}, X_{20}, X_{10}$, nombres de pièces à rendre
- **Domaines** :
 - $D_{X_2} = \{0, 1, \dots, E_2\}$
 - $D_{X_1} = \{0, 1, \dots, E_1\}$
 - $D_{X_{50}} = \{0, 1, \dots, C_{50}\}$
 - $D_{X_{20}} = \{0, 1, \dots, C_{20}\}$
 - $D_{X_{10}} = \{0, 1, \dots, C_{10}\}$

Correction : le retour de monnaie

<handout :0>

- **Variables** : $X_2, X_1, X_{50}, X_{20}, X_{10}$, nombres de pièces à rendre

- **Domaines** :

- $D_{X_2} = \{0, 1, \dots, E_2\}$
- $D_{X_1} = \{0, 1, \dots, E_1\}$
- $D_{X_{50}} = \{0, 1, \dots, C_{50}\}$
- $D_{X_{20}} = \{0, 1, \dots, C_{20}\}$
- $D_{X_{10}} = \{0, 1, \dots, C_{10}\}$

- **Contrainte** :

$$200X_2 + 100X_1 + 50X_{50} + 20X_{20} + 10X_{10} = T \quad P$$

Exercice : Send More Money

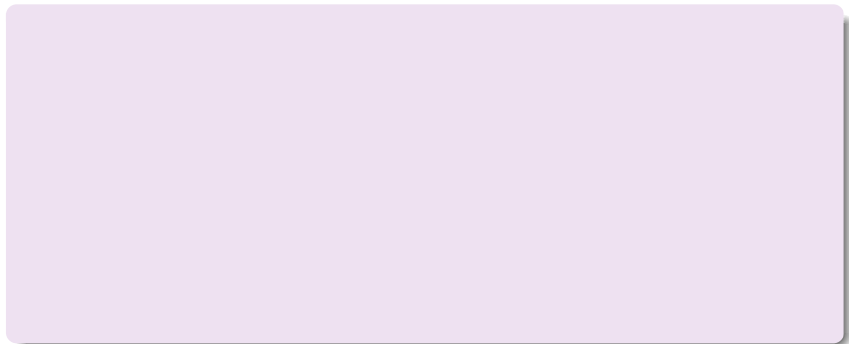
On considère l'addition suivante :

$$\begin{array}{rcccccc}
 & & S & E & N & D & \\
 + & & M & O & R & E & \\
 \hline
 = & M & O & N & E & Y &
 \end{array}$$

où chaque lettre représente un chiffre différent (compris entre 0 et 9). On souhaite connaître la valeur de chaque lettre, sachant que la première lettre de chaque mot représente un chiffre différent de 0.

- Modélisez ce problème sous la forme d'un CSP.

Correction : Send More Money



Correction : Send More Money

- **Variables** : S, E, N, D, M, O, R, Y

Correction : Send More Money

- **Variables** : S, E, N, D, M, O, R, Y
- **Domaines** :
 - $D_S = D_M = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
 - $D_E = D_N = D_D = D_O = D_R = D_Y = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Correction : Send More Money

- **Variables** : S, E, N, D, M, O, R, Y
- **Domaines** :
 - $D_S = D_M = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
 - $D_E = D_N = D_D = D_O = D_R = D_Y = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- **Contraintes** :
 - $1000(S + M) + 100(E + O) + 10(N + R) + (D + E) = 10000M + 1000O + 100N + 10E + Y$
 - $tousDifférents(S, E, N, D, M, O, R, Y)$

Programmation par contraintes : plan

1 Programmation par contrainte

2 Modélisation

3 Résolution

Résolution naïve

Enumération

- On génère toutes les affectations totales possibles
- On vérifie si elles sont consistantes
- si on en trouve une consistante, c'est gagné

Résolution naïve

Enumération

- On génère toutes les affectations totales possibles
 - On vérifie si elles sont consistantes
 - si on en trouve une consistante, c'est gagné
-
- Avantage : très facile à mettre en œuvre
 - Inconvénient : très gourmand en ressource temps

Résolution naïve

Un problème à n variables qui peuvent prendre 2 valeurs.
 10^9 affectations traitées par seconde.

n	nb d'affectations	temps
10	10^3	
20	10^6	
30	10^9	
40	10^{12}	
50	10^{15}	
60	10^{18}	
70	10^{21}	

... d'où l'intérêt de bien choisir la modélisation

Résolution naïve

Un problème à n variables qui peuvent prendre 2 valeurs.
 10^9 affectations traitées par seconde.

n	nb d'affectations	temps
10	10^3	10^{-6} secondes ✓
20	10^6	10^{-3} secondes ✓
30	10^9	1 seconde ✓
40	10^{12}	16 minutes
50	10^{15}	11 jours ✗
60	10^{18}	32 ans ✗✗
70	10^{21}	317 siècles ✗✗

... d'où l'intérêt de bien choisir la modélisation

Les remèdes

- S'arrêter quand une affectation partielle est inconsistante
 - **backtrack**
- Restreindre les domaines des variables durant l'exécution
 - **propagation de contraintes**
- Utiliser des heuristiques
- Utiliser nos connaissances sur le problème étudié
- ...

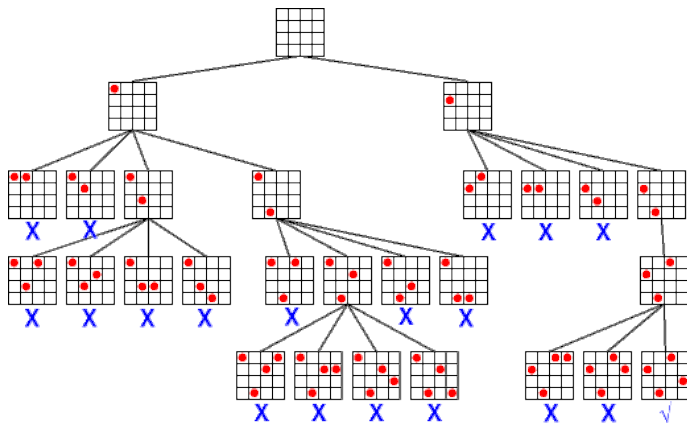
Backtrack

Principe :

- On parcourt l'arbre des affectations en profondeur
- Lorsqu'une affectation partielle est inconsistante, on n'explore pas le sous-arbre correspondant

Backtrack : jeu de dames

Exemple d'exécution



Backtrack

Backtrack

- ✓ Moins d'affectations considérées
 - ✗ Toutes les contraintes sont testées à chaque affectation même partielle
 - ✓ Facile à mettre en œuvre
- Importance de l'ordre des variables

Méthode naïve

- ✗ Toutes les affectations sont considérées
- ✓ On ne teste les contraintes que sur les affectations totales
- ✓ Très facile à mettre en œuvre

Propagation des contraintes

Principe :

- On restreint le domaine d'une variable
- avec les contraintes, on restreint les autres domaines

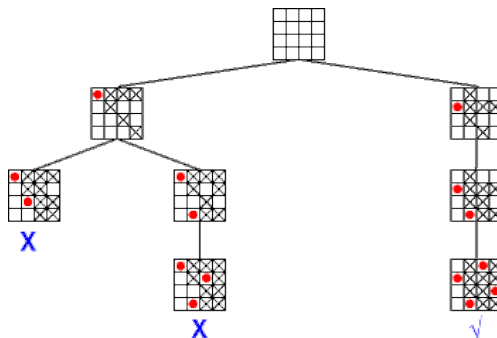
Exemple

- $0 < x \leq 10$ et $3 \leq y \leq 9$
- contrainte : $x > y$

Alors on peut restreindre le domaine de x à $4 \leq x \leq 10$

Propagation des contraintes

Exemple d'exécution



Propagation des contraintes

Choix

On propage :

- Quand un domaine est réduit
- Quand une des bornes du domaine est changée
- Quand un domaine est un singleton

On propage :

- Une fois : **nœud-consistance**
- Deux fois : **arc-consistance**
- Ou plus...

Propagation des contraintes

Nœud-consistance

Formellement, un CSP (X, D, C) est consistant de nœud si pour toute variable X_i de X , et pour toute valeur v de $D(X_i)$, l'affectation partielle (X_i, v) satisfait toutes les contraintes unaires de C .

Propagation des contraintes

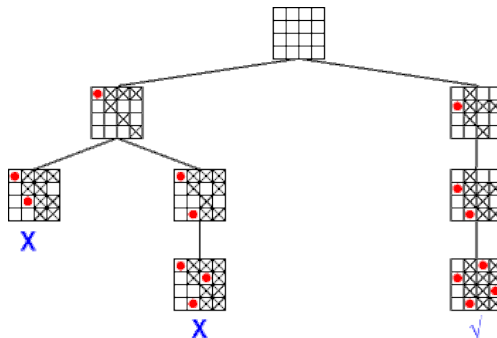
Nœud-consistance

Formellement, un CSP (X, D, C) est consistant de nœud si pour toute variable X_i de X , et pour toute valeur v de $D(X_i)$, l'affectation partielle (X_i, v) satisfait toutes les contraintes unaires de C .

Algorithmiquement, pour chaque variable X_i non affectée dans A , on enlève de $D(X_i)$ toute valeur v telle que l'affectation $A \cup \{(X_i, v)\}$ est inconsistante.

Propagation des contraintes

Nœud-consistance



Propagation des contraintes

Arc-consistance

Formellement, un CSP (X, D, C) est consistant d'arc si pour tout couple de variables (X_i, X_j) de X , et pour toute valeur v_i appartenant à $D(X_i)$, il existe une valeur v_j appartenant à $D(X_j)$ telle que l'affectation partielle $\{(X_i, v_i), (X_j, v_j)\}$ satisfasse toutes les contraintes binaires de C .

Propagation des contraintes

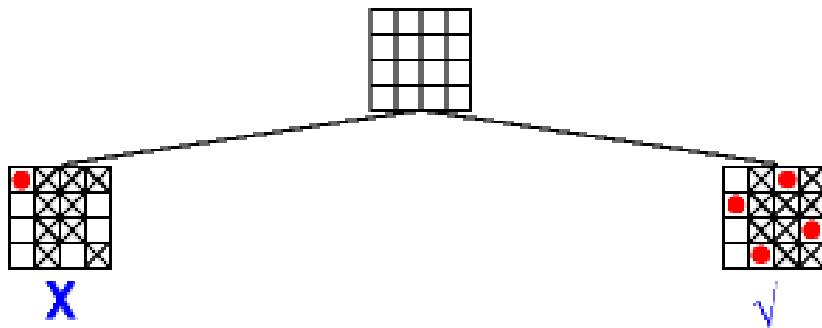
Arc-consistance

Formellement, un CSP (X, D, C) est consistant d'arc si pour tout couple de variables (X_i, X_j) de X , et pour toute valeur v_i appartenant à $D(X_i)$, il existe une valeur v_j appartenant à $D(X_j)$ telle que l'affectation partielle $\{(X_i, v_i), (X_j, v_j)\}$ satisfasse toutes les contraintes binaires de C .

Algorithmiquement, pour chaque variable X_i non affectée dans A , on enlève de $D(X_i)$ toute valeur v telle qu'il existe une variable X_j non affectée pour laquelle, pour toute valeur w de $D(X_j)$, l'affectation $A \cup \{(X_i, v), (X_j, w)\}$ soit inconsistante.

Propagation des contraintes

Arc-consistance : exemple d'exécution



Propagation de contraintes

Nœud-consistance :

- ✓ Moins d'affectations considérées
- ✗ Toutes les contraintes sont testées à chaque affectation même partielle
- ✓ Facile à mettre en œuvre

Arc-consistance :

- ✓ Beaucoup moins d'affectations considérées
- ✗ Etablir l'arc-consistance peut prendre beaucoup de temps selon les contraintes.
- ✗ Moins facile à mettre en œuvre

- La nœud-consistance est quasiment toujours plus efficace que le Backtrack
- Du fait de sa complexité l'arc-consistance se révèle moins efficace que la nœud-consistance

Résolution : bilan

Remarques :

- Backtrack = propagation de contrainte au niveau 0
- De la supériorité du cerveau : quel que soit l'algorithme choisi, la modélisation et l'ordre des variable ont leur importance
- Heuristiques : évaluation de la branche de l'arbre à inspecter en priorité
- Autres méthodes : backtrack à plusieurs étages
- Selon la nature du problème : simplexe, branch & bound... plus efficaces car non-génériques

Exemple : Affectation de stock

- N entrepôts (coût d'ouverture)
- M boutiques
- coûts d'acheminement d'un entrepôt à une boutique
- capacité : chaque entrepôt ne peut fournir qu'un certain nombre de boutiques

Question : avec un budget de mille euros, puis-je subvenir aux besoins de mes boutiques ?

Correction : Affectation de stock

- coût d'ouverture : $\text{ouvre}(i)$ pour $1 \leq i \leq N$
- coût d'acheminement : $\text{transfert}(i, j)$ pour $1 \leq i \leq N$, $1 \leq j \leq M$
- capacités : $\text{cap}(i)$ pour $1 \leq i \leq N$

Correction : Affectation de stock

- coût d'ouverture : $\text{ouvre}(i)$ pour $1 \leq i \leq N$
- coût d'acheminement : $\text{transfert}(i, j)$ pour $1 \leq i \leq N$,
 $1 \leq j \leq M$
- capacités : $\text{cap}(i)$ pour $1 \leq i \leq N$
- **Variables** : $\text{ouvert}(i)$ pour $1 \leq i \leq N$
 $\text{fournisseur}(j)$ pour $1 \leq j \leq M$
- **Domaines** : $\text{ouvert}(i) \in \{0, 1\}$
 $\text{fournisseur}(j) \in \{1, 2, \dots, N\}$

Correction : Affectation de stock

- **Variables** : ouvert(i) pour $1 \leq i \leq N$
fournisseur(j) pour $1 \leq j \leq M$
- **Domaines** : ouvert(i) $\in \{0, 1\}$
fournisseur(j) $\in \{1, 2, \dots, N\}$
- **Contraintes** :

Correction : Affectation de stock

- **Variables** : ouvert(i) pour $1 \leq i \leq N$
fournisseur(j) pour $1 \leq j \leq M$
- **Domaines** : ouvert(i) $\in \{0, 1\}$
fournisseur(j) $\in \{1, 2, \dots, N\}$
- **Contraintes** :
 - pour tout $1 \leq j \leq M$, ouvert(fournisseur(j)) = 1

Correction : Affectation de stock

- **Variables** : ouvert(i) pour $1 \leq i \leq N$
fournisseur(j) pour $1 \leq j \leq M$
- **Domaines** : ouvert(i) $\in \{0, 1\}$
fournisseur(j) $\in \{1, 2, \dots, N\}$
- **Contraintes** :
 - pour tout $1 \leq j \leq M$, ouvert(fournisseur(j)) = 1
 - pour tout $1 \leq i \leq N$, $\sum_{j | \text{fournisseur}(j)=i} 1 \leq \text{cap}(i)$

Correction : Affectation de stock

- **Variables** : ouvert(i) pour $1 \leq i \leq N$
fournisseur(j) pour $1 \leq j \leq M$
- **Domaines** : ouvert(i) $\in \{0, 1\}$
fournisseur(j) $\in \{1, 2, \dots, N\}$
- **Contraintes** :
 - pour tout $1 \leq j \leq M$, ouvert(fournisseur(j)) = 1
 - pour tout $1 \leq i \leq N$, $\sum_{j | \text{fournisseur}(j)=i} 1 \leq \text{cap}(i)$
 - $1000 \leq \sum_i \text{ouvre}(i) \text{ouvert}(i) + \sum_j \text{transfert}(\text{fournisseur}(j), j)$