

Programmation par contraintes

Arnaud Malapert

29 octobre 2015

Département d'informatique
Université Nice Sophia Antipolis



Plan du cours

Lecture 1 : Généralités

Lecture 2 : L'art de la modélisation

Lecture 3 : Méthodes de résolution

Lecture 1

Généralités

Programmation par contraintes

29 octobre 2015

Arnaud Malapert
Département d'informatique
Université Nice Sophia Antipolis

Plan

- 1 Introduction
- 2 Problème de satisfaction de contraintes
- 3 Solveurs de contraintes

Plan

1 Introduction

2 Problème de satisfaction de contraintes

3 Solveurs de contraintes

Computers are incredibly fast, accurate and stupid. Human beings are incredibly slow, inaccurate and brilliant. Together they are powerful beyond imagination.

Albert Einstein.

Constraint Programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming : the user states the problem, the computer solves it.

Eugene C. Freuder, Inaugural issue of the *Constraints Journal*, 1997.

Programmation par contraintes (PPC)

Modélisation et de résolution de problèmes combinatoires.

Problème combinatoire

Ensemble discret et dénombrable de solutions.

Satisfiabilité trouver une solution

Énumération trouver plusieurs/toutes les solutions

Optimisation trouver la meilleure solution

Programmation par contraintes (PPC)

Modélisation et de résolution de problèmes combinatoires.

Problème combinatoire

Ensemble discret et dénombrable de solutions.

Satisfiabilité trouver une solution

Énumération trouver plusieurs/toutes les solutions

Optimisation trouver la meilleure solution

Est-ce que ça vaut le coup d'oeil ?

- ▶ Langage de modélisation riche.
- ▶ Techniques de résolution puissantes et génériques.

Programmation par contraintes (PPC)

Modélisation et de résolution de problèmes combinatoires.

Problème combinatoire

Ensemble discret et dénombrable de solutions.

Satisfiabilité trouver une solution

Énumération trouver plusieurs/toutes les solutions

Optimisation trouver la meilleure solution

Est-ce que ça vaut le coup d'oeil ?

- ▶ Langage de modélisation riche.
- ▶ Techniques de résolution puissantes et génériques.

Au point de suivre un cours ?

- ▶ Concevoir de meilleurs (solvables) modèles.
- ▶ Meilleure exploitation de la technologie.

Du rêve à la réalité

- > Solveur, quelle est la solution de SEND+MORE=MONEY ?
- > Facile!
- > $[9,5,6,7] + [1,0,8,5] = [1,0,6,5,2]$

Du rêve à la réalité

- > Solveur, quelle est la solution de SEND+MORE=MONEY ?
- > Facile!
- > $[9,5,6,7] + [1,0,8,5] = [1,0,6,5,2]$

```
Model model = new CPModel();
IntegerVariable S = makeIntVar("S",0,9);
...
IntegerVariable Y = makeIntVar("Y",0,9);
```

Du rêve à la réalité

- > Solveur, quelle est la solution de SEND+MORE=MONEY ?
- > Facile!
- > $[9,5,6,7] + [1,0,8,5] = [1,0,6,5,2]$

```
Model model = new CPModel();
IntegerVariable S = makeIntVar("S",0,9);
...
IntegerVariable Y = makeIntVar("Y",0,9);
model.addConstraint(neq(S,0),neq(M,0) );
model.addConstraint(allDifferent(S,E,N,D,M,O,R,Y));
model.addConstraint(
eq( plus( scalar(new int[]{1000,100,10,1}, new IntegerVariable[]{S,E,N,D}),
scalar(new int[]{1000,100,10,1}, new IntegerVariable[]{M,O,R,E})),
scalar(new int[]{10000,1000,100,10,1}, new IntegerVariable[]{M,O,N,E,Y})));
```

Du rêve à la réalité

- > Solveur, quelle est la solution de SEND+MORE=MONEY ?
- > Facile!
- > $[9,5,6,7] + [1,0,8,5] = [1,0,6,5,2]$

```
Model model = new CPModel();
IntegerVariable S = makeIntVar("S",0,9);
...
IntegerVariable Y = makeIntVar("Y",0,9);
model.addConstraint(neq(S,0),neq(M,0) );
model.addConstraint(allDifferent(S,E,N,D,M,O,R,Y));
model.addConstraint(
eq( plus( scalar(new int[]{1000,100,10,1}, new IntegerVariable[]{S,E,N,D}),
scalar(new int[]{1000,100,10,1}, new IntegerVariable[]{M,O,R,E})),
scalar(new int[]{10000,1000,100,10,1}, new IntegerVariable[]{M,O,N,E,Y})));
```

Du rêve à la réalité

- > Solveur, quelle est la solution de SEND+MORE=MONEY ?
- > Facile!
- > $[9,5,6,7] + [1,0,8,5] = [1,0,6,5,2]$

```
Model model = new CPModel();
IntegerVariable S = makeIntVar("S",0,9);
...
IntegerVariable Y = makeIntVar("Y",0,9);
model.addConstraint(neq(S,0),neq(M,0) );
model.addConstraint(allDifferent(S,E,N,D,M,O,R,Y));
model.addConstraint(
eq( plus( scalar(new int[]{1000,100,10,1}), new IntegerVariable[]{S,E,N,D}),
scalar(new int[]{1000,100,10,1}, new IntegerVariable[]{M,O,R,E})),
scalar(new int[]{10000,1000,100,10,1}, new IntegerVariable[]{M,O,N,E,Y})));
Solver solver=new CPSolver();
solver.read(model);
```

Propagation initiale : S=9, E in 4..7, N in 5..8, M=1, O=0, [D,R,Y] in 2..8

Du rêve à la réalité

- > Solveur, quelle est la solution de SEND+MORE=MONEY ?
- > Facile!
- > $[9,5,6,7] + [1,0,8,5] = [1,0,6,5,2]$

```
Model model = new CPModel();
IntegerVariable S = makeIntVar("S",0,9);
...
IntegerVariable Y = makeIntVar("Y",0,9);
model.addConstraint(neq(S,0),neq(M,0) );
model.addConstraint(allDifferent(S,E,N,D,M,O,R,Y));
model.addConstraint(
eq( plus( scalar(new int[]{1000,100,10,1}, new IntegerVariable[]{S,E,N,D}),
scalar(new int[]{1000,100,10,1}, new IntegerVariable[]{M,O,R,E})),
scalar(new int[]{10000,1000,100,10,1}, new IntegerVariable[]{M,O,N,E,Y})));
Solver solver=new CPSolver();
solver.read(model);
solver.solve();
```

Propagation initiale : S=9, E in 4..7, N in 5..8, M=1, O=0, [D,R,Y] in 2..8
Recherche : S=9, E=5, N=6, D=7, M=1, O=0, R=8, Y=2

Programmation par contraintes



Modèle déclaratif.

- ▶ Description des propriétés et relations entre objets partiellement connus.
- ▶ Gestion d'information exacte, approchée, finie, infinie, complète et incomplètes.



Raisonnement automatique basé sur les contraintes.

- ▶ Propagation des nouvelles informations.
- ▶ Simplification extrait les informations implicites.



Résolution efficace de problèmes combinatoires.

- ▶ Combinaison aisée avec des techniques de recherche et d'optimisation.



Technologie récente et coûteuse.



Résolution de très grandes instances.



Optimisation sous contraintes.

Il était une fois ...

1960 Réseau de contraintes en IA

1970 Programmation logique (prolog)

1980 Programmation logique concurrente

1980 Programmation par contraintes

1990 Programmation par contraintes concurrente

1990 Applications commerciales (ILOG)

Aujourd'hui <http://openjvm.jvmhost.net/CPSolvers/>

Influences

- ▶ **Mathématiques**
 - ▶ Recherche opérationnelle
 - ▶ Théorie des graphes
 - ▶ Algèbre
- ▶ **Programmation Logique**
- ▶ **Informatique**
 - ▶ Automate
 - ▶ Preuve automatique
- ▶ **Économie**
- ▶ **Linguistique**

Applications

► Académiques

- ▶ sac-à-dos,
- ▶ voyageur de commerce,
- ▶ coloriage de graphe,
- ▶ Robotique,
- ▶ Agents,
- ▶ ...

► Industrielles

- ▶ ordonnancement,
- ▶ allocations de ressources
- ▶ emploi du temps
- ▶ design de circuit
- ▶ séquençage de l'ADN,
- ▶ ...

PLNE versus PPC

PLNE

- ▶ Méthodologie éprouvée pour la résolution de grandes instances
- ▶ Plusieurs méthodes de résolution pour chaque modèle
- ▶ Méthodes tournées vers l'optimisation, approche globale

PPC

- ▶ Méthode de modélisation adaptée aux problèmes composites
- ▶ Modèles déclaratifs et méthodes de résolutions flexibles
- ▶ Méthodes tournées vers la satisfaction, approche locale

PLNE versus PPC

PLNE

- ▶ Méthodologie éprouvée pour la résolution de grandes instances
- ▶ Plusieurs méthodes de résolution pour chaque modèle
- ▶ Méthodes tournées vers l'optimisation, approche globale

PPC

- ▶ Méthode de modélisation adaptée aux problèmes composites
- ▶ Modèles déclaratifs et méthodes de résolutions flexibles
- ▶ Méthodes tournées vers la satisfaction, approche locale

branch-and-bound=backtracking

- ▶ Séparation itérative (*branching*)
- ▶ Évaluation (*bounding/filtering*)
- ▶ Inférence (*cutting plane/propagation*)

PLNE versus PPC

PLNE

- opérations algébriques faites sur le système entier
- très bonnes bornes pour accélérer la preuve de l'optimalité.
- heuristiques de bonne qualité disponibles.
- modélisation bas niveau (contraintes linéaires)

PPC

- solutions réalisables
- structure du problème préservée
- information sur les domaines
- formulation efficace
- toutes les solutions
- hybridation
- propagation trop locale
- optimisation

Plan

1 Introduction

2 Problème de satisfaction de contraintes

3 Solveurs de contraintes

Définitions

Problème de satisfaction de contraintes

- ▶ un ensemble de variables $\{X_1, \dots, X_n\}$
- ▶ un ensemble de domaines discrets $\{D_1, \dots, D_n\}$
- ▶ Un ensemble de contraintes $\{C_1, \dots, C_m\}$

Affectation

Une affectation consiste à instancier des variables.

- ▶ partielle ou totale
- ▶ consistante ou inconsistante

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

Solution

Une solution est une affectation totale consistante.

Only two Ingredients !!!

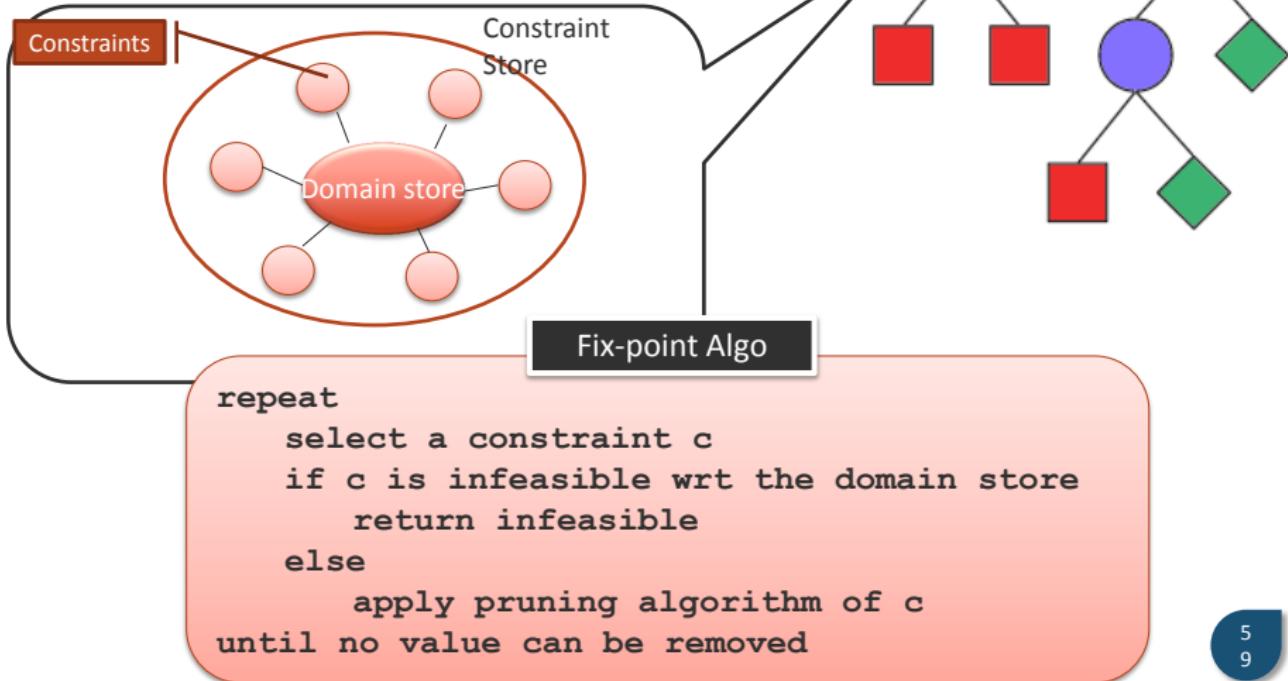


CP = Pruning + Search

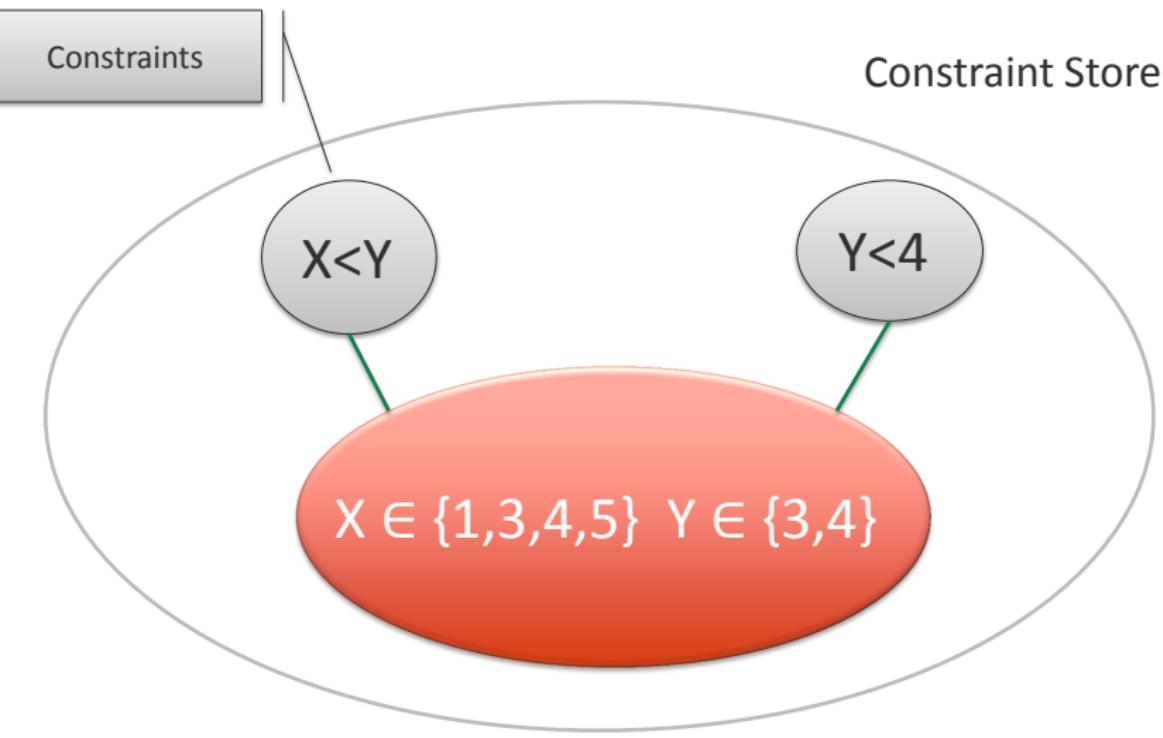
- Pruning = removing values from non feasible solutions
- Search = Systematic exploration of solution space

CP in One Slide

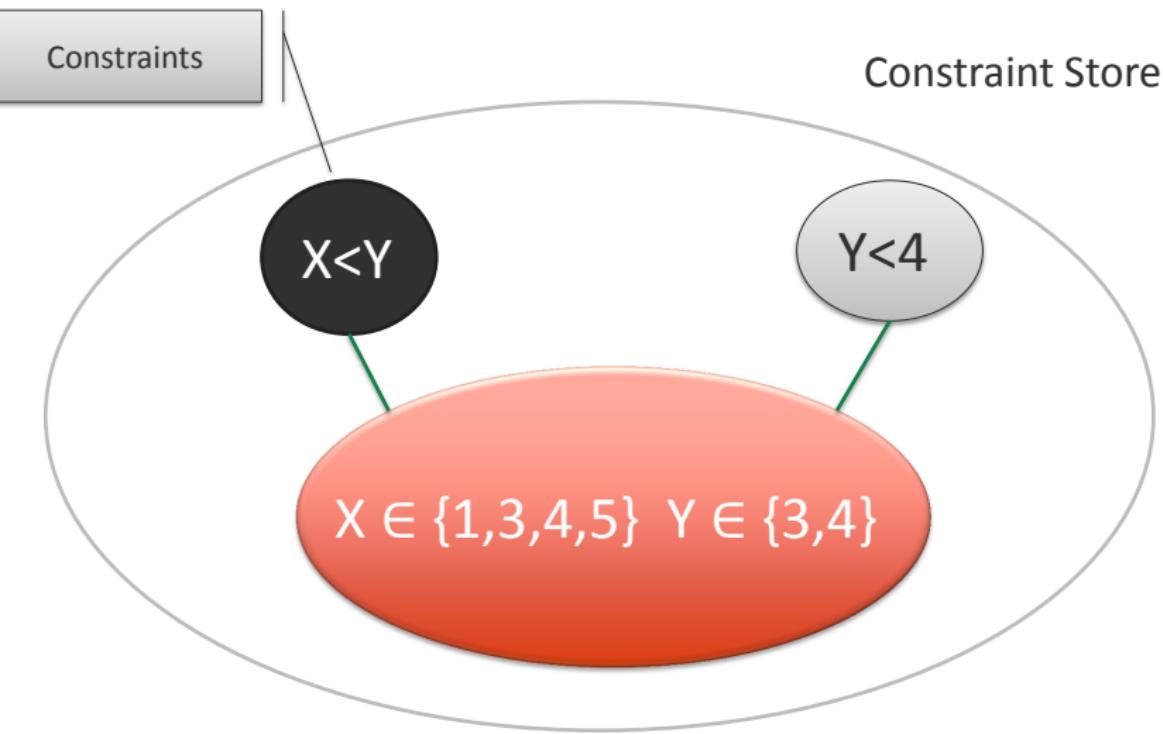
- CP = Pruning + Search



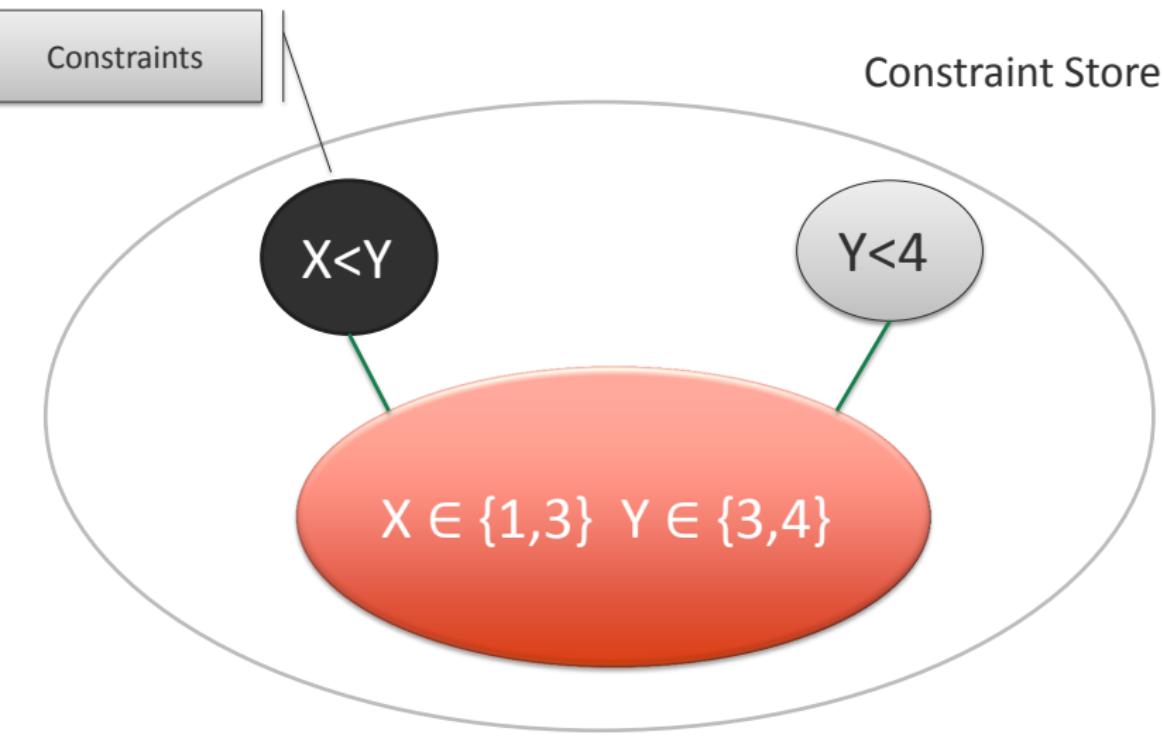
Fix Point Algorithm



Fix Point Algorithm



Fix Point Algorithm



Fix Point Algorithm

Constraints

Constraint Store

$X < Y$

$Y < 4$

$X \in \{1, 3\}$ $Y \in \{3, 4\}$

Fix Point Algorithm

Constraints

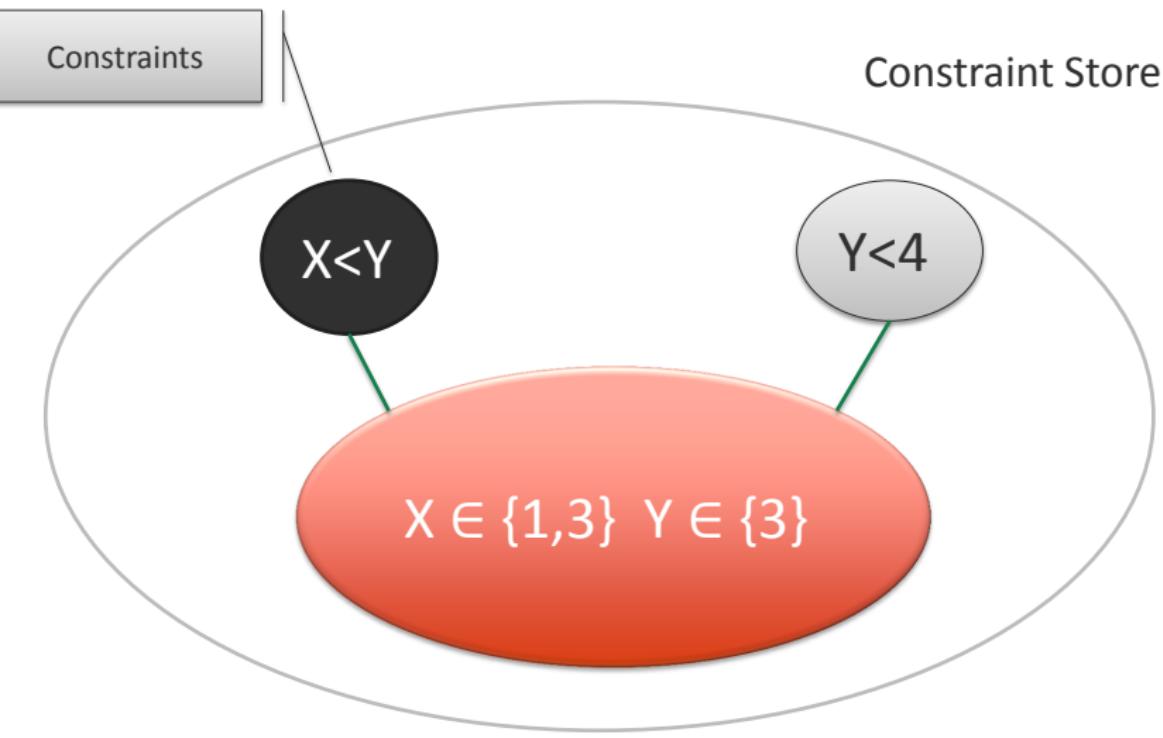
Constraint Store

$X < Y$

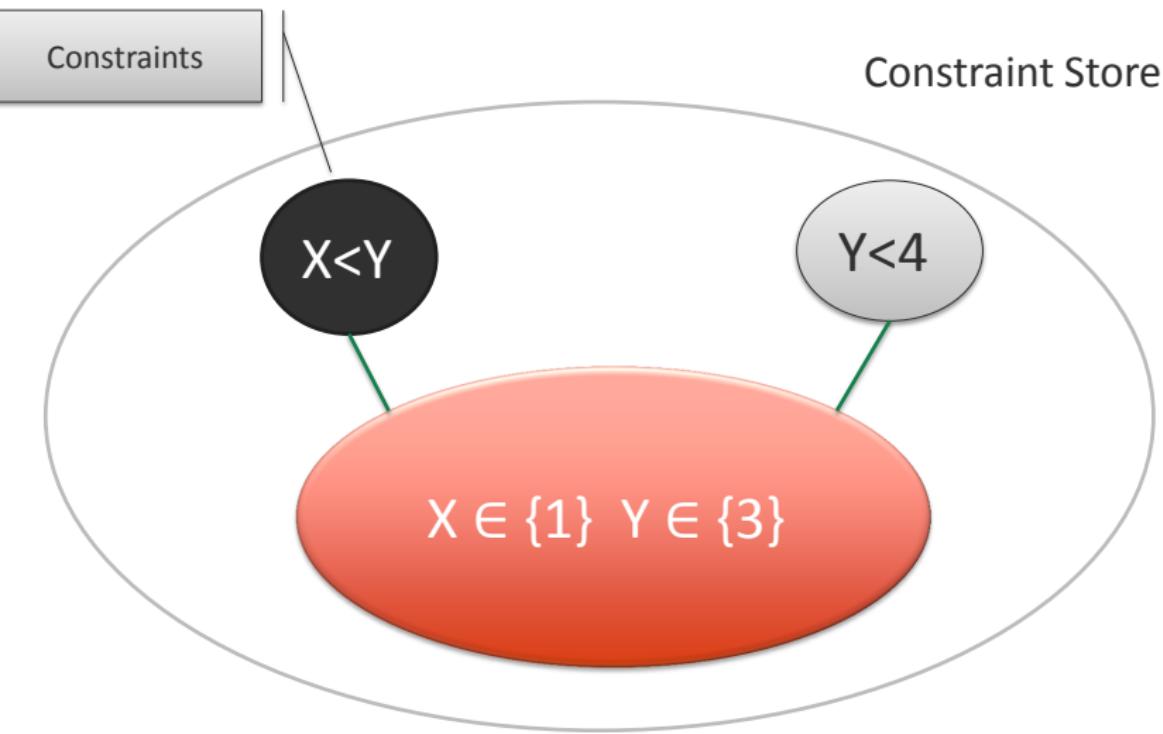
$Y < 4$

$X \in \{1,3\} \quad Y \in \{3\}$

Fix Point Algorithm



Fix Point Algorithm



Search Tree

Variables:

- $\text{Dom}(x_1)=[0..3]$
- $\text{Dom}(x_2)=\{2,4\}$
- $\text{Dom}(x_3)=\{1,4\}$

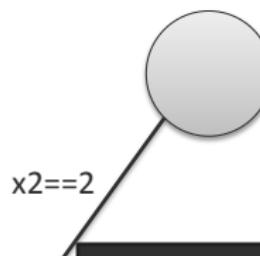
Fix Point

$\text{Dom}(x_1)=[0..2]$
 $\text{Dom}(x_2)=\{2,4\}$
 $\text{Dom}(x_3)=\{4\}$

Constraints:

$x_1+x_2=x_3$
 $x_1 \neq x_2$

$x_1 + x_2 = x_3$
 $x_1 \neq x_2$

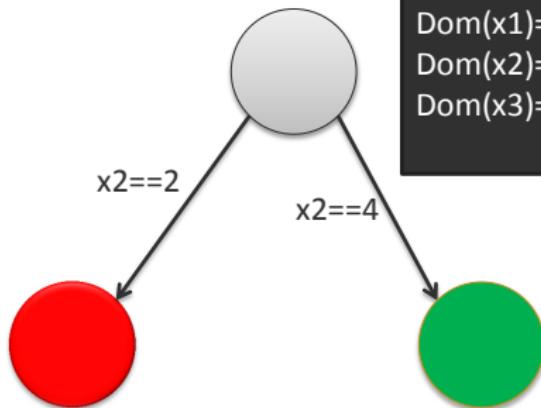


$\text{Dom}(x_1)=[0..2]$
 $\text{Dom}(x_2)=\{2,4\}$
 $\text{Dom}(x_3)=\{4\}$

$\text{Dom}(x_1)=[0..1]$
 $\text{Dom}(x_2)=\{2\}$
 $\text{Dom}(x_3)=\{\}$

$x_1 + x_2 = x_3$
 $x_1 \neq x_2$

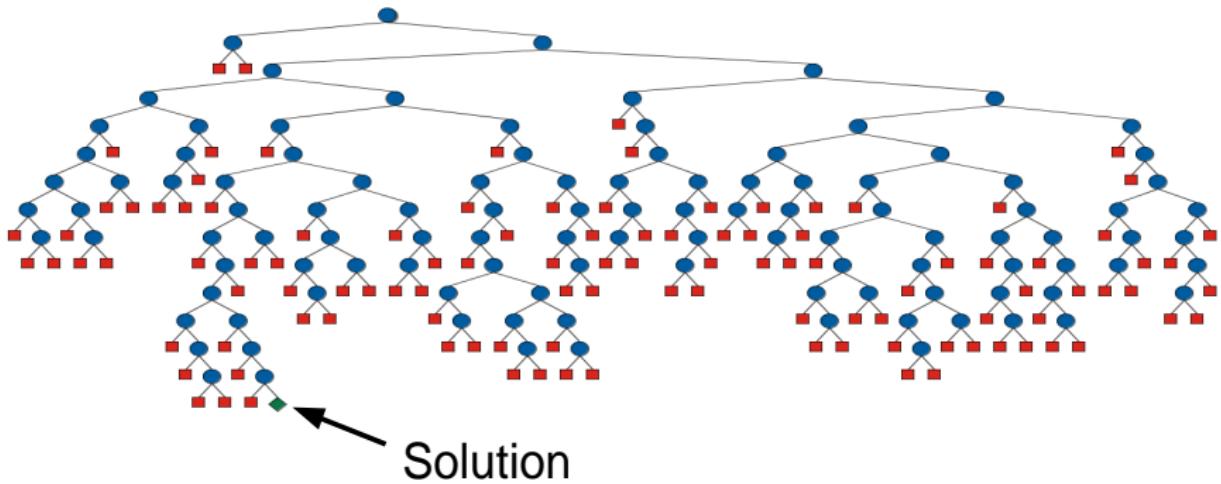
df



$\text{Dom}(x_1)=[0..2]$
 $\text{Dom}(x_2)=\{2,4\}$
 $\text{Dom}(x_3)=\{4\}$

$\text{Dom}(x_1)=[0]$
 $\text{Dom}(x_2)=\{4\}$
 $\text{Dom}(x_3)=\{4\}$

Search Tree



Domaine entier

Exemple

En pratique, la cardinalité du domaine est finie.

$$D = \{1, 2, 3, 5, 7, 9, 10\}$$

Domaine entier

Exemple

En pratique, la cardinalité du domaine est finie.

$$D = \{1, 2, 3, 5, 7, 9, 10\}$$

Opérations

- ▶ test d'appartenance de valeur(s)
- ▶ suppression de valeur(s)
- ▶ itération(s) sur des valeurs

Domaine entier

Exemple

En pratique, la cardinalité du domaine est finie.

$$D = \{1, 2, 3, 5, 7, 9, 10\}$$

Opérations

- ▶ test d'appartenance de valeur(s)
- ▶ suppression de valeur(s)
- ▶ itération(s) sur des valeurs

Représentation

Énumération occupation mémoire

Intervalle domaine non connexe

Liste, arbre complexité des structures de données.

Domaine ensembliste

Exemple

Par bijection, on peut se restreindre à l'étude des ensembles d'entiers.

$$D = \{\emptyset, \{1, 2\}, \{1, 2, 3, 4\}, \{2, 3\}, \{2, 4\}\}$$

Opérations

Elles s'appliquent à des **éléments** ou à des **ensembles**.

Domaine ensembliste

Exemple

Par bijection, on peut se restreindre à l'étude des ensembles d'entiers.

$$D = \{\emptyset, \{1, 2\}, \{1, 2, 3, 4\}, \{2, 3\}, \{2, 4\}\}$$

Opérations

Elles s'appliquent à des **éléments** ou à des **ensembles**.

Représentations

Énumération occupation mémoire exponentielle

- ▶ 2^n sous-ensembles de $[1, n]$

Noyau-Enveloppe perte d'information

- ▶ $K(D) = \cap_{S \in D} S = \emptyset$
- ▶ $E(D) = \cup_{S \in D} S = [1, 4]$

Représentation des domaines

Gestion des états/mondes

Pendant la résolution, on maintient d'une pile d'états.

- ▶ variables, domaines, contraintes.

Un compromis temps/mémoire

- la représentation des domaines influe considérablement sur les performances de la pile.
- l'action des contraintes peut différer selon la représentation des domaines.
- configuration automatique des solveurs.

Contraintes

C'est quoi ?

Une contrainte porte sur un ensemble de variables et restreint les valeurs que l'on peut affecter simultanément à ces variables

- ▶ $X + Y \leq Z$
- ▶ $A \cup B \neq C$
- ▶ $x^2 + y^2 = z^2$
- ▶ (ABC) forme un triangle isocèle
- ▶ x, y, z , distincts deux à deux.
- ▶ A, B, C forment une partition de E

Contraintes

Déclaration

Extension on énumère les n-uplets admis :

- ▶ $(x = 1 \wedge y = 2) \vee (x = 0 \wedge y = 3) \vee (x = 2)$

Intension formule mathématique :

- ▶ $x + y \leq z$

Arité d'une contrainte

unaire, binaire, n-aire

Exemples

non linéaire $x^2 + y^3 = z^2$

symbolique $(x = bleu) \vee (x = vert)$

logique $x = 3 \Rightarrow y < z$

globale `allDifferent(x1, ..., xn)`, ...

Quelques contraintes globales

table

(X_1, \dots, X_n) prennent leur valeurs parmi un ensemble de tuples.

globalCardinality

Les valeurs v_1, \dots, v_m apparaissent O_{v_1}, \dots, O_{v_m} dans X_1, \dots, X_n .

allDifferent

les valeurs de X_1, \dots, X_n sont distinctes.

element (1D/2D/3D)

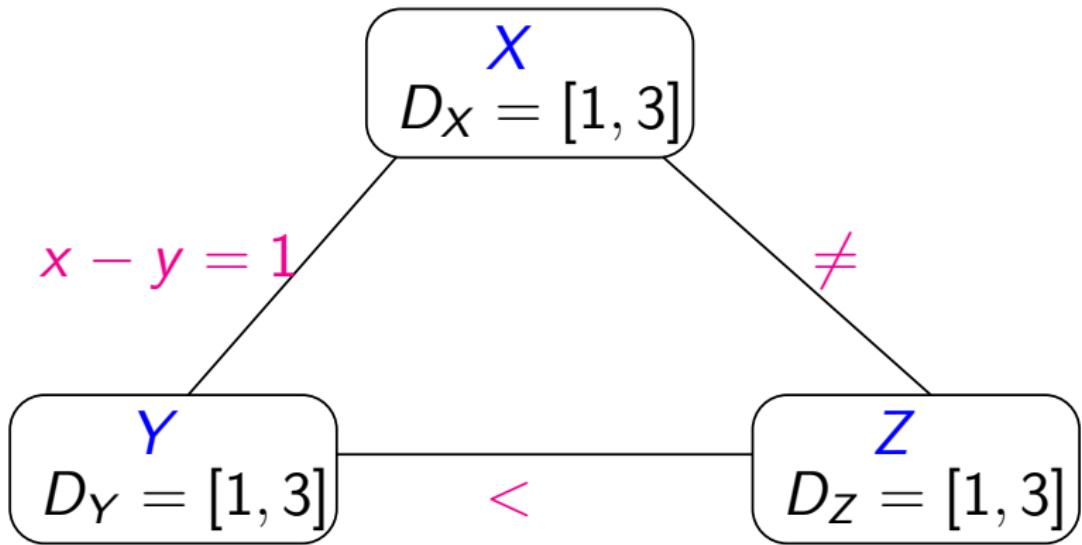
La valeur V est égale au I -ème élément de (X_1, \dots, X_n) .

lex

(X_1, \dots, X_n) est plus petit que (Y_1, \dots, Y_n) selon l'ordre lexico.

Pour les curieux : Global Constraint catalog

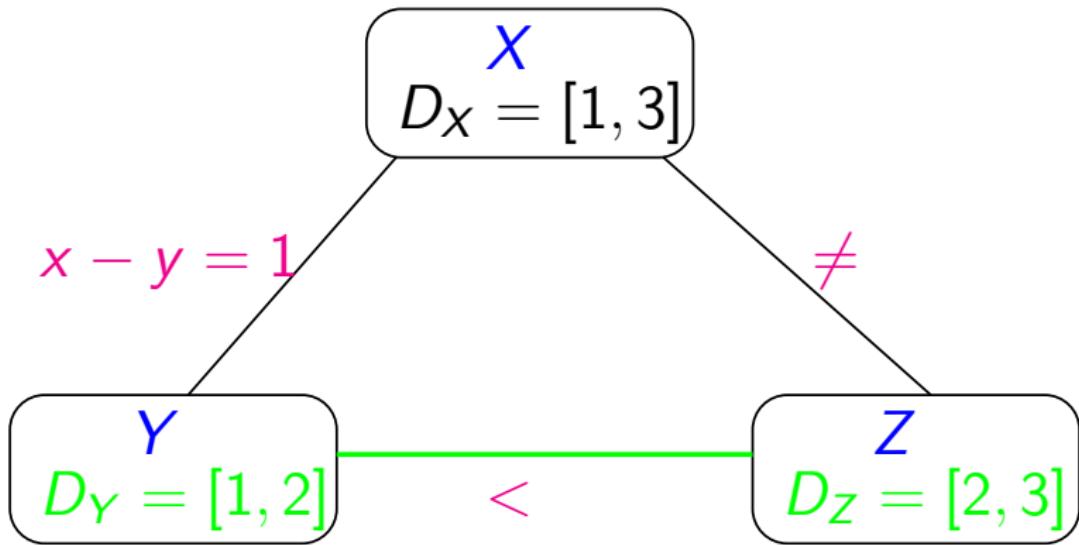
Résolution



Réseau de contraintes

- ▶ multi-graphe
- ▶ hyper-graphe

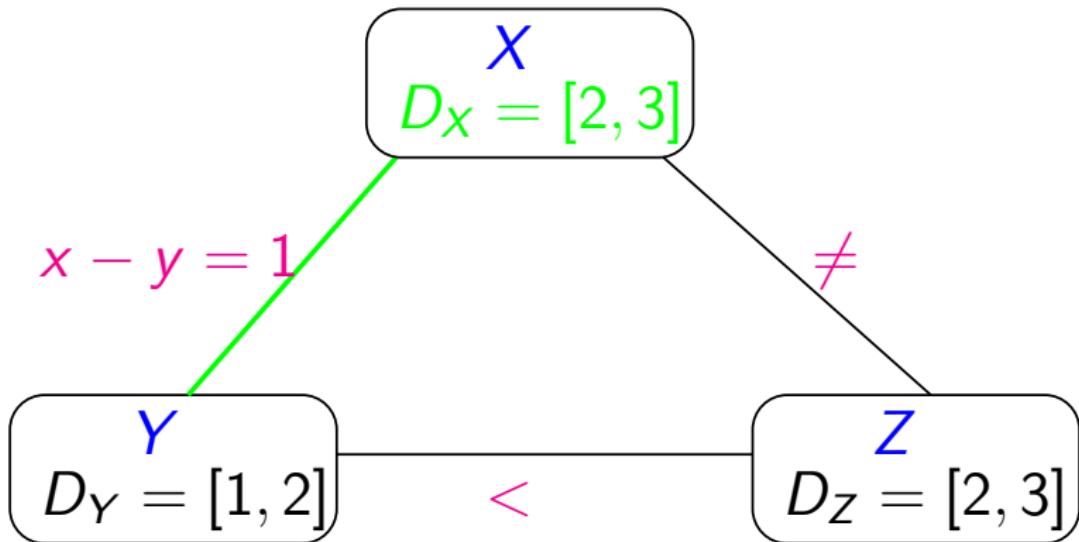
Résolution



Filtrage des contraintes

- ▶ réduction des domaines
- ▶ instantiation

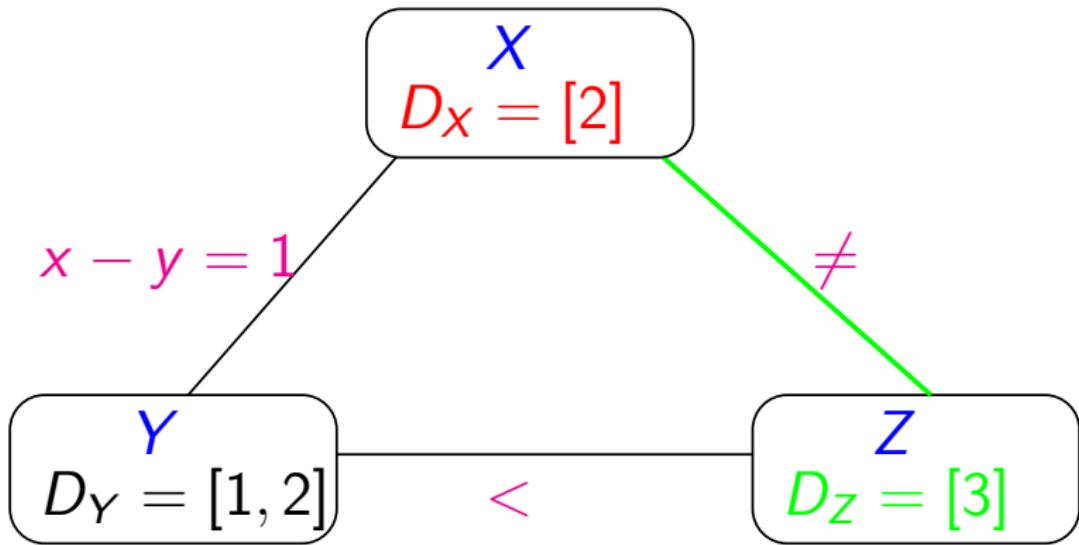
Résolution



Propagation de contraintes

- Le point fixe est indépendant de l'ordre de propagation.
- L'ordre peut influer sur la vitesse de convergence.

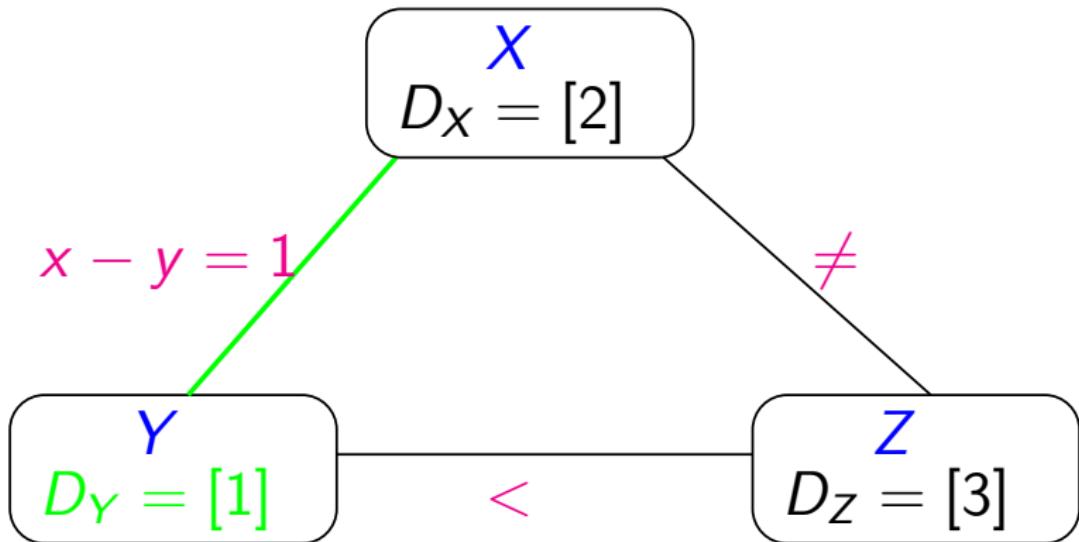
Résolution



Séparation itérative

- ▶ $\mathcal{P} \cup \{X = 2\}$
- ▶ ou $\mathcal{P} \cup \{X \neq 2\}$

Résolution



Séparation itérative

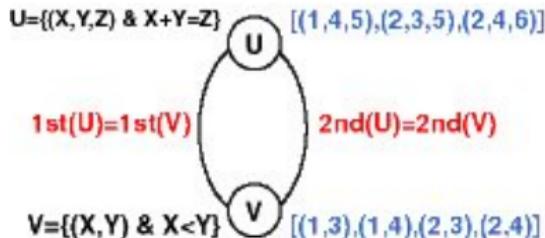
- $\mathcal{P} \cup \{X = 2\}$
- ou $\mathcal{P} \cup \{X \neq 2\}$

CSPs binaires et booléens

Problème de satisfaction de contraintes binaires

Tout CSP peut être converti en un CSP binaire : **encodage dual**.

- inversion du rôle des variables et des contraintes.

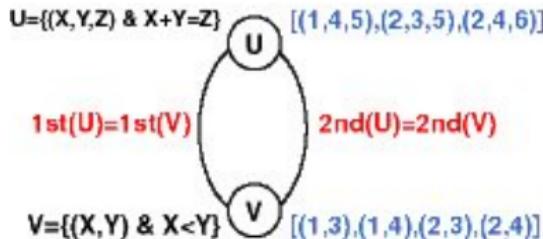


CSPs binaires et booléens

Problème de satisfaction de contraintes binaires

Tout CSP peut être converti en un CSP binaire : **encodage dual**.

- ▶ inversion du rôle des variables et des contraintes.



Problème de satisfaction de contraintes booléennes

Tout CSP peut être converti en un CSP booléen : **encodage SAT**.

- ▶ un booléen indique si une valeur est assignée à une variable.

Plan

1 Introduction

2 Problème de satisfaction de contraintes

3 Solveurs de contraintes

Solveur de contraintes

Réinventer la poudre ?

Il est inutile de programmer toutes les techniques en partant de rien.

Solveur de contraintes

Réinventer la poudre ?

Il est inutile de programmer toutes les techniques en partant de rien.

Solveur de contraintes

Un solveur se présente généralement sous la forme d'une **bibliothèque dans un langage de programmation** (Prolog, C, Java, . . .).

- ▶ **implémentation des structures de données**
 - ▶ variables, domaines, contraintes
- ▶ **cadre général de consistance**
- ▶ **stratégies de recherche**
- ▶ **interface flexible et évolutive**
 - ▶ variables, domaines
 - ▶ contraintes, algorithmes de filtrage
 - ▶ heuristiques, stratégie de recherche

Catalogue de solveurs

- ▶ SICStus (Prolog)
- ▶ ECLIPSe (Prolog)
- ▶ CHIP (C)
 - ▶ Prolog/C/C++
- ▶ ILOG (C++)
 - ▶ C++/Java/Python/OPL
- ▶ Gecode (C++)
- ▶ Mozart (Oz)
- ▶ Koalog (Java)
- ▶ Choco (Java)
- ▶ Jacop (Java)
- ▶ Comet

Catalogue de solveurs

- ▶ SICStus (Prolog)
- ▶ ECLIPSe (Prolog)
- ▶ CHIP (C)
 - ▶ Prolog/C/C++
- ▶ ILOG (C++)
 - ▶ C++/Java/Python/OPL
- ▶ Gecode (C++)
- ▶ Mozart (Oz)
- ▶ Koalog (Java)
- ▶ Choco (Java)
- ▶ Jacop (Java)
- ▶ Comet

Hybridation/Spécialisation

- ▶ intégration avec packages RO
 - ▶ programmation mathématique
 - ▶ recherche locale
- ▶ ordonnancement
 - ▶ activités
 - ▶ ressources, précédences
 - ▶ stratégies de recherche

Catalogue de solveurs

- ▶ SICStus (Prolog)
- ▶ ECLIPSe (Prolog)
- ▶ CHIP (C)
 - ▶ Prolog/C/C++
- ▶ ILOG (C++)
 - ▶ C++/Java/Python/OPL
- ▶ Gecode (C++)
- ▶ Mozart (Oz)
- ▶ Koalog (Java)
- ▶ Choco (Java)
- ▶ Jacop (Java)
- ▶ Comet

Hybridation/Spécialisation

- ▶ intégration avec packages RO
 - ▶ programmation mathématique
 - ▶ recherche locale
- ▶ ordonnancement
 - ▶ activités
 - ▶ ressources, précédences
 - ▶ stratégies de recherche

Pour les curieux

Catalogue of Constraint Programming Tools

En résumé

Cadre générale de la PPC

- ▶ La **consistance locale** relient les algorithmes de filtrage qui s'appliquent à chaque contrainte.
- ▶ La **recherche arborescente** résout les disjonctions restantes.

En résumé

Cadre générale de la PPC

- ▶ La **consistance locale** relient les algorithmes de filtrage qui s'appliquent à chaque contrainte.
- ▶ La **recherche arborescente** résout les disjonctions restantes.

Résolution d'un problème

- ▶ modèle déclaratif d'un problème sous la forme d'un CSP
- ▶ algorithmes de filtrage encodés dans des contraintes
- ▶ stratégies de recherche

Lecture 2

L'art de la modélisation

Programmation par contraintes

29 octobre 2015

Arnaud Malapert
Département d'informatique
Université Nice Sophia Antipolis

Plan

1 SEND+MORE=MONEY

2 Problème des dames

3 Séquence magique

4 Règles de Golomb

5 Suite de Langford

Constat

Tous les problèmes ne sont donnés pas sous la forme d'un CSP !

Recette

- ▶ formaliser le problème
- ▶ résoudre de petites instances « à la main »
- ▶ modèle de base
 - ▶ déclaration des variables
 - ▶ déclaration des contraintes
- ▶ reformulation
 - ▶ déclaration des variables auxiliaires
 - ▶ contraintes difficiles à exprimer ou peu efficace
- ▶ étude des modèles 0/1, dual et combiné
- ▶ étude des symétries

Plan

① SEND+MORE=MONEY

② Problème des dames

③ Séquence magique

④ Règles de Golomb

⑤ Suite de Langford

À vous de jouer

- > Solveur, quelle est la solution de SEND+MORE=MONEY ?
- > Facile!
- > $[9,5,6,7] + [1,0,8,5] = [1,0,6,5,2]$

SEND+MORE=MONEY

$$D + E = 10 \times r_1 + Y \quad (1)$$

$$r_1 + N + R = 10 \times r_2 + E \quad (2)$$

$$r_2 + E + O = 10 \times r_3 + N \quad (3)$$

$$r_3 + S + M = 10 \times M + O \quad (4)$$

$$S, M > 0 \quad (5)$$

$$\text{allDifferent}(S, E, N, D, M, O, R, Y) \quad (6)$$

$$r_1 \in [0, 1] \quad r_2 \in [0, 1] \quad r_3 \in [0, 1]$$

$$S \in [1, 9] \quad E \in [0, 9] \quad N \in [0, 9]$$

$$D \in [0, 9] \quad M \in [1, 9] \quad O \in [0, 9]$$

$$R \in [0, 9] \quad Y \in [0, 9]$$

SEND+MORE=MONEY

$$D + E = 10 \times r_1 + Y \quad (1)$$

$$(4) \quad M \leq \lfloor (\overline{r}_3 + \overline{S} - \underline{O}) \div 9 \rfloor \leq 1$$

$$r_1 + N + R = 10 \times r_2 + E \quad (2)$$

$$r_2 + E + O = 10 \times r_3 + N \quad (3)$$

$$r_3 + S + M = 10 \times M + O \quad (4)$$

$$S, M > 0 \quad (5)$$

$$\text{allDifferent}(S, E, N, D, M, O, R, Y)$$

(6)

$$r_1 \in [0, 1] \quad r_2 \in [0, 1] \quad r_3 \in [0, 1]$$

$$S \in [1, 9] \quad E \in [0, 9] \quad N \in [0, 9]$$

$$D \in [0, 9] \quad M = 1 \quad O \in [0, 9]$$

$$R \in [0, 9] \quad Y \in [0, 9]$$

SEND+MORE=MONEY

$$D + E = 10 \times r_1 + Y \quad (1)$$

$$r_1 + N + R = 10 \times r_2 + E \quad (2)$$

$$r_2 + E + O = 10 \times r_3 + N \quad (3)$$

$$r_3 + S + M = 10 \times M + O \quad (4)$$

$$S, M > 0 \quad (5)$$

$$\text{allDifferent}(S, E, N, D, M, O, R, Y) \quad (6)$$

$$r_1 \in [0, 1] \quad r_2 \in [0, 1] \quad r_3 \in [0, 1]$$

$$S \in [8, 9] \quad E \in [0, 9] \quad N \in [0, 9]$$

$$D \in [0, 9] \quad M = 1 \quad O \in [0, 1]$$

$$R \in [0, 9] \quad Y \in [0, 9]$$

$$(4) \quad M \leq \lfloor (\overline{r}_3 + \overline{S} - \underline{O}) \div 9 \rfloor \leq 1$$

$$(4) \quad O \leq \overline{r}_3 + \overline{S} - 9 \leq 1$$

$$(4) \quad S \geq 9 + \underline{O} - \overline{r}_3 \geq 8$$

SEND+MORE=MONEY

$$D + E = 10 \times r_1 + Y \quad (1)$$

$$r_1 + N + R = 10 \times r_2 + E \quad (2)$$

$$r_2 + E + O = 10 \times r_3 + N \quad (3)$$

$$r_3 + S + M = 10 \times M + O \quad (4)$$

$$S, M > 0 \quad (5)$$

$$\text{allDifferent}(S, E, N, D, M, O, R, Y) \quad (6)$$

$$(4) \quad M \leq \lfloor (\overline{r}_3 + \overline{S} - \underline{O}) \div 9 \rfloor \leq 1$$

$$(4) \quad O \leq \overline{r}_3 + \overline{S} - 9 \leq 1$$

$$(4) \quad S \geq 9 + \underline{O} - \overline{r}_3 \geq 8$$

$$(6) \quad O = 0; E, N, D, R, Y \geq 2$$

$$r_1 \in [0, 1] \quad r_2 \in [0, 1] \quad r_3 \in [0, 1]$$

$$S \in [8, 9] \quad E \in [2, 9] \quad N \in [2, 9]$$

$$D \in [2, 9] \quad M = 1 \quad O = 0$$

$$R \in [2, 9] \quad Y \in [2, 9]$$

SEND+MORE=MONEY

$$D + E = 10 \times r_1 + Y \quad (1)$$

$$r_1 + N + R = 10 \times r_2 + E \quad (2)$$

$$r_2 + E + O = 10 \times r_3 + N \quad (3)$$

$$r_3 + S + M = 10 \times M + O \quad (4)$$

$$S, M > 0 \quad (5)$$

$$\text{allDifferent}(S, E, N, D, M, O, R, Y) \quad (6)$$

$$(4) \quad M \leq \lfloor (\overline{r}_3 + \overline{S} - \underline{O}) \div 9 \rfloor \leq 1$$

$$(4) \quad O \leq \overline{r}_3 + \overline{S} - 9 \leq 1$$

$$(4) \quad S \geq 9 + \underline{O} - \overline{r}_3 \geq 8$$

$$(6) \quad O = 0; E, N, D, R, Y \geq 2$$

$$(3) \quad r_3 \leq \lfloor (\overline{r}_2 + \overline{E} - \underline{N}) \div 10 \rfloor \leq 0$$

$$r_1 \in [0, 1] \quad r_2 \in [0, 1] \quad r_3 = 0$$

$$S \in [8, 9] \quad E \in [2, 9] \quad N \in [2, 9]$$

$$D \in [2, 9] \quad M = 1 \quad O = 0$$

$$R \in [2, 9] \quad Y \in [2, 9]$$

SEND+MORE=MONEY

$$D + E = 10 \times r_1 + Y \quad (1)$$

$$r_1 + N + R = 10 \times r_2 + E \quad (2)$$

$$r_2 + E + O = 10 \times r_3 + N \quad (3)$$

$$r_3 + S + M = 10 \times M + O \quad (4)$$

$$S, M > 0 \quad (5)$$

$$\text{allDifferent}(S, E, N, D, M, O, R, Y) \quad (6)$$

$$(4) \quad M \leq \lfloor (\overline{r}_3 + \overline{S} - \underline{O}) \div 9 \rfloor \leq 1$$

$$(4) \quad O \leq \overline{r}_3 + \overline{S} - 9 \leq 1$$

$$(4) \quad S \geq 9 + \underline{O} - \overline{r}_3 \geq 8$$

$$(6) \quad O = 0; E, N, D, R, Y \geq 2$$

$$(3) \quad r_3 \leq \lfloor (\overline{r}_2 + \overline{E} - \underline{N}) \div 10 \rfloor \leq 0$$

$$(4) \quad S = 9$$

$$r_1 \in [0, 1] \quad r_2 \in [0, 1] \quad r_3 = 0$$

$$S = 9 \quad E \in [2, 9] \quad N \in [2, 9]$$

$$D \in [2, 9] \quad M = 1 \quad O = 0$$

$$R \in [2, 9] \quad Y \in [2, 9]$$

SEND+MORE=MONEY

$$D + E = 10 \times r_1 + Y \quad (1)$$

$$r_1 + N + R = 10 \times r_2 + E \quad (2)$$

$$r_2 + E + O = 10 \times r_3 + N \quad (3)$$

$$r_3 + S + M = 10 \times M + O \quad (4)$$

$$S, M > 0 \quad (5)$$

$$\text{allDifferent}(S, E, N, D, M, O, R, Y) \quad (6)$$

$$(4) \quad M \leq \lfloor (\overline{r}_3 + \overline{S} - \underline{O}) \div 9 \rfloor \leq 1$$

$$(4) \quad O \leq \overline{r}_3 + \overline{S} - 9 \leq 1$$

$$(4) \quad S \geq 9 + \underline{O} - \overline{r}_3 \geq 8$$

$$(6) \quad O = 0; E, N, D, R, Y \geq 2$$

$$(3) \quad r_3 \leq \lfloor (\overline{r}_2 + \overline{E} - \underline{N}) \div 10 \rfloor \leq 0$$

$$(4) \quad S = 9$$

$$(6) \quad E, N, D, R, Y \leq 8$$

$$r_1 \in [0, 1] \quad r_2 \in [0, 1] \quad r_3 = 0$$

$$S = 9 \quad E \in [2, 8] \quad N \in [2, 8]$$

$$D \in [2, 8] \quad M = 1 \quad O = 0$$

$$R \in [2, 8] \quad Y \in [2, 8]$$

SEND+MORE=MONEY

$$D + E = 10 \times r_1 + Y \quad (1)$$

$$r_1 + N + R = 10 \times r_2 + E \quad (2)$$

$$r_2 + E + O = 10 \times r_3 + N \quad (3)$$

$$r_3 + S + M = 10 \times M + O \quad (4)$$

$$S, M > 0 \quad (5)$$

$$\text{allDifferent}(S, E, N, D, M, O, R, Y) \quad (6)$$

$$(4) \quad M \leq \lfloor (\overline{r}_3 + \overline{S} - \underline{O}) \div 9 \rfloor \leq 1$$

$$(4) \quad O \leq \overline{r}_3 + \overline{S} - 9 \leq 1$$

$$(4) \quad S \geq 9 + \underline{O} - \overline{r}_3 \geq 8$$

$$(6) \quad O = 0; E, N, D, R, Y \geq 2$$

$$(3) \quad r_3 \leq \lfloor (\overline{r}_2 + \overline{E} - \underline{N}) \div 10 \rfloor \leq 0$$

$$(4) \quad S = 9$$

$$(6) \quad E, N, D, R, Y \leq 8$$

$$(3) \text{ ET } (6) \quad r_2 = 1 \wedge N \geq 3$$

$$r_1 \in [0, 1] \quad r_2 = 1 \quad r_3 = 0$$

$$S = 9 \quad E \in [2, 8] \quad N \in [3, 8]$$

$$D \in [2, 8] \quad M = 1 \quad O = 0$$

$$R \in [2, 8] \quad Y \in [2, 8]$$

SEND+MORE=MONEY

$$D + E = 10 \times r_1 + Y \quad (1)$$

$$r_1 + N + R = 10 \times r_2 + E \quad (2)$$

$$r_2 + E + O = 10 \times r_3 + N \quad (3)$$

$$r_3 + S + M = 10 \times M + O \quad (4)$$

$$S, M > 0 \quad (5)$$

$$\text{allDifferent}(S, E, N, D, M, O, R, Y) \quad (6)$$

$$r_1 = 1 \qquad r_2 = 1 \qquad r_3 = 0$$

$$S = 9 \qquad E \in [2, 8] \quad N \in [3, 8]$$

$$D \in [2, 8] \qquad M = 1 \qquad O = 0$$

$$R = 8 \qquad Y \in [2, 8]$$

$$(4) \quad M \leq \lfloor (\overline{r}_3 + \overline{S} - \underline{O}) \div 9 \rfloor \leq 1$$

$$(4) \quad O \leq \overline{r}_3 + \overline{S} - 9 \leq 1$$

$$(4) \quad S \geq 9 + \underline{O} - \overline{r}_3 \geq 8$$

$$(6) \quad O = 0; E, N, D, R, Y \geq 2$$

$$(3) \quad r_3 \leq \lfloor (\overline{r}_2 + \overline{E} - \underline{N}) \div 10 \rfloor \leq 0$$

$$(4) \quad S = 9$$

$$(6) \quad E, N, D, R, Y \leq 8$$

$$(3) \text{ ET } (6) \quad r_2 = 1 \wedge N \geq 3$$

$$(2) \quad R \geq 9 - \overline{r}_1 \geq 8$$

$$(2) \quad r_1 \geq 9 - \overline{R} \geq 1$$

SEND+MORE=MONEY

$$D + E = 10 \times r_1 + Y \quad (1)$$

$$r_1 + N + R = 10 \times r_2 + E \quad (2)$$

$$r_2 + E + O = 10 \times r_3 + N \quad (3)$$

$$r_3 + S + M = 10 \times M + O \quad (4)$$

$$S, M > 0 \quad (5)$$

$$\text{allDifferent}(S, E, N, D, M, O, R, Y) \quad (6)$$

$$r_1 = 1 \qquad r_2 = 1 \qquad r_3 = 0$$

$$S = 9 \qquad E \in [5, 7] \quad N \in [3, 7]$$

$$D \in [2, 7] \qquad M = 1 \qquad O = 0$$

$$R = 8 \qquad Y \in [2, 7]$$

$$(4) \quad M \leq \lfloor (\overline{r}_3 + \overline{S} - \underline{O}) \div 9 \rfloor \leq 1$$

$$(4) \quad O \leq \overline{r}_3 + \overline{S} - 9 \leq 1$$

$$(4) \quad S \geq 9 + \underline{O} - \overline{r}_3 \geq 8$$

$$(6) \quad O = 0; E, N, D, R, Y \geq 2$$

$$(3) \quad r_3 \leq \lfloor (\overline{r}_2 + \overline{E} - \underline{N}) \div 10 \rfloor \leq 0$$

$$(4) \quad S = 9$$

$$(6) \quad E, N, D, R, Y \leq 8$$

$$(3) \text{ ET } (6) \quad r_2 = 1 \wedge N \geq 3$$

$$(2) \quad R \geq 9 - \overline{r}_1 \geq 8$$

$$(2) \quad r_1 \geq 9 - \overline{R} \geq 1$$

$$(6) \quad E, N, D, Y \leq 7$$

SEND+MORE=MONEY

$$D + E = 10 \times r_1 + Y \quad (1)$$

$$r_1 + N + R = 10 \times r_2 + E \quad (2)$$

$$r_2 + E + O = 10 \times r_3 + N \quad (3)$$

$$r_3 + S + M = 10 \times M + O \quad (4)$$

$$S, M > 0 \quad (5)$$

$$\text{allDifferent}(S, E, N, D, M, O, R, Y) \quad (6)$$

$$r_1 = 1 \qquad r_2 = 1 \qquad r_3 = 0$$

$$S = 9 \qquad E \in [5, 7] \quad N \in [3, 7]$$

$$D \in [5, 7] \qquad M = 1 \qquad O = 0$$

$$R = 8 \qquad Y \in [2, 4]$$

$$(4) \quad M \leq \lfloor (\overline{r}_3 + \overline{S} - \underline{O}) \div 9 \rfloor \leq 1$$

$$(4) \quad O \leq \overline{r}_3 + \overline{S} - 9 \leq 1$$

$$(4) \quad S \geq 9 + \underline{O} - \overline{r}_3 \geq 8$$

$$(6) \quad O = 0; E, N, D, R, Y \geq 2$$

$$(3) \quad r_3 \leq \lfloor (\overline{r}_2 + \overline{E} - \underline{N}) \div 10 \rfloor \leq 0$$

$$(4) \quad S = 9$$

$$(6) \quad E, N, D, R, Y \leq 8$$

$$(3) \text{ ET } (6) \quad r_2 = 1 \wedge N \geq 3$$

$$(2) \quad R \geq 9 - \overline{r}_1 \geq 8$$

$$(2) \quad r_1 \geq 9 - \overline{R} \geq 1$$

$$(6) \quad E, N, D, Y \leq 7$$

$$(1) \quad (D, E) \geq 10 + \underline{Y} - (\overline{E}, \overline{D}) \geq 5$$

$$(1) \quad Y \leq \overline{E} + \overline{D} - 10 \leq 4$$

$$9\text{END} + 108\text{E} = 10\text{NEY}$$

$$D + E = 10 + Y \quad (7)$$

$$N = E + 1 \quad (8)$$

$$\text{allDifferent}(E, N, D, Y) \quad (9)$$

$$E \in [5, 7] \quad N \in [3, 7]$$

$$D \in [5, 7] \quad Y \in [2, 4]$$

$$9\text{END} + 108\text{E} = 10\text{NEY}$$

$$D + E = 10 + Y \quad (7)$$

$$N = E + 1 \quad (8)$$

$$\text{allDifferent}(E, N, D, Y) \quad (9)$$

$$(8) \quad N \geq E + 1 \geq 6$$

$$(8) \quad E \leq N - 1 \leq 6$$

$$E \in [5, 6] \quad N \in [6, 7]$$

$$D \in [5, 7] \quad Y \in [2, 4]$$

$$9\text{END} + 108\text{E} = 10\text{NEY}$$

$$D + E = 10 + Y \quad (7)$$

$$N = E + 1 \quad (8)$$

$$\text{allDifferent}(E, N, D, Y) \quad (9)$$

$$(8) \quad N \geq \underline{E} + 1 \geq 6$$

$$(8) \quad E \leq \bar{N} - 1 \leq 6$$

$$(7) \quad D \geq 10 + \underline{Y} - \bar{E} \geq 6$$

$$(7) \quad Y \leq \bar{D} + \bar{E} - 10 \geq 3$$

$$E \in [5, 6] \quad N \in [6, 7]$$

$$D \in [6, 7] \quad Y \in [2, 3]$$

$$9\text{END} + 108\text{E} = 10\text{NEY}$$

$$D + E = 10 + Y \quad (7)$$

$$N = E + 1 \quad (8)$$

$$\text{allDifferent}(E, N, D, Y) \quad (9)$$

$$(8) \quad N \geq \underline{E} + 1 \geq 6$$

$$(8) \quad E \leq \bar{N} - 1 \leq 6$$

$$(7) \quad D \geq 10 + \underline{Y} - \bar{E} \geq 6$$

$$(7) \quad Y \leq \bar{D} + \bar{E} - 10 \geq 3$$

$$(9) \quad E = 5$$

$$E = 5 \quad N \in [6, 7]$$

$$D \in [6, 7] \quad Y \in [2, 3]$$

$$9\text{END} + 108\text{E} = 10\text{NEY}$$

$$D + E = 10 + Y \quad (7)$$

$$N = E + 1 \quad (8)$$

$$\text{allDifferent}(E, N, D, Y) \quad (9)$$

$$E = 5 \quad N = 6$$

$$D \in [6, 7] \quad Y \in [2, 3]$$

$$(8) \quad N \geq \underline{E} + 1 \geq 6$$

$$(8) \quad E \leq \bar{N} - 1 \leq 6$$

$$(7) \quad D \geq 10 + \underline{Y} - \bar{E} \geq 6$$

$$(7) \quad Y \leq \bar{D} + \bar{E} - 10 \geq 3$$

$$(9) \quad E = 5$$

$$(8) \quad N = 6$$

$$9\text{END} + 108\text{E} = 10\text{NEY}$$

$$D + E = 10 + Y \quad (7)$$

$$N = E + 1 \quad (8)$$

$$\text{allDifferent}(E, N, D, Y) \quad (9)$$

$$(8) \quad N \geq \underline{E} + 1 \geq 6$$

$$(8) \quad E \leq \bar{N} - 1 \leq 6$$

$$(7) \quad D \geq 10 + \underline{Y} - \bar{E} \geq 6$$

$$(7) \quad Y \leq \bar{D} + \bar{E} - 10 \geq 3$$

$$(9) \quad E = 5$$

$$(8) \quad N = 6$$

$$(9) \quad D = 7$$

$$E = 5 \quad N = 6$$

$$D = 7 \quad Y \in [2, 3]$$

$$9\text{END} + 108\text{E} = 10\text{NEY}$$

$$D + E = 10 + Y \quad (7)$$

$$N = E + 1 \quad (8)$$

$$\text{allDifferent}(E, N, D, Y) \quad (9)$$

$$(8) \quad N \geq \underline{E} + 1 \geq 6$$

$$(8) \quad E \leq \bar{N} - 1 \leq 6$$

$$(7) \quad D \geq 10 + \underline{Y} - \bar{E} \geq 6$$

$$(7) \quad Y \leq \bar{D} + \bar{E} - 10 \geq 3$$

$$(9) \quad E = 5$$

$$(8) \quad N = 6$$

$$(9) \quad D = 7$$

$$(7) \quad Y = 2$$

$$E = 5 \quad N = 6$$

$$D = 7 \quad Y = 2$$

SOLUTION

Plan

1 SEND+MORE=MONEY

2 Problème des dames

3 Séquence magique

4 Règles de Golomb

5 Suite de Langford

CSP, vos papiers !

Sur un échiquier de taille $n \times n$, placer un **nombre maximum** de dames de telle sorte qu'aucune dame ne puisse en capturer une autre.

- ▶ Une dame peut en capturer une autre si elles sont sur la même ligne, colonne, ou diagonale.

CSP, vos papiers !

Sur un échiquier de taille $n \times n$, placer un **nombre maximum** de dames de telle sorte qu'aucune dame ne puisse en capturer une autre.

- ▶ Une dame peut en capturer une autre si elles sont sur la même ligne, colonne, ou diagonale.

Méthode simple

Transformer le problème d'optimisation en une séquence de problèmes de satisfaction de contraintes :

- ▶ 1 dame sur l'échiquier ?
- ▶ 2 dames sur l'échiquier ?
- ▶ ...
- ▶ n dames sur l'échiquier ?

CSP, vos papiers !

Sur un échiquier de taille $n \times n$, placer un **nombre maximum** de dames de telle sorte qu'aucune dame ne puisse en capturer une autre.

- ▶ Une dame peut en capturer une autre si elles sont sur la même ligne, colonne, ou diagonale.

Méthode simple

Transformer le problème d'optimisation en une séquence de problèmes de satisfaction de contraintes :

- ▶ 1 dame sur l'échiquier ?
- ▶ 2 dames sur l'échiquier ?
- ▶ ...
- ▶ n dames sur l'échiquier ?

Exercice 1

Résoudre le problème à la main pour $n = 2, 3, 4, 5$.

Algorithme glouton ($n \geq 4$)

List of numbers for vertical positions (rows) of queens with horizontal position (column) simply increasing.

- ▶ If the remainder from dividing N by 6 is not 2 or 3 then the list is simply all even numbers followed by all odd numbers $\leq n$.
- ▶ Otherwise, write separate lists of even and odd numbers.
 - ▶ 2,4,6,8 – 1,3,5,7
- ▶ If the remainder is 2, swap 1 and 3 in odd list and move 5 to the end.
 - ▶ 3,1,7,5
- ▶ If the remainder is 3, move 2 to the end of even list and 1,3 to the end of odd list.
 - ▶ 4,6,8,2 – 5,7,9,1,3
- ▶ Append odd list to the even list and place queens in the rows given by these numbers, from left to right.
 - ▶ a2, b4, c6, d8, e3, f1, g7, h5

Exemples

Algorithme glouton ($n \geq 4$)

List of numbers for vertical positions (rows) of queens with horizontal position (column) simply increasing.

- ▶ If the remainder from dividing N by 6 is not 2 or 3 then the list is simply all even numbers followed by all odd numbers $\leq n$.
- ▶ Otherwise, write separate lists of even and odd numbers.
 - ▶ 2,4,6,8 – 1,3,5,7
- ▶ If the remainder is 2, swap 1 and 3 in odd list and move 5 to the end.
 - ▶ 3,1,7,5
- ▶ If the remainder is 3, move 2 to the end of even list and 1,3 to the end of odd list.
 - ▶ 4,6,8,2 – 5,7,9,1,3
- ▶ Append odd list to the even list and place queens in the rows given by these numbers, from left to right.
 - ▶ a2, b4, c6, d8, e3, f1, g7, h5

Exemples

14 dames (reste 2) : 2, 4, 6, 8, 10, 12, 14, 3, 1, 7, 9, 11, 13, 5.

Algorithme glouton ($n \geq 4$)

List of numbers for vertical positions (rows) of queens with horizontal position (column) simply increasing.

- ▶ If the remainder from dividing N by 6 is not 2 or 3 then the list is simply all even numbers followed by all odd numbers $\leq n$.
- ▶ Otherwise, write separate lists of even and odd numbers.
 - ▶ 2,4,6,8 – 1,3,5,7
- ▶ If the remainder is 2, swap 1 and 3 in odd list and move 5 to the end.
 - ▶ 3,1,7,5
- ▶ If the remainder is 3, move 2 to the end of even list and 1,3 to the end of odd list.
 - ▶ 4,6,8,2 – 5,7,9,1,3
- ▶ Append odd list to the even list and place queens in the rows given by these numbers, from left to right.
 - ▶ a2, b4, c6, d8, e3, f1, g7, h5

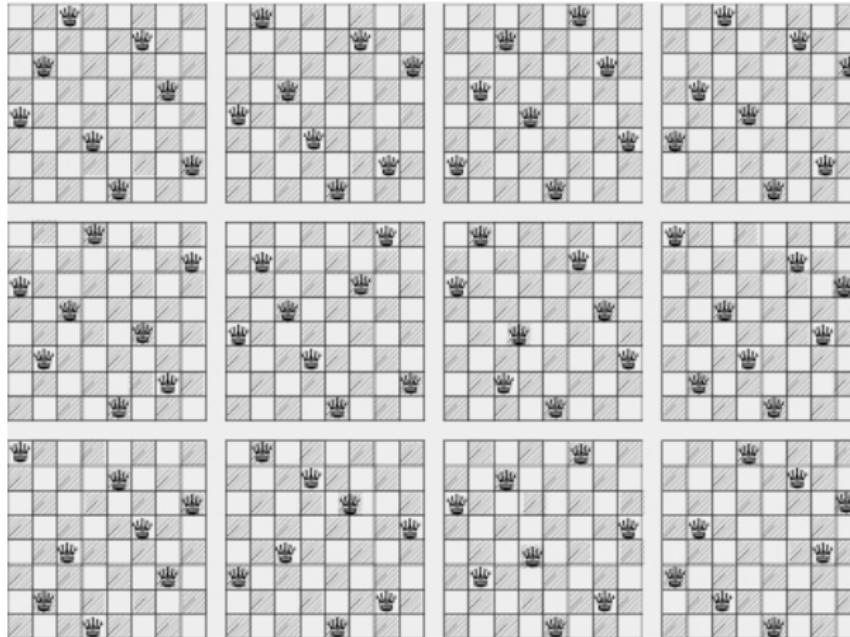
Exemples

14 dames (reste 2) : 2, 4, 6, 8, 10, 12, 14, 3, 1, 7, 9, 11, 13, 5.

15 dames (reste 3) : 4, 6, 8, 10, 12, 14, 2, 5, 7, 9, 11, 13, 15, 1, 3.

Problème des n dames (n queens problem)

Sur un échiquier de taille $n \times n$, placer n dames de telle sorte qu'aucune dame ne puisse en capturer une autre.



Modèle booléen

- ▶ La variable $q_{ij} \in [0, 1]$ indique si une reine est présente sur la case (i, j) .

Modèle lignes/colonnes

- ▶ La variable $l_i \in [1, n]$ indique la **ligne** de la dame i .
- ▶ La variable $c_i \in [1, n]$ indique la **colonne** de la dame i .

$$c_i \neq c_j \quad 1 \leq i < j \leq n$$

$$l_i \neq l_j \quad 1 \leq i < j \leq n$$

$$l_i + (c_j - c_i) \neq l_j \quad 1 \leq i \neq j \leq n$$

$$c_i + (l_i - l_j) \neq c_j \quad 1 \leq i \neq j \leq n$$

Modèle lignes (binaire)

- ▶ la dame i se trouve sur la ligne i
- ▶ La variable $c_i \in [1, n]$ représente la colonne de la dame i .

$$c_i \neq c_j \quad 1 \leq i < j \leq n$$

$$c_i \neq c_j + (j - i) \quad 1 \leq i < j \leq n$$

$$c_i \neq c_j - (j - i) \quad 1 \leq i < j \leq n$$

Modèle lignes (global)

- ▶ la dame i se trouve sur la ligne i
- ▶ La variable $c_i \in [1, n]$ représente la colonne de la dame i .
- ▶ Les variables auxiliaires m_i et p_i représentent les projections de la dame i sur la colonne 0 selon les diagonales.
- ▶ Des contraintes globales remplacent les cliques d'inégalités binaires.

`allDifferent(c_1, \dots, c_n)`

$m_i = c_i - i$

$1 \leq i \leq n$

`allDifferent(m_1, \dots, m_n)`

$p_i = c_i + i$

$1 \leq i \leq n$

`allDifferent(p_1, \dots, p_n)`

Dualité

Exercice 2

Proposer un modèle **dual**.

Exercice 3

Proposer un modèle **combiné**, i.e. le modèles primal et le dual.

Dualité

Exercice 2

Proposer un modèle **dual**.

Solution

- ▶ Inversion du rôle des lignes et des colonnes.
- ▶ la dame j se trouve sur la colonne j
- ▶ La variable $I_j \in [1, n]$ représente la ligne de la dame j .

Exercice 3

Proposer un modèle **combiné**, i.e. le modèles primal et le dual.

Dualité

Exercice 2

Proposer un modèle **dual**.

Solution

- ▶ Inversion du rôle des lignes et des colonnes.
- ▶ la dame j se trouve sur la colonne j
- ▶ La variable $I_j \in [1, n]$ représente la ligne de la dame j .

Exercice 3

Proposer un modèle **combiné**, i.e. le modèles primal et le dual.

Solution

$$I_j = i \Leftrightarrow c_i = j \quad 1 \leq i \leq n$$

Symétries

Une **solution unique ou fondamentale** a au plus 8 variantes (incluant elle-même) :

- ▶ **rotation** d'un angle de 90, 180, ou 270 degrés et ensuite
- ▶ **réflexion** en miroir selon un axe fixe.

# - - - -	- - - - #	- # - - -	# - - - -
- - # - -	- # - - -	- - - # -	- - - # -
- - - - #	- - - # -	# - - - -	- # - - -
- # - - -	# - - - -	- - # - -	- - - - #
- - - # -	- - # - -	- - - - #	- - # - -
- - - # -	- - # - -	- - - - #	- - # - -
- # - - -	# - - - -	- - # - -	- - - - #
- - - - #	- - - # -	# - - - -	- # - - -
- - # - -	- # - - -	- - - # -	- - - # -
# - - - -	- - - - #	- # - - -	# - - - -

Dénombrement

Invariant par rotation de 90 degrés

2 : Identité ; miroir.

$$2x + 8x = 10$$

-	#	-	-	-	#	-	-	-	-
-	-	-	-	#	-	-	#	-	-
-	-	#	-	-	-	-	-	-	#
#	-	-	-	-	-	#	-	-	-
-	-	-	#	-	-	-	-	#	-

Dénombrement

Invariant par rotation de 90 degrés

2 : Identité ; miroir.

Invariant par rotation de 180 degrés

4 : identité ; réflexion ; rotation 90 degrés ; réflexion de la rotation

$$2x + 8x = 10$$

-	#	-	-	-	#	-	-	-	-
-	-	-	-	#	-	-	#	-	-
-	-	#	-	-	-	-	-	-	#
#	-	-	-	-	-	#	-	-	-
-	-	-	#	-	-	-	-	#	-

Dénombrement

Invariant par rotation de 90 degrés

2 : Identité ; miroir.

Invariant par rotation de 180 degrés

4 : identité ; réflexion ; rotation 90 degrés ; réflexion de la rotation

Invariant par miroir

Impossible, sinon deux reines seraient placées en face l'une de l'autre.

$$2x + 8x = 10$$

$$\begin{array}{r} - \# - - - \\ - - - - \# \\ - - \# - - \\ \# - - - - \\ - - - \# - \end{array} \quad \begin{array}{r} \# - - - - \\ - - \# - - - \\ - - - - \# \\ - \# - - - \\ - - - \# - \end{array}$$

Élimination des symétries

Rotation de 180 degrés

$$\wedge_{j=1}^{k-1} (c_j = n + 1 - c_{n+1-j}) \Rightarrow c_k \leq n + 1 - c_{n+1-k} \quad 1 \leq k \leq \lfloor \frac{n}{2} \rfloor$$

8 dames

$$c_1 = n + 1 - c_8 \quad \wedge \quad c_2 \leq n + 1 - c_7$$

- - # - | - - - - - - # - | - - - -
- - - - | # - - - - - - - | - - # -
- - - - | - - - # - # - - | - - - -
- - - - # | - - - - - - - - | - - - #
- - - | - - - - - - - - | # - - -
- - - - | - - # - # - - - | - - - -
- # - - | - - - - - - - - # | - - - -
- - - - | - # - - - - - - | - # - -

Élimination des symétries

Exercice 4 (Rotation de 90 degrés)

- ① Proposer des contraintes pour le modèle primal.
- ② Proposer des contraintes pour le modèle combiné.

Solution

$$\wedge_{j=1}^{k-1} (c_j = p \wedge c_p = n + 1 - j) \Rightarrow c_k \leq n + 1 - k \quad 1 \leq k \leq n - 1$$

- - # - | - - - - - - - - # | - - - -
- - - - | # - - - - # - - | - - - -
- - - - | - - - # - - - - | - - - - #
- - - - # | - - - - - - - - - | # - - -
- - - | - - - - - - - - | - - # -
- - - - | - - - # - # - - - | - - - -
- # - - | - - - - - - # - | - - - -
- - - - | - # - - - - - - | - # - -

Plan

1 SEND+MORE=MONEY

2 Problème des dames

3 Séquence magique

4 Règles de Golomb

5 Suite de Langford

Énoncé du problème

Une **séquence magique de longueur n** est

- ▶ une séquence d'entiers x_0, \dots, x_{n-1} compris entre 0 et $n - 1$ telle que
- ▶ le nombre i ($i = 0, \dots, n - 1$) apparaisse exactement x_i fois dans la séquence.

Séquence magique ($n = 10$)

i	0	1	2	3	4	5	6	7	8	9
x_i	6	2	1	0	0	0	1	0	0	0

- ▶ 0 apparaît **six** fois,
- ▶ 1 apparaît **deux** fois,
- ▶ 2 apparaît **une** fois,
- ▶ 6 apparaît **une** fois

Énoncé du problème

Une **séquence magique de longueur n** est

- ▶ une séquence d'entiers x_0, \dots, x_{n-1} compris entre 0 et $n - 1$ telle que
- ▶ le nombre i ($i = 0, \dots, n - 1$) apparaisse exactement x_i fois dans la séquence.

Séquence magique ($n = 10$)

i	0	1	2	3	4	5	6	7	8	9
x_i	6	2	1	0	0	0	1	0	0	0

- ▶ 0 apparaît **six** fois,
- ▶ 1 apparaît **deux** fois,
- ▶ 2 apparaît **une** fois,
- ▶ 6 apparaît **une** fois

Exercice 5

Résoudre le problème à la main pour $n = 1, 2, 3, 4, 5$.

Modèle de base

x_i ($i = 0, \dots, n - 1$) est égale au nombre d'occurrences de la valeur i dans le tableau $[x_0, \dots, x_{n-1}]$.

$$x_i = |\{j \mid 0 \leq j \leq n - 1 \wedge x_j = i\}| \quad 0 \leq i \leq n - 1$$

Modèle de base

x_i ($i = 0, \dots, n - 1$) est égale au nombre d'occurrences de la valeur i dans le tableau $[x_0, \dots, x_{n-1}]$.

$$x_i = |\{j \mid 0 \leq j \leq n - 1 \wedge x_j = i\}| \quad 0 \leq i \leq n - 1$$

Rappel : globalCardinality

Les valeurs v_1, \dots, v_m apparaissent O_{v_1}, \dots, O_{v_m} dans X_1, \dots, X_n .

Contrainte redondante

- ▶ une contrainte redondante est impliquée par une ou plusieurs autres.
- ▶ Elle n'est pas nécessaire, mais on espère qu'elle va aider à la résolution.

Contrainte redondante sur la somme

La somme des occurrences des valeurs dans la séquence est égale au nombre d'éléments de la séquence :

$$\sum_{i=0}^{n-1} x_i = n.$$

n = 10

$$6 + 2 + 1 + 0 + 0 + 0 + 1 + 0 + 0 + 0 = 10$$

Contrainte redondante sur la somme des produits

La somme des produits des valeurs et de leurs occurrences est égale au nombre d'éléments de la séquence :

$$\sum_{i=0}^{n-1} i \times x_i = n.$$

$n = 10$

0,0,0,0,0,0 , 1,1, 2, 0,0,0, 6, 0,0,0

Propagation de contraintes ($n = 10$)

Propagation de contraintes ($n = 10$)

$$D(x_0) = [0, 9]$$

$$D(x_1) = [0, 9]$$

$$D(x_2) = [0, 5]$$

$$D(x_3) = [0, 3]$$

$$D(x_4) = [0, 2]$$

$$D(x_5) = [0, 2]$$

$$D(x_6) = [0, 1]$$

$$D(x_7) = [0, 1]$$

$$D(x_8) = [0, 1]$$

$$D(x_9) = [0, 1]$$

Propagation de contraintes ($n = 10$)

$$D(x_0) = [0, 9]$$

$$D(x_1) = [0, 9]$$

$$D(x_2) = [0, 5]$$

$$D(x_3) = [0, 3]$$

$$D(x_4) = [0, 2]$$

$$D(x_5) = [0, 2]$$

$$D(x_6) = [0, 1]$$

$$D(x_7) = [0, 1]$$

$$D(x_8) = [0, 1]$$

$$D(x_9) = [0, 1]$$

Shaving ou SAC

Pour chaque **variable** $x \in X$ et
chaque **valeur** $v \in D(x)$:

- ▶ propager la décision $x = v$.
- ▶ Si une contradiction est détectée, alors $x \neq v$.

Recommencer jusqu'au point fixe.

Propagation de contraintes ($n = 10$)

$$D(x_0) = [0, 9]$$

$$D(x_1) = [0, 9]$$

$$D(x_2) = [0, 5]$$

$$D(x_3) = [0, 3]$$

$$D(x_4) = [0, 2]$$

$$D(x_5) = [0, 2]$$

$$D(x_6) = [0, 1]$$

$$D(x_7) = [0, 1]$$

$$D(x_8) = [0, 1]$$

$$D(x_9) = [0, 1]$$

Shaving ou SAC

Pour chaque **variable** $x \in X$ et chaque **valeur** $v \in D(x)$:

- ▶ propager la décision $x = v$.
- ▶ Si une contradiction est détectée, alors $x \neq v$.

Recommencer jusqu'au point fixe.

Performance

Search 8 nœuds et 8 backtracks.

Shaving 13 affectations.

Le mot de la fin

Pour $n > 6$, le motif suivant représente l'unique (?) séquence magique :

$$n - 4, 2, 1, \dots, 1, 0, 0, 0.$$

Plan

1 SEND+MORE=MONEY

2 Problème des dames

3 Séquence magique

4 Règles de Golomb

5 Suite de Langford

Règle de Golomb

Une **règle de Golomb** $R(n, l)$ est :

- ▶ un ensemble de n entiers $\{a_1, \dots, a_n\}$
- ▶ de longueur $l = \max_{i,j \in [1,n]} (a_i - a_j)$ et tel que
- ▶ les différences $a_i - a_j (i, j \in [1, n])$ soient distinctes.

Une règle est optimale si elle est de longueur minimale.

Plus grande règle optimale : $R(26, 492)$

0 1 33 83 104 110 124 163 185 200 203 249 251 258 314 318 343
356 386 430 440 456 464 475 487 492

Applications

radioastronomie, cristallographie

Règle de Golomb

Une règle de Golomb $R(n, l)$ est :

- ▶ un ensemble de n entiers $\{a_1, \dots, a_n\}$
- ▶ de longueur $l = \max_{i,j \in [1,n]} (a_i - a_j)$ et tel que
- ▶ les différences $a_i - a_j (i, j \in [1, n])$ soient distinctes.

Une règle est optimale si elle est de longueur minimale.

Plus grande règle optimale : $R(26, 492)$

0 1 33 83 104 110 124 163 185 200 203 249 251 258 314 318 343
356 386 430 440 456 464 475 487 492

Applications

radioastronomie, cristallographie

Exercice 6

Résoudre le problème à la main pour $n = 1, 2, 3, 4, 5$.

Algorithme glouton : $R(n, 2^{n-1} - 1)$

Règle exponentielle

$$a_i = 2^i - 1 \quad i \geq 0$$

0, 1, 3, 7, 15, 31, 63, ...

Preuve

formelle décomposition en facteurs premiers.

intuitive décomposition binaire.

$$\begin{aligned} a_i - a_j &= 2^i - 2^j = 2^j(2^{i-j} - 1) \\ &= (\underbrace{1 \dots 1}_{i-j} \underbrace{0 \dots 0}_j)_2 \end{aligned}$$

Élimination des symétries

Élimination des symétries

Monotonie

$$a_1 < a_2 < \dots < a_n$$

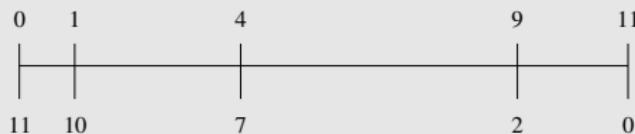
Translation

$$a_1 = 0$$

Réflexion : $a'_i = a_n - a_{n-i}$

La première différence est plus petite que la dernière :

$$a_2 - a_1 < a_n - a_{n-1}.$$



Élimination des symétries

Monotonie

$$a_1 < a_2 < \dots < a_n$$

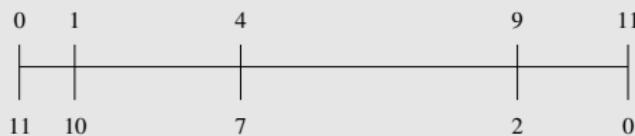
Translation

$$a_1 = 0$$

Réflexion : $a'_i = a_n - a_{n-i}$

La première différence est plus petite que la dernière :

$$a_2 - a_1 < a_n - a_{n-1}.$$



Exercice 7

Prouver que deux marques ne peuvent pas être à la même position.

Modèle de base

$a_i \in [0, l]$ la position de la i -ème marque ($i = 1, \dots, n$) de la règle.

$d_{ij} \in [1, l]$ la différence entre les positions des marques i et j
($1 \leq i < j \leq n$).

$$d_{ij} = a_j - a_i$$

$$1 \leq i < j \leq n$$

$$\text{allDifferent}([d_{ij}]_{1 \leq i < j \leq n})$$

Modèle de base

$a_i \in [0, l]$ la position de la i -ème marque ($i = 1, \dots, n$) de la règle.

$d_{ij} \in [1, l]$ la différence entre les positions des marques i et j
 $(1 \leq i < j \leq n)$.

$$d_{ij} = a_j - a_i$$

$$1 \leq i < j \leq n$$

$$\text{allDifferent}([d_{ij}]_{1 \leq i < j \leq n})$$

Exercice 8

Proposer des contraintes redondantes.

Modèle de base

$a_i \in [0, l]$ la position de la i -ème marque ($i = 1, \dots, n$) de la règle.

$d_{ij} \in [1, l]$ la différence entre les positions des marques i et j
($1 \leq i < j \leq n$).

$$d_{ij} = a_j - a_i$$

$$1 \leq i < j \leq n$$

$$\text{allDifferent}([d_{ij}]_{1 \leq i < j \leq n})$$

Exercice 8

Proposer des contraintes redondantes.

Solution

- ▶ Triplet de différences : $d_{ij} = d_{ik} + d_{kj}$ $1 \leq i < k < j \leq n$
- ▶ Différences consécutives : $d_{ij} = \sum_{k=i}^{j-1} d_{k(k+1)}$ $1 \leq i < j \leq n$

Bornes inférieures

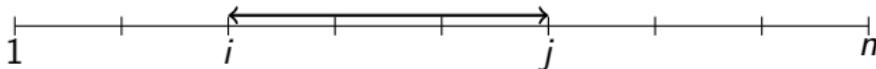
Quelles propriétés doit vérifier la partie de la règle comprise entre les marques a_i et a_j ?



La **sous-règle** comprise entre les marques x_i et x_j **est aussi une règle de Golomb** (pas forcément optimale) avec $j - i + 1$ marques.

Bornes inférieures

Quelles propriétés doit vérifier la partie de la règle comprise entre les marques a_i et a_j ?



La **sous-règle** comprise entre les marques x_i et x_j **est aussi une règle de Golomb** (pas forcément optimale) avec $j - i + 1$ marques.

- ▶ Sa longueur est supérieure à la somme des $j - i$ plus petits entiers.

$$d_{ij} = a_j - a_i = \sum_{k=i}^{j-1} d_{k(k+1)} \geq \sum_{k=1}^{j-i} k$$

Bornes inférieures

Quelles propriétés doit vérifier la partie de la règle comprise entre les marques a_i et a_j ?



La **sous-règle** comprise entre les marques x_i et x_j est aussi une règle de Golomb (pas forcément optimale) avec $j - i + 1$ marques.

- ▶ Sa longueur est supérieure à la somme des $j - i$ plus petits entiers.

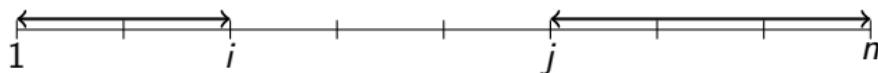
$$d_{ij} = a_j - a_i = \sum_{k=i}^{j-1} d_{k(k+1)} \geq \sum_{k=1}^{j-i} k$$

- ▶ Soit L_m la longueur optimale de la règle de Golomb d'ordre $m < n$.

$$d_{ij} \geq L_{j-i+1}$$

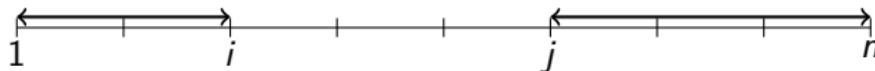
Borne supérieure

Quelle propriété doit vérifier le complémentaire de la sous-règle comprise entre les marques x_i et x_j ?



Borne supérieure

Quelle propriété doit vérifier le complémentaire de la sous-règle comprise entre les marques x_i et x_j ?



- ▶ Les différences consécutives du complémentaire sont distinctes.

$$d_{ij} = d_{1n} - d_{1i} - d_{jn} = x_n - \sum_{k=1}^i d_{k(k+1)} - \sum_{k=j}^n d_{k(k+1)}$$

$$d_{ij} \geq x_n - \sum_{k=1}^{m-1-j+i} k$$

- ▶ Le complémentaire ne vérifie pas nécessairement la propriété de Golomb.

Plan

1 SEND+MORE=MONEY

2 Problème des dames

3 Séquence magique

4 Règles de Golomb

5 Suite de Langford

Pour commencer

Exercice 9

Trouver une séquence de huit nombres telle que :

- ▶ Chaque nombre i ($i = 1, \dots, 4$) apparaît exactement deux fois ;
- ▶ deux occurrences du nombre i sont à une distance i

Indice : commencez avec [1, 3].

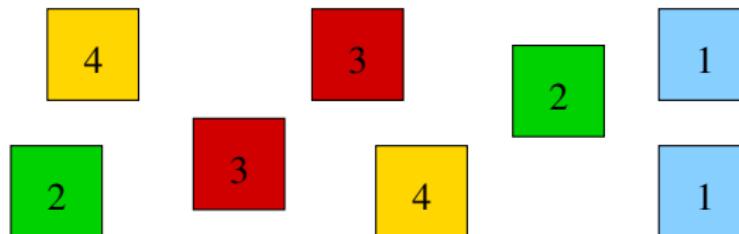


FIGURE: Empilez correctement les cubes. Solutions uniques de $L(2,3)$ et $L(2,4)$

Pour commencer

Exercice 9

Trouver une séquence de huit nombres telle que :

- ▶ Chaque nombre i ($i = 1, \dots, 4$) apparaît exactement deux fois ;
- ▶ deux occurrences du nombre i sont à une distance i

Indice : commencez avec [1, 3].

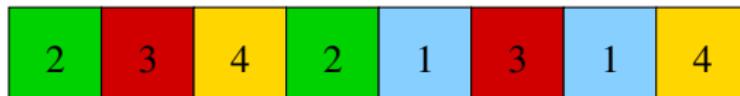
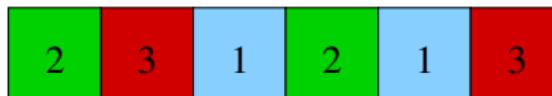


FIGURE: Empilez correctement les cubes. Solutions uniques de L(2,3) et L(2,4)

Énoncé du problème

Une suite de Langford $L(k, n)$ est :

- ▶ une séquence $k \times n$ nombres telle que
- ▶ chacune des k occurrences successives de $i \in [1, n]$
- ▶ soient séparées par une distance i .

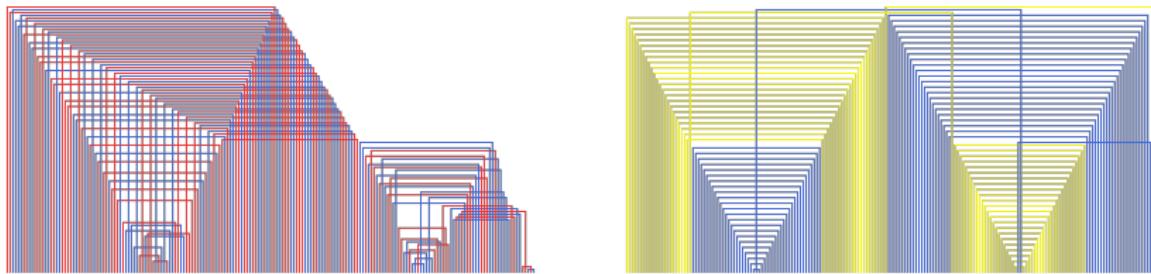


FIGURE: 2 solutions de $L(2, 96)$ issues de Langford's Problem Graphics Gallery.
Les traits bleus représentent les couples de nombres pairs et les traits rouges/jaunes représentent les couples de nombres impairs.

Modèle primal

$x_{ij} \in [1, k \times n]$ est la position dans la séquence de la j -ème occurrence du nombre i

$$\text{allDifferent}([x_{ij}]_{j \in [1, k], i \in [1, n]})$$

$$x_{ij+1} = x_{ij} + i + 1$$

Élimination des symétries

Exercice 10

- ① Est-ce que le problème a des symétries ?
- ② Proposer une méthode d'élimination des symétries pour $L(3, 9)$.

Élimination des symétries

Exercice 10

- ① Est-ce que le problème a des symétries ?
- ② Proposer une méthode d'élimination des symétries pour $L(3, 9)$.

Solution (Réflexion)

Non trivial : $X_{92} < 14 \vee (X_{92} = 14 \wedge X_{82} < 14)$.

- ▶ La 2ème occurrence de 9 est dans la première moitié.
- ▶ ou la 2ème occurrence de 9 est exactement au milieu et la 2ème occurrence de 8 est dans la première moitié),

Modèles duals

Permutation !

y_p représente le p -ème nombre de la séquence.

Modèles duals

Permutation !

y_p représente le p -ème nombre de la séquence.

Modèle dual 1

$y_p = j \times n + i$ si la j -ème occurrence de i occupe la position p .

`allDifferent($y_1, \dots, y_{k \times n}$)`

$$y_p = v \Rightarrow y_{p+(v \bmod n)+1} = v + n$$

Modèles duals

Permutation !

y_p représente le p -ème nombre de la séquence.

Modèle dual 1

$y_p = j \times n + i$ si la j -ème occurrence de i occupe la position p .

`allDifferent($y_1, \dots, y_{k \times n}$)`

$y_p = v \Rightarrow y_{p+(v \bmod n)+1} = v + n$

Modèle dual 2

$y_p = i$ si une occurrence du nombre i occupe la position p .

`globalCardinality([$y_1, \dots, y_{k \times n}$], [1, ..., n], [1, ..., k])`

$y_p = i \Rightarrow y_{p+i+1} = i$

Modèles combinés

Contraintes de liaison

Unicité de la solution du modèle primal et le modèle dual.

Modèles combinés

Contraintes de liaison

Unicité de la solution du modèle primal et le modèle dual.

Modèle Combiné 1

$$x_{ij} = p \Leftrightarrow y_p = j \times n + i$$

Modèle Combiné 2

$$x_{ij} = p \Leftrightarrow y_p = i$$

Lecture 3

Méthodes de résolution

Programmation par contraintes

29 octobre 2015

Arnaud Malapert
Département d'informatique
Université Nice Sophia Antipolis

Plan

- 1 Recherche systématique
- 2 Consistances locales
- 3 Contraintes globales
- 4 Algorithmes de recherche

Plan

- 1 Recherche systématique
- 2 Consistances locales
- 3 Contraintes globales
- 4 Algorithmes de recherche

Résolution d'un CSP

Complexité

- ▶ Prouver la consistance d'un CSP (satisfiabilité) est NP-complet.
- ▶ Exhiber une solution d'un CSP est NP-difficile dans le cas général.
 - ▶ Solution admissible ou optimale.

Résolution d'un CSP

Complexité

- ▶ Prouver la consistance d'un CSP (satisfiabilité) est NP-complet.
- ▶ Exhiber une solution d'un CSP est NP-difficile dans le cas général.
 - ▶ Solution admissible ou optimale.

Recherche systématique

Generate-and-Test vérification du viol d'une contrainte après instantiation de toutes les variables du problème.

Test-and-Generate (backtrack) vérification du viol d'une contrainte après instantiation de toutes les variables inhérentes à la contrainte.

Résolution d'un CSP

Complexité

- ▶ Prouver la consistance d'un CSP (satisfiabilité) est NP-complet.
- ▶ Exhiber une solution d'un CSP est NP-difficile dans le cas général.
 - ▶ Solution admissible ou optimale.

Recherche systématique

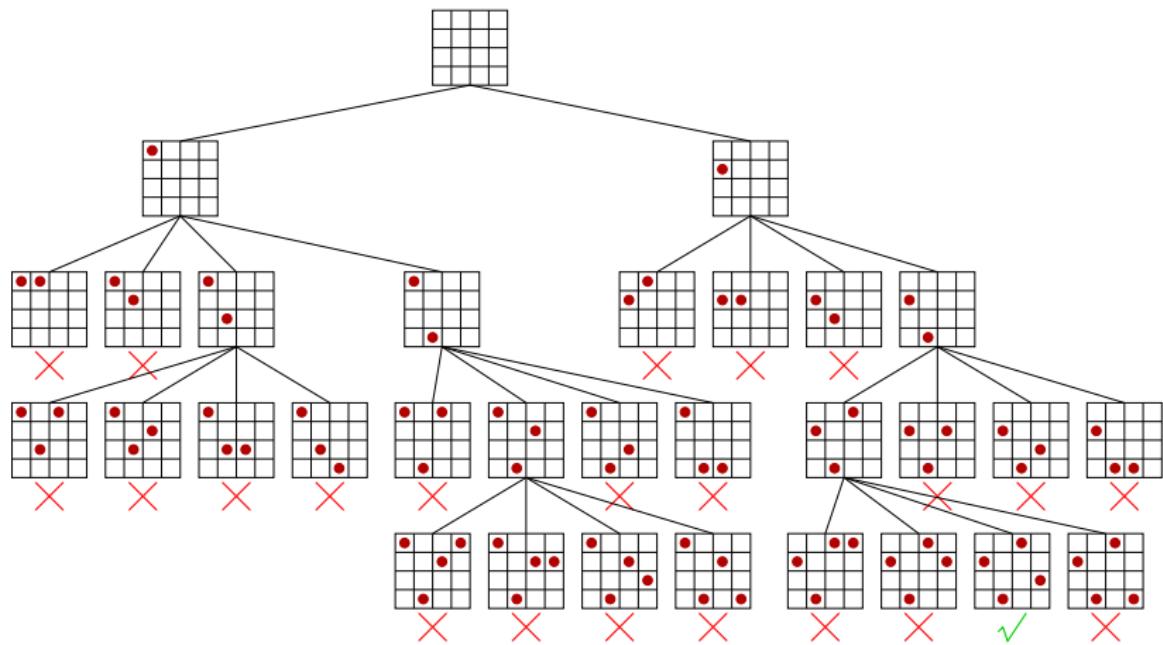
Generate-and-Test vérification du viol d'une contrainte après instantiation de toutes les variables du problème.

Test-and-Generate (backtrack) vérification du viol d'une contrainte après instantiation de toutes les variables inhérentes à la contrainte.

Exercice 11

Appliquer Test-and-Generate sur le problème des 3 dames.

Backtrack sur les 4 dames



En bref

Generate-and-Test

-  Toutes les affectations sont considérées
-  Les contraintes sont testées uniquement à chaque affectation totale.

Test-and-Generate

-  Moins d'affectations sont considérées.
-  Les contraintes sont testées à chaque affectation partielle.

En bref

Generate-and-Test

-  Toutes les affectations sont considérées
-  Les contraintes sont testées uniquement à chaque affectation totale.

Test-and-Generate

-  Moins d'affectations sont considérées.
-  Les contraintes sont testées à chaque affectation partielle.

-  Facile à mettre en œuvre.
-  **Complexité exponentielle.**
-  Échecs tardifs et découverte redondante d'inconsistance locale.
-  Perte d'informations triviales sur l'inconsistance d'affectation partielle.

Plan

1 Recherche systématique

2 Consistances locales

3 Contraintes globales

4 Algorithmes de recherche

Principes

Les contraintes sont utilisées activement pour supprimer les inconsistances.

- ▶ Une fonction `revise()` est associée à chaque contrainte.
- ▶ Différents niveaux de consistance possibles pour une même contrainte.

Calcul un **point fixe global** par des techniques de **consistance locale**.

- ▶ En général, les algorithmes de consistance sont **incomplets**.

Exemple : $x < y$

`revise()`

$$x_{max} \leftarrow y_{max} - 1 \quad (10)$$

$$y_{min} \leftarrow x_{min} + 1 \quad (11)$$

Exemple : $x < y$

`revise()`

$$x_{max} \leftarrow y_{max} - 1 \quad (10)$$

$$y_{min} \leftarrow x_{min} + 1 \quad (11)$$

Exemple

$$D_x = \{1, 2, 3\} \wedge D_y = \{1, 2, 3\}$$

$$(10) \Rightarrow D_x = \{1, 2\}$$

$$(11) \Rightarrow D_Y = \{2, 3\}$$

$$(10), (11) \Rightarrow \emptyset(\text{point fixe})$$

Exemple (BC) : $x = y + c$

revise()

$$x_{min} \leftarrow y_{min} + c \quad (12)$$

$$x_{max} \leftarrow y_{max} + c \quad (13)$$

$$y_{min} \leftarrow x_{min} - c \quad (14)$$

$$y_{max} \leftarrow x_{max} - c \quad (15)$$

Exemple (BC) : $x = y + c$

revise()

$$x_{min} \leftarrow y_{min} + c \quad (12)$$

$$x_{max} \leftarrow y_{max} + c \quad (13)$$

$$y_{min} \leftarrow x_{min} - c \quad (14)$$

$$y_{max} \leftarrow x_{max} - c \quad (15)$$

Exemple : $x = y + 6$

$$D_x = [1, 20] \quad \wedge \quad D_y = [1, 18]$$

$$(12) \Rightarrow D_x = [7, 20]$$

$$(15) \Rightarrow D_Y = [1, 14]$$

(12), (13), (14), (15) \Rightarrow point fixe

Exemple (AC) : $x = y + c$

`revise()`

$$\exists y \in D_y, (y + c) \notin D_x \Rightarrow y \notin D_y \quad (16)$$

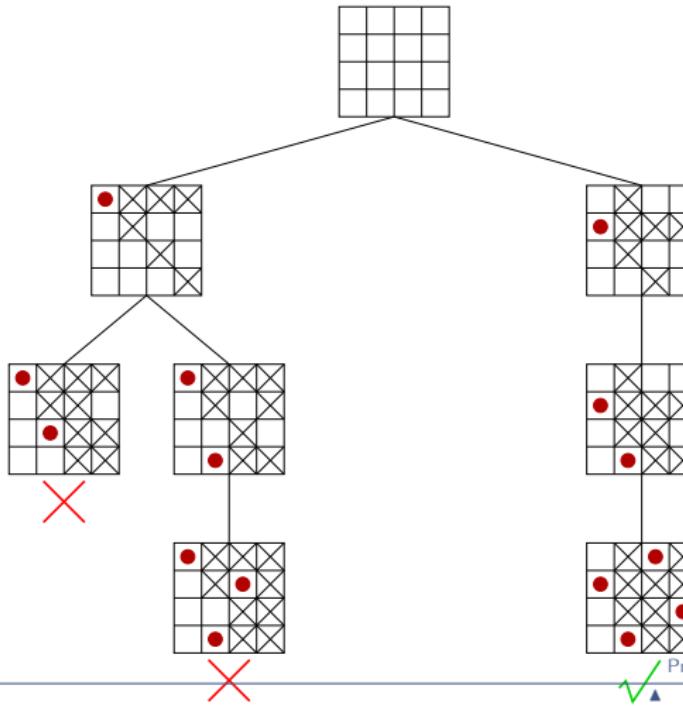
$$\exists x \in D_x, (x - c) \notin D_y \Rightarrow x \notin D_x \quad (17)$$

Exercice 12

Proposer un algorithme effectuant l'AC.

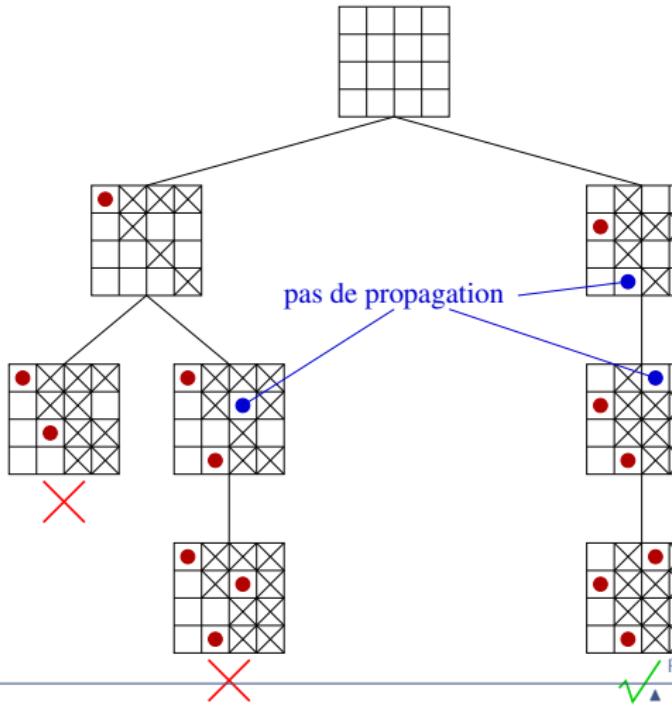
Consistance de nœud

Un **CSP** est dit **consistant de nœud** si pour toute variable X et pour toute valeur $v \in D_X$, l'affectation partielle (X, v) satisfait toutes les contraintes unaires.



Consistance de nœud

Un **CSP** est dit **consistant de nœud** si pour toute variable X et pour toute valeur $v \in D_X$, l'affectation partielle (X, v) satisfait toutes les contraintes unaires.



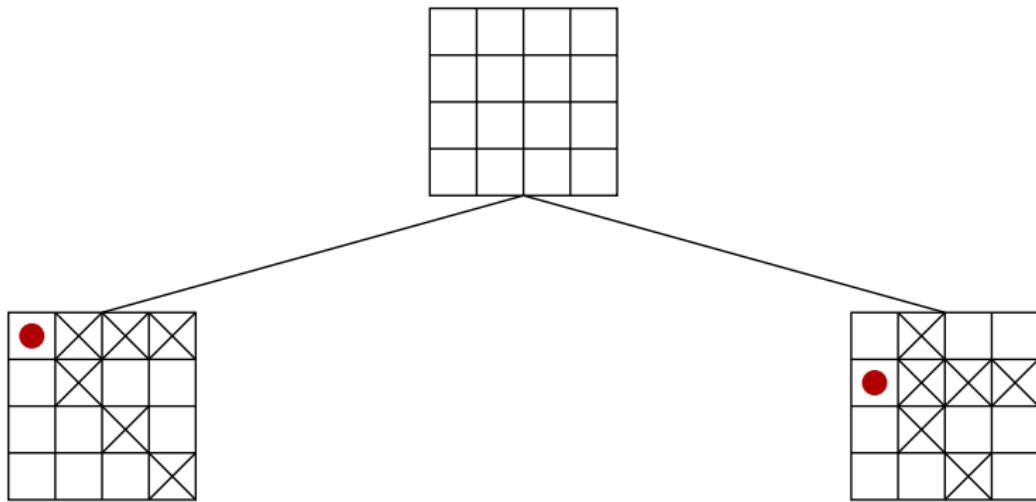
Arc-consistance

- ▶ Une **contrainte** est dite **arc-consistante** si pour chaque valeur de chaque variable, il existe une affectation des autres variables telle que la contrainte soit satisfaite
 - ▶ La valeur est dans le **support** de la contrainte.
- ▶ Un **CSP** est **arc-consistant** si toutes ses contraintes sont arc-consistantes.
 - ▶ Effectuer la révision de chaque contrainte une fois n'est pas suffisant.

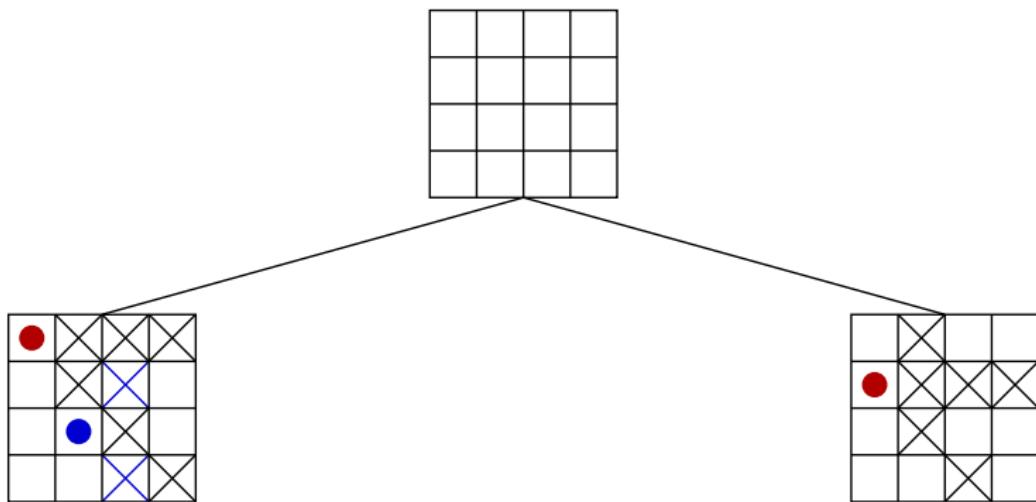
AC-1

Répéter la révision de toutes les contraintes jusqu'à ce qu'aucun domaine n'ait changé.

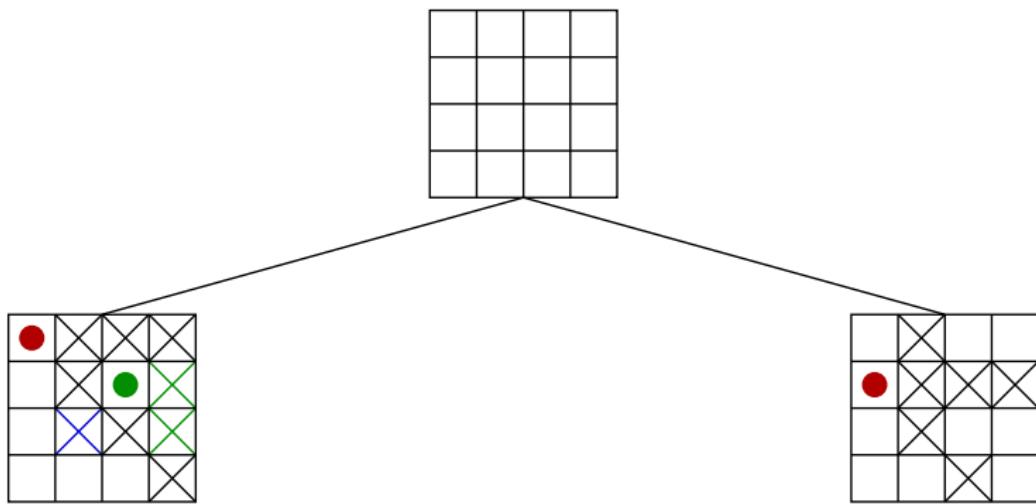
Arc-consistance : modèle binaire des 4 dames



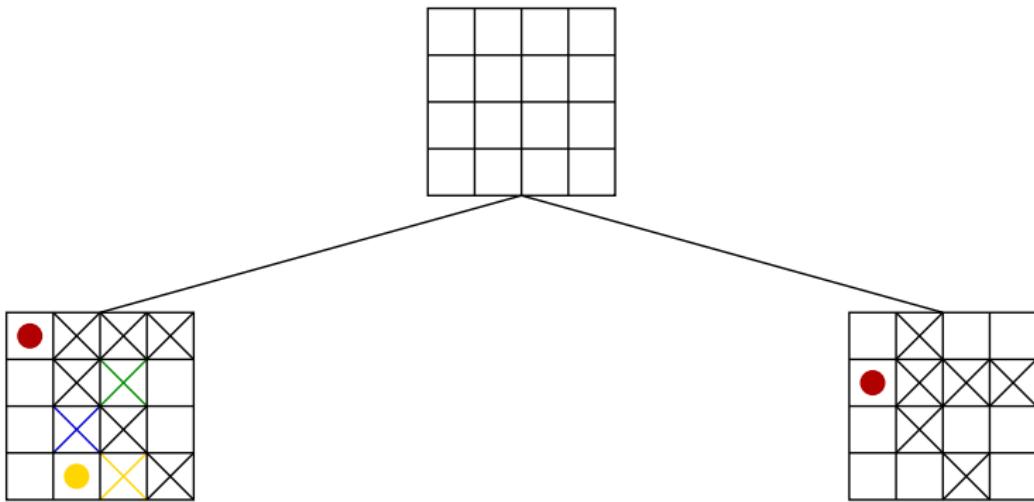
Arc-consistance : modèle binaire des 4 dames



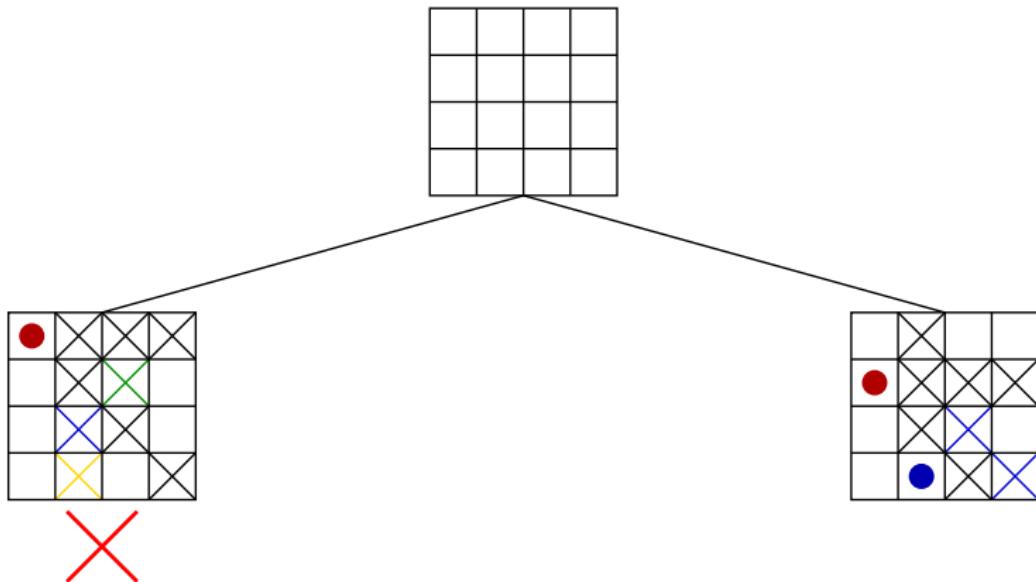
Arc-consistance : modèle binaire des 4 dames



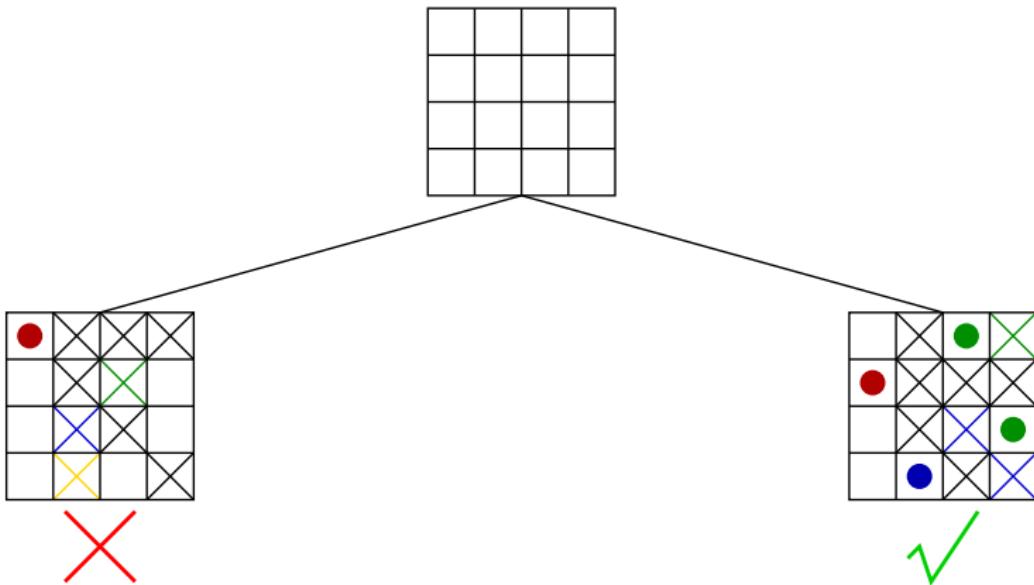
Arc-consistance : modèle binaire des 4 dames



Arc-consistance : modèle binaire des 4 dames



Arc-consistance : modèle binaire des 4 dames



Autres algorithmes d'AC

AC-3

- ▶ Utiliser une **queue de contraintes** qui doivent être révisées
- ▶ Quand le domaine d'une variable a changé, seules les contraintes portant sur cette variable sont ajoutées à la queue

Autres algorithmes d'AC

AC-3

- ▶ Utiliser une **queue de contraintes** qui doivent être révisées
- ▶ Quand le domaine d'une variable a changé, seules les contraintes portant sur cette variable sont ajoutées à la queue

En pratique, AC-3 -> AC-8

- ▶ Utiliser une **queue de variables** dont le domaine a changé.
- ▶ On filtre le réveil d'une contrainte en fonction d'**événements**
 - ▶ réduction, suppression, instanciation, ...

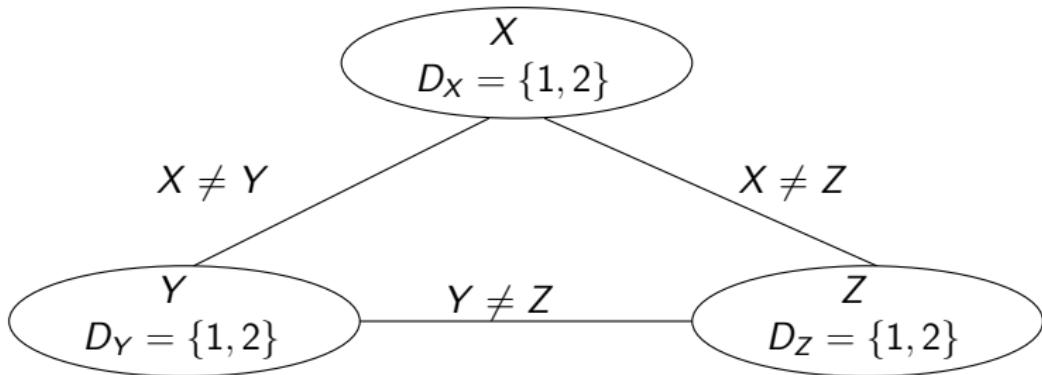
Consistance de bornes

un **CSP** est dit **A-consistant de bornes** pour une consistance donnée A si et seulement si pour chaque valeur x_{min} et x_{max} sont A -consistant.

- ▶ Variables avec de grands domaines.
- ▶ Algorithme de filtrage ne créant pas de trous.

AC est incomplète

AC ne détecte pas les inconsistances dues à un ensemble de contraintes.



Consistance de chemin

Les variables (X_0, X_1, \dots, X_m) sont consistantes de chemin si et seulement si pour chaque pair $x \in D_0$, $y \in D_m$ satisfaisant toutes les contraintes binaires entre X_0 et X_m , il existe une affectation des variables X_1, \dots, X_{m-1} telle que toutes les contraintes binaires X_i, X_{i+1} sont satisfaites.

- ▶ uniquement les contraintes binaires entre voisins
- ▶ l'exploration des chemins de longueur 2 est suffisante (Montanary, 1974)

Consistance de singleton

un **CSP** est dit **singleton A-consistant** pour une consistance donnée A si et seulement si pour chaque valeur $x \in D_X$ le problème $P_{|X=v|}$ est A -consistant.

- ▶ On supprime des valeurs uniquement
- ▶ Facile à mettre en œuvre (meta-programmation).
- ▶ Lenteur d'exécution.

Plan

1 Recherche systématique

2 Consistances locales

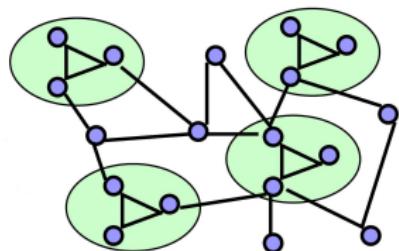
3 Contraintes globales

4 Algorithmes de recherche

Un monde global

Les contraintes agissent sur de petits ensembles de variables.

- 😊 contraintes hétérogènes ⇒ aide à la modélisation.
- 😢 pas de vue globale ⇒ réduction des domaines affaiblie.



Un monde global

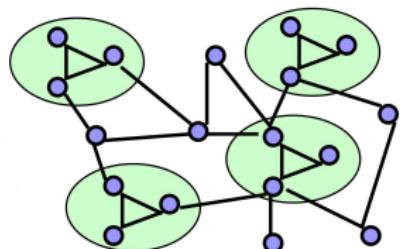
Les contraintes agissent sur de petits ensembles de variables.



contraintes hétérogènes ⇒ aide à la modélisation.



pas de vue globale ⇒ réduction des domaines affaiblie.

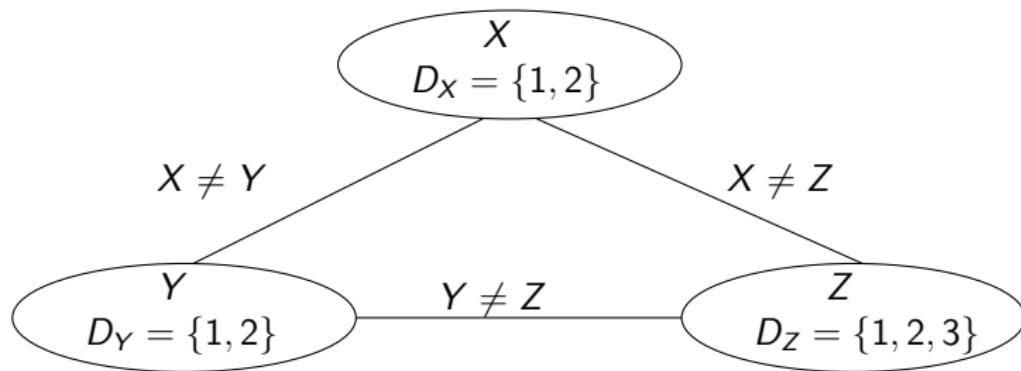


Contrainte globale

- ▶ Raisonnement sur un sous-problème ou
- ▶ utilisation d'information sémantique
- ▶ pour diminuer le temps de calcul ou
- ▶ augmenter l'efficacité du filtrage

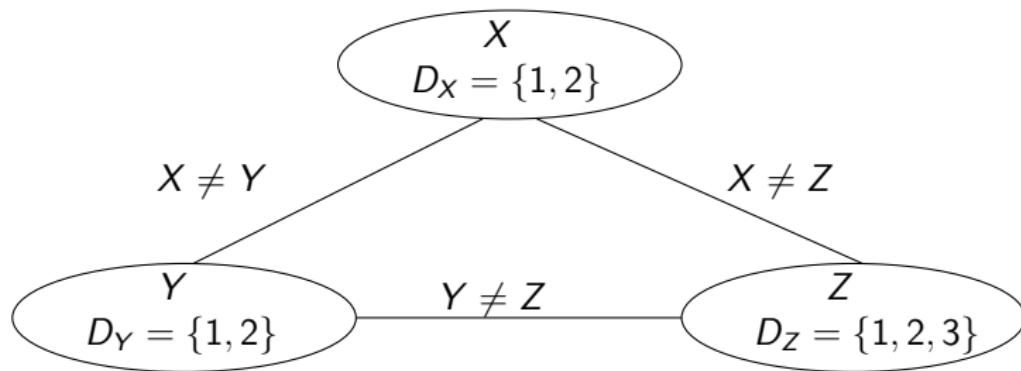
Un monde global

- ▶ AC ne déduit toujours rien ...
- ▶ PC supprime certaines valeurs des domaines au prix d'une lenteur certaine.
- ▶ On cherche un compromis ...



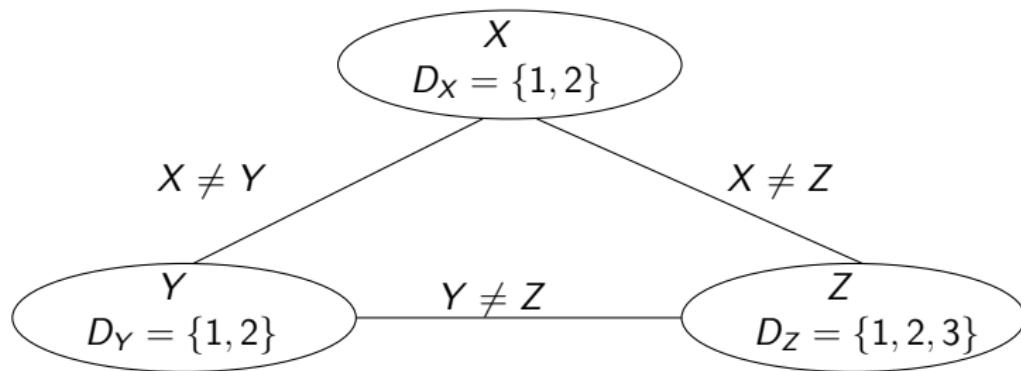
Un monde global

- ▶ AC ne déduit toujours rien ...
- ▶ PC supprime certaines valeurs des domaines au prix d'une lenteur certaine.
- ▶ On cherche un compromis ...



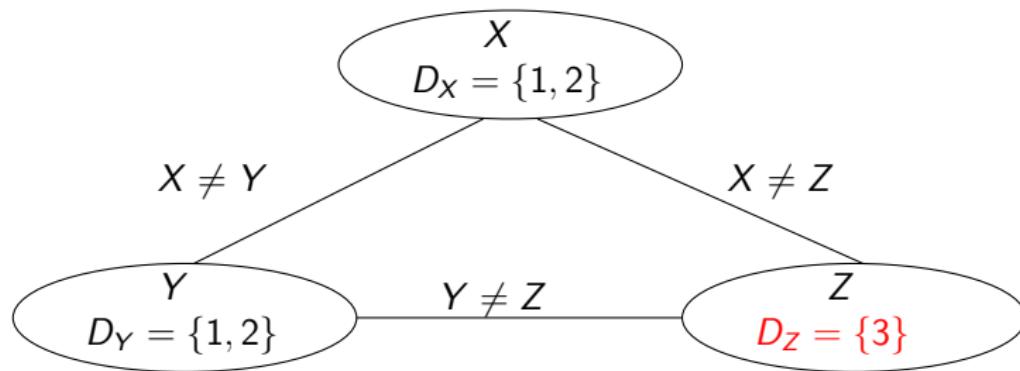
Un monde global

- ▶ AC ne déduit toujours rien ...
- ▶ PC supprime certaines valeurs des domaines au prix d'une lenteur certaine.
- ▶ On cherche un compromis ...



Un monde global

- ▶ AC ne déduit toujours rien ...
- ▶ PC supprime certaines valeurs des domaines au prix d'une lenteur certaine.
- ▶ On cherche un compromis ...



Rappel de théorie des graphes

Couplage

Étant donné, un graphe simple $G = (X, U)$ non orienté, un couplage est un ensemble d'arêtes $K \subseteq U$ tel que deux arêtes quelconques de U ne sont pas adjacentes.

Graphe biparti

Un graphe $G = (X, U)$ est dit biparti si l'ensemble des sommets X peut être partitionné en deux sous-ensembles X_1 et X_2 de telle sorte que, pour toute arête $(i, j) \in U : i \in X_1 \Rightarrow j \in X_2$ ou $i \in X_2 \Rightarrow j \in X_1$.

Chaîne alternée

Une chaîne élémentaire de G dont les arêtes sont alternativement dans K et dans $U - K$.

Couplage maximal

Transfert le long d'un chaîne alternée L

Si chaque extrémité est

- ▶ ou bien est un sommet insaturé,
- ▶ ou bien est telle que l'unique arête de K lui soit incidente.

Alors, un transfert le long L qui donne un nouveau couplage :

- ▶ échange entre les arêtes de K et celle de $U - K$ dans L .

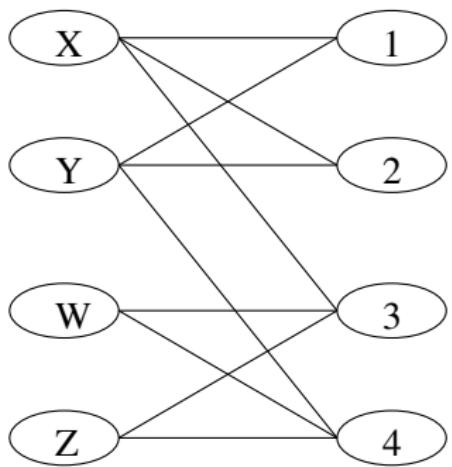
Chaîne alternée augmentante

Une chaîne alternée augmentante relie deux sommets insaturés de G .
Un transfert augmente la cardinalité du couplage d'une unité.

Couplage maximal

Un couplage K est maximal si et seulement si il n'existe pas de chaîne alternée augmentante.

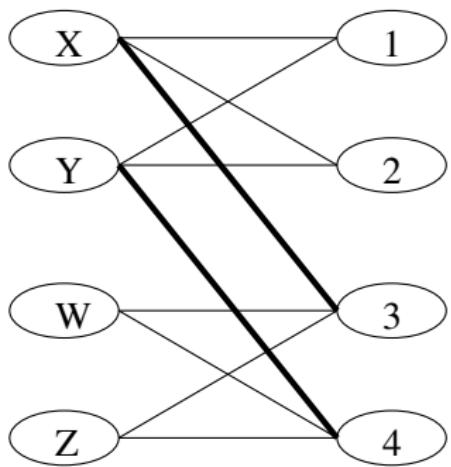
AllDifferent : filtrage



Principe

- ▶ Calculer un couplage maximal.

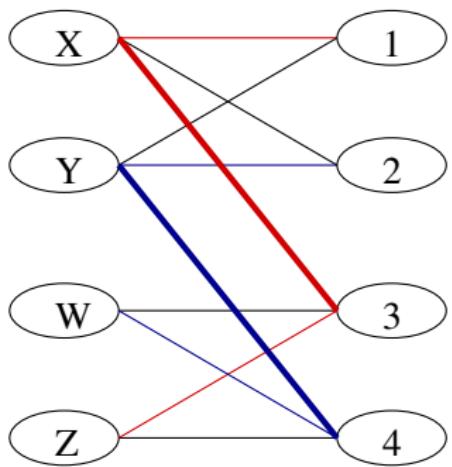
AllDifferent : filtrage



Principe

- ▶ Calculer un couplage maximal.

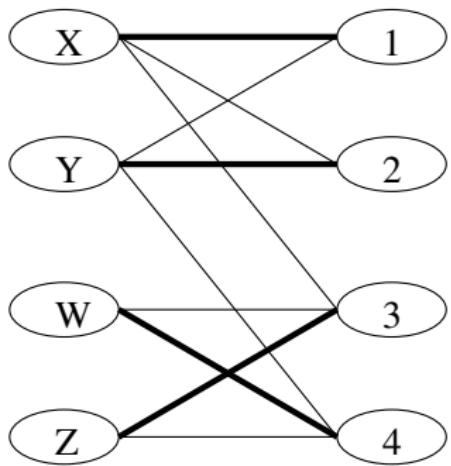
AllDifferent : filtrage



Principe

- ▶ Calculer un couplage maximal.

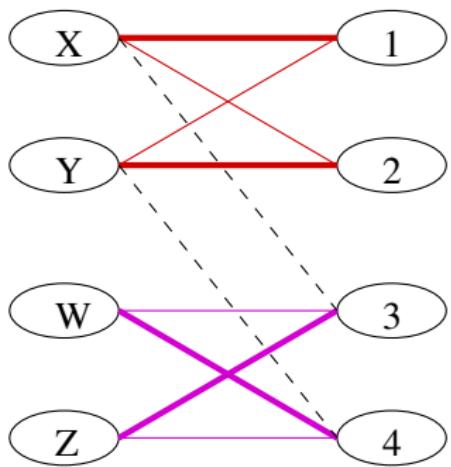
AllDifferent : filtrage



Principe

- ▶ Calculer un couplage maximal.

AllDifferent : filtrage



Principe

- ▶ Calculer un couplage maximal.
- ▶ enlever toutes les arêtes qui n'appartiennent à aucun couplage maximal.

Plan

- 1 Recherche systématique
- 2 Consistances locales
- 3 Contraintes globales
- 4 Algorithmes de recherche

Algorithme de recherche

Problème

- ▶ Les méthodes de recherche systématique sont inefficaces.
- ▶ Les techniques de propagation sont incomplètes.

Combinaison des deux

prospectif évitement des conflits, look ahead.
rétrospectif réparation des conflits, look back.

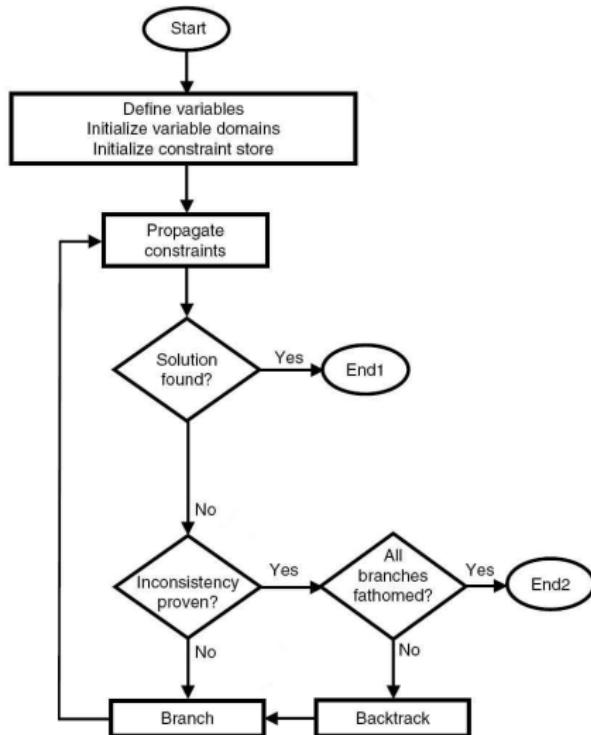
Résolution des disjonctions restantes (labeling)

standard labeling $X = v \vee X \neq v$, $v \in D_X$.

domain splitting $X < v \vee X \geq v$, $v \in D_X$.

variable ordering $X < Y \vee X \geq Y$.

Principe de l'algorithme de recherche



Heuristiques de sélection

Sélection de variable : first-fail

Choisir la variable dont l'affectation a la plus grande probabilité de provoquer un échec.

- ▶ plus petit domaine, plus contraintes, ...

Définit la **forme de l'arbre de recherche**.

Heuristiques de sélection

Sélection de variable : first-fail

Choisir la variable dont l'affectation a la plus grande probabilité de provoquer un échec.

- ▶ plus petit domaine, plus contraintes, ...

Définit la **forme de l'arbre de recherche**.

Sélection de valeur : succeed-first

Choisir la valeur qui a la plus grande probabilité d'appartenir à une solution.

- ▶ plus petite, plus grande, ...
- ▶ dépendant du problème

Définit l'**ordre d'exploration des branches**

Heuristiques de sélection

Sélection de variable : first-fail

Choisir la variable dont l'affectation a la plus grande probabilité de provoquer un échec.

- ▶ plus petit domaine, plus contraintes, ...

Définit la **forme de l'arbre de recherche**.

Sélection de valeur : succeed-first

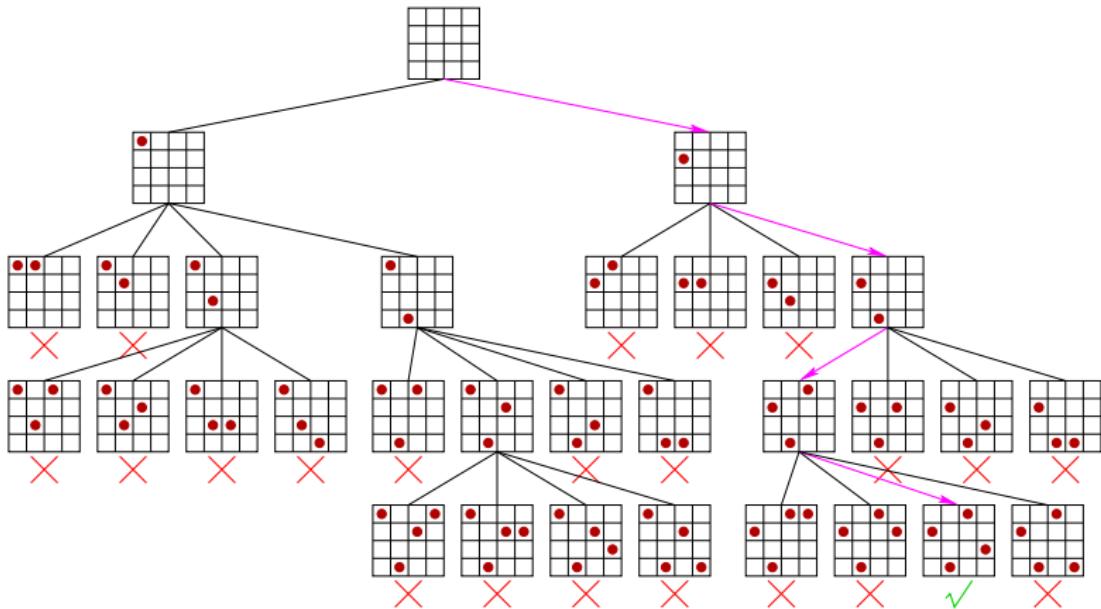
Choisir la valeur qui a la plus grande probabilité d'appartenir à une solution.

- ▶ plus petite, plus grande, ...
- ▶ dépendant du problème

Définit l'**ordre d'exploration des branches**

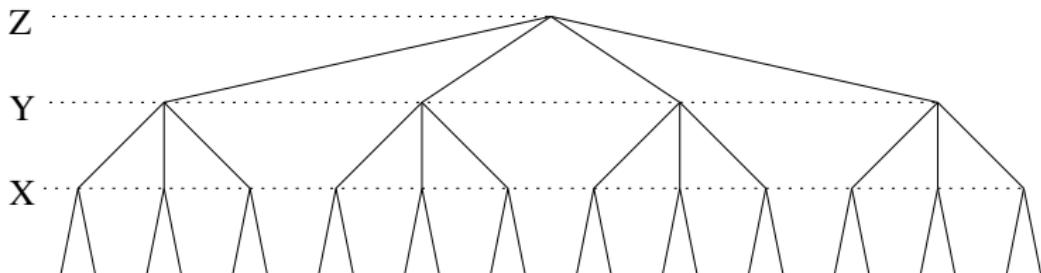
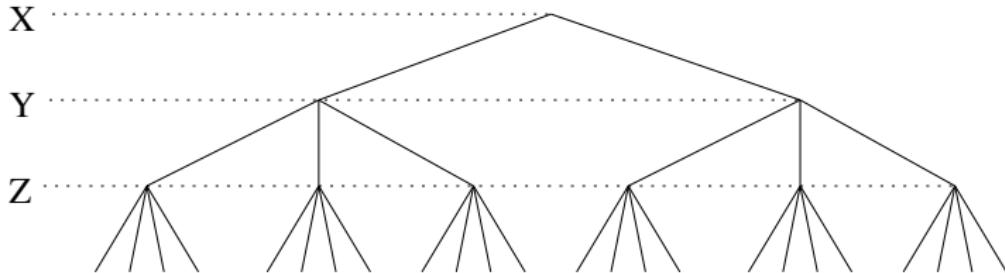
Moins d'informations au début de la recherche !

Oracle

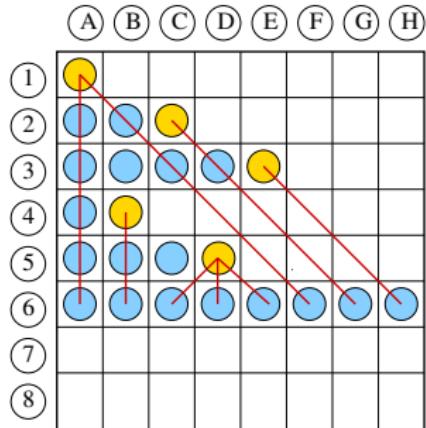
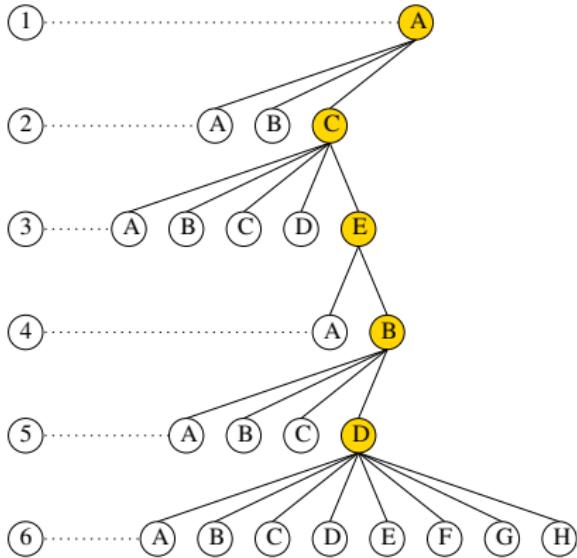


Exemples d'arbre de recherche

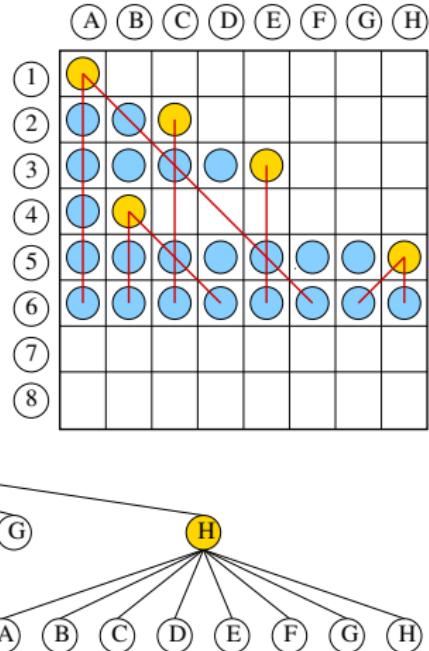
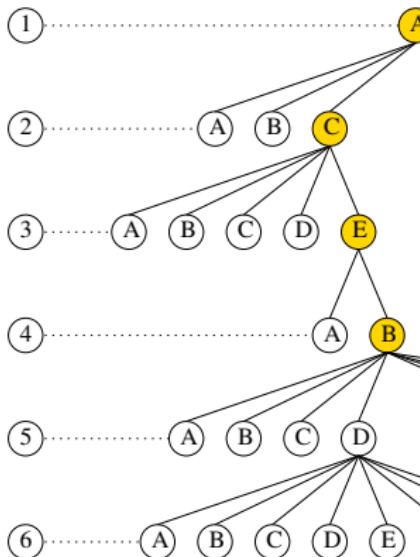
$$D_X = [1, 2] \wedge D_Y = [1, 3] \wedge D_Z = [1, 4]$$



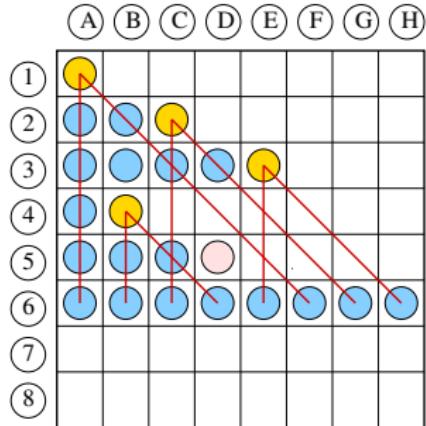
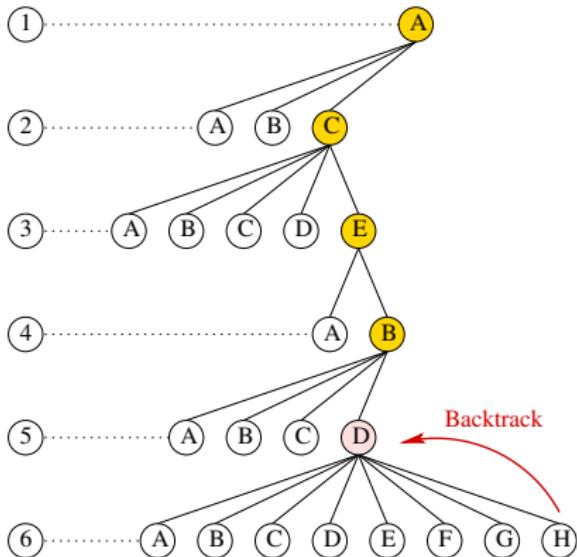
Trashing



Trashing



Trashing

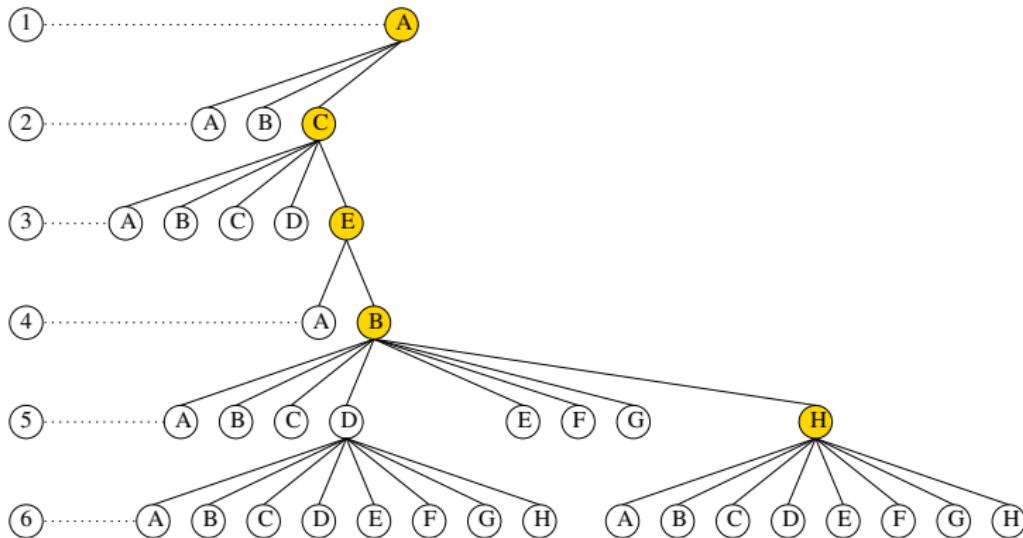


La reine sur la 5ème ligne n'entre pas dans le conflit.

Algorithmes de recherche rétrospectifs

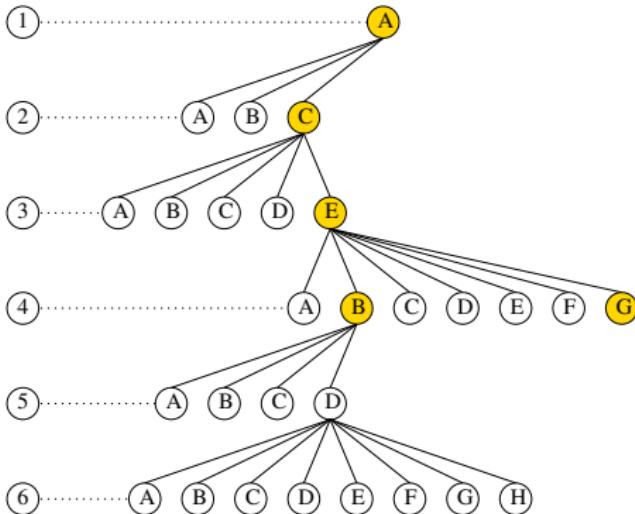
Backtrack chronologique

On revient sur l'avant-dernier choix.



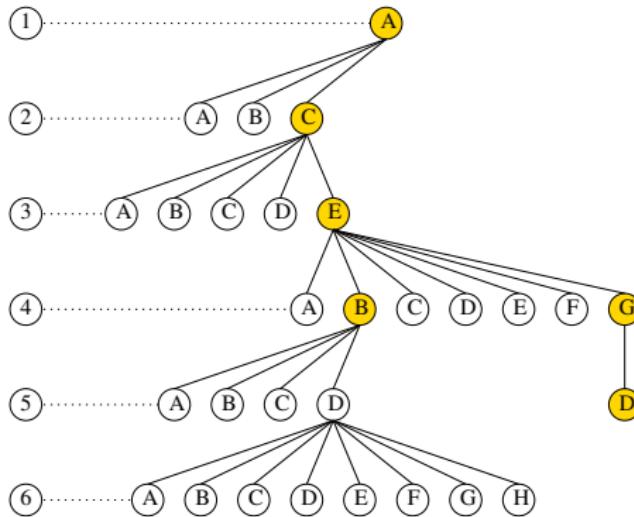
Conflit Directed Backjumping (Gaschnig, 1979)

- ▶ Chaque échec sur le choix d'une valeur est expliqué par les précédents choix qui entre en conflit.
- ▶ Si toutes les valeurs d'un domaine ont été testées sans succès. L'explication de cette échec est l'union des explications des valeurs du domaine.
- ▶ **On revient sur le choix le plus récent de cette explication.**



Dynamic backtracking (Ginsberg, 1993)

- ▶ On revient sur le choix le plus récent de l'explication
- ▶ sans remettre en cause ce qui ne dépend pas de ce choix.
- ▶ Une réparation plutôt qu'un saut.



Backmarking (Haralick & Elliot, 1980)

- ▶ mémoriser les explications (no-goods).
- ▶ propager ces explications (SAT)

Backmarking (Haralick & Elliot, 1980)

- ▶ mémoriser les explications (no-goods).
- ▶ propager ces explications (SAT)

Intérêts

- ▶ Explication des retraits (précieux pour la mise au point).
- ▶ Traitement des problèmes sur-contraints.
- ▶ Résolution interactive.

Algorithmes incomplets

Une **limite/cutoff** arrête l'exploration d'un (sous-)arbre.

- ▶ temps, backtracks, noeuds, ...

Recherche tronquée

- ▶ restreindre la profondeur de l'arbre.
- ▶ restreindre la largeur de l'arbre.

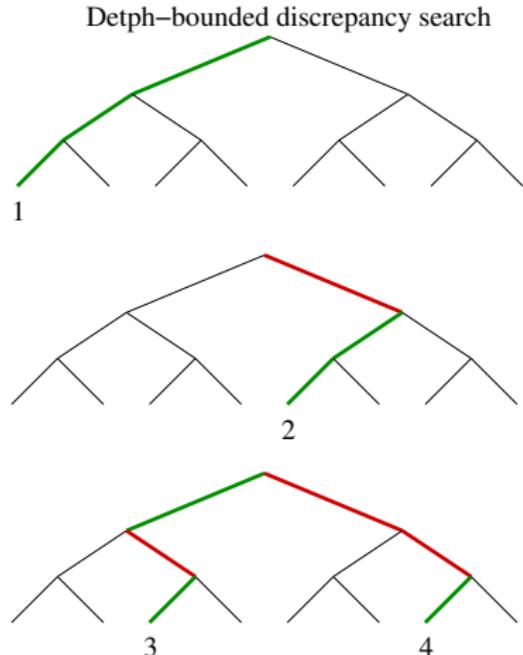
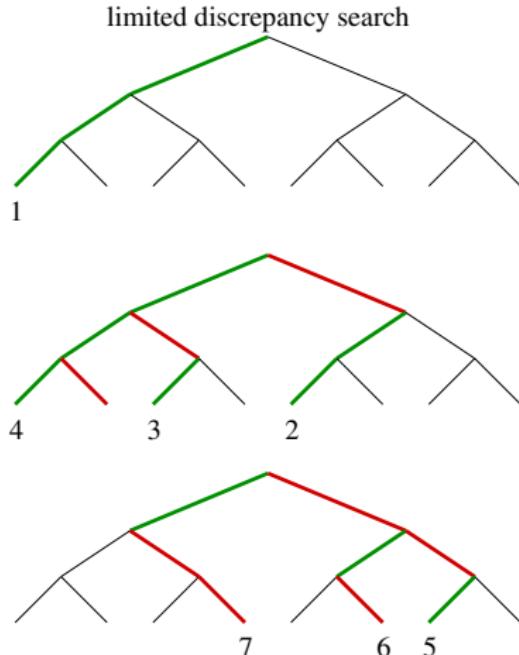
Redémarrages

Recommencer la recherche quand aucune solution n'est trouvée avec une limite agrandie (politique de redémarrage).

- ▶ Diversification : randomisation ou apprentissage.
- ▶ Mémoriser les conflits (backmarking).

Limited Discrepancies Search (LDS)

- ▶ une heuristique n'est pas infaillible,
- ▶ mais si l'heuristique est adaptée, on ne doit pas trop s'en éloigner



References I

Remerciements

Je me suis inspiré de nombreux cours glanés sur internet.
R. Debruyne, I. Gent, H. Simonis, P. Schaus, J. Hooker, ...



Barták, R. (1999).

Constraint Programming : In Pursuit of the Holy Grail.
Theoretical Computer Science, 17(12) :555–564.



Bordeaux, L., Hamadi, Y., and Zhang, L. (2005).

Propositional satisfiability and constraint programming : A comparative survey.
ACM Comput. Surv., 38 :2006.



Galinier, P., Jaumard, B., Morales, R., and Pesant, G. (2001).

A Constraint-Based Approach to the Golomb Ruler Problem.
In *Proceedings of the 3rd International Workshop on integration of AI and OR techniques*.



Gent, I. P. and Smith, B. M. (2000).

Symmetry Breaking in Constraint Programming.
In Horn, W., editor, *Proceedings ECAI 2000*, pages 599–603. IOS Press.

References II



Rossi, F., Beek, P. v., and Walsh, T. (2006).
Handbook of Constraint Programming (Foundations of Artificial Intelligence).
Elsevier Science Inc., New York, USA.



Smith, B. M., Stergiou, K., and Walsh, T. (1999).
Modelling the Golomb Ruler Problem.
Technical report.