

# Programmation Par Contraintes

## Cours 2 - Arc-Consistance et autres amusettes

David Savourey

CNRS, École Polytechnique

Séance 2

inspiré des cours de Philippe Baptiste, Ruslan Sadykov et de la thèse d'Hadrien  
Cambazard

# Sommaire

Arc-Consistance

Algos d'AC

Algorithmes prospectifs

Stratégies de branchement

Au delà de l'arc-consistance

# Sommaire

Arc-Consistance

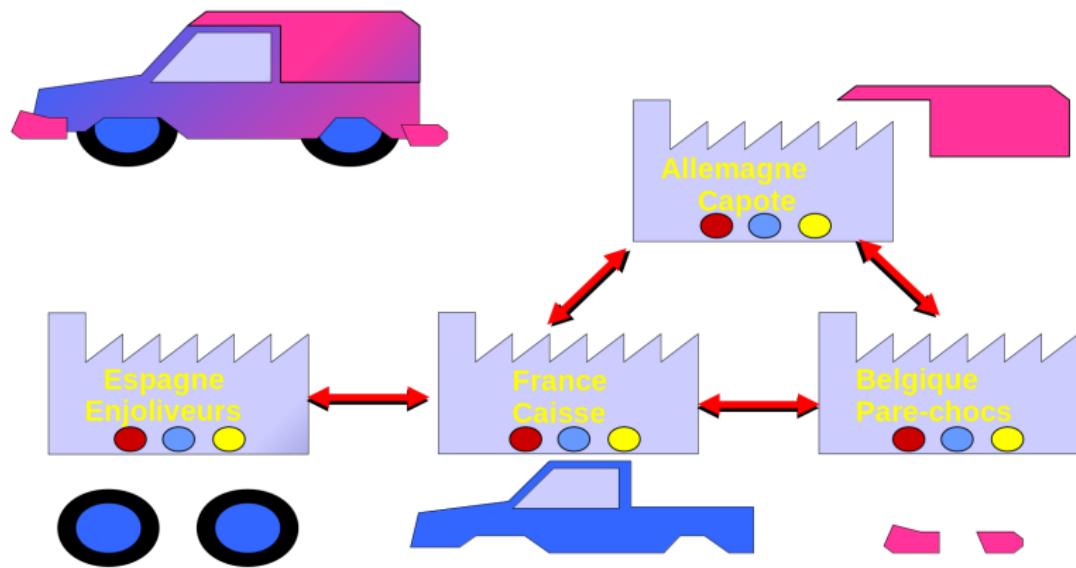
Algos d'AC

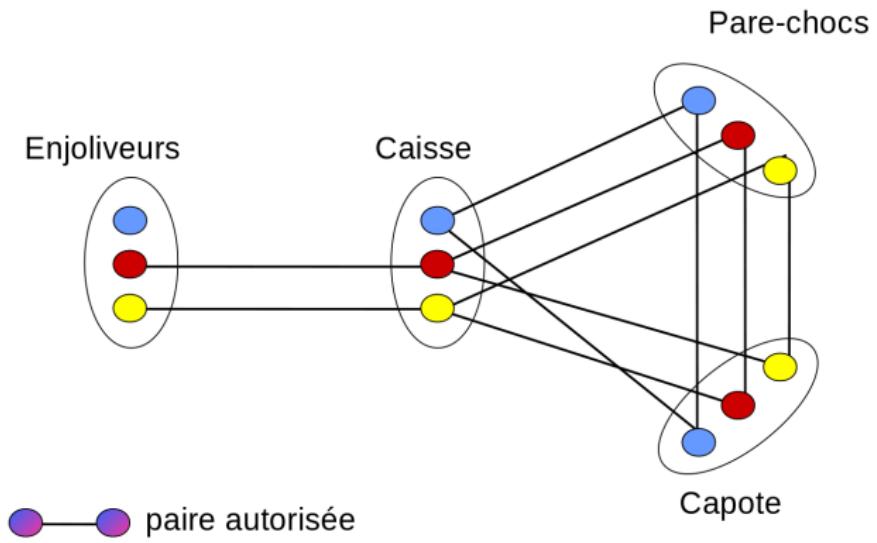
Algorithmes prospectifs

Stratégies de branchement

Au delà de l'arc-consistance

## Un exemple trivial





## CSP n-aires, CSP binaires

- ▶ on peut toujours binariser un CSP
- ▶ la plupart des techniques développées travaillent sur des CSP binaires
- ▶ dans la suite, on considérera des CSP binaires uniquement

# Un exemple de binarisation

Soit le CSP suivant :

- ▶  $A, B, C, D$
- ▶  $D_A = D_B = D_C = D_D = \{0, 1, 2\}$
- ▶  $A + B + C = 2, B + C + D = 5$

Version binaire équivalente :

- ▶  $A, B, C, D, X, Y$
- ▶  $D_A = D_B = D_C = D_D = \{0, 1, 2\},$   
 $D_X = \{011, 101, 110, 002, 020, 200\}$  et  $D_Y = \{122, 221, 212\}$
- ▶  $C(A, X) = \{(0, 011), (0, 002), (0, 020), (1, 101), (1, 110), (2, 200)\}$   
 $C(B, X) = \{(1, 011), (0, 002), (2, 020), (0, 101), (1, 110), (0, 200)\}$   
 $C(C, X) = \{(1, 011), (2, 002), (0, 020), (1, 101), (0, 110), (0, 200)\}$   
 $C(B, Y) = \{(1, 122), (2, 221), (2, 212)\}$   
 $C(C, Y) = \{(2, 122), (2, 221), (1, 212)\}$   
 $C(D, Y) = \{(2, 122), (1, 221), (2, 212)\}$

## Résolution naïve : Generate & Test



# Résolution naïve : Generate & Test



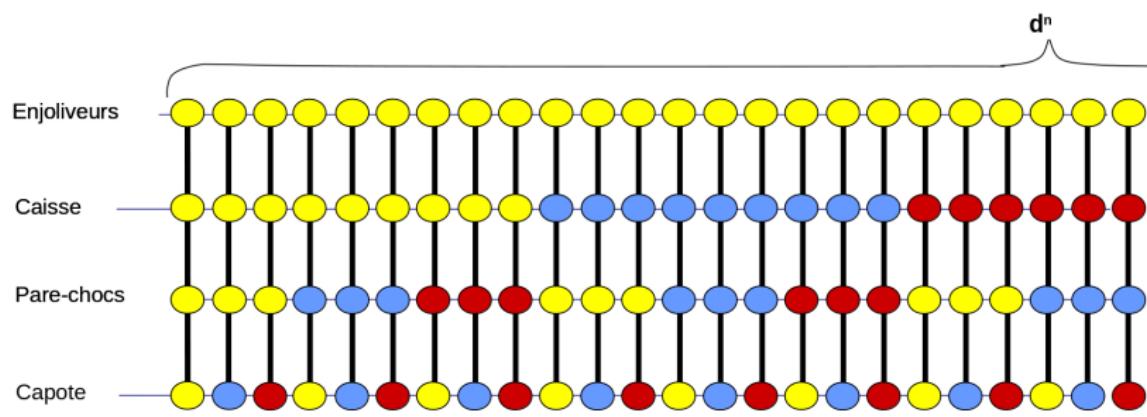
## Résolution naïve : Generate & Test



## Résolution naïve : Generate & Test



## Résolution naïve : Generate & Test



## Mieux : Backtrack

## Enjoliveurs

---

Digitized by srujanika@gmail.com

## Caisse

---

---

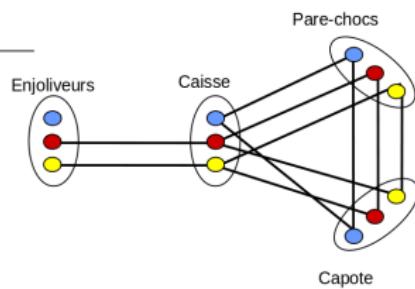
Pare-chocs

---

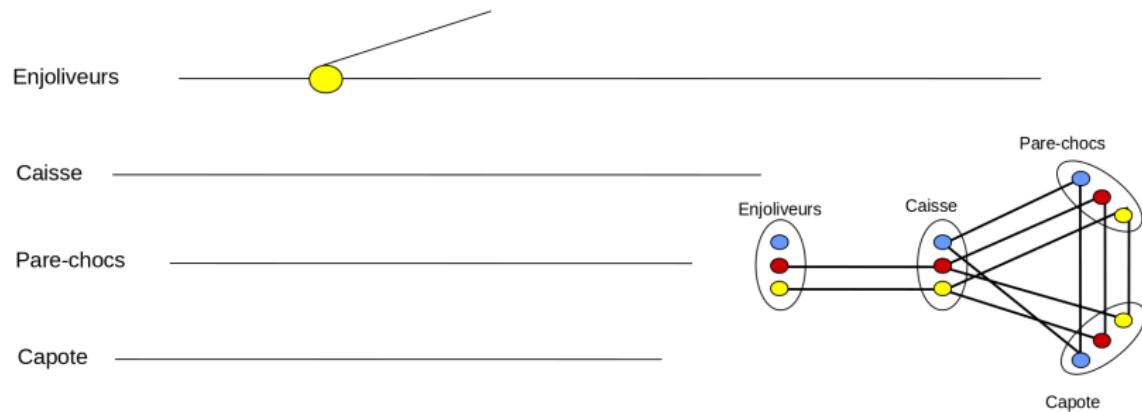
Capote

---

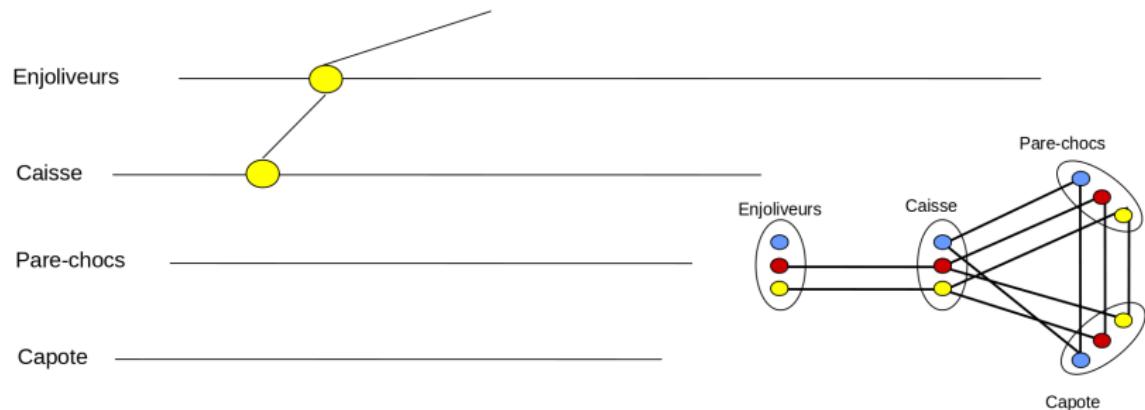
---



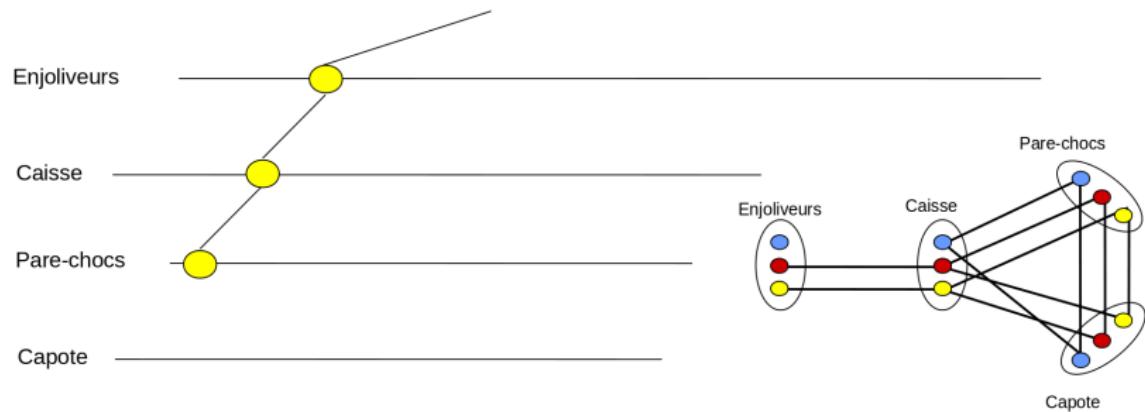
# Mieux : Backtrack



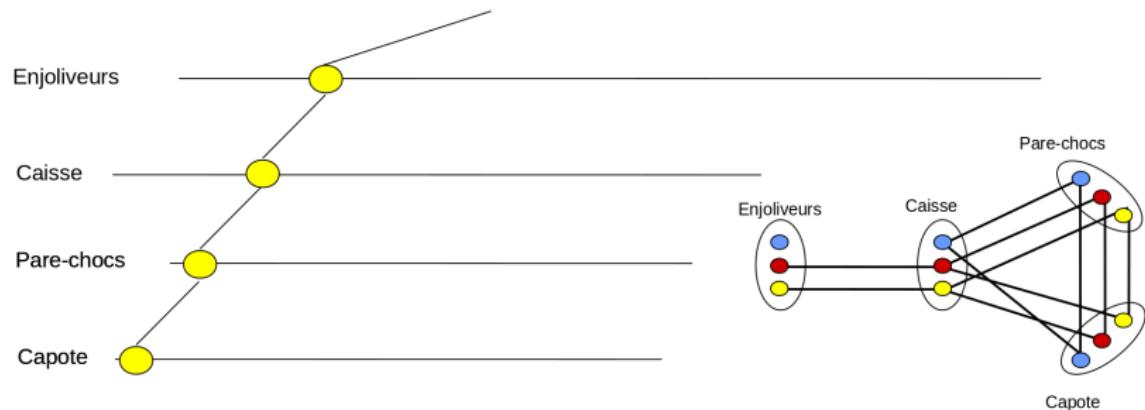
# Mieux : Backtrack



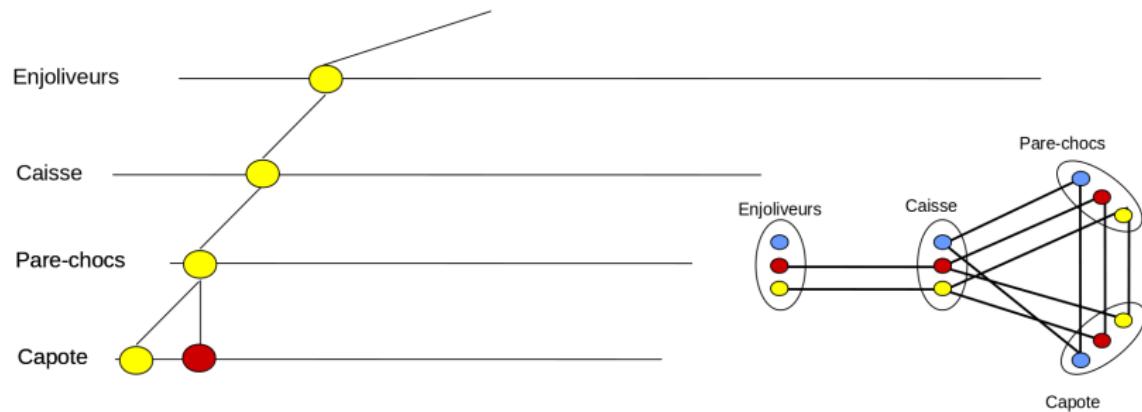
# Mieux : Backtrack



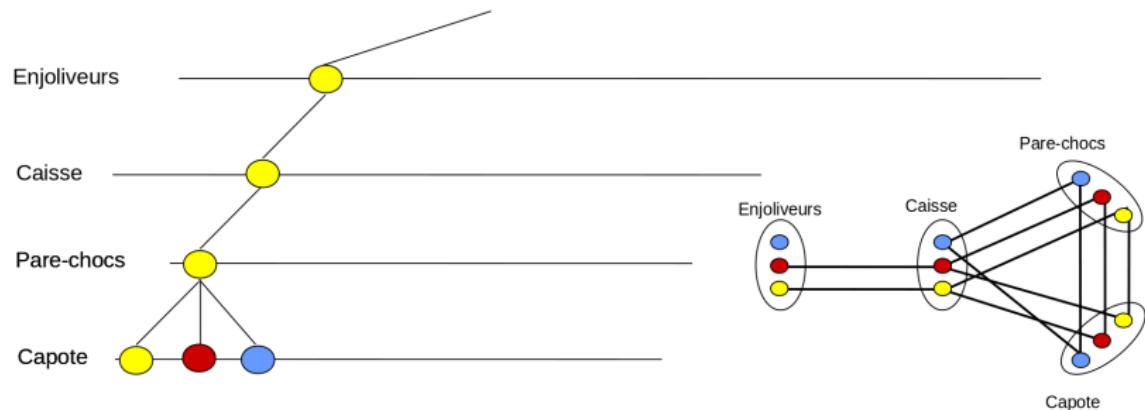
# Mieux : Backtrack



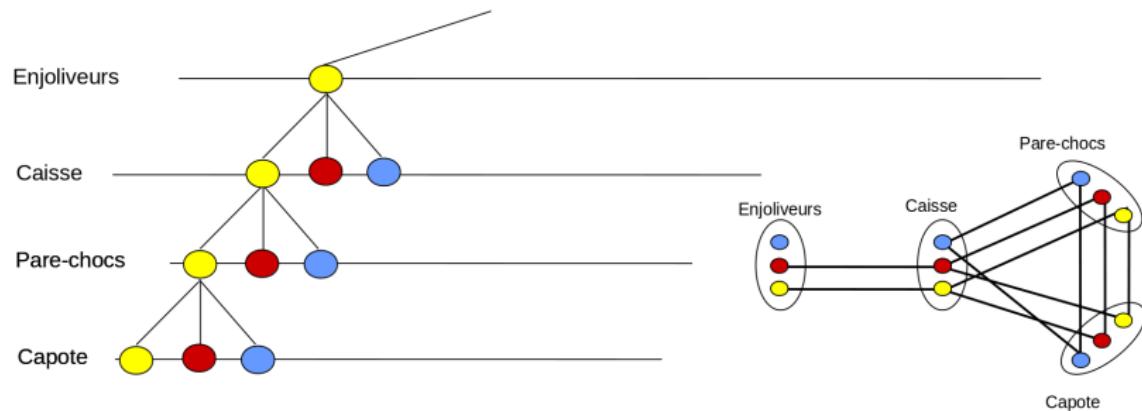
# Mieux : Backtrack



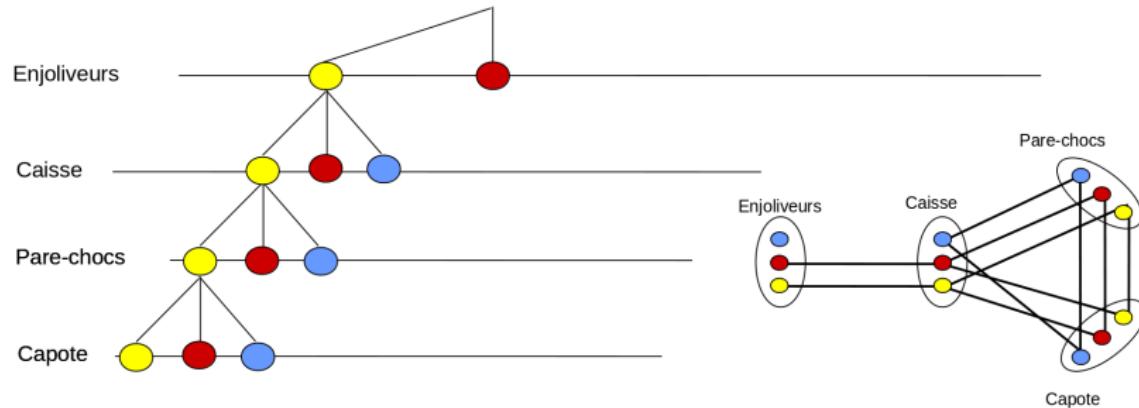
# Mieux : Backtrack



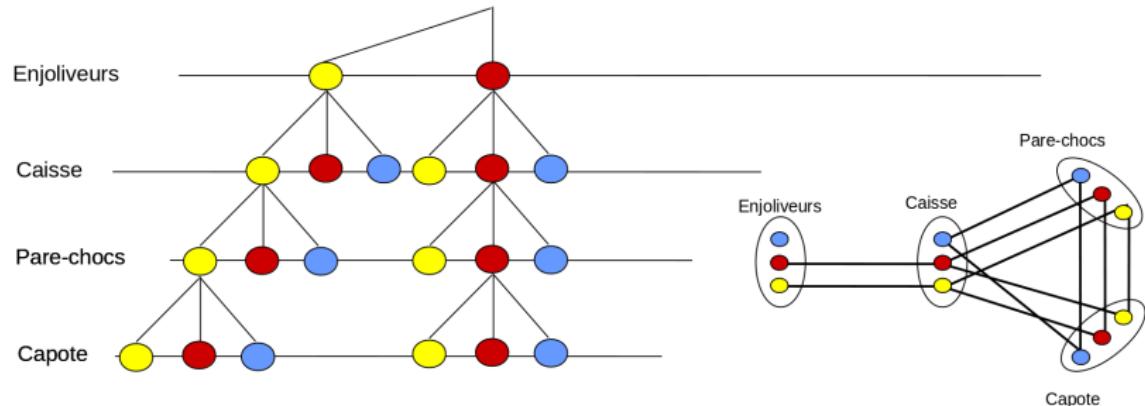
# Mieux : Backtrack



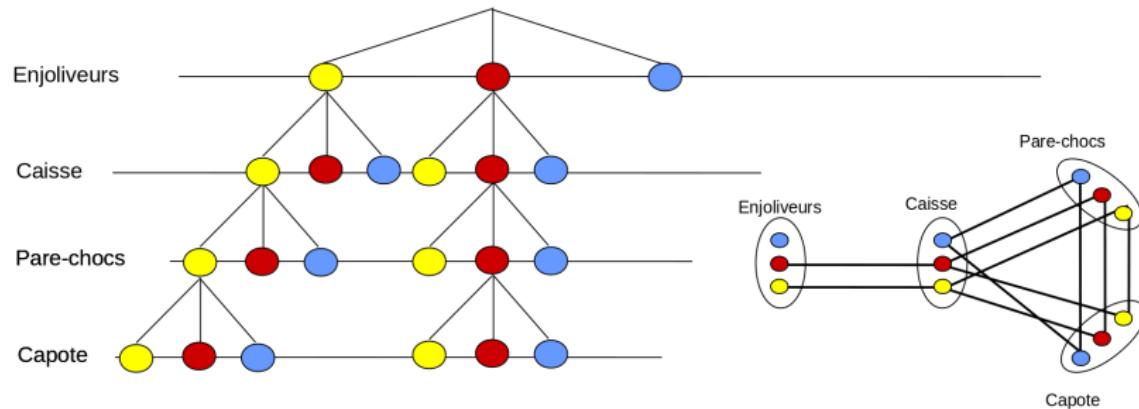
# Mieux : Backtrack



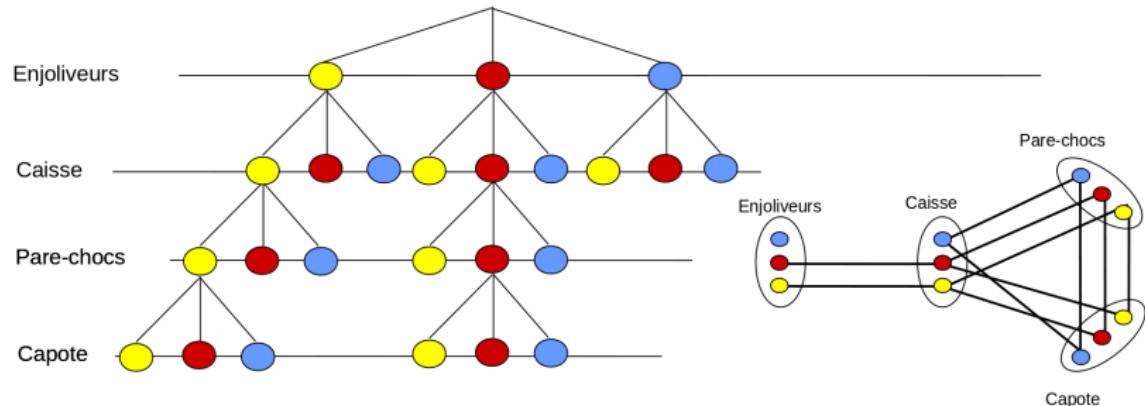
# Mieux : Backtrack



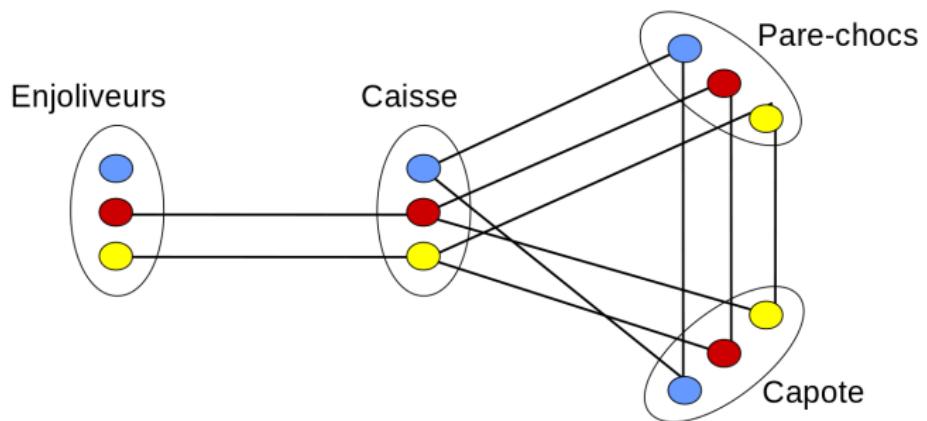
# Mieux : Backtrack



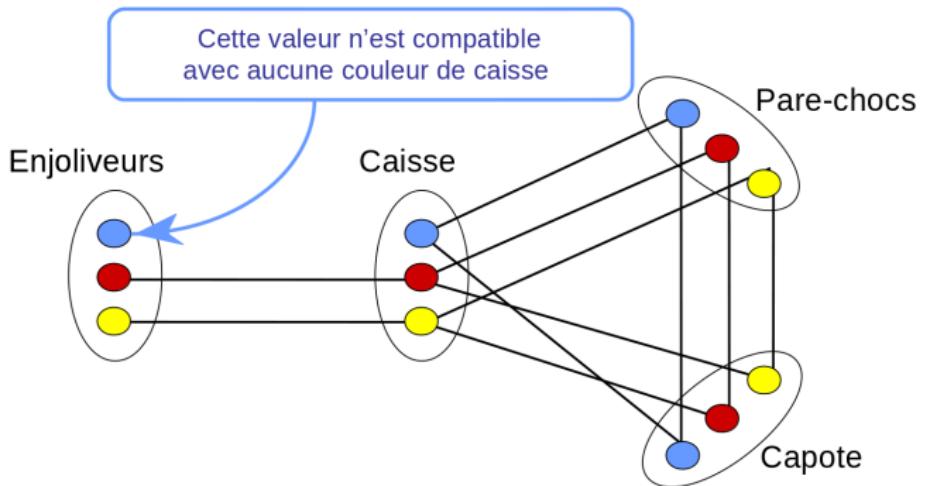
# Mieux : Backtrack



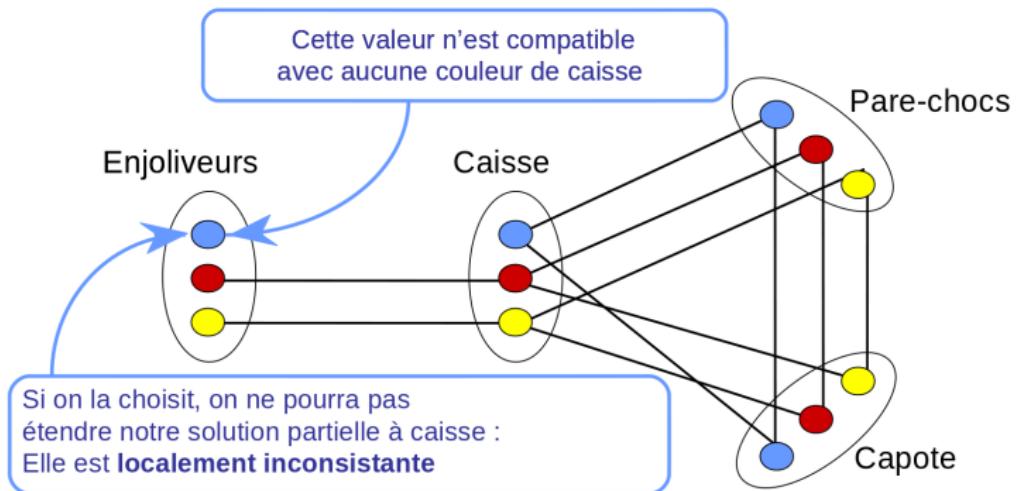
# Espace de recherche



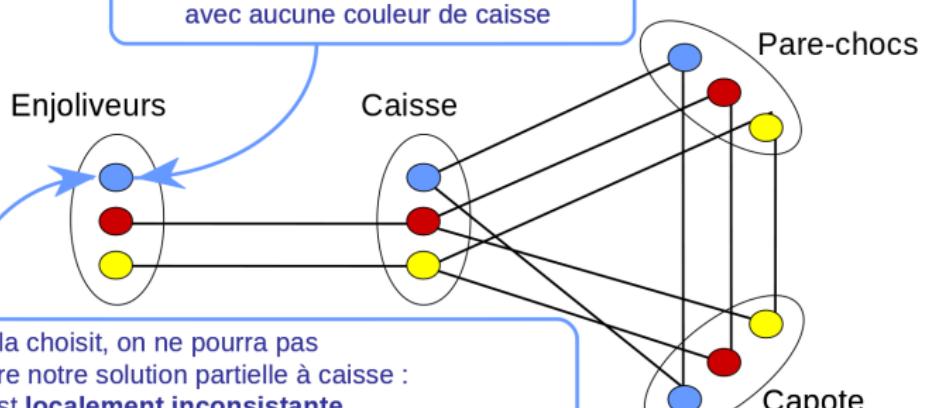
# Espace de recherche



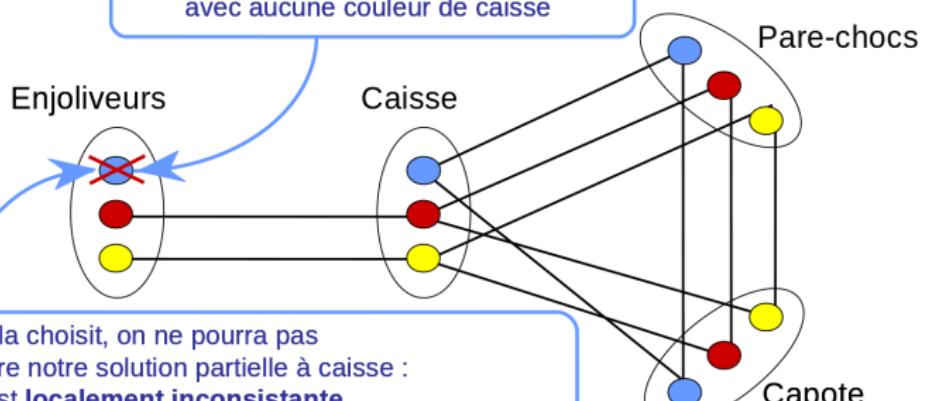
# Espace de recherche



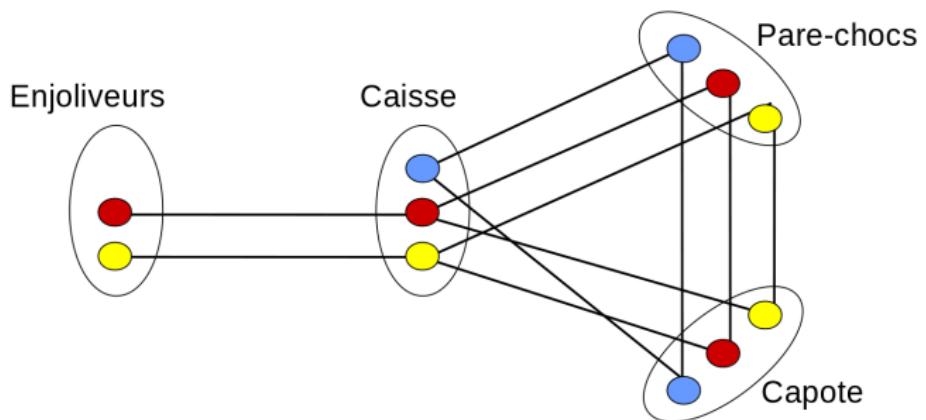
# Espace de recherche



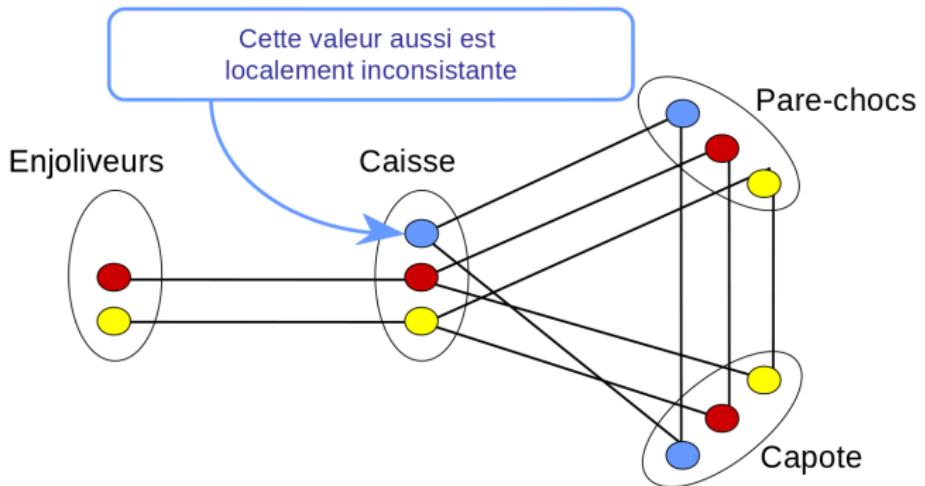
# Espace de recherche



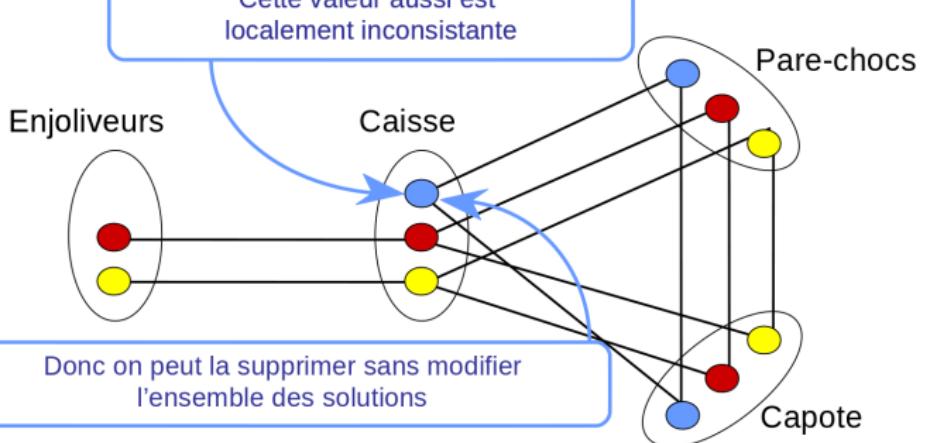
# Espace de recherche



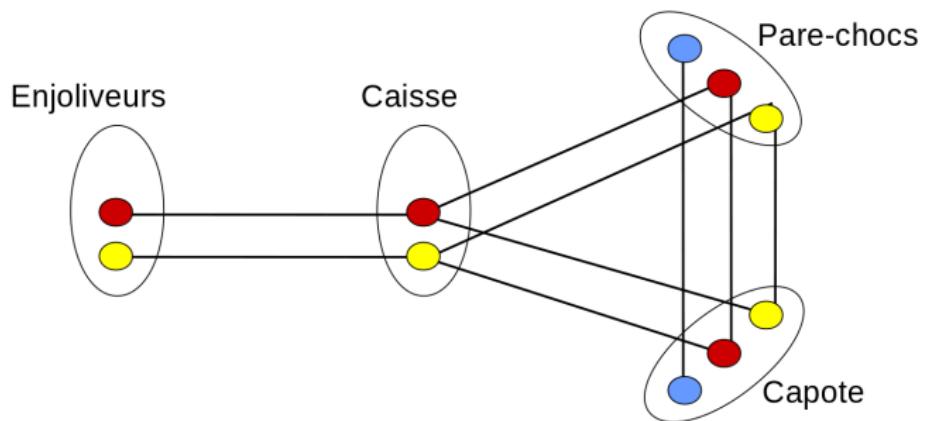
# Espace de recherche



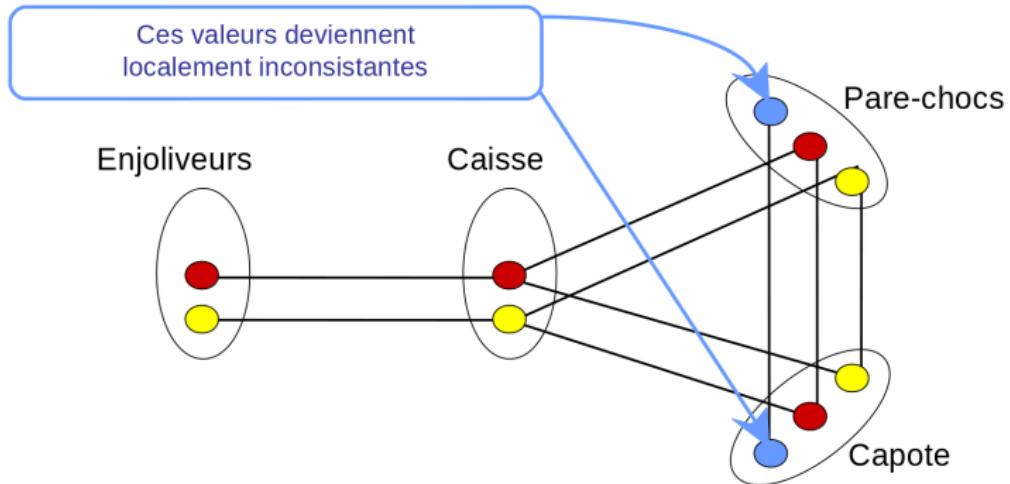
# Espace de recherche



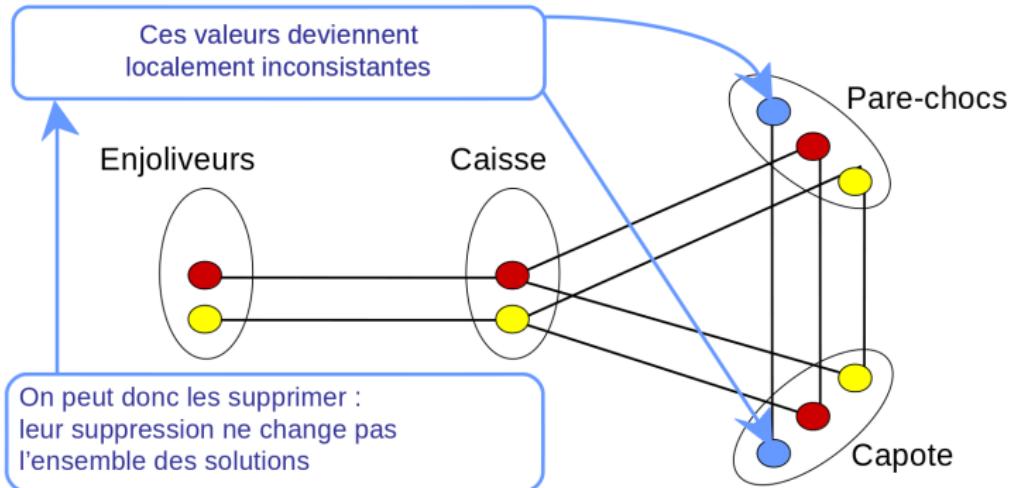
# Espace de recherche



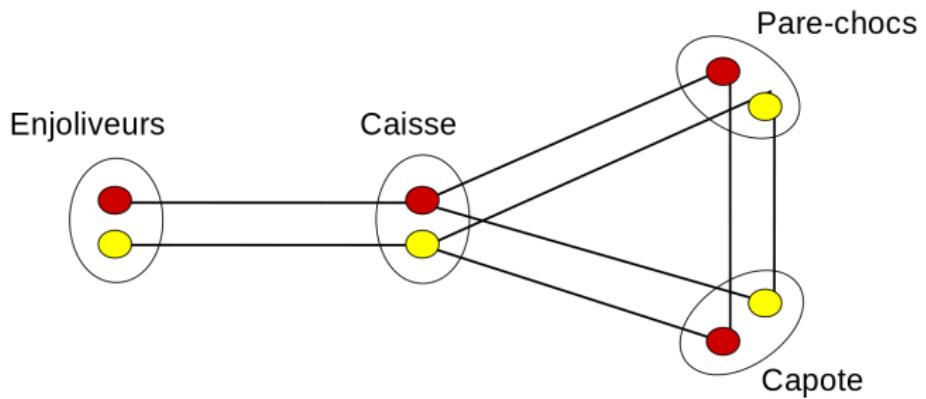
# Espace de recherche



# Espace de recherche

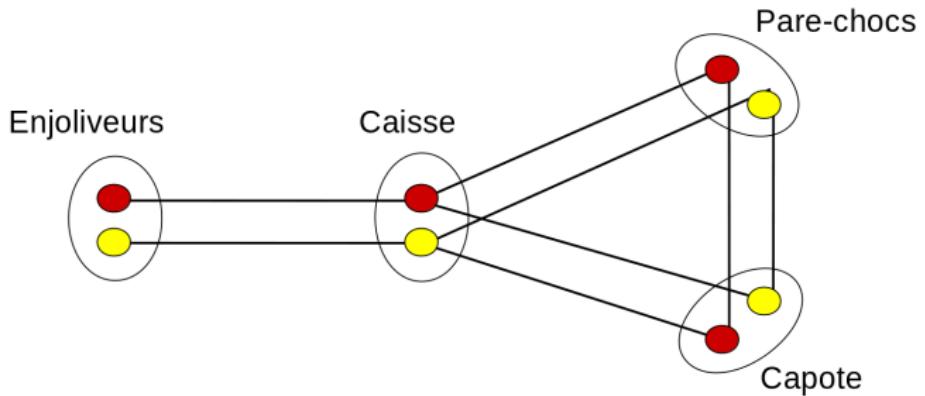


# Espace de recherche



# Espace de recherche

On n'a pas changé l'ensemble des solutions :  
On a un réseau de contraintes équivalent

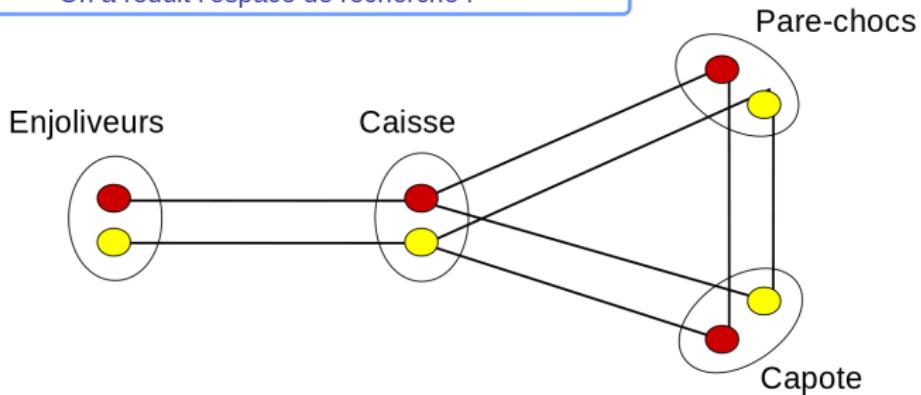


# Espace de recherche

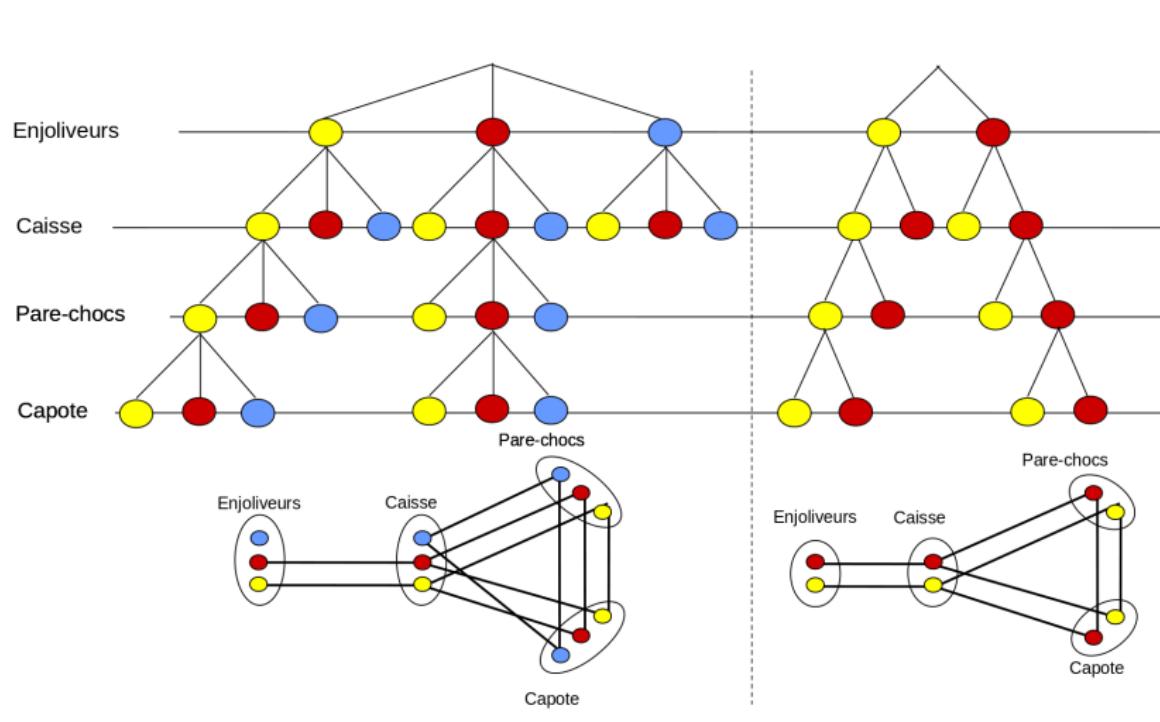
On n'a pas changé l'ensemble des solutions :

On a un réseau de contraintes **équivalent**

On a réduit l'espace de recherche !



# Arbre de recherche



## Consistance locale

- ▶ consistance globale = recherche de solutions
- ▶ consistance locale : on résout des sous-problèmes de taille fixée

Attention, dans la suite on considèrera les contraintes de manière directionnelle : une contrainte portant sur les variables  $x$  et  $y$  générera les deux contraintes  $C_{x,y}$  et  $C_{y,x}$ .

## Arc-Consistance : Définitions

- ▶ Nous venons de réaliser la fermeture arc-consistante de notre CSP
- ▶ La valeur  $a$  de la variable  $x$  est arc-consistante ssi elle possède au moins une valeur compatible (support) dans chaque domaine voisin.
- ▶  $\langle x, a \rangle$  est arc-consistante ssi  $\forall C_{x,y} \exists b \in D_y$  tq.  $C_{x,y}(a, b)$
- ▶  $\langle x, a \rangle$  est arc-inconsistante ssi  $\exists C_{x,y} \forall b \in D_y$  tq.  $\neg C_{x,y}(a, b)$
- ▶ une contrainte est arc-consistante si toutes les valeurs de ses variables sont arc-consistantes.
- ▶ un CSP est un arc-consistant si toutes ses contraintes sont arc-consistantes.

## Intérêt de l'arc-consistance

- ▶ réduit la combinatoire
- ▶ ne touche pas aux solutions valides
- ▶ il existe des algorithmes polynomiaux pour propager l'arc-consistance dans un CSP.

# Sommaire

Arc-Consistance

Algos d'AC

Algorithmes prospectifs

Stratégies de branchement

Au delà de l'arc-consistance

# Algorithmes d'arc-consistance

- ▶ AC-1
- ▶ AC-3
- ▶ AC-4
- ▶ AC-6, AC-2000
- ▶ tous font la même chose, mais plus ou moins vite.

On notera :

- ▶  $n$  : le nombre de variables
- ▶  $e$  : le nombre de contraintes
- ▶  $d$  : la taille du plus grand des domaines

---

## Algorithme 1 : AC1

---

**Données :** Un CSP  $\langle X, D, C \rangle$

*term*  $\leftarrow$  FAUX ;

**tant que** *term* = FAUX **faire**

*term*  $\leftarrow$  VRAI ;

**pour chaque** *contrainte*  $C_{x,y} \in C$  **faire**

Vérifier que chaque valeur pour la variable  $x$  est supportée  
par une valeur de la variable  $y$  ;

**si** une valeur  $v$  n'a pas de support **alors**

$D_x \leftarrow D_x \setminus \{v\}$  ;

*term*  $\leftarrow$  FAUX ;

# AC-1

- ▶ tant qu'on bouge quelque chose, on refait tout
- ▶ Complexité temporelle : ?

# AC-1

- ▶ tant qu'on bouge quelque chose, on refait tout
- ▶ Complexité temporelle :  $O(ned^3)$

---

**Algorithme 2 : AC3**

---

**Données :** Un CSP  $\langle X, D, C \rangle$

$aTester \leftarrow \{\}$  ;

**pour chaque** contrainte  $C_{x,y} \in C$  **faire**

$aTester \leftarrow aTester \cup \{(x, y)\}$  ;

**tant que**  $aTester$  non vide **faire**

Enlever un couple  $(x, y)$  de  $aTester$  ;

Vérifier que chaque valeur pour la variable  $x$  est supportée par une valeur de la variable  $y$  ;

**si** une valeur  $v$  n'a pas de support **alors**

$D_x \leftarrow D_x \setminus \{v\}$  ;

**pour chaque** contrainte  $C_{z,x} \in C$  avec  $z \neq y$  **faire**

$aTester \leftarrow aTester \cup \{(z, x)\}$  ;

- ▶ quand on supprime une valeur à la variable  $x$ , on ne réexamine que les variables liées à  $x$  par une contrainte
- ▶ complexité temporelle : ?

## AC-3

- ▶ quand on supprime une valeur à la variable  $x$ , on ne réexamine que les variables liées à  $x$  par une contrainte
- ▶ complexité temporelle :  $O(ed^3)$

## AC-4 : initialisation

### Algorithme 3 : initAC4

**Données :** Un CSP  $\langle X, D, C \rangle$

$Q \leftarrow \{\} ; S \leftarrow \{\} ;$

**pour chaque** *contrainte*  $C_{x,y} \in C$  **faire**

**pour chaque**  $a \in D_x$  **faire**

$total \leftarrow 0 ;$

**pour chaque**  $b \in D_y$  **faire**

**si**  $(a, b) \in C_{x,y}$  **alors**

$total \leftarrow total + 1 ;$

$S(\langle y, b \rangle) \leftarrow S(\langle y, b \rangle) \cup \{\langle x, a \rangle\} ;$

$Count(x, y, a) \leftarrow total ;$

**si**  $Count(x, y, a) = 0$  **alors**

$D_x \leftarrow D_x \setminus \{a\} ;$

$Q \leftarrow Q \cup \{\langle x, a \rangle\} ;$

Retourner  $Q$  ;

---

**Algorithme 4 : AC4**

---

$Q \leftarrow initAC4() ;$   
**tant que**  $Q$  non vide **faire**  
    Enlever un élément  $\langle y, b \rangle$  de  $Q$  ;  
    **pour chaque**  $\langle x, a \rangle \in S(\langle y, b \rangle)$  **faire**  
         $Count(x, y, a) \leftarrow Count(x, y, a) - 1$  ;  
        **si**  $Count(x, y, a) = 0$  et  $a \in D_x$  **alors**  
             $D_x \leftarrow D_x \setminus \{a\}$  ;  
             $Q \leftarrow Q \cup \{\langle x, a \rangle\} ;$

---

- ▶ on mémorise tous les supports des couples  $\langle x, a \rangle$ .
- ▶ on supprime une valeur  $a$  quand elle n'a plus de support.
- ▶ on propage à tous les couples  $\langle y, b \rangle$  que  $\langle x, a \rangle$  supportait.
- ▶ complexité temporelle : ?

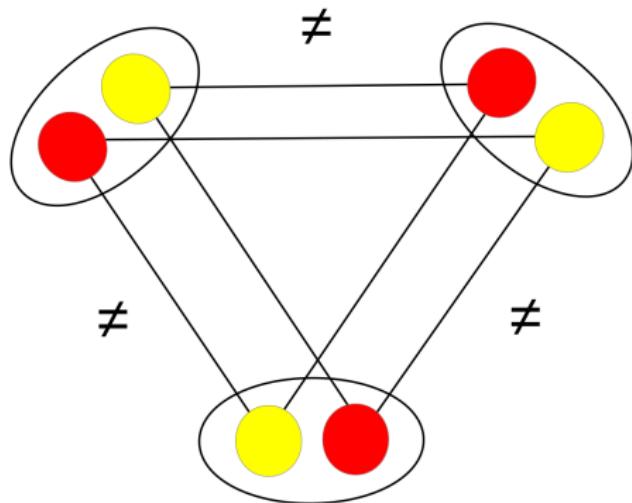
- ▶ on mémorise tous les supports des couples  $\langle x, a \rangle$ .
- ▶ on supprime une valeur  $a$  quand elle n'a plus de support.
- ▶ on propage à tous les couples  $\langle y, b \rangle$  que  $\langle x, a \rangle$  supportait.
- ▶ complexité temporelle :  $O(ed^2)$

## Bilan des algos

- ▶ il existe encore d'autres algos : AC-6, AC-2000, etc.
- ▶ AC-4 plus dur à implémenter qu'AC-1 et AC-3
- ▶ AC-3 et AC-4 sont les plus utilisés

## Attention!

L'arc-consistance est une consistance locale. Un CSP qui est arc-consistant peut être globalement inconsistent!



# Sommaire

Arc-Consistance

Algos d'AC

Algorithmes prospectifs

Stratégies de branchement

Au delà de l'arc-consistance

# Algorithmes prospectifs

- ▶ *Look-Ahead* in English
- ▶ idée : anticiper certaines prises de décisions dans le backtrack
- ▶ version light : Forward-Checking
- ▶ version lourde : Maintain-Arc-Consistency

# Forward-Checking

Dès qu'une variable  $x$  est instanciée à la valeur  $a$ , on filtre toutes les valeurs incompatibles avec  $\langle x, a \rangle$  **mais on ne propage pas plus.**

---

## Algorithme 5 : ForwardChecking

---

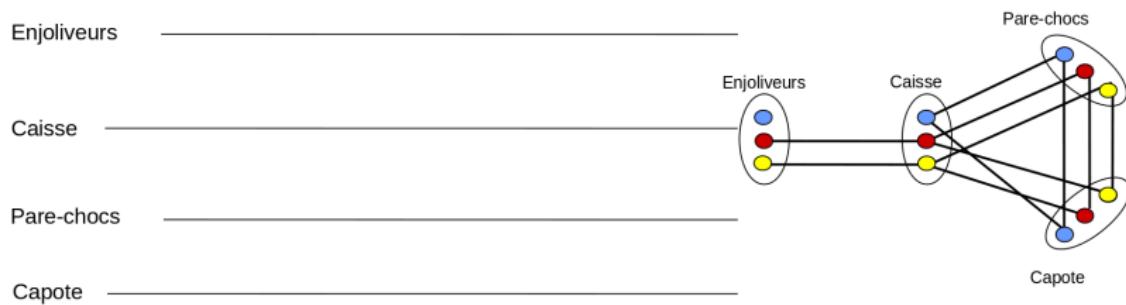
**Données :** Une instantiation partielle  $I$  où on vient de fixer  $\langle x, a \rangle$   
**pour chaque variable**  $y \notin I$  tq  $\exists C_{x,y}$  faire

**pour chaque valeur**  $b \in D_y$  faire

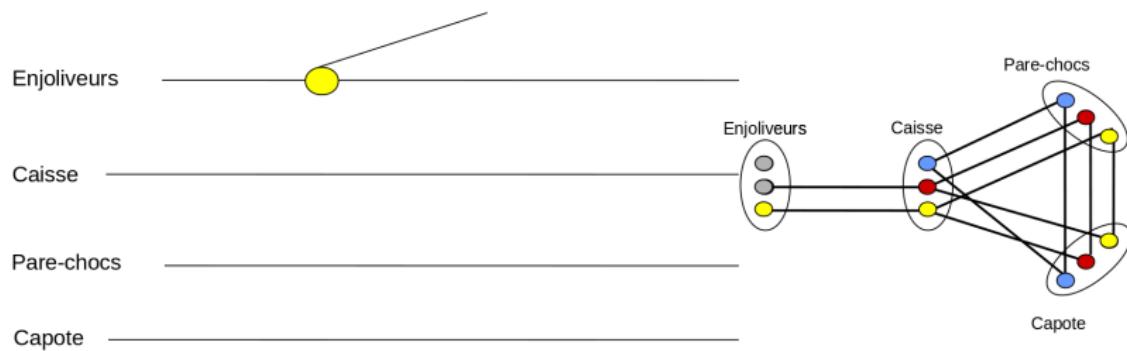
**si**  $\neg C_{x,y}(a, b)$  **alors**

$D_y \leftarrow D_y \setminus \{b\}$  ;

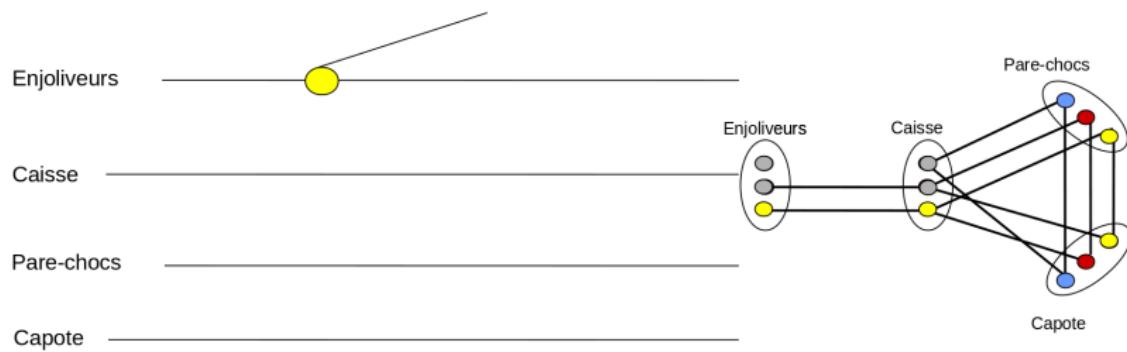
# Forward-Checking



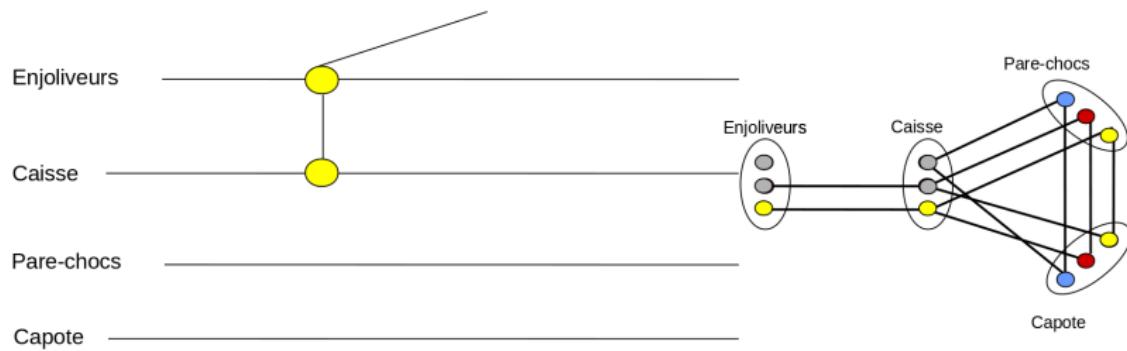
# Forward-Checking



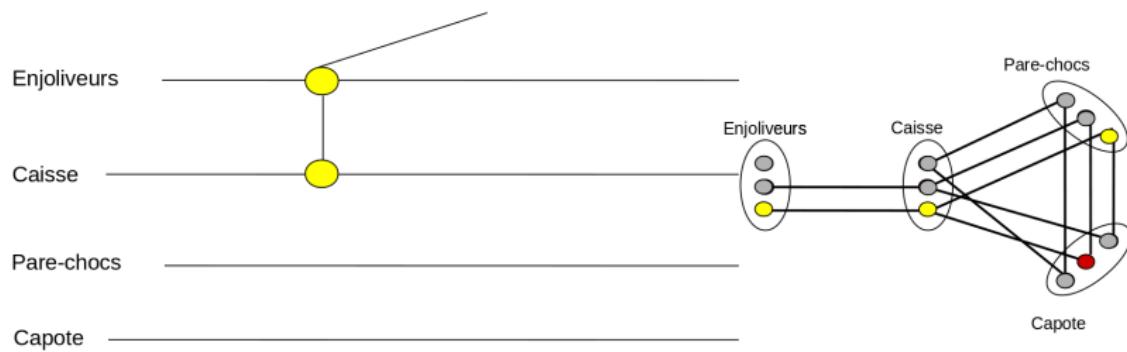
# Forward-Checking



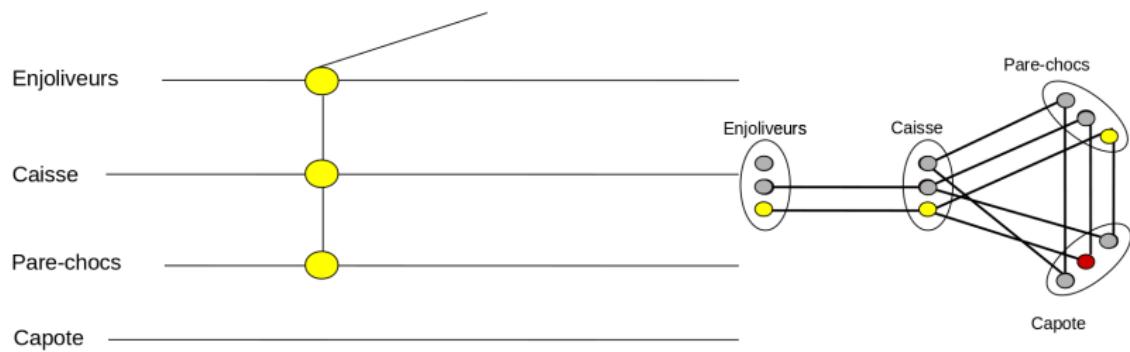
# Forward-Checking



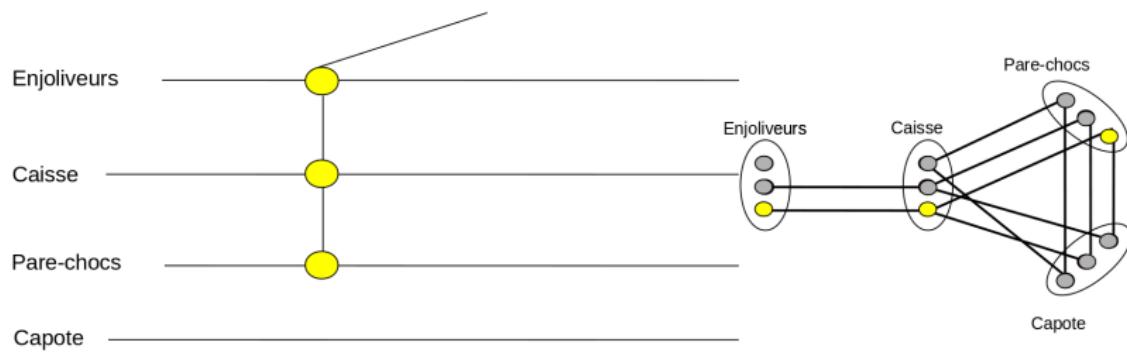
# Forward-Checking



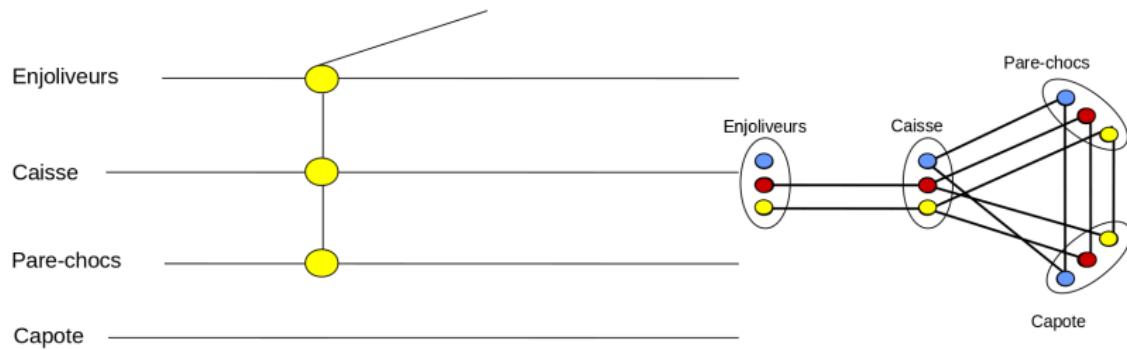
# Forward-Checking



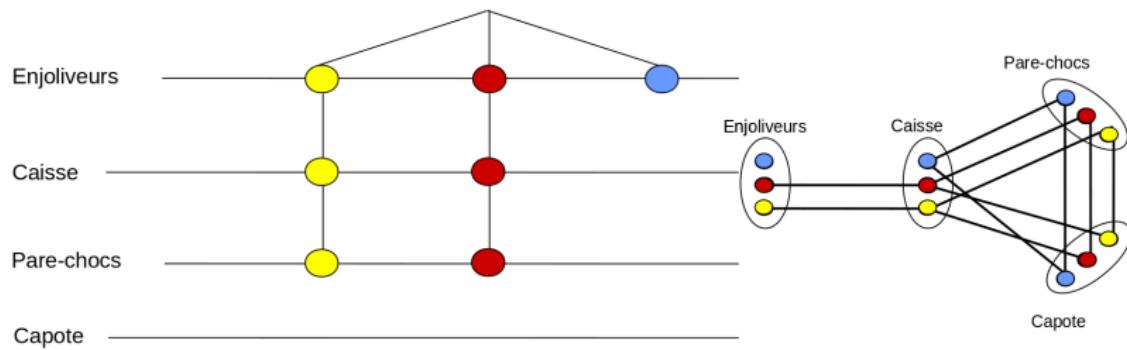
# Forward-Checking



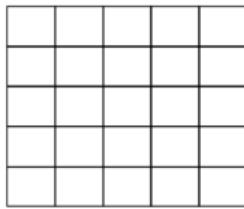
# Forward-Checking



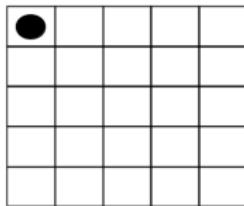
# Forward-Checking



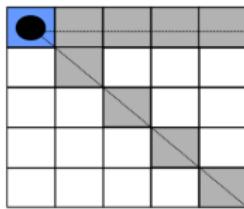
## Exemple de FC sur les n-Reines



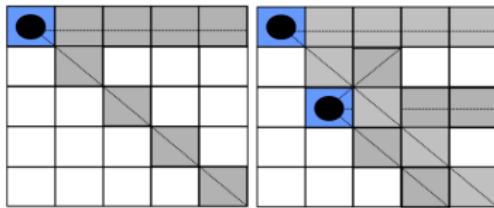
## Exemple de FC sur les n-Reines



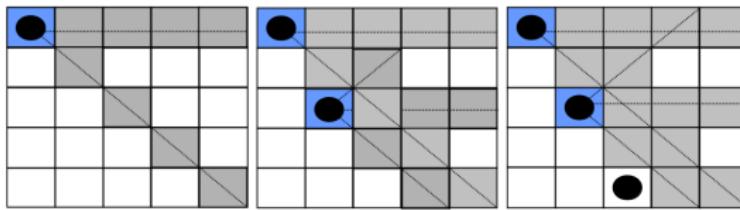
## Exemple de FC sur les n-Reines



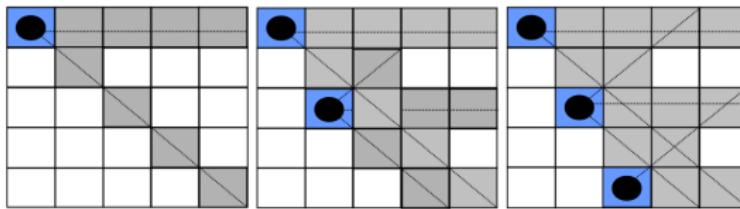
# Exemple de FC sur les n-Reines



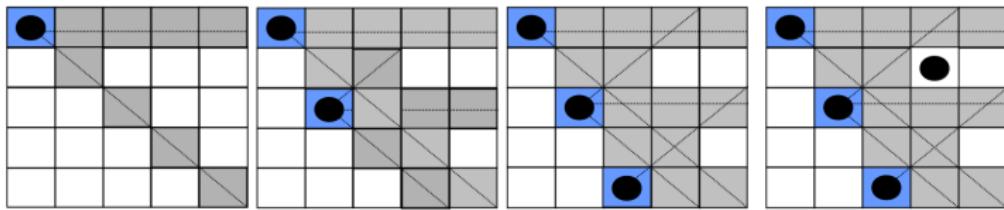
## Exemple de FC sur les n-Reines



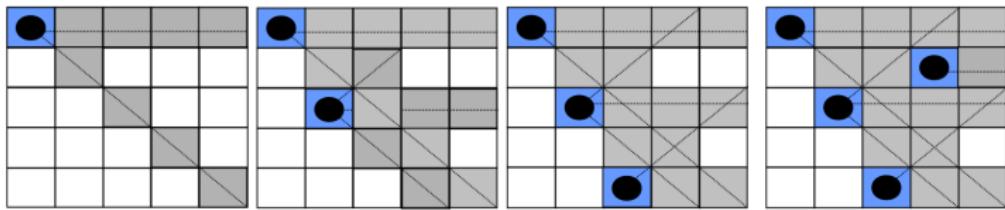
# Exemple de FC sur les n-Reines



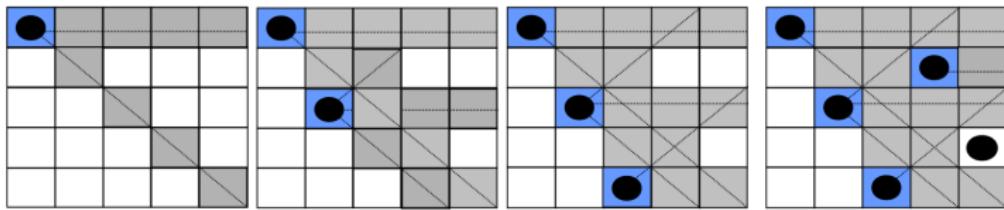
## Exemple de FC sur les n-Reines



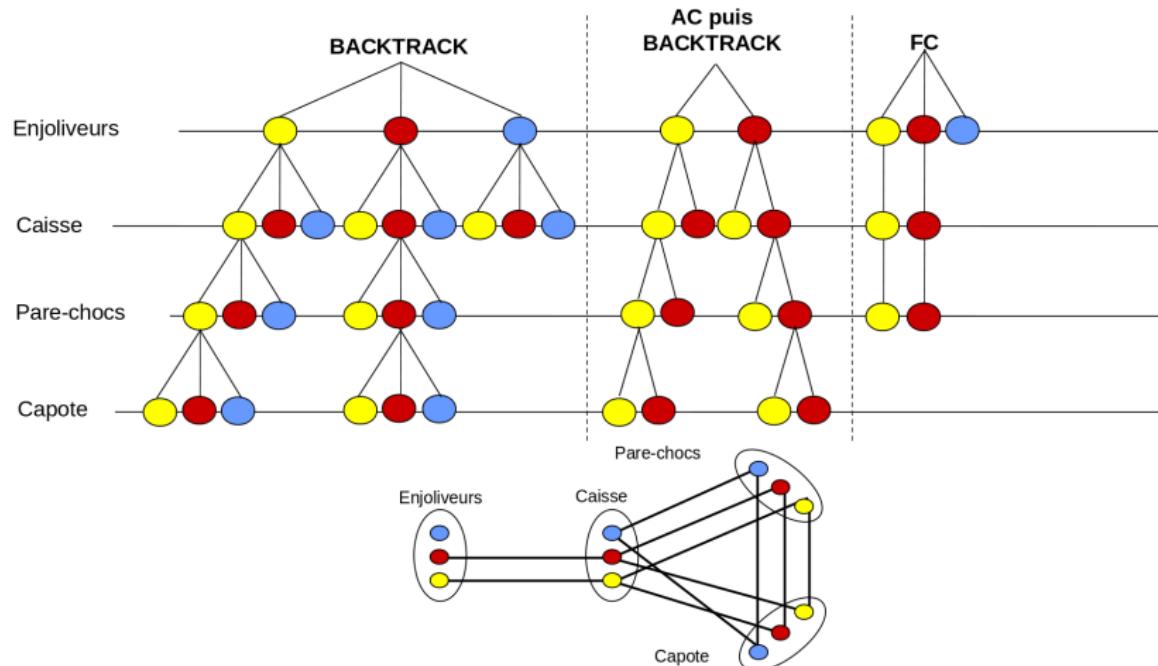
## Exemple de FC sur les n-Reines



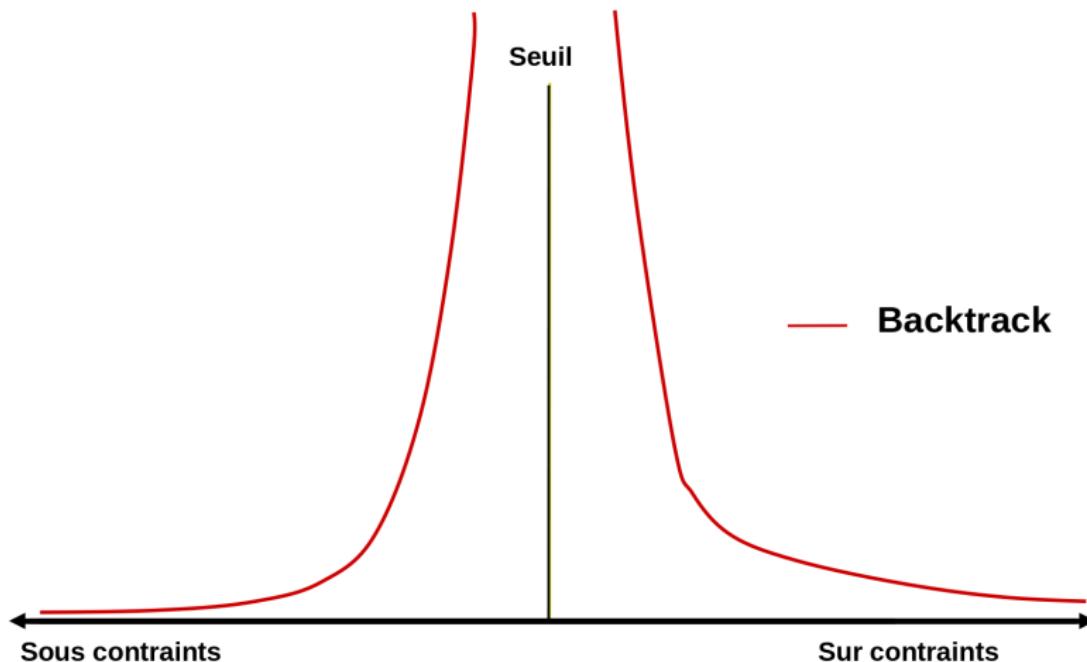
## Exemple de FC sur les n-Reines



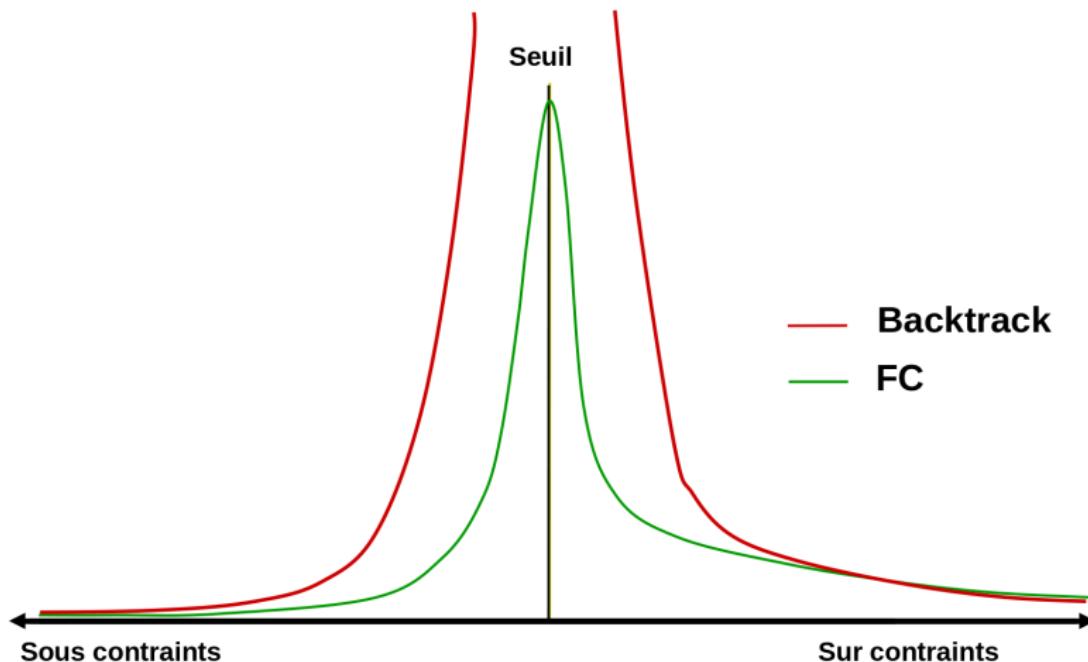
# Arbres de recherche



## Coût moyen d'une recherche



# Coût moyen d'une recherche



# Maintain-Arc-Consistency

- ▶ après chaque instantiation, on réalise la fermeture arc-consistante
- ▶ s'appelle aussi *Full Look Ahead*

---

## Algorithme 6 : MaintainArcConsistency

---

**Données :** Une instantiation partielle  $I$  où on vient de fixer  $\langle x, a \rangle$   
Appliquer AC-4 ;

---

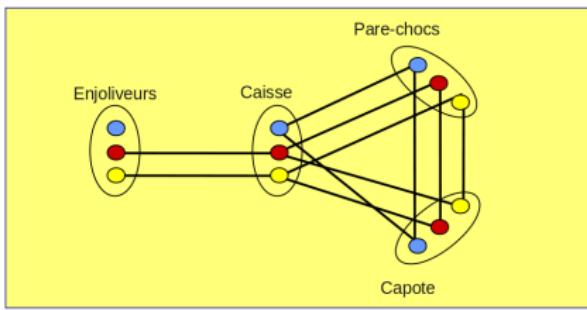
# Maintain-Arc-Consistency

Enjoliveurs \_\_\_\_\_

Caisse \_\_\_\_\_

Pare-chocs \_\_\_\_\_

Capote \_\_\_\_\_



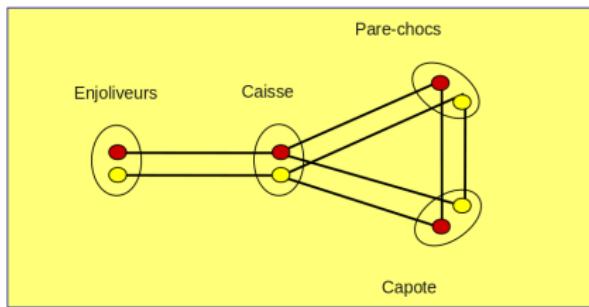
# Maintain-Arc-Consistency

Enjoliveurs \_\_\_\_\_

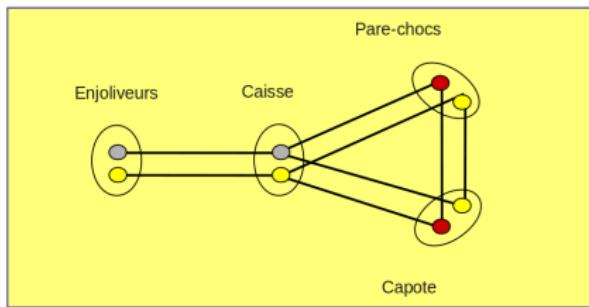
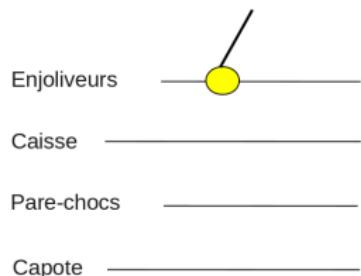
Caisse \_\_\_\_\_

Pare-chocs \_\_\_\_\_

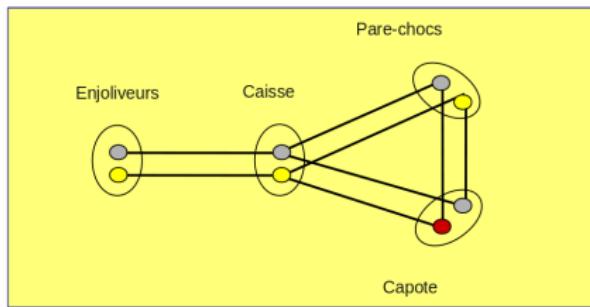
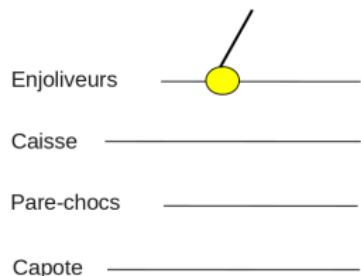
Capote \_\_\_\_\_



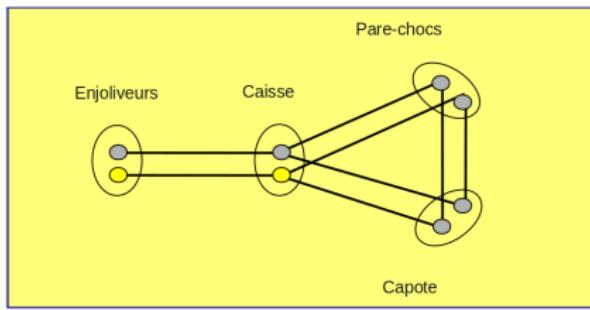
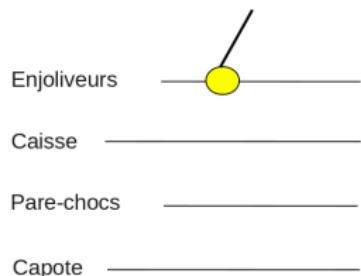
# Maintain-Arc-Consistency



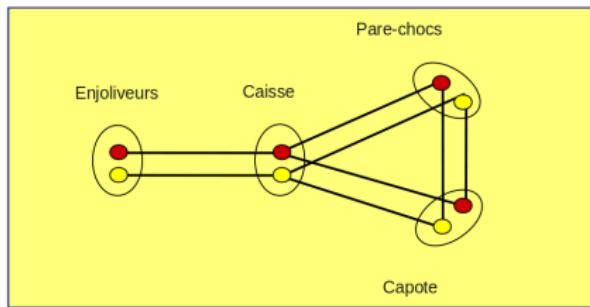
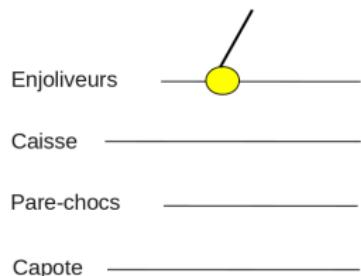
# Maintain-Arc-Consistency



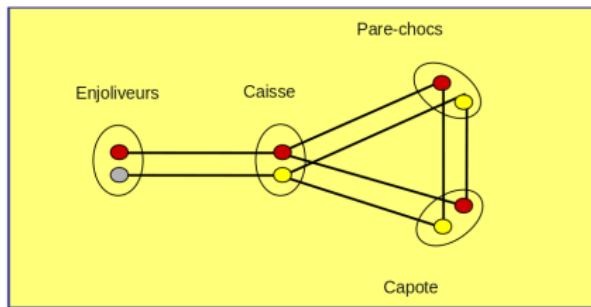
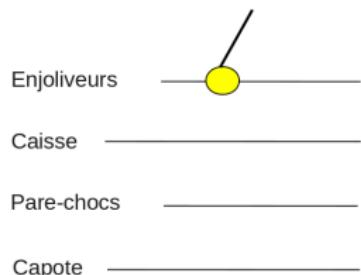
# Maintain-Arc-Consistency



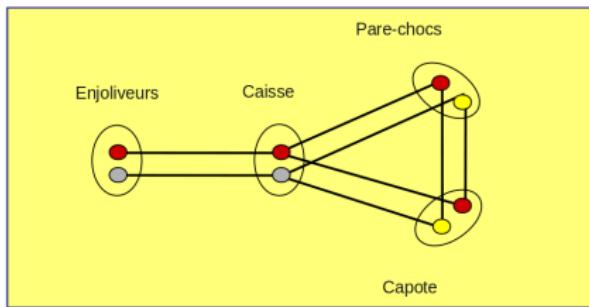
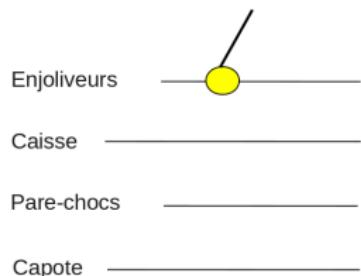
# Maintain-Arc-Consistency



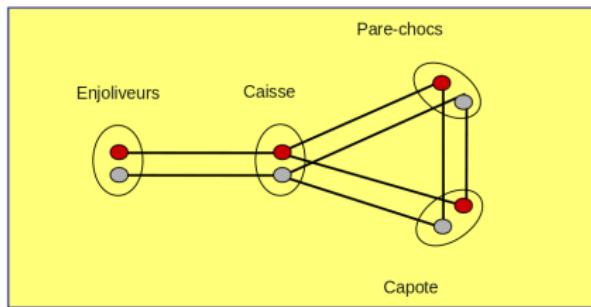
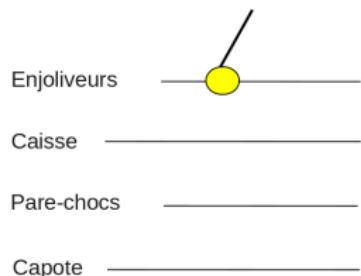
# Maintain-Arc-Consistency



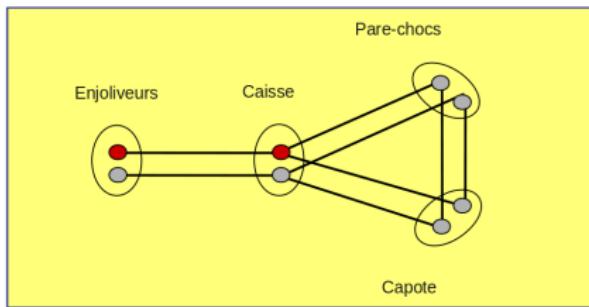
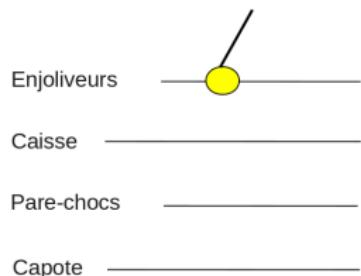
# Maintain-Arc-Consistency



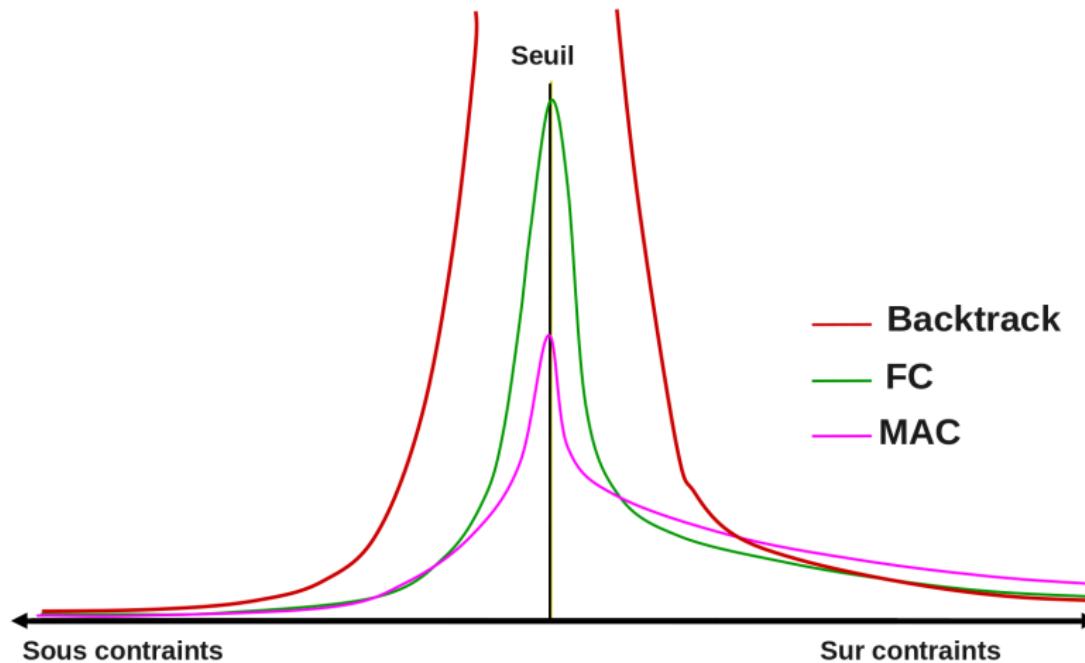
# Maintain-Arc-Consistency



# Maintain-Arc-Consistency



# Coût moyen d'une recherche



## Conclusion sur la consistance locale

- ▶ la consistance locale doit accélérer la recherche de solutions
- ▶ il faut comparer le temps qu'elle coûte au temps d'exploration qu'elle évite
- ▶ en pratique, souvent Forward Checking

# Sommaire

Arc-Consistance

Algos d'AC

Algorithmes prospectifs

Stratégies de branchement

Au delà de l'arc-consistance

## Rappel backtrack

---

**Données :** Une instantiation partielle  $i$

**si**  $i$  viole une contrainte **alors**

    └ Retourner FAUX ;

**si**  $i$  est complète **alors**

    └ Retourner VRAI ;

Choisir une variable  $x$  non instanciée ;

**pour chaque valeur**  $v$  dans  $D_x$  **faire**

    └  $j \leftarrow i \cup \langle x, v \rangle$  ;

**si**  $Backtrack(j)$  **alors**

        └ Retourner VRAI ;

Retourner FAUX ;

---

# Stratégies de branchement

- ▶ dans le backtrack, tout n'est pas précisé
- ▶ ordre des variables
- ▶ ordre des valeurs
- ▶ choisir ces ordres définit une stratégie de recherche
- ▶ on parle aussi d'heuristiques
- ▶ **impacte fortement le temps de résolution**
- ▶ ordre des variables plus d'impact que l'ordre des valeurs
- ▶ 2 types : statique ou dynamique

## Ordre des variables

- ▶ principe : aller le plus vite vers une contradiction.
- ▶ variable de domaine minimal
- ▶ variable la plus contrainte
- ▶ instancier en dernier les variables non liées entre elles

## Ordre des valeurs

- ▶ principe : choisir les valeurs les plus prometteuses
- ▶ valeur la plus supportée
- ▶ valeur qui induit le moins de filtrage
- ▶ valeur qui conduit à des sous-problèmes faciles
- ▶ parfois, séparer le domaine en deux

# Sommaire

Arc-Consistance

Algos d'AC

Algorithmes prospectifs

Stratégies de branchement

Au delà de l'arc-consistance

# Nœuds, arc et chemins

On se situe toujours dans le cadre de CSP binaires.

- ▶ **Nœud-Consistance** : les contraintes unaires sont satisfaites par toutes les valeurs des domaines
- ▶ **Arc-Consistance** : déjà vu.
- ▶ **Chemin-Consistance** : le couple de variable  $(x, y)$  est chemin-consistant avec la variable  $z$  ssi  $\forall(a, b) \in D_x \times D_y$  tel que  $C_{x,y}(a, b)$ ,  $\exists c \in D_z$  tel que  $C_{x,z}(a, c)$  et  $C_{y,z}(b, c)$ .

## k-consistance

On se situe toujours dans le cadre de CSP binaires.

- ▶ généralisation des cas précédents : nœud =1, arc=2, chemin=3
- ▶ Un CSP est  $k$ -consistant ssi toute instantiation partielle de  $k - 1$  variables consistante peut-être étendue à une instantiation partielle de  $k$  variables, en ajoutant n'importe quelle variable non instanciée.
- ▶ Complexité en  $O(n^k d^k)$
- ▶  $n$ -consistance = consistance globale = résolution du CSP

## AC et CSP n-aires

- ▶ on peut étendre la définition de l'arc-consistance aux CSP n-aires.
- ▶ Pour toute contrainte n-aire, pour toute variable impliquée dans cette contrainte, pour chaque valeur dans le domaine de cette variable, il existe des valeurs pour les autres variables de la contrainte telles que la contrainte est satisfaite.
- ▶ ça s'appelle l'**arc-consistance généralisée**.
- ▶ ça s'appelle aussi l'**hyper-arc consistance**.