Programmation par Contraintes

4A STI

B. Nguyen

Basé sur les Tr. Par Yacine Zemali

Bibliographie : *Principles of Constraint Programming*, K. R. Apt.

Introduction

Problème de satisfaction de contraintes

- Les « Constraint Satisfaction Problem » (CSP) regroupent des problèmes d'IA et de RO :
  - satisfiabilité d'un ensemble de clauses
  - Ordonnancement
- Plusieurs types de problèmes :
  - trouver une solution (satisfiabilité)
  - la meilleure solution (optimisation)
  - savoir si une solution existe
  - . . .

Contraintes : premier exemple

- Une contrainte correspond à l'énoncé d'une propriété relative à différents objets
- x:y + z = y met en relation les variables x, y et z en limitant leurs valeurs
- La résolution d'un pb de *satisfaction de contraintes* revient à trouver des valeurs dans les domaines des variables x, y et z qui vérifient l'équation (i.e qui satisfont la contrainte)

Contraintes : domaines d’utilisation

- Les contraintes ont un domaine d’utilisation très vaste :
  - représentation de lois physiques :U = R.I; P = M.G …
  - connaissances symboliques : logique propositionnelle
  - propriétés variées : géométriques, thermiques,...

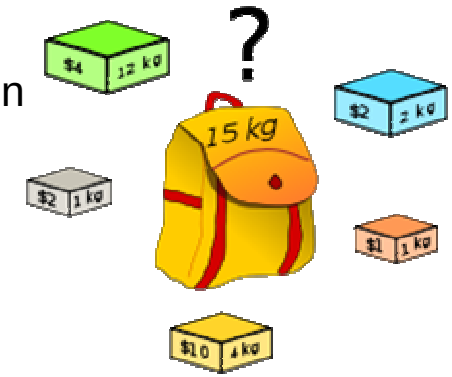
→ il existe de nombreux types de contraintes

Contraintes : Types

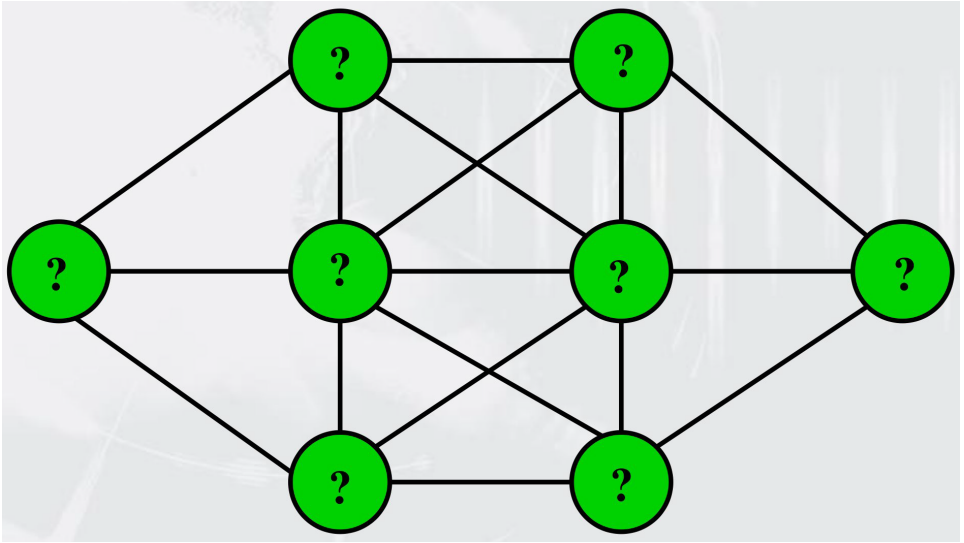
On peut tenter de les classifier suivant :

- la nature des domaines dans lesquels les objets prennent leur valeur (ensemble de symboles, intervalle ou partie de R, de N, intervalle temporel…)
- le langage / modèle dans lequel elles sont exprimées (contraintes linéaires, quadratiques, exprimées en extension…)
- les situation qu'elles décrivent (contraintes temporelles, géométriques,. . .)

- La nature des contraintes affecte la difficulté du problème de satisfaction :
  - complexité polynomiale pour des problèmes de satisfaction ou d'optimisation d'une fonction linéaire sous contraintes linéaires dans R ou Q (programmation linéaire, simplexe)
  - le problème devient NP-complet lors du passage à des domaines discrets (Programmation Lineaire en Nombres Entiers ou PLNE / ILP)
- Exemple : Knapsack : n objets de poids p_i et valeur w_i
  - Maximiser : $z(X) = \sum_{\{i, x_i=1\}} p_i = \sum_{i=1}^{n} x_i p_i$
  - Sous contrainte : $w(X) = \sum_{i=1}^{n} x_i w_i \leq W$

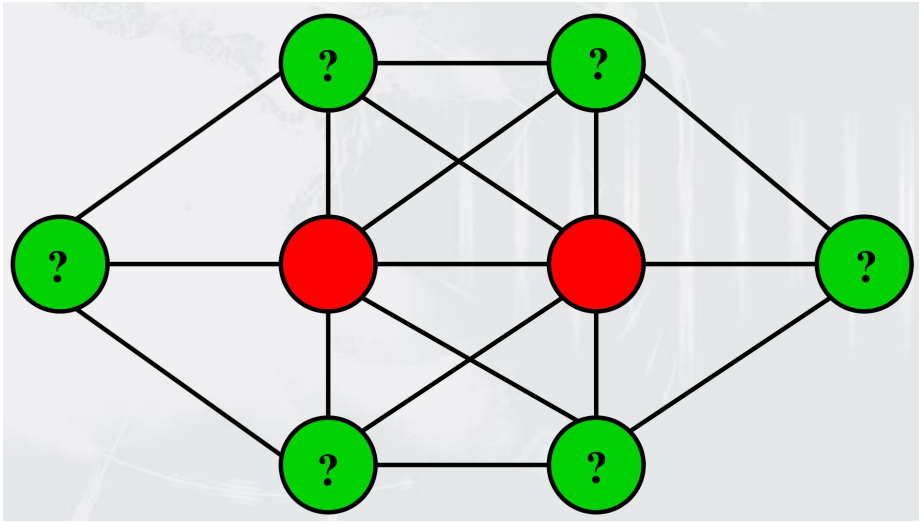


Exemple : puzzle



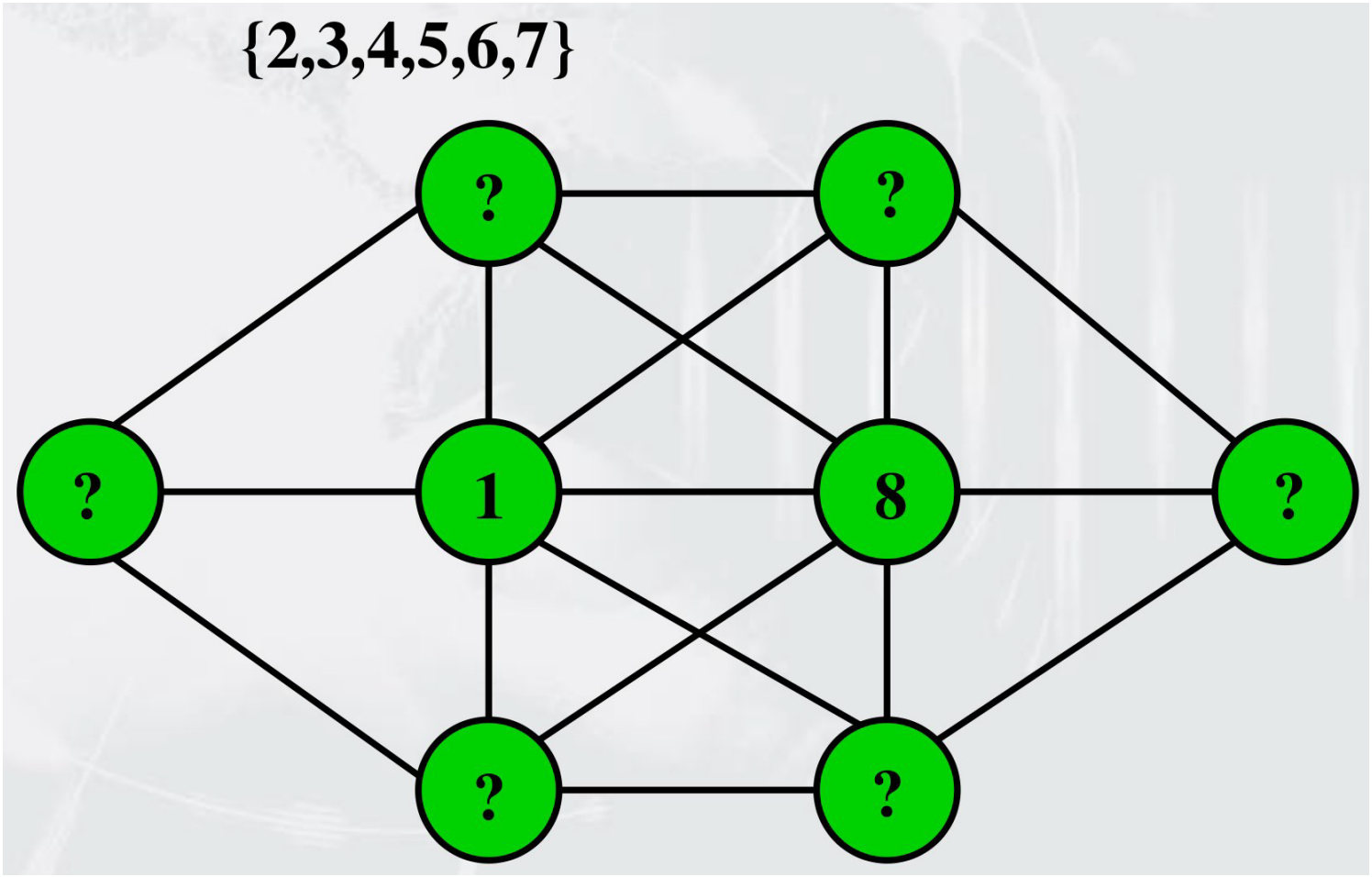
Problème : Placer les chiffres de 1 à 8 de telle sorte qu'aucun des nœuds connecté n'a de valeurs consécutives

Analyse : nœuds les plus difficiles à chiffrer ?

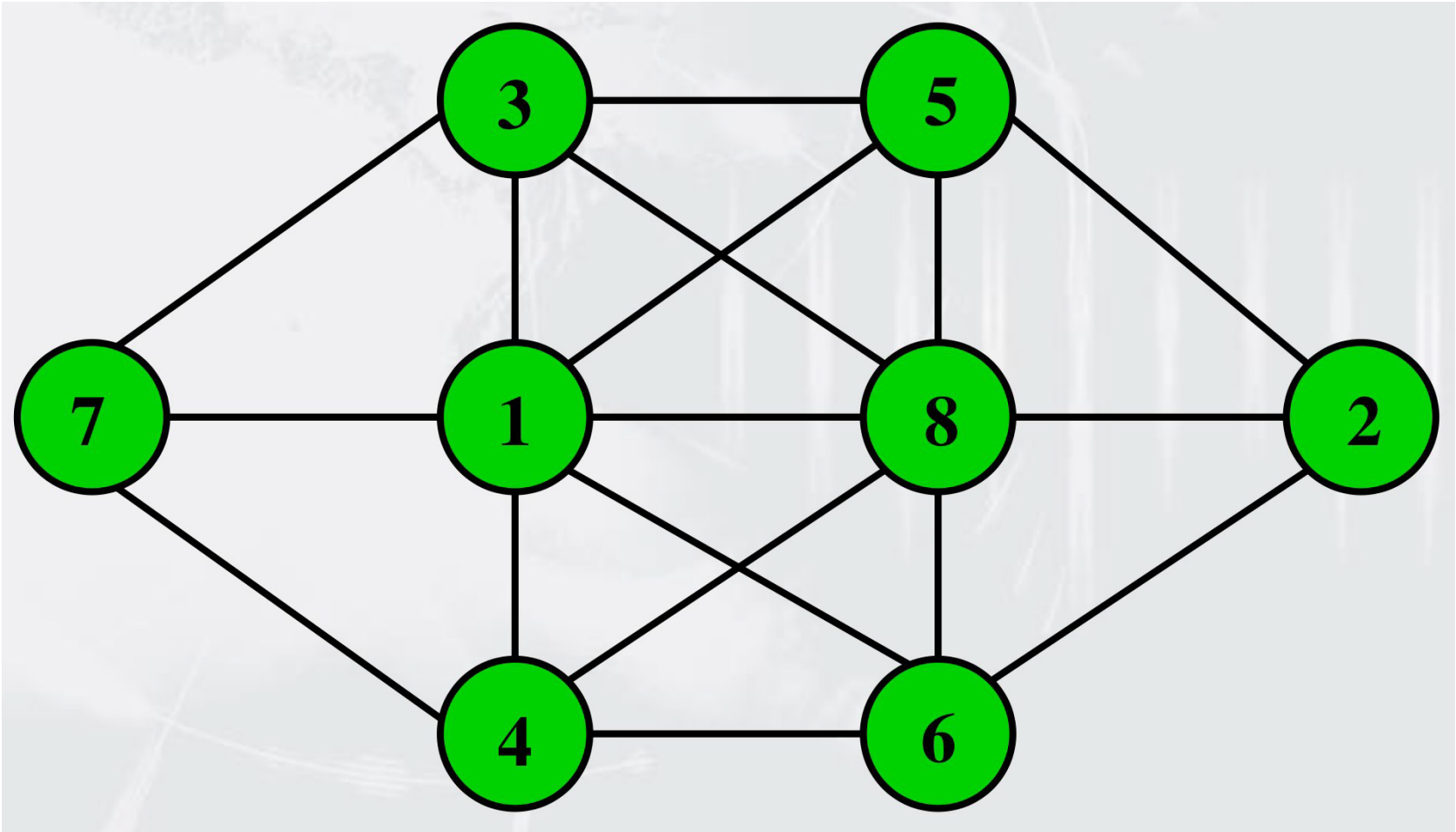


Valeurs moins contraignantes ?
→ 1 et 8 : ils n'ont qu'un seul successeur

Inférence et propagation



Solution



Exemple : Cryptarithmétique

```
   TWO
+  TWO
-------
  FOUR
```

Exprimer les contraintes pour chaque chiffre du résultat, en introduisant les retenues…

O + O = 10X + R
W + W + X = 10Y + U
T + T + Y = 10F + O

F = 1, O = 4, R = 8, T = 7, W = 3, U = 6 donne 734+734=1468

Langage et Notations

N-uplets

- nous utiliserons le langage des n-uplets pour exprimer un CSP
- par abus d’écriture nous utiliserons des fonctions ensemblistes sur des n-uplets
- nous considérerons alors qu’un n-uplet représente l’ensemble de ses éléments

Définition : CSP

Définition 1

Un problème de satisfaction de contraintes ou CSP $\mathcal{P} = (X, D, C)$ est défini par :

- *une séquence $X = (x_1, \ldots, x_n)$ de n variables ;*
- *une séquence $D = (d_1, \ldots, d_n)$ de n domaines finis pour les variables de X ;*
- *une séquence $C = (c_1, \ldots, c_m)$ de m contraintes. Chaque contrainte est définie par (v_i, r_i) :*
  - *v_i est une séquence de variables $(x_{i_1}, \ldots, x_{i_{n_i}}) \subset X$ sur lesquelles porte la contrainte c_i. Arité de $c_i = |v_i|$.*
  - *r_i est une relation définie par un sous-ensemble du produit cartésien $d_{i_1} \times \ldots \times d_{i_{n_i}}$. Il représente les n-uplets des valeurs autorisées pour ces variables.*

Définition : CSP binaire

Définition 2

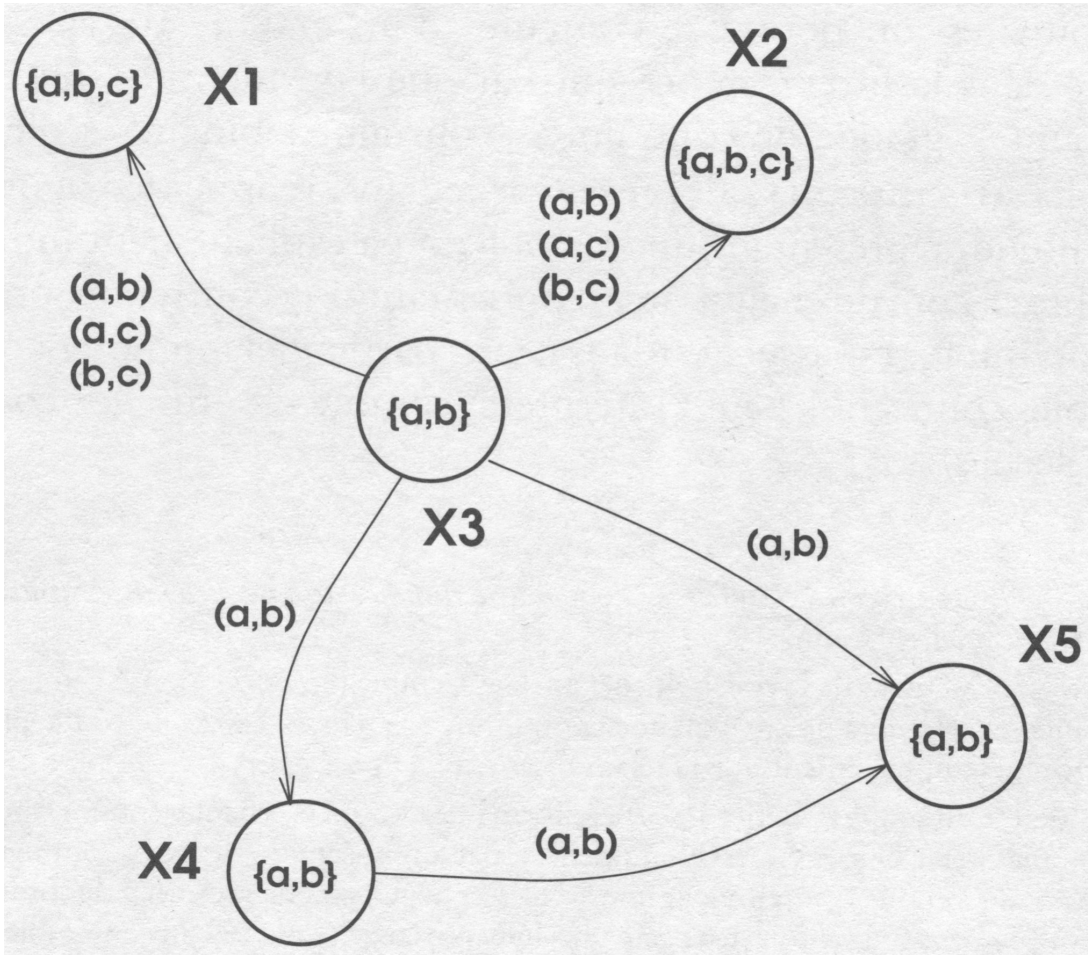
CSP binaire Un CSP binaire est un CSP (X, D, C) *dont toutes les contraintes* $c_i \in C$ *ont une arité égale à 2.*

Variables et Domaine

Variables	Domaines
x_1	$\{a, b, c\}$
x_2	$\{a, b, c\}$
x_3	$\{a, b\}$
x_4	$\{a, b\}$
x_5	$\{a, b\}$

Contraintes	Variables	Relations
$c_1 = (v_1, r_1)$	$v_1 = (x_3, x_1)$	$r_1 = \{(a, b), (a, c), (b, c)\}$
$c_2 = (v_2, r_2)$	$v_2 = (x_3, x_2)$	$r_2 = \{a, b), (a, c), (b, c)\}$
$c_3 = (v_3, r_3)$	$v_3 = (x_3, x_4)$	$r_3 = \{(a, b)\}$
$c_4 = (v_4, r_4)$	$v_4 = (x_3, x_5)$	$r_4 = \{(a, b)\}$
$c_5 = (v_5, r_5)$	$v_5 = (x_4, x_5)$	$r_5 = \{(a, b)\}$

Représentation sous forme de graphe

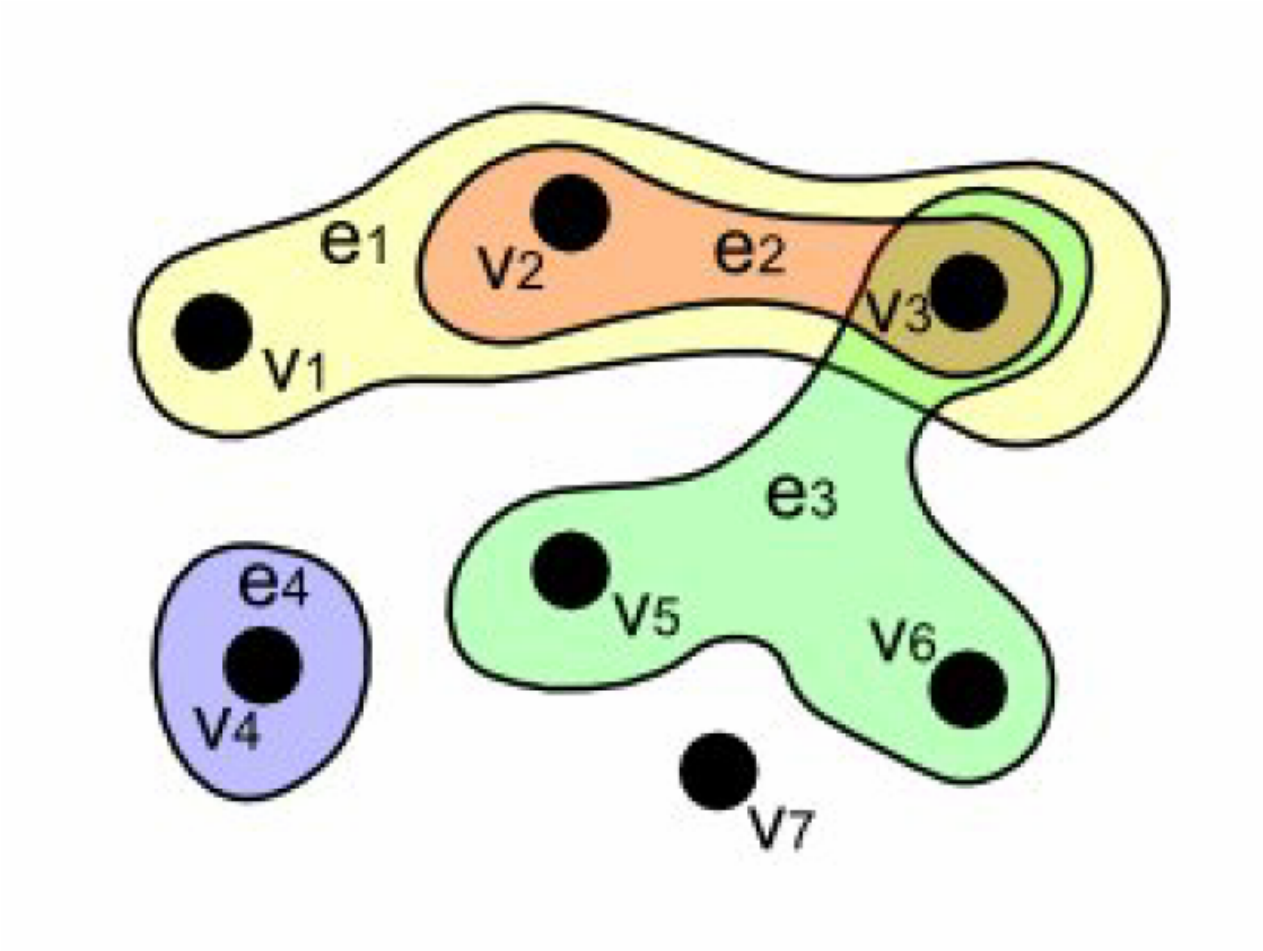


Existe-t-il des solutions possible au problème ?

Représentation sous forme d'*hypergraphe* pour CSP non-binaire

- correspondance CSP <-> hyper-graphe
- sommets = variables
- hyper-arrêtes = contraintes
- hyper-arrêtes = ensemble de n sommets connectés
- **dans le cas d'un CSP binaire, un graphe suffit**
- arrêtes = ensemble de 2 sommets connectés

Exemple



Sémantique

Donnons du sens …

- Instanciation = interprétation en logique propositionnelle
- Il s'agit de donner un sens aux variables

Définition 3 (Instanciation)

Soit un CSP $\mathcal{P} = (X, D, C)$*, on appelle instanciation* $\mathscr{A}$ *de* $Y = x_{y_1}, \ldots, x_{y_{|Y|}} \subset X$ *une application qui associe à chaque variable* $x_{y_i} \in Y$ *une valeur* $\mathscr{A}(x_{y_i}) \in d_{y_i}$ *(le domaine de* x_{y_i}*)*

Instanciation

nous ferons référence à une instanciation $\mathscr{A}$

- en tant qu'application (couples $(x_{y_i}, \mathscr{A}(x_{y_i}))$

$$\mathscr{A} = \{x_{y_1} \rightarrow \mathscr{A}(x_{y_1}), \ldots, x_{y_{|Y|}} \rightarrow \mathscr{A}(x_{y_{|Y|}})\}$$

- en tant que séquence de valeurs

$$\{\mathscr{A}(x_{y_1}), \ldots, \mathscr{A}(x_{y_{|Y|}})\}$$

une instanciation $\mathscr{A}$ de domaine X est dite complète (partielle sinon)

l'application de $\mathscr{A}$ sur un ensemble (resp. séquence) de variables V à pour résultat l'ensemble (resp. séquence) des images des éléments de V par $\mathscr{A}$

Satisfaction de contraintes

Définition 4 (Satisfaction de contrainte)

Soit un CSP $\mathcal{P} = (X, D, C)$, une instanciation $\mathscr{A}$ de Y satisfait la contrainte $c_i = (v_i, r_i)$ de C (on note $\mathscr{A} \models c_i$) ssi $v_i \subset Y$ et $\mathscr{A}(v_i) \in r_i$.
Une instanciation $\mathscr{A}$ de Y viole c_i ssi $v_i \subset Y$ et $\mathscr{A}(v_i) \notin r_i$.

Retour sur l'exemple

Contraintes	Variables	Relations
$c_1 = (v_1, r_1)$	$v_1 = (x_3, x_1)$	$r_1 = \{(a, b), (a, c), (b, c)\}$
$c_2 = (v_2, r_2)$	$v_2 = (x_3, x_2)$	$r_2 = \{a, b), (a, c), (b, c)\}$
$c_3 = (v_3, r_3)$	$v_3 = (x_3, x_4)$	$r_3 = \{(a, b)\}$
$c_4 = (v_4, r_4)$	$v_4 = (x_3, x_5)$	$r_4 = \{(a, b)\}$
$c_5 = (v_5, r_5)$	$v_5 = (x_4, x_5)$	$r_5 = \{(a, b)\}$

- **rappel :** la relation associée à une contrainte définit l'ensemble des valeurs qui sont autorisées
- dans notre exemple précédent, l'affectation :

$$\mathscr{A} = \{x_1 \to b, x_2 \to a, x_3 \to a\}$$

- satisfait c_1 car $\mathscr{A}((x_3, x_1)) = (a, b) \in r_1$
- viole c_2 car $\mathscr{A}((x_3, x_2)) = (a, a) \notin r_2$
- ne viole pas et ne satisfait pas c_3 car $v_3 \notin \{x_1, x_2, x_3\}$

Autre exemple

Instanciation Consistante

Définition 5 (Instanciation consistante)

Soit un CSP $\mathcal{P} = (X, D, C)$, une instanciation $\mathscr{A}$ des variables de $Y \subset X$ est dite consistante ssi :

$$\forall c_i = (v_i, r_i) \in C \text{ telle que } v_i \subset Y, \mathscr{A} \models c_i$$

- une instanciation est consistante si elle ne viole aucune contrainte
- vérifier qu'une instanciation $\mathscr{A}$ sur Y est consistante est un problème facile (polynomial) $\Rightarrow$ il suffit de vérifier que chaque c_i n'est pas violée par $\mathscr{A}$

Solution

Définition 6 (Solution(s) d'un CSP)

Une solution $\mathscr{S}$ de $\mathcal{P} = (X, D, C)$ est une instanciation consistante des variables de X. On dit alors que l'instanciation $\mathscr{S}$ satisfait $\mathcal{P}$ (on note $\mathscr{S} \models \mathcal{P}$).
L'ensemble des solutions de $\mathcal{P}$ est noté $S_{\mathcal{P}}$.

- Vérifier qu'une instanciation complète est une solution est un problème **facile**.

Consistance d'un CSP

Définition 7 (Solution(s) d'un CSP)

Un CSP $\mathcal{P} = (X, D, C)$ est dit consistant ssi $S_{\mathcal{P}} \neq \varnothing$.

- Le problème de décision "*un CSP est-il consistant ?*" est NP-complet.

Instanciation globalement consistante

Définition 8 (Instanciation globalement consistante)

Soit un CSP $\mathcal{P} = (X, D, C)$, une instanciation $\mathscr{A}$ de $Y \subset X$ est dite globalement consistante ssi $\exists \mathscr{S} \in S_{\mathcal{P}}$ telle que $\mathscr{A} \subset \mathscr{S}$.

- instanciation $\mathscr{A}$ non globalement consistante $\Rightarrow$ il n'est pas possible d'étendre $\mathscr{A}$ en une solution
- vérification difficile (problème NP-complet)
- difficulté de la vérification inversement proportionnelle à la taille de l'instanciation $\mathscr{A}$
- vérifier la consistance globale de l'instanciation vide $\mathscr{A} = \varnothing$ est équivalent à vérifier la consistance du CSP

Exemple : Firme automobile (Bessière 1992)

- Une firme automobile veut sortir un nouveau modèle de voiture fabriqué dans toute l'europe :
  - les portières et le capot sont faits à Lille, où le constructeur ne dispose que de peinture rose, rouge et noire ;
  - la carrosserie est faite à Hambourg, où l'on a de la peinture blanche, rose, rouge et noire ;
  - les pare-chocs, faits à Palerne sont toujours blancs ;
  - la bâche du toit-ouvrant, qui est faite à Madrid, ne peut être que rouge ;
  - les enjoliveurs sont faits à Athènes, où l'on a de la peinture rose et rouge.

- Le concepteur impose quelques contraintes quant à l'agencement des couleurs :
  - la carrosserie doit être de la même couleur que les portières ;
  - les portières doivent être de la même couleur que le capot ;
  - le capot doit être de la même couleur que la carrosserie ;
  - les enjoliveurs doivent être plus clair que la carrosserie ;
  - les pare-chocs doivent être plus clair que la carrosserie ;
  - le toit-ouvrant doit être plus clair que la carrosserie.

Modélisation

- les contraintes portent sur au plus 2 variables → CSP binaire
- 6 variables x1; x2; x3; x4; x5; x6 pour modéliser les 6 composants de la voiture
- domaines des variables → 4 couleurs *ordonnées* (notées a, b, c, d)

Composant	Variable
pare-chocs	x_1
portières	x_2
carrosserie	x_3
enjoliveurs	x_4
capot	x_5
toit-ouvrant	x_6

Couleur	Symbole
blanc	*a*
rose	*b*
rouge	*c*
noir	*d*

Variables et Domaine

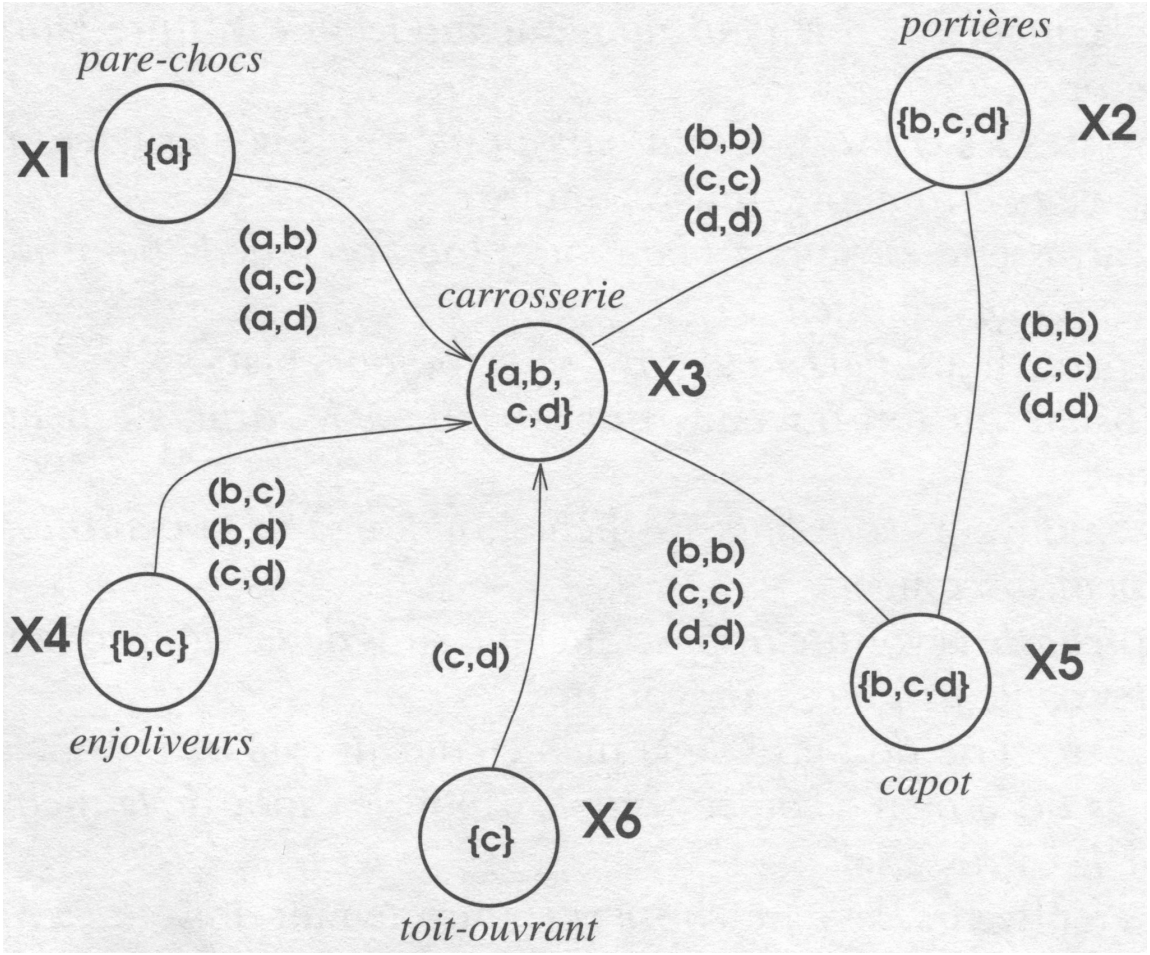
Variables	Domaines
pare-chocs	blanc
portières	rose, rouge, noire
carrosserie	blanc, rose, rouge, noire
enjoliveurs	rose, rouge
capot	rose, rouge, noire
toit-ouvrant	rouge

Variables	Domaines
x_1	$\{a\}$
x_2	$\{b, c, d\}$
x_3	$\{a, b, c, d\}$
x_4	$\{b, c\}$
x_5	$\{b, c, d\}$
x_6	$\{c\}$

Graphe du CSP

- arc → contrainte dont la relation associée peut s'interpréter comme relation d'ordre
- arrête → contrainte dont la relation associée peut s'interpréter comme relation d'égalité

Graphe du CSP



Solutions

Ce CSP admet 2 solutions :

- $\mathscr{S}_1 = \{x_1 \to a, x_2 \to d, x_3 \to d, x_4 \to b, x_5 \to d, x_6 \to c\}$
- $\mathscr{S}_2 = \{x_1 \to a, x_2 \to d, x_3 \to d, x_4 \to c, x_5 \to d, x_6 \to c\}$

On peut en déduire par exemple que :

- l'instanciation $\{x_2 \to d, x_3 \to d, x_6 \to c\}$ est globalement consistante
- l'instanciation $\{x_1 \to a, x_3 \to c, x_4 \to b\}$ ne l'est pas

Variables	Instance
pare-chocs	blanc
portières	noire
carrosserie	noire
enjoliveurs	rose
capot	noire
toit-ouvrant	rouge

$\mathscr{S}_1$

Variables	Instance
pare-chocs	blanc
portières	noire
carrosserie	noire
enjoliveurs	rouge
capot	noire
toit-ouvrant	rouge

$\mathscr{S}_2$

Equivalence de CSP

Définition 9 (Équivalence de CSP)

Les CSP $\mathcal{P} = (X, D, C)$ et $\mathcal{P}' = (X, D', C')$ sont dits équivalents (on note $\mathcal{P} \equiv \mathcal{P}'$) ssi $S_{\mathcal{P}} = S_{\mathcal{P}'}$.

- 2 CSP équivalents $\Rightarrow$ on peut résoudre l'un à la place de l'autre
- sur l'exemple précédent, on peut supprimer b et c de d_3 pour obtenir un CSP $\mathcal{P}'$ équivalent
- $\mathcal{P}'$ est plus simple que $\mathcal{P}$: le nombre d'instanciations complètes est plus faible (et donc l'espace de recherche aussi)

Contrainte Induite

Définition 10 (Contrainte induite)

Soit un CSP $\mathcal{P} = (X, D, C)$ et une contrainte c définie par un ensemble $v_c \subset X$ de variables qu'elle connecte et une relation r_c, on dit que c est induite par $\mathcal{P}$ ssi $\forall \mathcal{S} \in S_{\mathcal{P}}, \mathcal{S} \models c$.

- une contrainte c est induite par un CSP $\mathcal{P}$ si elle est satisfaite par toutes les solutions de $\mathcal{P}$
- notion liée à la notion d'instanciation globalement consistante : une contrainte est induite par un CSP ssi seuls des n-uplets représentant une instanciation non globalement consistante sont absents de sa relation associée

Exercice : formalisation

Reformuler de manière formelle la phrase “une contrainte est induite par un CSP ssi seuls des n-uplets représentant une instanciation non globalement consistante sont absents de sa relation associée”

Contrainte Induite

- une contrainte induite peut être rajouté au CSP $\mathcal{P}$ sans modifier $S_{\mathcal{P}}$
- dans notre exemple, on peut rajouter une contrainte c portant sur x_3 dont la relation associée est $\{(a), (d)\}$
- c est une contrainte induite
- l'ajout de c nous donne le même CSP $\mathcal{P}'$ obtenu par suppression de b et c de d_3
- $\mathscr{A}_1 = \{x_3 \rightarrow b\}$ et $\mathscr{A}_1 = \{x_3 \rightarrow c\}$ ne sont pas globalement consistante

Consistance Globale

Définition 11 (Consistance globale)

Un CSP $\mathcal{P} = (X, D, C)$ est dit globalement consistant ssi toute instanciation consistante est globalement consistante.

- toute instanciation partielle consistante peut alors être étendue de façon gloutonne en une solution
- la construction d'une solution d'un CSP globalement consistant est donc un problème trivial
- cela n'est pas le cas des exemples vus jusqu'ici

Minimalité

Définition 12 (Consistance globale)

Un CSP $\mathcal{P} = (X, D, C)$ *est dit minimal ssi :*

$$\forall c_i = (v_i, r_i) \in C, \bigcup_{\mathscr{S} \in S_{\mathcal{P}}} [\mathscr{S}(v_i)] = r_i$$

- un CSP est minimal si toutes ses contraintes sont les contraintes minimales (au sens de l'inclusion sur les relations associées) induites par lui-même
- pour une contrainte $c_i = (v_i, r_i)$ il n'est pas possible d'induire de nouvelle contrainte $c'_i = (v_i, r'_i)$
- un CSP binaire globalement consistant est nécessairement minimal (réciproque fausse)
- l'existance de solution pour un CSP minimal est un problème trivial (mais pas la construction de solution)

Minimalité : Exercice

Soit $\mathcal{P} = (X, D, C)$ un CSP minimal, expliquer de manière informelle pourquoi pour une contrainte $c_i = (v_i, r_i)$ il n'est pas possible d'induire de nouvelle contrainte $c'_i = (v_i, r'_i)$

Enoncés de problèmes

Problèmes

- le problème habituellement considéré est le problème de satisfaction
- consiste à exhiber une solution du CSP
- problème NP-difficile dans le cas général (i.e. au moins aussi difficile qu'un problème NP-complet)

Problèmes

- Pour un CSP donné, de nombreux autres problèmes peuvent être considérés :
  - prouver qu'un CSP est consistant (pb de satisfiabilité),NP-complet
  - exhiber toutes les solutions d'un CSP
  - exhiber une solution qui maximise un ou plusieurs critères
  - calculer ou estimer le nombre de solutions d'un CSP
  - trouver pour une (ou plusieurs) variable(s) un ensemble de valeurs qui figurent dans toutes les solutions
  - . . .

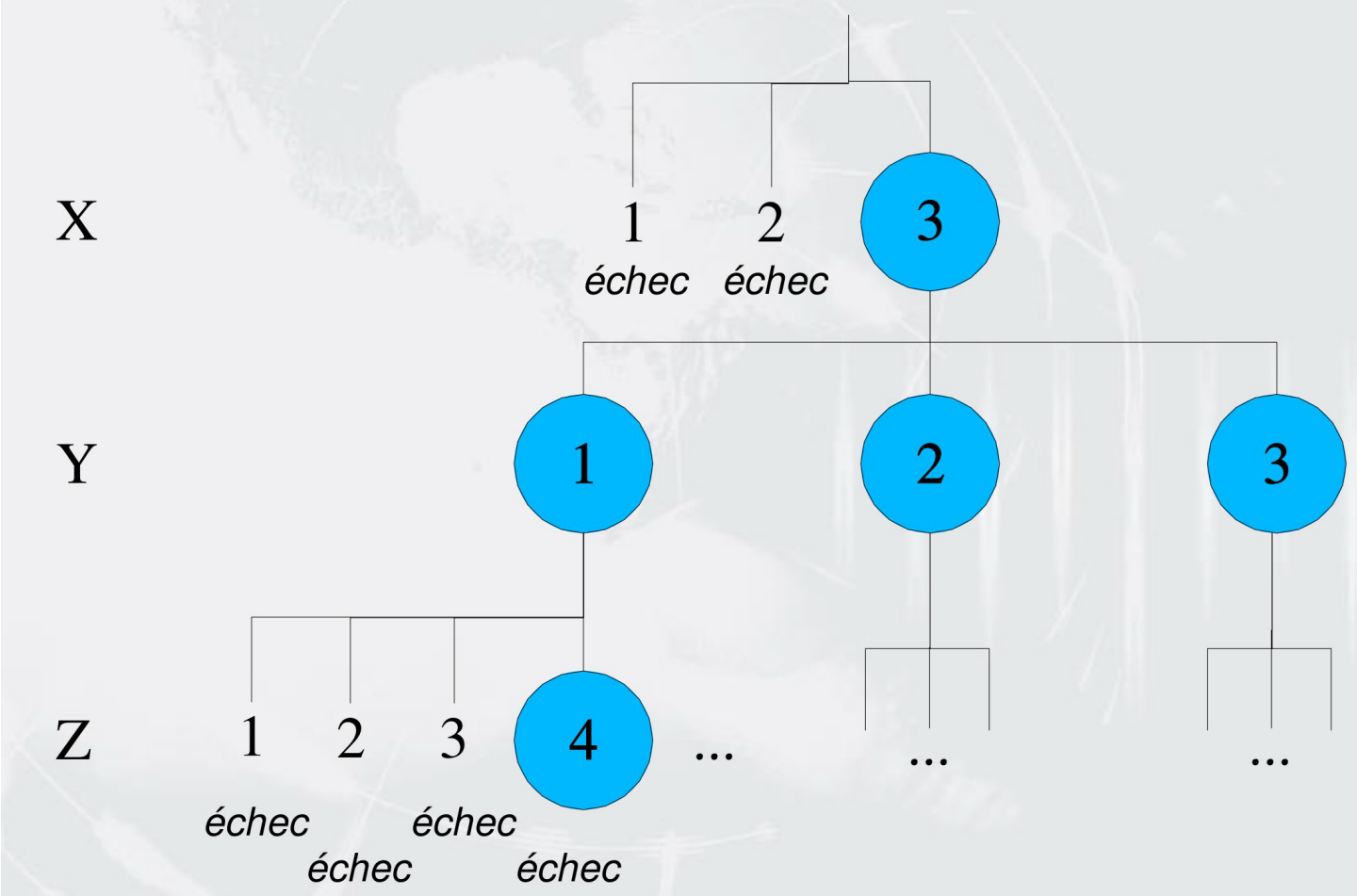
Méthode (cf cours d'IA)

- recherche dans l'espace d'état (un arbre)
- chaque état est une instanciation A
- règles de production permettant d'étendre l'instanciation A sur une variable avec toutes les valeurs possibles de son domaine
- les feuilles de l'arbre exploré sont des instanciations complètes
- on réalise un parcours en profondeur d'abord (pas d'intérêt pour largeur d'abord car on connaît la profondeur des solutions)

Algo Generate and Test

- l'algo précèdent est appelé *generate and test*
- sur un CSP de n variables, avec des domaines de cardinal d et m contraintes, l'ensemble des terminaux a un cardinal d^n
- pour obtenir toutes les solutions, on fera au pire cas $m * d^n$ évaluations
- exemple : $10^{-6}s$ pour évaluer une contrainte, sur un CSP de 20 variables, avec des domaines de cardinal 5 et 20 contraintes, il faudrait 60 ans pour obtenir toutes les solutions. . .

Algo Generate and Test



Contraintes superflues : on ne teste qu'à la fin !

Problèmes de grande taille

- nombre d'atomes dans l'univers 2.10^{80}
- c'est également la taille de l'espace de recherche engendré par un CSP :
  - à 266 variables booléennes
  - à 80 variables sur un domaine de taille 10
  - à 40 variables sur un domaine de taille 100
  - à 27 variables sur un domaine de taille 1000

Amélioration et test de consistance

- dès qu'une contrainte est violée, aucune instanciation fille d'un noeud ne pourra être consistante
- il est possible de vérifier qu'une contrainte est violée seulement lorsque toutes les variables sur lesquelles elle porte ont été instanciées

Propriété 1

Toute instanciation non consistante est non globalement consistante.

Amélioration : Backtrack

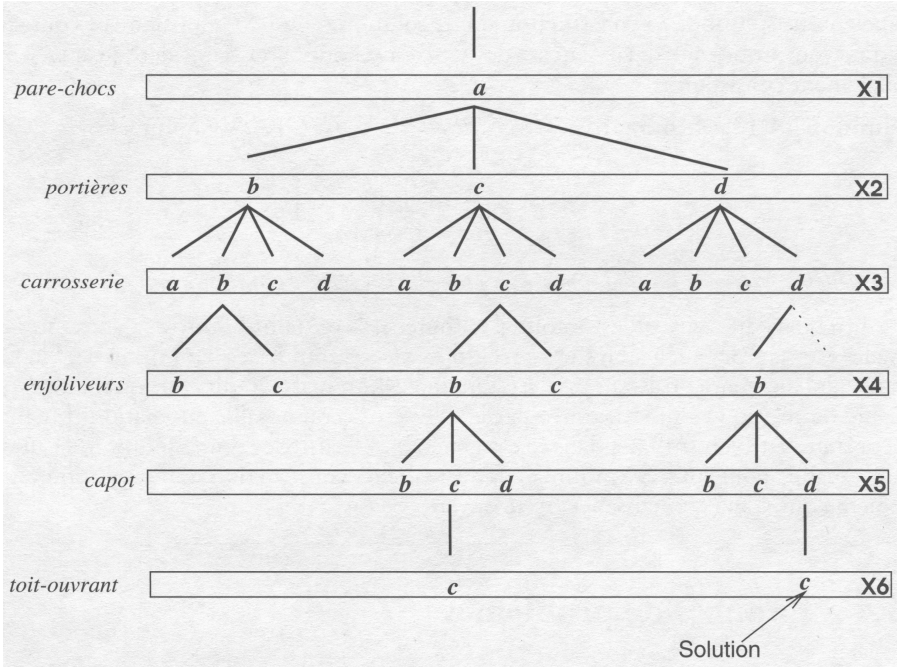
- vérifier la satisfaction de chaque contrainte dès que possible
- algorithme Backtrack, BT, test and generate
- parcours en profondeur d'abord
- l'arrivée sur un noeud rendant l'instanciation non
- consistante entraîne un retour arrière sur un noeud père
- fonction récursive prenant en entrée :
  - une séquence V de variables à instancier (initialement X en entier)
  - une instanciation A (initialement vide)

Algorithme 1: Backtrack

```
backtrack(V, 𝒜);
si V = ∅ alors
    𝒜 est une solution;
sinon
    soit x_i ∈ V;
    pour chaque ν ∈ d_i faire
        si 𝒜 ∪ {x_i → ν} est consistante alors
            backtrack(V − {x_i}, 𝒜 ∪ {x_i → ν});
```

Sur l’exemple « voiture »

- instanciation des variables dans l’ordre (x1; x2; x3; x4; x5; x6)
- il faut explorer 29 noeuds avant de trouver une solution
- il faut effectuer 30 vérifications de contraintes avant de trouver une solution



Défauts et améliorations

- parcours aveugle de l'espace sans prise en compte
- d'informations "évidentes" sur la non consistance globale
- d'instanciations partielles
- découverte répétée des mêmes inconsistances locales
- pas d'ordonnancement dans les évaluations de variable
- . . .

Dans les semaines à venir nous nous intéresserons principalement aux points suivants :

- définition de problèmes polynomiaux correspondant à des versions affaiblies de la propriété de satisfiabilité ! propriétés de consistance locale
- plusieurs améliorations de l'efficacité de l'algo Backtrack
- méthodes de décomposition de CSP en sous-CSP plus simples, si possible dont la satisfiabilité soit un problème polynomial