Organisation

- `http://www.liafa.jussieu.fr/~haberm/cours/prologconstraints/`
  e-mail : Peter.Habermehl@liafa.jussieu.fr
- Contrôle des connaissances :
  Note finale première session = 1/2 Note partiel + 1/2 Note examen
  Note finale deuxième session = examen
- TD/TP : Jeudi 8h30 ou 10h30, Salle 108
- Avertissement : Les transparents ne contiennent pas tout.
- Bibliographie
  - Marriott and Stuckey. Programming with Constraints, an introduction. The MIT Press. 1998. `www.cs.mu.oz.au/~pjs/book/book.html`
  - Programmation Logique par Contraintes. François Fages, Collection "Cours de l'Ecole Polytechnique", Ellipses, 1996.

Plan du cours

- Introduction
- Contraintes sur un domaine fini
- Simplification de contraintes
- Contraintes linéaires
- Contraintes sur un domaine fini numérique
- Programmation logique avec contraintes
- Modélisation

Qu'est-ce qu'une contrainte ?

- Exemples : $X = Y + 2$, $list(a, list(b, Y)) = list(a, L)$
- Variable : X, Y, L
- Symbole de fonction : $+, -, sin, cos, ||, list, \ldots$
- Symbole de relation : $=, \leq, \neq, \ldots$
- Domaine de contraintes : D. Il détermine la sémantique d'une contrainte.
- Contrainte simple :
  - Symbole de relation avec arguments
  - p.e. $X \geq 42$, $X = Y + 2$
- Contrainte : conjonction de contraintes simples
  $C = c_1 \wedge c_2 \wedge \cdots \wedge c_k$ (par exemple $X \geq 42 \wedge X = Y + 2$)

Contraintes

- Arité d'une contrainte (unaire, binaire, ternaire, etc.)
- Contraintes numériques (linéaires), booléennes, de Herbrand, etc.
- Contraintes spéciales : $true$, $false$
- Affectation (totale, partielle)
- Une affectation θ viole une contrainte simple, si elle la rend fausse.
- Une affectation θ est consistante pour une contrainte générale, si elle ne viole aucune de ses contraintes simples.
- Solution : une affectation totale et consistante
  - p.e. $X \geq 42 \wedge X = Y + 2$ a une solution $\theta = \{X \leftarrow 43, Y \leftarrow 41\}$

Contraintes

- Une contrainte est satisfaisable, si elle a une solution.
- L'ordre des contraintes simples peut être important, certains algorithmes dépendent de l'ordre.
- Pour $C = c_1 \ \wedge \ c_2 \ \wedge \ \cdots \ \wedge \ c_k$ on définit $ensemble(C) = \{c_1, c_2, \ldots, c_k\}$.
- Deux contraintes sont équivalentes si elles ont le même ensemble de solutions.

Problème de satisfaction de contraintes (CSP)

- Les variables du problème avec leur domaines
- Une contrainte C

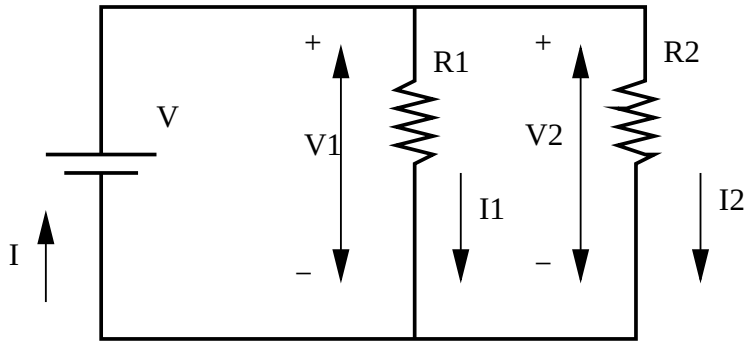
Questions :

- C est satisfaisable ?
- Donnez une solution, si C en a.

Un solutionneur de contraintes répond à la première question. Mais souvent aussi à la deuxième.

Modélisation avec contraintes

- Les contraintes décrivent le comportement idéalisé d'un système d'objets
- Exemple :



$V1 = I1 * R1 \ \wedge \ V2 = I2 * R2 \ \wedge \ V - V1 = 0 \ \wedge \ V - V2 = 0 \ \wedge$
$V1 - V2 = 0 \ \wedge \ I - I1 - I2 = 0 \ \wedge \ -I + I1 + I2 = 0$

Satisfaction de contraintes

- Comment résoudre le problème de satisfaction de contrainte ?
- Approche : essayer toutes les affectations
- ne marchera pas pour les réelles, entiers, etc.
- pour les domaines finis, on va essayer d'être plus intelligent.

Contraintes sur un domaine fini

- Solutionneur "génère et teste"
- Solutionneur par retour en arrière
- Consistance d'arc et de noeuds
- Heuristiques
- Consistance de bornes
- Consistance généralisée

Contraintes sur un domaine fini

- Une classe importante de domaine de contraintes
- Utilisée pour modéliser des problème avec des choix
- Ordonnancement, Emploi du temps, routage, etc.
- Beaucoup d'applications industrielles

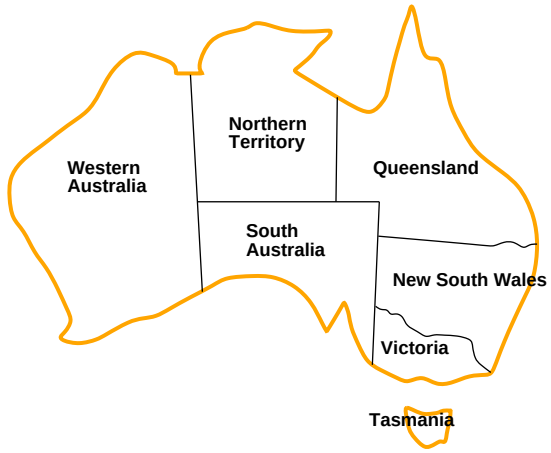
Problème de satisfaction de contraintes

- Une contrainte C sur des variables $x_1, \ldots, x_n$
- Un domaine $D(x_i)$ pour chaque variable
- Une contrainte C est implicitement donné par

$$C \wedge x_1 \in D(x_1) \wedge \cdots \wedge x_n \in D(x_n)$$

- Contraintes binaires : Graphe de contraintes

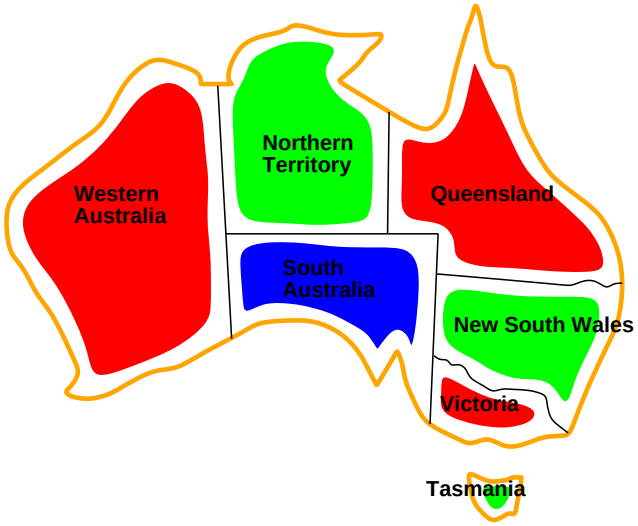
Exemple: Colorer une carte



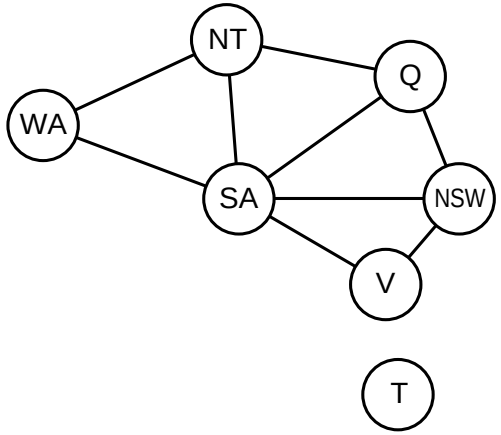
$WA \neq NT \wedge WA \neq SA \wedge NT \neq SA \wedge NT \neq Q \wedge SA \neq Q \wedge SA \neq V \wedge Q \neq NSW \wedge NSW \neq V$

$D(WA) = D(NT) = D(SA) = D(Q) = D(V) = D(NSW) = D(T) = \{rouge, jaune, bleu\}$

Exemple: Colorer une carte



Graphe de contraintes



Exemple: Les 4 reines

- Placer 4 reines sur un échiquier de taille 4x4 de sorte qu'aucune reine est en prise
- Quatre variables Q_1, Q_2, Q_3, Q_4 qui représentent la ligne de la reine dans chaque colonne. Domaine de chaque variable: $\{1, 2, 3, 4\}$
- Les contraintes: $Q1 \neq Q2 \ \wedge \ Q1 \neq Q3 \ \wedge \ Q1 \neq Q4 \ \wedge$ $Q2 \neq Q3 \ \wedge \ Q2 \neq Q4 \ \wedge \ Q3 \neq Q4$
- $Q1 \neq Q2 + 1 \ \wedge \ Q1 \neq Q3 + 2 \ \wedge \ Q1 \neq Q4 + 3 \ \wedge$ $Q2 \neq Q3 + 1 \ \wedge \ Q2 \neq Q4 + 2 \ \wedge \ Q3 \neq Q4 + 1$
- $Q1 \neq Q2 - 1 \ \wedge \ Q1 \neq Q3 - 2 \ \wedge \ Q1 \neq Q4 - 3 \ \wedge$ $Q2 \neq Q3 - 1 \ \wedge \ Q2 \neq Q4 - 2 \ \wedge \ Q3 \neq Q4 - 1$

Exemple: sac du contrebandier

- Contrebandier avec un sac de capacité 9.
- Il doit choisir des objets pour faire un profit d'au moins 30
objet	profit	taille
whisky	15	4
parfum	10	3
cigarettes	7	2

  $4W + 3P + 2C \leq 9 \ \wedge \ 15W + 10P + 7C \geq 30$
- Domaines des variables ?

Solutionneur génère et teste

- Le plus simple est d'énumérer les affectations possibles
- Le solutionneur génère et teste :
  - énumère une par une les valeurs des variables une par une
  - Quand chaque variable a une valeur, on teste, si la contrainte est satisfaite ou pas.
- On peut améliorer cette technique en testant à chaque fois, si l'affectation partielle entraîne déjà la non-satisfaisabilité.

Solutionneur simple par retour en arrière

- Le solutionneur simple par retour en arrière:
  - énumère une par une les valeurs des variables une par une
  - vérifie qu'aucune contrainte simple est fausse à chaque étape
  - On peut facilement tester la satisfaisabilité d'une contrainte simple sans variables
  - $partsat(C)$ retourne faux, si C n'est pas satisfaisable à cause d'une contrainte simple sans variables qui n'est pas satisfaisable. Sinon $partsat(C)$ retourne vrai.

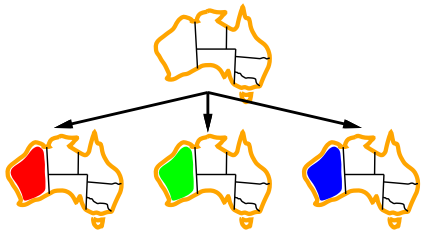
Solutionneur par retour en arrière

$backsolve(C, D)$

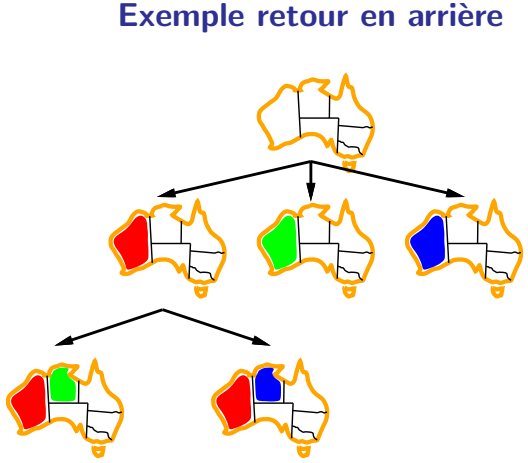
- Si $variables(C)$ est vide, alors retourne $partsat(C)$
- Choisir x dans $variables(C)$
- Pour chaque valeur d dans $D(x)$
  - Soit C_1 la contrainte C où x est remplacé par d
  - Si $partsat(C_1)$ alors
    - Si $backsolve(C_1, D)$ alors retourne $vrai$
- retourne $faux$

Exemple retour en arrière

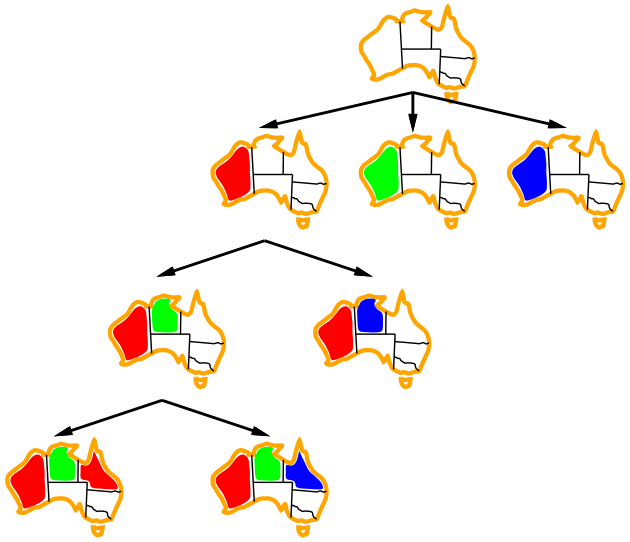
Exemple retour en arrière



Exemple retour en arrière



Exemple retour en arrière



Consistance de noeud et d'arc

- Idée: Trouver un CSP équivalent au CSP d'origine qui a des domaines de variables plus petits
- On considère les contraintes simples une par une
- Consistance de noeud : $(variables(c) = \{x\})$: enlever chaque valeur du domaine de x qui rend c insatisfaisable
- Consistance d'arc : $(variables(c) = \{x, y\})$: enlever chaque valeur de $D(x)$ pour laquelle il n'y a pas de valeur dans $D(y)$ qui satisfait c et vice-versa

Consistance de noeud

- Une contrainte simple c est noeud-consistante avec domaine D, si $|variables(c)| \neq 1$ ou
  - si $variables(c) = \{x\}$, alors pour chaque d dans $D(x)$, $x \leftarrow d$ est une solution de c
- Un CSP est noeud-consistant, si chaque contrainte simple est noeud-consistante

Comment obtenir un CSP noeud-consistant ?

$noeudcons(C, D)$

- Pour chaque contrainte simple c dans C
  - $D := noeudconssimple(c, D)$
- retourne D

$noeudconssimple(c, D)$

- Si $|variables(c)| = 1$ alors
  - Soit $\{x\} = variables(c)$
    $D(x) := \{d \in D(x) | \{x \leftarrow d\}$ est une solution de $c\}$
- retourne D

Arc consistance

- Une contrainte simple est arc-consistante avec domaine D, si $|variables(c)| \neq 2$ ou
  - $variables(c) = \{x, y\}$ et pour chaque d dans $D(x)$, il y a $e \in D(y)$ tel que $\{x \leftarrow d, y \leftarrow e\}$ est une solution de c
  - et d'une façon similaire pour y
- Un CSP est arc-consistant, si chaque contrainte simple est arc-consistante

Comment obtenir un CSP arc-consistant ?

$arcconssimple(c, D)$

- si $|variables(c)| = 2$ alors
  - $D(x) := \{d \in D(x) | \exists e \in D(y)$ t.q. $\{x \leftarrow d, y \leftarrow e\}$ est une solution de $c\}$
  - $D(y) := \{e \in D(y) | \exists d \in D(x)$ t.q. $\{x \leftarrow d, y \leftarrow e\}$ est une solution de $c\}$
- retourne D

Enlève des valeurs non arc-consistantes avec c

Comment obtenir un CSP arc-consistant ?

$arccons(C, D)$

- Répète
  - $W := D$
  - Pour chaque contrainte simple c de C
    - $D := arcconssimple(c, D)$
- jusqu'à $W = D$
- retourne D

version naïve

Utiliser noeud et arc consistance

- On peut définir des solutionneurs
- Deux domaines importants
  - domaine faux: une variable a un domaine vide
  - domaine simple: toutes les variables ont un domaine singleton (de taille un)
- étendre satisfaisabilité sur des CSPs avec des domaines simples

Solutionneur noeud et arc consistance

Solutionneur incomplet

- $D := noeudcons(C, D)$
- $D := arccons(C, D)$
- Si D est un domaine faux, alors retourne $faux$
- Si D est un domaine simple, alors retourne $satisfaisable(C, D)$
- sinon retourne $inconnu$

Comment définir un solutionneur complet en utilisant arc et noeud consistance ?

Retour en arrière avec consistance

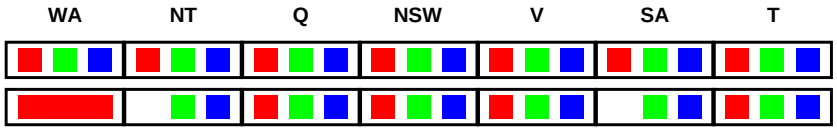
- Combiner le solutionneur par retour en arrière avec consistance
- Appliquer noeud (et/ou) arc consistance avant de lancer le solutionneur par retour en arrière et après chaque fois qu'une variable est affectée par le solutionneur

Exemple avec noeud consistance uniquement

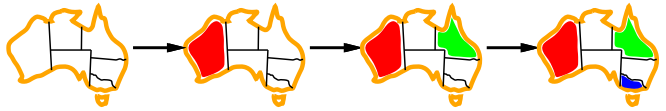
Exemple avec noeud consistance

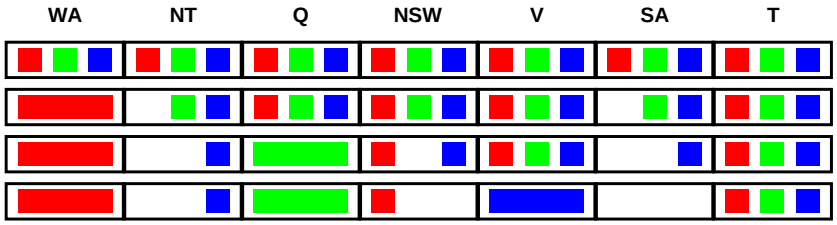
Ce problème est-t-il arc-consistant ?

Exemple avec noeud consistance



Exemple avec noeud consistance



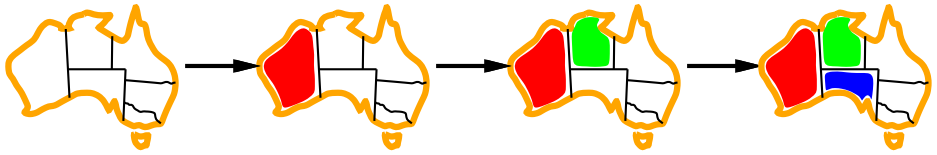


Heuristiques

- On peut utiliser des heuristiques pour choisir la variable à affecter et la valeur.
- statique/dynamique

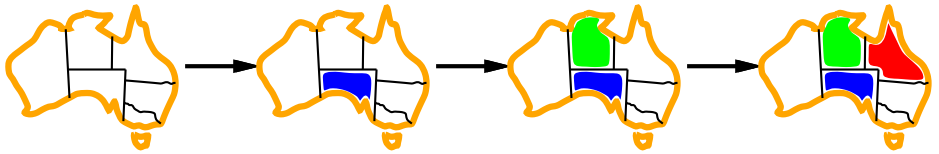
La variable la plus contrainte

- Choisir la variable avec le plus petit nombre de valeurs légales



La variable la plus contraignante

- En cas d'égalité pour la variable la plus contrainte
- Choisir la variable qui a le plus de contraintes avec les variables qui restent



La valeur la moins contraignante

- Pour une variable, choisir la valeur qui contraint le moins possible les variables qui restent