

Une introduction aux problèmes de satisfaction de contraintes

CNAM — Cycle C
Jérôme GENSEL
Université Pierre Mendès France
équipe SIGMA - Laboratoire LSR IMAG
e-mail : Jerome.Gensel@imag.fr
Tél : 04.76.82.72.80

1

Plan

- Introduction
- Eléments de la théorie des CSP
- CSP dynamiques
- CSP à intervalles
- Méthodes heuristiques
- Outils et langages de programmation par contraintes
- Conclusion

2

Introduction

3

Satisfaction de Contraintes

- Un ensemble de techniques de résolution de problèmes appelés Problèmes de Satisfaction de Contraintes ou CSP
- Mise en œuvre à travers des outils informatiques (langages, bibliothèques) de Programmation Par Contraintes (PPC)
- S'appuie sur une théorie élaborée dans le domaine de l'Intelligence Artificielle
- S'attaque à des problèmes en général combinatoires, dits NP-complets (pas d'algorithme de complexité polynomiale en la taille de leurs données)

4

Satisfaction de Contraintes

- Les origines
 - interfaces graphiques
 - Sketchpad 1963
 - ThingLab 1981
 - Intelligence Artificielle
 - reconnaissance d'objets 3 D par contour (scene labelling) 1972
 - la programmation logique
 - de l'unification à la résolution de contraintes 1985
 - Recherche opérationnelle et mathématiques discrètes
 - Problèmes combinatoires NP-Complets

5

Satisfaction de Contraintes

- Théorie de la satisfaction de contraintes
 - domaine de l'Intelligence Artificielle qui a vu le jour en 1977 en évolution constante depuis (forte croissance depuis 1990 liée à la demande industrielle)
- Son domaine d'étude
 - les algorithmes de consistance
 - les algorithmes de résolution
 - propriétés particulières de certains problèmes de satisfaction de contraintes (CSP)

6

Satisfaction de Contraintes

- D'abord appliquée à des problèmes 'jouets' (SEND+MORE = MONEY , N-reines, ...), maintenant orientée sur des CSP de taille réelle (aménagement intérieur d'un avion à l'Aérospatiale, planification du réseau France Telecom...)
- Les techniques de satisfaction de contraintes sont issues d'une théorie qui s'échafaude en intelligence artificielle en empruntant des résultats de :
 - la théorie des graphes
 - l'arithmétique des intervalles
 - l'analyse numérique
 - la recherche opérationnelle

7

Programmation par Contraintes

- Citation : "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 Freuder, 1997
- Objectif :
 - fournir un moyen d'exprimer dans un formalisme déclaratif (dans un langage de modélisation souple, proche des concepts de l'utilisateur) les contraintes qui définissent un problème (composant langage)
 - fournir les algorithmes capables de le résoudre, lorsque c'est possible (composant solveur)

8

Un domaine d'âge mûr

- Aujourd'hui, la Satisfaction de Contraintes et sa mise en œuvre informatique, la Programmation par Contraintes, ont atteint une certaine maturité (recherche + applications industrielles)
- Grâce à des outils comme Ilog Solver, CHIP, Prolog IV, de nombreux secteurs économiques y ont recours

9

Quels types de problèmes ?

- Planification et ordonnancement : planification d'itinéraires, ordonnancement d'ateliers
- Gestion du temps : emploi du temps, rotations d'équipes ...
- Gestion et affectation de ressources : gestion de personnel, de moyens de transport ...
- Optimisation : optimisation de production, d'investissements, de coûts ...
- Gestion de la cohérence d'informations : interfaces graphiques, édition électronique ...
- Langage naturel : relations syntaxiques, sous-catégorisation
- Jeux...

10

Dans quels secteurs d'activité ?

- Industrie : ordonnancement (Sextant), transport de matières nucléaires (CEA) ...
- Transport : rotation d'équipages (Corsair), gestion de port (port de Singapour) ...
- Télécommunications : composants visuels (Hewlett-Packard) ...
- Biologie moléculaire : prédiction de structure protéique (Esprit) ...
- Environnement : pollution des eaux (ville de Venise), recyclage (Michelin) ...
- Banques : aide à l'attribution de crédits (Crédit Lyonnais, Société Générale) ...

11

Eléments de la théorie des CSP

12

Qu'est-ce qu'un CSP ?

- Définition [Mackworth, 1977]
 - un CSP est un triplet (X, D, C) où :
 - $X = \{x_1, \dots, x_n\}$: les variables du CSP
 - $D = \{d_1, \dots, d_n\}$: les domaines des variables du CSP avec $d_i = \text{dom}(x_i)$
 - $C = \{c_1, \dots, c_m\}$: les contraintes du CSP avec $c_j \in C$ et $\text{var}(c_j) = \{x_{j_1}, \dots, x_{j_k}\} \in X$

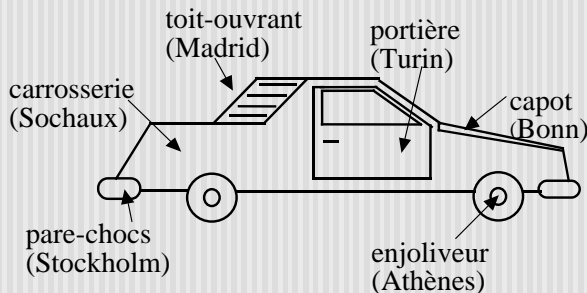
13

Qu'est-ce qu'un CSP ?

- variables de type entier, réel (flottant), rationnel, booléen, objet, ensemble,...
- domaines finis (énumérés ou intervalles) ou infinis (intervalles) ou continus
- contraintes décrites
 - en extension : sous-ensemble des k-uplets $(v_{j_1}, \dots, v_{j_k})$ du produit cartésien $d_1 \times d_2 \times \dots \times d_k$ qui satisfont c_j (où $v_{j_i} \in \text{dom}(x_{j_i})$ et $x_{j_i} \in \text{var}(c_j)$)
 - exemple : $C = \{c : \{(5, 3, 2), (4, 3, 1)\}\}$,
 - $X = \{x_1, x_2, x_3\}$, $D = \{\{5, 4\}, \{3\}, \{1, 2, 3\}\}$
 - en intension : expression mathématique entre les x_{j_1}, \dots, x_{j_k}
 - exemple : $C = \{c : x_1 = x_2 + x_3\}$, $X = \{x_1, x_2, x_3\}$,
 - $D = \{\{5, 4\}, \{3\}, \{1, 2, 3\}\}$

14

Un exemple : la voiture européenne



- d'après C. Bessière

15

Un exemple : la voiture européenne

- Enoncé : Peut-on construire une voiture européenne qui réponde aux critères esthétiques suivants :
 - Les pare-chocs, les enjoliveurs et le toit-ouvrant doivent être plus clairs que la carrosserie, et
 - Les portières, la carrosserie et le capot doivent être de la même couleur

16

Un exemple : la voiture européenne

- Sachant que sur chaque site de fabrication d'une pièce de l'automobile, on ne dispose que de certaines teintes :
 - à Stockholm: blanc
 - à Madrid : rouge
 - à Athènes : rose et rouge
 - à Sochaux : blanc, rose, rouge et noir
 - à Bonn et à Turin : rose, rouge et noir

17

Modélisation en termes de CSP

- Déterminer le triplet (X, D, C)
 - Les variables sont les 6 pièces à assembler (les données) et leur domaine contiennent couleurs possibles associées (les valeurs)
 - pare-chocs → domaine = {blanc}
 - toit-ouvrant → domaine = {rouge}
 - enjoliveur → domaine = {rose, rouge}
 - carrosserie → domaine = {blanc, rose, rouge, noir}
 - portière → domaine = {rose, rouge, noir}
 - capot → domaine = {rose, rouge, noir}

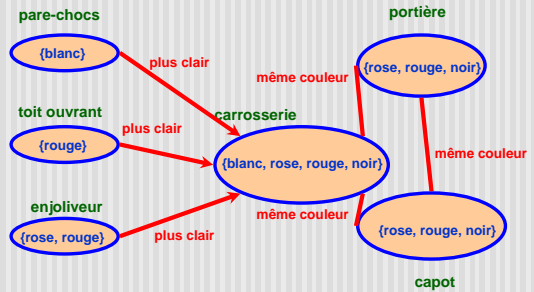
18

Modélisation en termes de CSP

- Déterminer le triplet (X, D, C)
 - Les contraintes : des critères esthétiques....
 - pare-chocs, enjoliveurs et toit-ouvrant plus clairs que la carrosserie → 3 contraintes
 - portières, carrosserie et capot de la même couleur → 3 contraintes

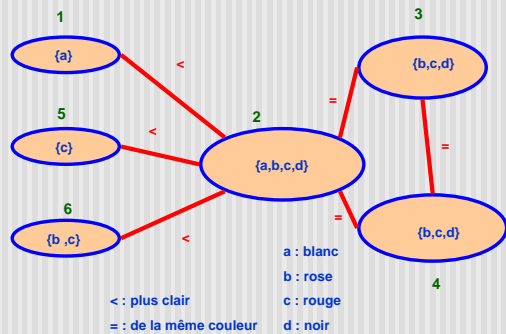
19

Modélisation graphique symbolique



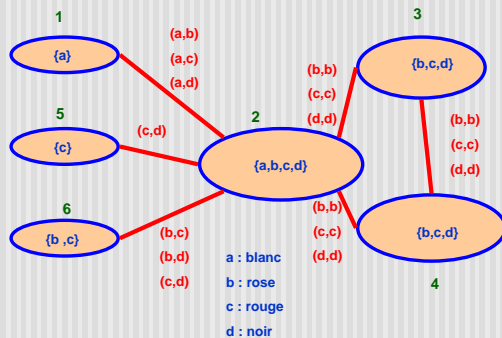
20

Modélisation graphique alphanumérique



21

Modélisation graphique alphanumérique



22

Bilan de la modélisation

- Au total
 - CSP symbolique ou alphanumérique à 6 variables, 6 domaines et 6 contraintes
- Questions équivalentes
 - Peut-on peindre cette voiture ?
 - Ce modèle est-il concevable en fonction des contraintes et des ressources ?
 - Le CSP a-t-il une solution ?
- Vers la résolution

23

Résoudre un CSP

- Résoudre un CSP (X, D, C) , c'est trouver, s'ils existent, les n-uplets de valeurs (v_1, \dots, v_n) , avec $v_i \in d_i = \text{dom}(x_i)$, $x_i \in X$, $d_i \in D$, qui satisfont toutes les contraintes $c_j \in C$
- Des énoncés dérivés
 - Existe-t-il une solution ?
 - Telle valeur figure-t-elle dans une solution ?
 - Trouver toutes les valeurs possibles pour une variable
 - Trouver la "meilleure" solution \Rightarrow fonction de coût
- Toutes ces tâches sont, en général, NP-complètes

24

De l'instanciation à la solution

- **Instanciation partielle de $\{x_1, \dots, x_k\}$**
 - affectation de valeurs aux variables x_1, \dots, x_k , $k < n$
 - $\{(1,a), (2,b)\}$ et $\{(1,a), (2,c)\}$ sont des instanciations partielles des variables 1 et 2 du CSP donné en exemple
- **Solution**
 - instanciation totale (sur X) de toutes les variables telle que toutes les contraintes de C soient satisfaites
 - $\{(1,a) ; (2,d) ; (3,d) ; (4,d) ; (5,c) ; (6,b)\}$ est une solution du CSP donné en exemple

25

Techniques de résolution

- But : énumérer les domaines des variables afin de produire, si elle(s) existe(nt), la (ou les) solution
- Ces algorithmes sont des méthodes exhaustives d'énumération plus ou moins "intelligentes", plus ou moins performantes
-
- 2 types d'algorithmes de recherche de solution
 - les algorithmes rétrospectifs
 - les algorithmes prospectifs

26

Algorithmes rétrospectifs

- Les algorithmes rétrospectifs
 - backtrack
 - backjump
 - backchecking
 - backmarking
 - dynamic backtracking
 - ...
- Principe commun
 - Les instanciations des variables sont construites incrémentalement, lorsqu'il n'y a plus de valeurs possible pour une variable en cours d'instanciation, on revient en arrière

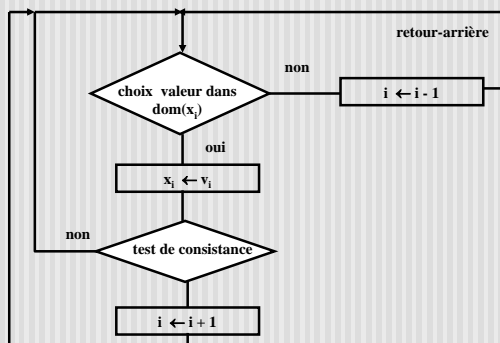
27

Le backtrack chronologique

- Principe
 - Tenter d'étendre l'instanciation $(v_1, v_2, \dots, v_{i-1})$ des variables x_1, x_2, \dots, x_{i-1} en cherchant une valeur v_i du domaine de x_i compatible avec l'instanciation $(v_1, v_2, \dots, v_{i-1})$.
 - Si une telle valeur v_i n'existe pas alors on effectue un retour arrière à la variable x_{i-1}
 - Sinon on poursuit sur x_{i+1}

28

Le backtrack chronologique



29

Arbre de recherche (backtrack)

- sur l'exemple
 - (1,a)
 - (2,a) x
 - (2,b)
 - (3,b) x
 - (3,c) x
 - (3,d) x
 - (2,c)
 - (3,b) x
 - (3,c) x
 - (3,d) x
 - (2,d)
 - (3,b) x
 - (3,c) x
 - (3,d) x
 - (4,b) x
 - (4,c) x
 - (4,d) x
 - (5,c) x
- méthode peu efficace...

30

Améliorations du backtrack

- Tirer parti des informations implicites contenues dans les situations d'échec
 - pour économiser plus tard des essais de valeurs (conflict-set, backchecking, backmarking)
 - pour sélectionner un meilleur point de retour (backjumping)

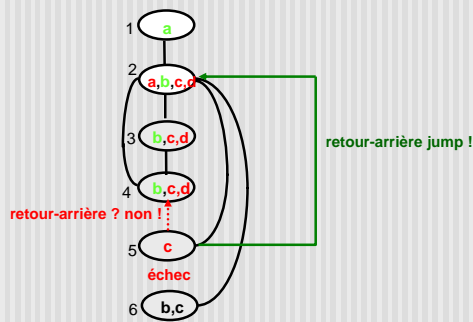
31

Le backjump

- Principe
 - en cas d'échec sur la variable en cours d'instanciation, trouver la variable déjà instanciée la plus judicieuse pour effectuer un retour arrière et prendre la valeur suivante dans son domaine
 - Quelle est cette variable ?
 - pour chacune des valeurs, déterminer la plus proche variable dans chaque contrainte violée
 - pour chacune des valeurs, sélectionner la variable la plus éloignée
 - dans ces variables choisir la plus proche

32

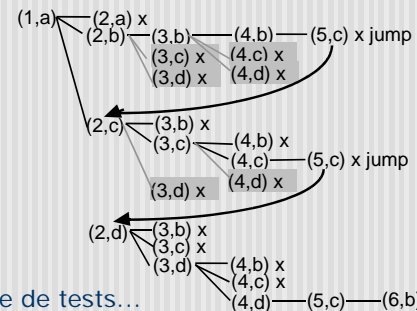
Le backjump



33

Arbre de recherche (backjump)

- sur l'exemple



- économie de tests...

34

Algorithmes prospectifs

- les algorithmes prospectifs
 - forward-checking
 - partial look-ahead
 - full look-ahead...
- Ces algorithmes se distinguent par la force du filtrage appliqué
- Principe commun
 - A chaque étape 1...k...n
 - 1) instancier la variable x_k
 - 2) filtrer les domaines des variables $x_i, i > k$
 - 3) si $\exists i > k$ t.q. $\text{dom}(x_i) = \emptyset$ alors on essaie la valeur suivante pour x_k s'il y en a une, sinon retour-arrière

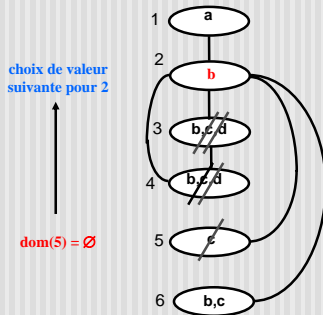
35

Le Forward-Checking

- Principe
 - A chaque étape 1...k...n
 - 1) instancier la variable x_k
 - 2) $\forall x_i, i > k$ et $\forall c_j, x_i$ et $x_k \in \text{var}(c_j)$ enlever toutes les valeurs de x_i incompatibles avec la valeur v_k choisie pour x_k
 - 3) si $\exists i > k$ t.q. $\text{dom}(x_i) = \emptyset$ alors on essaie la valeur suivante pour x_k s'il y en a une, sinon retour-arrière

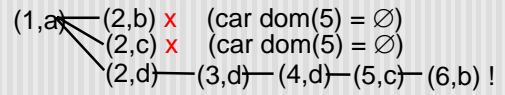
36

Le Forward-Checking



37

Arbre de recherche (forward-checking)



- efficace aussi...

38

Le Partial Look-Ahead

■ Principe

- A chaque étape 1...k...n
 - 1) instancier la variable x_k
 - 2) $\forall x_i, i > k$ et $\forall c_j, x_i$ et $x_k \in \text{var}(c_j)$ enlever toutes les valeurs de x_i incompatibles avec la valeur v_k choisie pour x_k
 - 2') $\forall x_i, \forall x_j, \forall c_i, x_i$ et $x_j \in \text{var}(c_j), i > j$ et $j \geq k$, enlever toutes les valeurs de x_j qui n'ont pas de valeurs compatibles dans x_i
 - 3) si $\exists i$ t.q. $\text{dom}(x_i) = \emptyset$ alors on essaie la valeur suivante pour x_k s'il y en a une, sinon retour-arrière

39

Le Full Look-Ahead

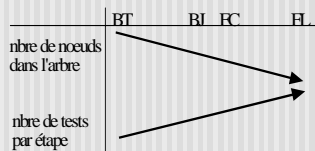
■ Principe

- A chaque étape 1...k...n
 - 1) instancier la variable x_k
 - 2) $\forall x_i, i > k$ et $\forall c_j, x_i$ et $x_k \in \text{var}(c_j)$ enlever toutes les valeurs de x_i incompatibles avec la valeur v_k choisie pour x_k
 - 2') $\forall x_i, \forall x_j, \forall c_i, x_i, x_j \in \text{var}(c_j), i \geq k$ et $j \geq k$, enlever toutes les valeurs de x_j qui n'ont pas de valeurs compatibles dans x_i (x_j)
 - 3) si $\exists i$ t.q. $\text{dom}(x_i) = \emptyset$ alors on essaie la valeur suivante pour x_k s'il y en a une, sinon retour-arrière

40

Complexité

- Ces algorithmes se distinguent par la force du filtrage appliqué



- Un filtre puissant réduit la taille de l'espace de recherche mais ralentit la recherche...

41

Ordonnement

- L'ordre de choix des variables ou des valeurs influe sur le temps de calcul d'une solution
- Ordonnement des variables
 - « First fail principle » : choisir dans le CSP la variable qui a le plus de chances de conduire à un échec pour minimiser les coûts de retour-arrière...
 - Souvent, on prend la variable x pour laquelle le ratio $|\text{dom}(x)|/|\text{contr}(x)|$ est le plus petit
- Ordonnement des valeurs (+ délicat)
 - Le type des contraintes est à considérer et, plus généralement, il faut avoir une bonne expérience du CSP...

42

Ordonnement statique

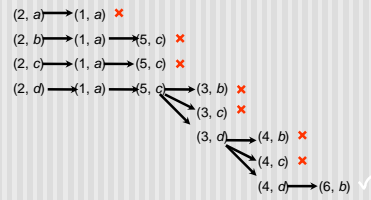
- Exemple avec $|dom(x)|/|contr(x)|$ croissant
- résolution avec le Backtrack Chronologique
- ordonnancement obtenu

2(4/5), 1(1/1), 5(1/1), 3(3/2), 4(3/2), 6(2/1)

43

Ordonnement statique

2(4/5), 1(1/1), 5(1/1), 3(3/2), 4(3/2), 6(2/1)



44

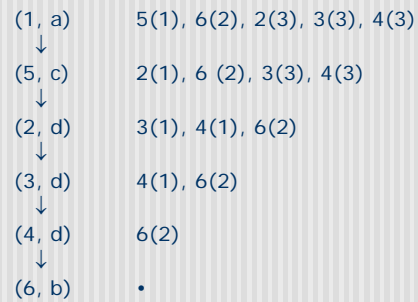
Ordonnement dynamique

- Exemple avec plus petit domaine,
 - résolution avec le Forward-Checking
 - au départ
- 1(1), 5(1), 6(2), 3(3), 4(3), 2(4)

45

Ordonnement dynamique

1(1), 5(1), 6(2), 3(3), 4(3), 2(4)



46