Problèmes de satisfaction de contraintes

1

Plan

- Description des CSP
- L'exploration avec backtracking
- La méthode du Forward checking
- Cohérence des arcs
- Gestion de contraintes spécifiques
- Recherche locale
- Utiliser la structure des problèmes

2

Problèmes de satisfaction de contraintes

- Définit par:
  - Un ensemble de variables: $X_1, X_2, \ldots, X_n$
  - Un ensemble de contraintes: $C_1, C_2, \ldots, C_m$
- Chaque variable X_i a un domaine non vide de valeurs possibles.
- Un état est une affectation de valeurs pour quelques unes ou toutes les variables.
- Affectation consistante: aucune violation de contraintes.
- Affectation complète: toutes les variables ont une valeur.
- Solution: Affectation complète et consistante.
- Si solution optimale, il convient d'ajouter une fonction objective

3

Exemple: coloration de carte

- Variables: *WA*, *NT*, *SA*, *Q*, *NSW*, *V*, *T*
- Domaines: D_i = {*rouge*, *vert*, *bleu*}
- Contraintes: Les régions adjacentes doivent avoir des couleurs différentes.
  - WA ≠ NT
  - ou, (WA, NT) ε {(rouge, vert), (rouge, bleu), etc.}

4

Exemple: coloration de carte

- Solution: Une affectation qui satisfait toutes les contraintes:
  - {*WA = rouge*, *NT = vert*, *SA = bleu*, *Q = rouge*, *NSW = vert*, *V = rouge*, *T = vert*} en est une

5

Types de CSP

- Variables discrètes
  - Domaines finis; nombre de variables *n* et taille max du domaine de n'importe quelle variable est *d*; alors O(d^n) affectations complètes.
    - CSP booléens
  - Domaines infinis (entiers, chaînes de caractères, etc.)
    - Ex: Ordonnancement de tâches; les variables sont les jours de début et de fin des tâches.
    - Demande un langage de contraintes, ex. $débutTâche_1 + 5 < débutTâche_3$
    - Contraintes linéaires: soluble; contraintes non linéaires: non décidable

6

Types de CSP

- Variables continues
  - Ex: temps de début et de fin des observations du télescope Hubble
  - Contraintes linéaires: soluble en temps polynomial par rapport au nombre de variables.

7

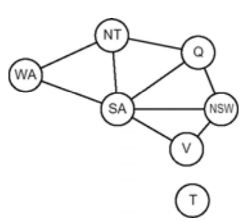
Types de contraintes

- Unaire: Les contraintes ne concernent qu'une variable.
  - Ex: *SA ≠ vert*
- Binaire: Les contraintes concernent deux variables.
  - Ex: *SA ≠ WA*
- Ordre plus élevé: Les contraintes concernent 3 variables ou plus.
  - Ex: cryptarithmétique
- Contraintes de préférence
  - Ex: *rouge* est mieux que *vert*
  - Souvent représenté par un coût pour chaque affectation de variable
  - Problèmes d'optimisation de contraintes

8

CSP binaire

- CSP où toutes les contraintes sont reliées à deux variables.
- Peuvent être représentés sous forme de graphe.
- Certains algorithmes généraux de résolution de CSP utilisent la structure de graphe pour accélérer la recherche.
  - Ex: Tasmania est un sous problème indépendant.

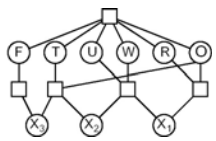


9

Cryptarithmétique

- Variables: $F, T, U, W, R, O, X_1, X_2, X_3$
- Domaines: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
- Contraintes:
  - Différentes(F, T, U, W, R, O)
  - $O + O = R + 10 * X_1$
  - $X_1 + W + W = U + 10 * X_2$
  - $X_2 + T + T = O + 10 * X_3$
  - $X_3 = F$

```
   T W O
 + T W O
 -------
 F O U R
```



10

Cryptarithmétique

```
   S E N D
   9 5 6 7
 + M O R E
   1 0 8 5
  ----------
 = M O N E Y
   1 0 6 5 2
```

11

Applications des CSP

- Problèmes d'assignation
  - Ex: Qui enseigne quel cours?
- Problèmes d'horaire
  - Ex: Quelle classe est offerte, quand et où?
- Planification pour le transport
- Planification à l'intérieur d'une usine

12

Formulation de recherche standard

- États: les valeurs assignées jusqu'à maintenant
- État initial: l'affectation vide {}
- Fonction successeurs: Affecter une valeur à une variable non affectée qui ne crée pas de conflit
- Test de but: l'affectation est complète
- Commentaires:
  - Même formulation pour tous les CSP
  - Toutes les solutions à n variables apparaissent à la profondeur n.
    - Profondeur d'abord (cette recherche profite de l'avantage de la commutativité en CSP)
  - Le chemin n'est pas important, donc on peut aussi utiliser des algorithmes itératifs.

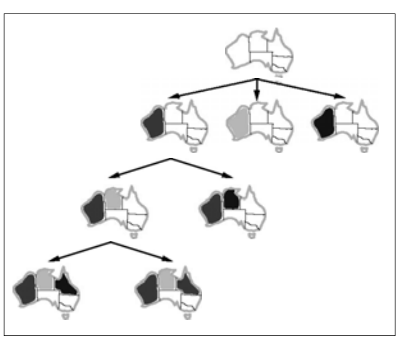
13

Recherche à retour arrière

- L'assignation des variables est commutative
  - Ex: [*WA* = *rouge* suivi de *NT* = *vert*] est la même chose que [*NT* = *vert* suivi de *WA* = *rouge*]
- À chaque nœud, on a seulement besoin de considérer l'affectation d'une seule variable.
  - $b = d$ et il y a d^n feuilles
- La recherche en profondeur d'abord pour les CSP avec l'assignation d'une seule variable à chaque tour est appelée exploration avec backtracking.
- Le retour arrière s'effectue lorsqu'il n'y a plus d'affectations possibles (on retourne à la variable précédente et on essaye une autre valeur).

14

Exemple retour arrière



15

Améliorer l'efficacité de la recherche par retour arrière

- Quelle variable devrait être assignée et dans quel ordre ses valeurs devraient être essayées ?
- Quelles sont les implications des affectations de la variable courante sur les autres variables non affectées ?
- Lorsqu'un chemin échoue, est-ce que la recherche pourrait éviter cet échec dans les chemins suivants ?

16

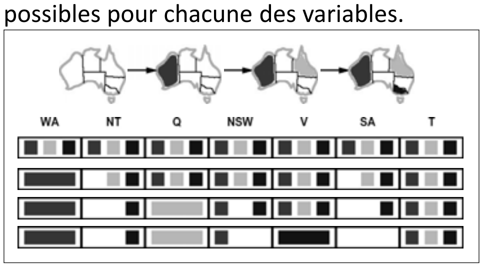
Ordre des variables et des valeurs

- Choix des variables:
  - Heuristique du nombre de valeurs restantes minimum (minimum remaining values (MRV)).
    - Choisir la variable avec le moins de valeurs possibles.
  - Heuristique du degré
    - Choisir la variable présente dans le plus de contraintes.
- Choix des valeurs:
  - Heuristique des valeurs les moins contraignantes (least-constraining-value)
    - Choisir la valeur qui va enlever le moins de choix pour les variables voisines.

17

Forward checking

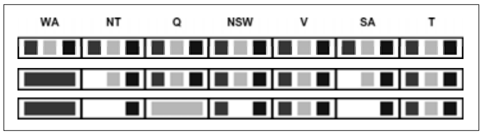
- Maintient en tout temps, toutes les valeurs possibles pour chacune des variables.



18

Propagation de contraintes

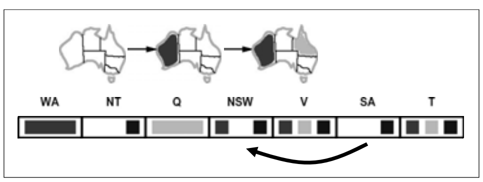
- Le « forward checking » ne permet pas de détecter tous les problèmes.



- NT et SA ne peuvent pas tous les deux être bleus!
- La propagation de contraintes renforce rapidement les contraintes locales.

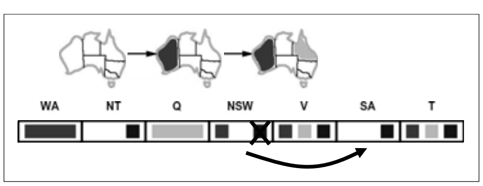
Cohérence des arcs

- Un arc entre *X* et *Y* est cohérent si pour toutes les valeurs *x* de *X*, il y a au moins une valeur possible *y* de *Y*.



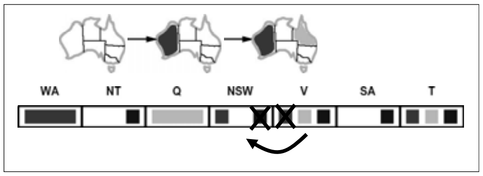
Cohérence des arcs

- Un arc entre *X* et *Y* est cohérent si pour toutes les valeurs *x* de *X*, il y a au moins une valeur possible *y* de *Y*.



Cohérence des arcs

- Un arc entre *X* et *Y* est cohérent si pour toutes les valeurs *x* de *X*, il y a au moins une valeur possible *y* de *Y*.

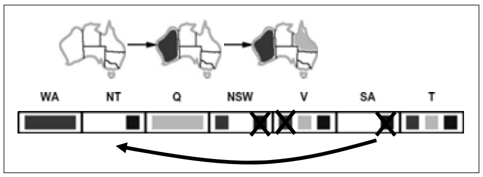


- Si X perd une valeur, les voisins de X doivent être revérifiés

22

Cohérence des arcs

- Un arc entre *X* et *Y* est cohérent si pour toutes les valeurs *x* de *X*, il y a au moins une valeur possible *y* de *Y*.



23

Cohérence des arcs

- Détecte les erreurs plus vite que le « forward checking »
- Peut être exécuté comme un pré-processus ou après chaque assignation.

24

Gestion de contraintes spécifiques

- Certains types de contraintes surviennent souvent dans les problèmes réels
  - On peut donc utiliser des algorithmes spécifiques à ces types de contraintes qui sont plus performant que les algorithmes généraux.

Gestion de contraintes spécifiques

- Contrainte *toutes différentes*
  - Toutes les variables doivent avoir des valeurs distinctes.
  - S'il y a *m* variables, *n* valeurs possibles et *m* > *n*, alors les contraintes ne peuvent pas être satisfaites.
  - Algorithme:
    - Enlever toutes les variables qui n'ont qu'une valeur dans leur domaine et enlever cette valeur de toutes les autres variables restantes.
    - Continuer tant qu'il y a des variables avec un domaine contenant qu'une seule valeur.
    - Si un domaine vide est produit, ou qu'il y a plus de variables que de valeurs disponibles, alors une incohérence a été détectée.

Gestion de contraintes spécifiques

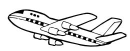
- Contrainte sur les ressources
  - Limite sur le nombre de ressources. Par exemple:
    - Il y a un maximum de 10 personnes pour effectuer 4 tâches.
    - Si le domaine de chacune des tâches est {3, 4, 5, 6}, alors les contraintes ne pourront pas être satisfaites.
    - On peut aussi réduire les domaines de valeurs. Par exemple, si le domaine des tâches est {2, 3, 4, 5, 6}, alors les valeurs 5 et 6 peuvent être enlevées des domaines.

Gestion de contraintes spécifiques

- Propagation de bornes
  - Lorsque les domaines sont plus grands, on utilise la propagation de bornes. Par exemple:

Capacité: 165 passagers

Domaine: [0, 165]



Capacité: 385 passagers

Domaine: [0, 385]

Contrainte: 420 passagers à transporter

Nouveau domaine: [35, 165]

Nouveau domaine: [255, 385]

28

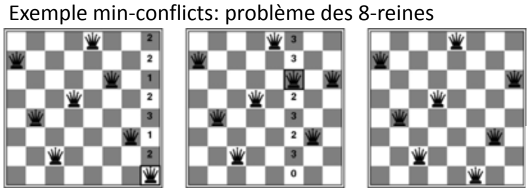
Recherche local pour les CSP

- Fonctionnement:
  - On commence en assignant une valeur à chacune des variables.
  - Ensuite, la fonction de successeurs change la valeur d'une variable à la fois.
- Heuristique « min-conflicts »
  - Choisir la valeur qui donne le moins de conflits
- Utile pour les problèmes « online ».

29

Recherche local pour les CSP

- Exemple min-conflicts: problème des 8-reines

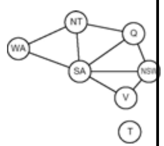


- Avec min-conflicts, peut résoudre 1 millions de reines en 50 étapes, en moyenne.
- Pour le télescope Hubble, le temps de la planification d'une semaine est passé de trois semaines à 10 minutes en utilisant min-conflicts!

30

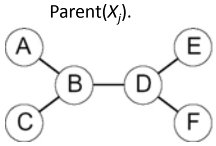
La structure des problèmes

- Utiliser la structure du graphe des contraintes pour trouver des solutions rapidement.
- Les composantes connexes sont des sous-problèmes indépendants.
  - On résout les sous-problèmes indépendamment
  - Supposons que chaque sous-problème contient *c* variables sur un total de *n*, alors le coût de la solution en pire cas est de $n/c * d^c$. (linéaire en *n*)
    - Si $n = 80$, $d = 2$ et $c = 20$
    - d^{80} = 4 milliards d'années à 10 millions de nœuds/sec
    - $4 * 2^{20}$ = 0,4 secondes à 10 millions de nœuds/sec



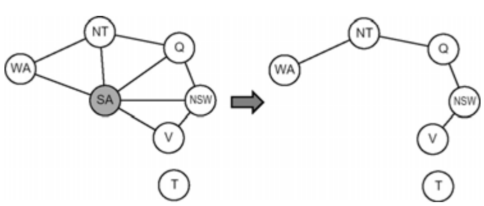
La structure des problèmes

- Si le graphe ne contient pas de cycle, alors le CSP peut être résolut en $O(nd^2)$
  - Choisir un nœud comme racine et ordonner les autres nœuds de manière à ce que le parent précède toujours.
  - Pour j de n à 2, EnleverIncohérence(Parent(X_j), X_j)
  - Pour j de 1 à n, affecter une valeur à X_j consistante avec Parent(X_j).



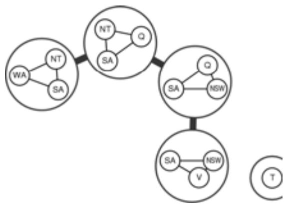
La structure des problèmes

- Si la structure est proche d'un arbre, on peut fixer la valeur de certains nœuds pour obtenir une structure d'arbre.



La structure des problèmes

- Décomposition en un arbre de sous problèmes
  - Chaque variable du problème original doit être dans au moins un sous problème
  - Chaque contrainte doit être dans au moins un sous problème
  - Si une variable apparaît plus d'une fois, les sous problèmes doivent être collés
- *O(nd^y)* où *y* est la grandeur du plus grand sous problème



34