Résolution de Problèmes
Programmation par Contraintes

Marie Pelleau
marie.pelleau@univ-cotedazur.fr

Notes

Algorithme glouton

Principe

- À chaque étape, on fait un choix, celui qui semble le meilleur à cet instant
- Construit une solution pas à pas
  - sans revenir sur ses décisions
  - en effectuant à chaque étape le choix qui semble le meilleur
  - en espérant obtenir un résultat optimum global
- Approche gloutonne
  - suivant les problèmes pas de garantie d'optimalité (heuristique gloutonne)
  - peu coûteuse (comparée à une énumération exhaustive)
  - choix intuitif

Notes

Recherche locale

Principe

- On part d'une solution initiale
- À chaque étape, on modifie la solution
  - en essayant d'améliorer la valeur de la fonction objectif
  - en espérant obtenir l'optimum global
- Approche locale
  - suivant les problèmes pas de garantie d'optimalité (heuristique)
  - peu coûteuse

Remarque

S'il n'existe pas de fonction objectif *Constraint Based Local Search*

Notes

Programmation par Contraintes

- Recherche Arborescente
  - trouver une solution
  - trouver l'ensemble des solutions
  - trouver une solution optimale
  - prouver la non existence de solution
- Approche complète
  - garantie d'optimalité
  - plus coûteuse

Notes

Send More Money

Description

$$\begin{array}{r} S\,E\,N\,D \\ +\ \ M\,O\,R\,E \\ \hline M\,O\,N\,E\,Y \end{array}$$

Contraintes possibles

$$\begin{array}{lrcccccccc} & & & s{*}1000 & + & e{*}100 & + & n{*}10 & + & d \\ C_1: & & + & m{*}1000 & + & o{*}100 & + & r{*}10 & + & e \\ & = m*10000 & + & o*1000 & + & n{*}100 & + & e{*}10 & + & y \end{array}$$

$C_2: s \neq e$	$C_3: s \neq n$	$C_4: s \neq d$	$C_5: s \neq m$	$C_6: s \neq o$
$C_7: s \neq r$	$C_8: s \neq y$	$C_9: e \neq n$	$C_{10}: e \neq d$	$C_{11}: e \neq m$
$C_{12}: e \neq o$	...	$C_{27}: o \neq r$	$C_{28}: o \neq y$	$C_{29}: r \neq y$

Notes

Programmation par Contraintes

Send More Money

Description

$$\begin{array}{r} r_4\,r_3\,r_2\,r_1\ \ \\ S\,E\,N\,D \\ +\ \ M\,O\,R\,E \\ \hline M\,O\,N\,E\,Y \end{array}$$

Contraintes possibles

$$\begin{array}{lrl} C_1: & d + e = y + 10 * r_1 & r_1 \in \{0,1\} \\ C_2: & r_1 + n + r = e + 10 * r_2 & r_2 \in \{0,1\} \\ C_3: & r_2 + e + o = n + 10 * r_3 & r_3 \in \{0,1\} \\ C_4: & r_3 + s + m = o + 10 * r_4 & r_4 \in \{0,1\} \\ C_5: & r_4 = m & \end{array}$$

$C_6: s \neq e$	$C_7: s \neq n$	$C_8: s \neq d$	$C_9: s \neq m$	$C_{10}: s \neq o$
$C_{11}: s \neq r$	$C_{12}: s \neq y$	$C_{13}: e \neq n$	$C_{14}: e \neq d$	$C_{15}: e \neq m$
$C_{16}: e \neq o$	...	$C_{31}: o \neq r$	$C_{32}: o \neq y$	$C_{33}: r \neq y$

Programmation par Contraintes 5 / 41

Notes

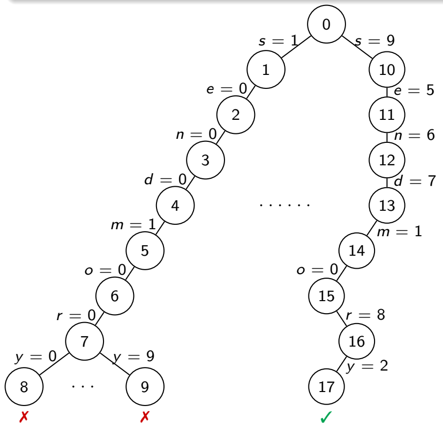
Comment résoudre un CSP ?

Notes

Generate and Test

Méthode Naïve

Générer toutes les affectations possibles et vérifier si elles correspondent à des solutions



Remark

Pour trouver une seule solution, on va générer :

- $9^2 * 10^6 = 81\ 000\ 000$ feuilles avec la première modélisation
- $2^4 * 9^2 * 10^6 = 1\ 296\ 000\ 000$ avec la seconde

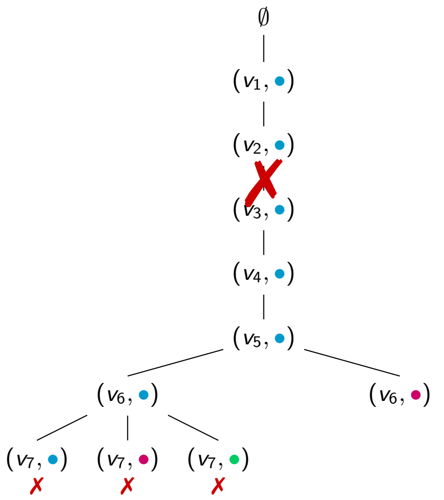
On peut faire mieux ?

Notes

Generate and Test

Coloriage de carte

- $\mathcal{V} = \{v_1, \ldots, v_7\}$
- $D_1 = \cdots = D_7 = \{\bullet, \bullet, \bullet\}$
- $C_1 : v_1 \neq v_2$
  $C_2 : v_1 \neq v_3$
  $C_3 : v_2 \neq v_3$
  $C_4 : v_2 \neq v_4$
  $C_5 : v_3 \neq v_4$
  $C_6 : v_3 \neq v_5$
  $C_7 : v_3 \neq v_6$
  $C_8 : v_4 \neq v_5$
  $C_9 : v_5 \neq v_6$
  $C_{10} : v_6 \neq v_7$

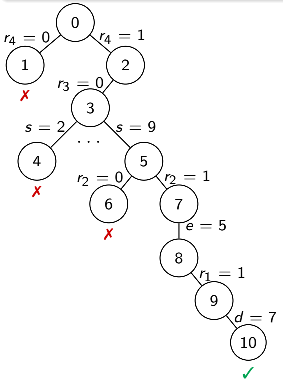


Notes

Forward Checking

Dès qu'une variable est affectée on essaye de filtrer les valeurs pour les autres variables

- On remplace la variable par sa valeur dans toutes les contraintes
- On peut filtrer si une contrainte ne contient plus qu'une variable



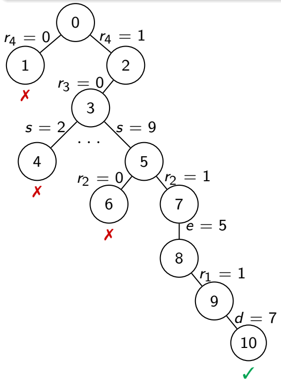
- $\{C_4 : r_3 + s + m = o + 10 * r_4,$
  $C_5 : r_4 = m\}$
  - $r_4 = 0$
    $\Rightarrow r_3 + s + m = o \wedge m = 0$ ✗
  - $r_4 = 1$
    $\Rightarrow r_3 + s + m = o + 10 \wedge m = 1$
    $\Rightarrow r_3 + s + 1 = o + 10$

Notes

Forward Checking

Dès qu'une variable est affectée on essaye de filtrer les valeurs pour les autres variables

- On remplace la variable par sa valeur dans toutes les contraintes
- On peut filtrer si une contrainte ne contient plus qu'une variable



Remark

Pour trouver une seule solution, on va générer :

- 483 840 feuilles avec la première modélisation
- 57 avec la seconde

Pourquoi attendre une affectation ?

Notes

Méthode avec Filtrage

Les **2 étapes clés** de la programmation par contraintes !

Propagation

Supprime des domaines les valeurs inconsistantes, c'est-à-dire les valeurs ne pouvant être dans une solution

Exploration

Affecte une valeur à une variable

Notes

Propagation

Consistance pour une contrainte

Différentes consistances :

- Consistance d'arc généralisée [Mackworth, 1977b]
- Consistance de chemin [Montanari, 1974]
- Consistance de borne [van Hentenryck et al., 1995]
- ...

Toutes reposent sur la notion de support

Notes

Propagation

Consistance pour une contrainte

Definition (Support)

Soient $v_1, \ldots, v_n$ des variables de domaines discrets finis $D_1, \ldots, D_n$ et C une contrainte. La valeur $x_i \in D_i$ a un **support** ssi $\forall j \in [1, n], j \neq i, \exists x_j \in D_j$ tel que $C(x_1, \ldots, x_n)$ soit vrai

Exemple

$C : r_4 = m$ avec $D_{r_4} = [0, 1]$ et $D_m = [1, 9]$

- 1 pour r_4 a un support : 1 pour m car $C(1, 1)$ est vrai
- 0 pour r_4 n'a pas de support : $\forall x_m \in D_m, C(0, x_m)$ est faux

Notes

Propagation

Consistance pour une contrainte

Definition (Support)

Soient $v_1, \ldots, v_n$ des variables de domaines discrets finis $D_1, \ldots, D_n$ et C une contrainte. La valeur $x_i \in D_i$ a un **support** ssi $\forall j \in [1, n], j \neq i, \exists x_j \in D_j$ tel que $C(x_1, \ldots, x_n)$ soit vrai

Exemple

$C : v_1 \neq v_2$ avec $D_1 = D_2 = \{\bullet, \bullet, \bullet\}$

- $\bullet$ pour v_1 a un support : $\bullet$ pour v_2 car $C(\bullet, \bullet)$ est vrai
- $\bullet$ pour v_1 a un support : $\bullet$ pour v_2 car $C(\bullet, \bullet)$ est vrai
- $\bullet$ pour v_1 a un support : $\bullet$ pour v_2 car $C(\bullet, \bullet)$ est vrai

Notes

Consistances

Definition (Consistance de bornes)

Soient $v_1, \ldots, v_n$ des variables de domaines discrets finis $D_1, \ldots, D_n$ et C une contrainte. Les domaines sont dits **borne-consistants** (BC) pour C ssi $\forall i \in [1, n], D_i = [a_i, b_i], a_i$ et b_i ont un support.

Exemple

Considérons deux variables v_1, v_2 de domaines $D_1 = D_2 = [-1, 4]$ et la contrainte $v_1 = 2v_2$. Les domaines borne-consistants pour cette contrainte sont $D_1 = [0, 4]$ et $D_2 = [0, 2]$

Notes

Consistances

Definition (Consistance d'arc généralisée)

Soient $v_1, \dots, v_n$ des variables de domaines discrets finis $D_1, \dots, D_n$ et C une contrainte. Les domaines sont dits **arc-consistants généralisés** (GAC) pour C ssi $\forall i \in [1, n], \forall x \in D_i, x$ a un support.

Exemple

Considérons deux variables v_1, v_2 de domaines $D_1 = D_2 = [-1, 4]$ et la contrainte $v_1 = 2v_2$. Les domaines arc-consistants pour cette contrainte sont $D_1 = \{0, 2, 4\}$ et $D_2 = \{0, 1, 2\}$

Notes

Consistance d'arc

Plusieurs implémentations

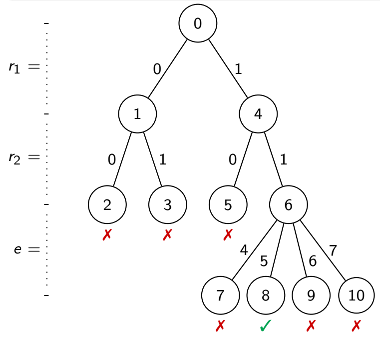
- AC1 et AC3 [Mackworth, 1977a]
- AC4 [Mohr and Henderson, 1986]
- AC5 [van Hentenryck et al., 1992]
- AC6 [Bessière, 1994]
- AC7 [Bessière et al., 1999]
- AC2001 [Bessière and Régin, 2001]
- AC3.2 et AC3.3 [Lecoutre et al., 2003]

Notes

Maintaining Generalized Arc Consistency

On alterne deux phases

- Propagation, on utilise l'arc-consistance généralisée
- Exploration, on fait un choix



Remark

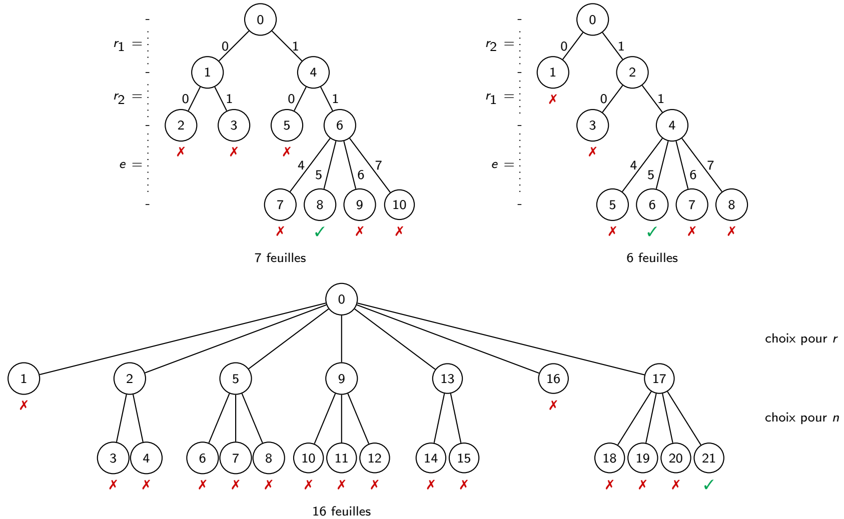
Pour trouver une seule solution, on va générer :

- 6 feuilles avec la première modélisation
- 7 avec la seconde

Notes

Stratégie d'exploration

L'ordre des variables compte



Notes

Stratégie d'exploration

Choisir la variable

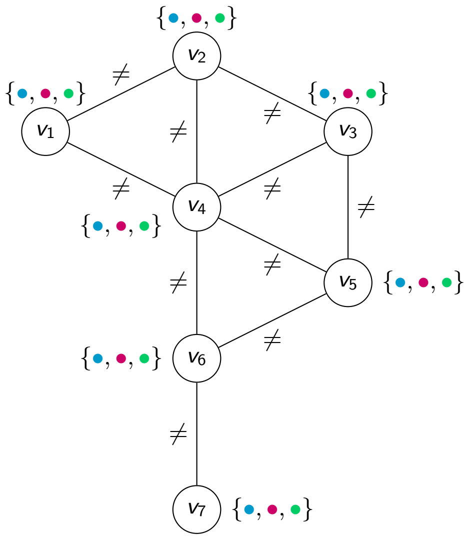
- Ayant le plus petit domaine (dom), First-fail [Haralick and Elliott, 1979]
- Apparaissant dans le plus grand nombre de contraintes (deg)
- $\mathrm{dom} + \mathrm{deg}$ [Brélaz, 1979]
- $\mathrm{dom}/\mathrm{deg}$ [Bessière and Régin, 1996]
- $\mathrm{dom}/w\mathrm{deg}$ [Boussemart et al., 2004]
- ...

Notes

Stratégie d'exploration

Coloriage de carte

- $\mathcal{V} = \{v_1, \dots, v_7\}$
- $D_1 = \cdots = D_7 = \{\bullet, \bullet, \bullet\}$
- $C_1 : v_1 \neq v_2$
  $C_2 : v_1 \neq v_4$
  $C_3 : v_2 \neq v_3$
  $C_4 : v_2 \neq v_4$
  $C_5 : v_3 \neq v_4$
  $C_6 : v_3 \neq v_5$
  $C_7 : v_4 \neq v_5$
  $C_8 : v_4 \neq v_6$
  $C_9 : v_5 \neq v_6$
  $C_{10} : v_6 \neq v_7$

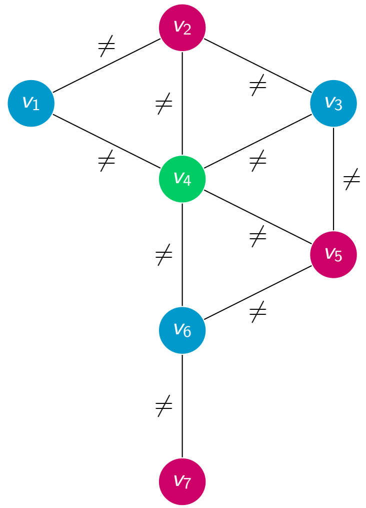


Notes

Stratégie d'exploration - dom

Coloriage de carte

- $\mathcal{V} = \{v_1, \dots, v_7\}$
- $D_1 = \dots = D_7 = \{{\color{cyan}\bullet}, {\color{magenta}\bullet}, {\color{green}\bullet}\}$
- $C_1 : v_1 \neq v_2$
  $C_2 : v_1 \neq v_4$
  $C_3 : v_2 \neq v_3$
  $C_4 : v_2 \neq v_4$
  $C_5 : v_3 \neq v_4$
  $C_6 : v_3 \neq v_5$
  $C_7 : v_4 \neq v_5$
  $C_8 : v_4 \neq v_6$
  $C_9 : v_5 \neq v_6$
  $C_{10} : v_6 \neq v_7$



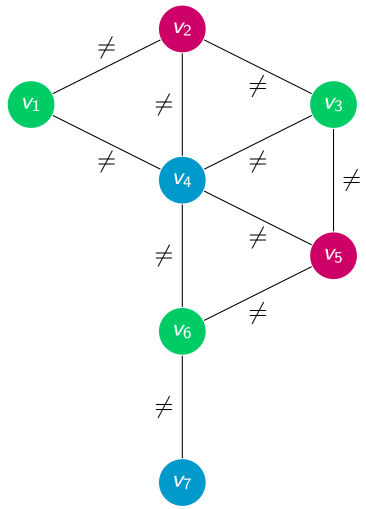
$(v_1, {\color{cyan}\bullet})$
$(v_2, {\color{magenta}\bullet})$
$(v_4, {\color{green}\bullet})$
$(v_3, {\color{cyan}\bullet})$
$(v_5, {\color{magenta}\bullet})$
$(v_6, {\color{cyan}\bullet})$
$(v_7, {\color{magenta}\bullet})$

Notes

Stratégie d'exploration - deg

Coloriage de carte

- $\mathcal{V} = \{v_1, \dots, v_7\}$
- $D_1 = \dots = D_7 = \{{\color{cyan}\bullet}, {\color{magenta}\bullet}, {\color{green}\bullet}\}$
- $C_1 : v_1 \neq v_2$
  $C_2 : v_1 \neq v_4$
  $C_3 : v_2 \neq v_3$
  $C_4 : v_2 \neq v_4$
  $C_5 : v_3 \neq v_4$
  $C_6 : v_3 \neq v_5$
  $C_7 : v_4 \neq v_5$
  $C_8 : v_4 \neq v_6$
  $C_9 : v_5 \neq v_6$
  $C_{10} : v_6 \neq v_7$



$(v_4, {\color{cyan}\bullet})$
$(v_2, {\color{magenta}\bullet})$
$(v_1, {\color{green}\bullet})$
$(v_3, {\color{green}\bullet})$
$(v_5, {\color{magenta}\bullet})$
$(v_6, {\color{green}\bullet})$
$(v_7, {\color{cyan}\bullet})$

Notes

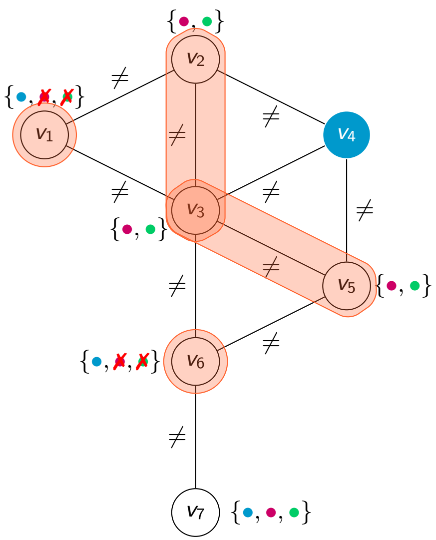
Ça marche tout le temps ?

Notes

Limites

Coloriage de carte

- $\mathcal{V} = \{v_1, \dots, v_7\}$
- $D_1 = \dots = D_7$ $= \{\bullet, \bullet, \bullet\}$
- $C_1 : v_1 \neq v_2$
  $C_2 : v_1 \neq v_3$
  $C_3 : v_2 \neq v_3$
  $C_4 : v_2 \neq v_4$
  $C_5 : v_3 \neq v_4$
  $C_6 : v_3 \neq v_5$
  $C_7 : v_3 \neq v_6$
  $C_8 : v_4 \neq v_5$
  $C_9 : v_5 \neq v_6$
  $C_{10} : v_6 \neq v_7$

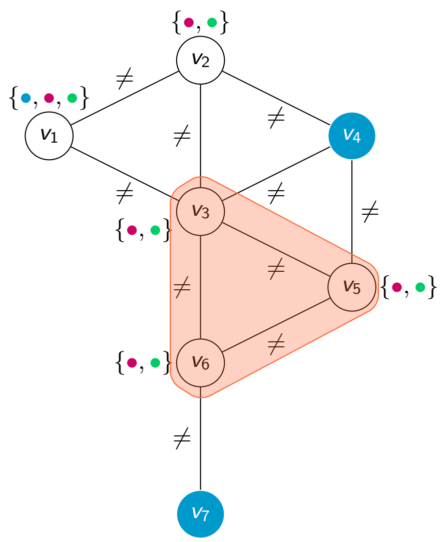


Notes

Limites

Coloriage de carte

- $\mathcal{V} = \{v_1, \dots, v_7\}$
- $D_1 = \dots = D_7$
  $= \{\bullet, \bullet, \bullet\}$
- $C_1 : v_1 \neq v_2$
  $C_2 : v_1 \neq v_3$
  $C_3 : v_2 \neq v_3$
  $C_4 : v_2 \neq v_4$
  $C_5 : v_3 \neq v_4$
  $C_6 : v_3 \neq v_5$
  $C_7 : v_3 \neq v_6$
  $C_8 : v_4 \neq v_5$
  $C_9 : v_5 \neq v_6$
  $C_{10} : v_6 \neq v_7$



Notes

Contraintes globales

Permet de représenter un ensemble de contraintes

- Facilite la modélisation
- Algorithme dédié pour supprimer les valeurs inconsistantes des domaines

Catalogue des contraintes [Beldiceanu et al., 2010]

Les plus connues

- alldifferent
- cycle
- global_cardinality
- nvalue
- element

Notes

Contrainte alldifferent

Présentée la première fois dans [Lauriere, 1978]
Retourne **vrai** si toutes les variables sont différentes deux à deux

Exemple

On peut ré-écrire les contraintes de différences du problème send + more = money
$\text{alldifferent}(s, e, n, d, m, o, r, y)$

Notes

Contrainte alldifferent

Coloriage de carte

- $\mathcal{V} = \{v_1, \ldots, v_7\}$
- $D_1 = \cdots = D_7 = \{\bullet, \bullet, \bullet\}$
- $C_1 : v_1 \neq v_2$
  $C_2 : v_1 \neq v_3$
  $C_3 : v_2 \neq v_3$
  $C_4 : v_2 \neq v_4$
  $C_5 : v_3 \neq v_4$
  $C_6 : v_3 \neq v_5$
  $C_7 : v_3 \neq v_6$
  $C_8 : v_4 \neq v_5$
  $C_9 : v_5 \neq v_6$
  $C_{10} : v_6 \neq v_7$

- $\mathcal{V} = \{v_1, \ldots, v_7\}$
- $D_1 = \cdots = D_7 = \{\bullet, \bullet, \bullet\}$
- $C_1 : \text{alldifferent}(v_1, v_2, v_3)$
  $C_2 : \text{alldifferent}(v_2, v_3, v_4)$
  $C_3 : \text{alldifferent}(v_3, v_4, v_5)$
  $C_4 : \text{alldifferent}(v_3, v_5, v_6)$
  $C_5 : v_6 \neq v_7$

Notes

Contrainte alldifferent

- Pas que du sucre syntaxique
  - Arc-consistance
    - Développé indépendamment par [Costa, 1994] et [Régin, 1994]
    - Repose sur la théorie des graphes
  - Borne-consistance
    - Développé par [Puget, 1998] puis amélioré par [Mehlhorn and Thiel, 2000] et [Lopez-Ortiz et al., 2003]
    - Repose sur la notion d'intervalle de Hall

Notes

Graphe des valeurs

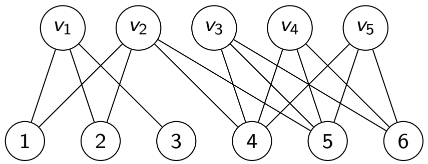
Definition (Graphe des valeurs)

À partir des variables et des domaines d'un CSP on peut créer un graphe biparti, appelé **graphe des valeurs**

- Les sommets correspondent aux variables et aux valeurs
- Une arête relie une variable v_i et une valeur x si $x \in D_i$

Example

- $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$
- $D_1 = \{1, 2, 3\}$, $D_2 = \{1, 2, 4, 5\}$, et $D_3 = D_4 = D_5 = \{4, 5, 6\}$

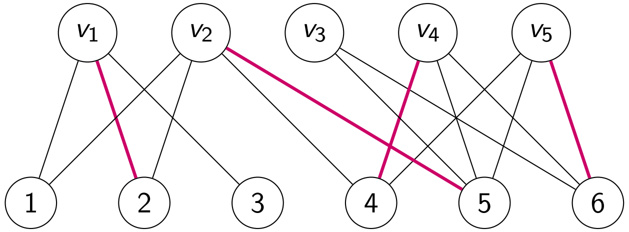


Notes

Théorie des graphes

Definition (Couplage)

Étant donné un graphe $G = (V, E)$, un sous-ensemble M des arêtes E est appelé **couplage** ssi deux arêtes ne partage pas de sommet



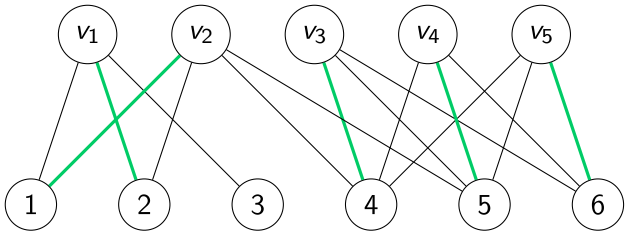
Notes

Contraintes globales | alldifferent

Théorie des graphes

Definition (Couplage maximal)

Un couplage est dit **maximal** si il contient le plus d'arêtes possibles

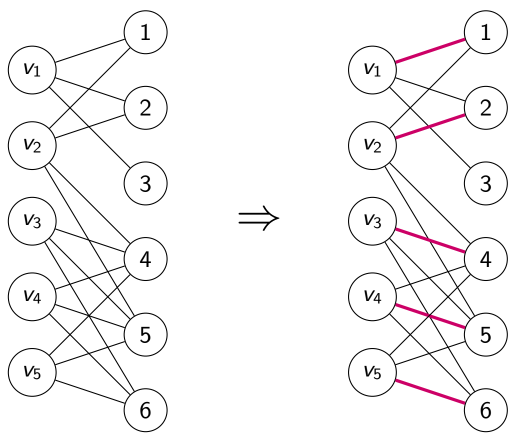


L'algorithme de Hopcroft-Karp [Hopcroft and Karp, 1973] permet de calculer le couplage maximal dans un graphe biparti

Programmation par Contraintes 28 / 41

Notes

Algorithme de Hopcroft-Karp



Notes

Composante fortement connexe

Definition (Graphe orienté)

Un graphe **orienté** $G = (V, E)$ est un graphe dont les arêtes ont une direction, on les appelle des **arcs**

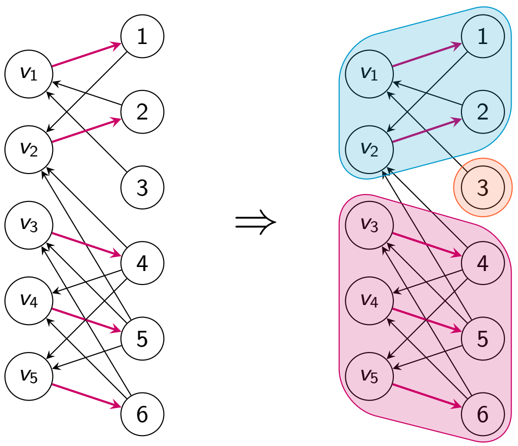
Definition (Composante fortement connexe)

Étant donné un graphe **orienté** $G = (V, E)$, une **composante fortement connexe** est un ensemble maximal de sommets tel que pour chaque sommet de l'ensemble il existe un chemin vers les autres sommets de l'ensemble

L'algorithme de Tarjan [Tarjan, 1972] permet de calculer efficacement les composantes fortement connexes dans un graphe

Notes

Algorithme de Tarjan



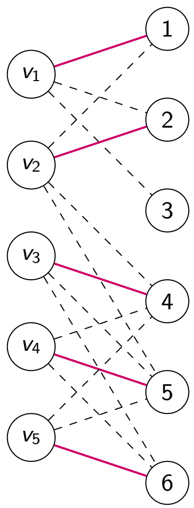
Notes

alldifferent : propagation pour l'arc-consistance

Exemple

- $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$
- $D_1 = \{1, 2, 3\}$, $D_2 = \{1, 2, 4, 5\}$, et $D_3 = D_4 = D_5 = \{4, 5, 6\}$

- On trouve un couplage maximal ⇒ **une solution**



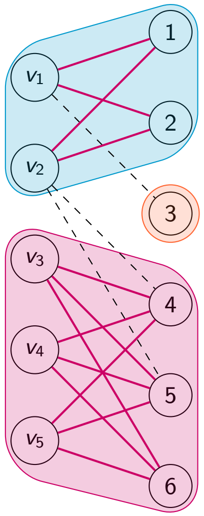
Notes

alldifferent : propagation pour l'arc-consistance

Exemple

- $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$
- $D_1 = \{1, 2, 3\}$, $D_2 = \{1, 2, 4, 5\}$, et $D_3 = D_4 = D_5 = \{4, 5, 6\}$

- On trouve un couplage maximal ⇒ **une solution**
- On cherche les composantes fortement connexes ⇒ **les permutations**



Notes

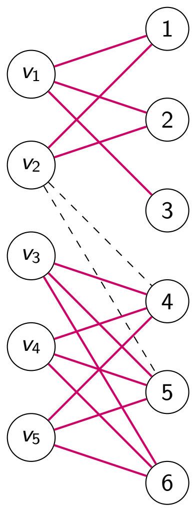
Contraintes globales alldifferent

alldifferent : propagation pour l'arc-consistance

Exemple

- $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$
- $D_1 = \{1, 2, 3\}$, $D_2 = \{1, 2, 4, 5\}$, et $D_3 = D_4 = D_5 = \{4, 5, 6\}$

- On trouve un couplage maximal ⇒ **une solution**
- On cherche les composantes fortement connexes ⇒ **les permutations**
- On ajoute les valeurs isolées aux domaines initiaux



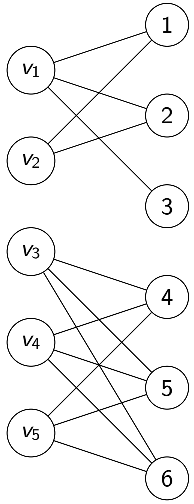
Notes

alldifferent : propagation pour l'arc-consistance

Exemple

- $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$
- $D_1 = \{1, 2, 3\}$, $D_2 = \{1, 2\}$, et $D_3 = D_4 = D_5 = \{4, 5, 6\}$

- On trouve un couplage maximal $\Rightarrow$ **une solution**
- On cherche les composantes fortement connexes $\Rightarrow$ **les permutations**
- On ajoute les valeurs isolées aux domaines initiaux



Notes

Intervalle de Hall

Definition

Soit $(v_1, \ldots, v_n)$ des variables de domaines discrets finis $(D_1, \ldots, D_n)$. Étant donné un intervalle I, on définit $K_I = \{v_i \mid D_i \subseteq I\}$. I est un **intervalle de Hall** si $|I| = |K_I|$.

Exemple

On considère le problème suivant :

- $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$
- $D_1 = [1, 3]$, $D_2 = [1, 5]$, et $D_3 = D_4 = D_5 = [4, 6]$
- $I = [4, 6]$ est un intervalle de Hall car $K_I = \{v_3, v_4, v_5\}$ on a bien $|I| = |K_I|$
- $I = [1, 3]$ n'est pas un intervalle de Hall car $K_I = \{v_1\}$ et $|I| \neq |K_I|$

Notes

alldifferent : propagation pour la borne-consistance

- Pour chaque borne inférieure a et borne supérieure b des domaines, on regarde si $I = [a, b]$ est un intervalle de Hall
- Si I est de Hall on peut supprimer des domaines des variables de $\mathcal{V} \setminus K_I$ les valeurs de I

Exemple

On considère le problème suivant :

- $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$
- $D_1 = [1, 3]$, $D_2 = [1, 3]$, et $D_3 = D_4 = D_5 = [4, 6]$
- $I = [1, 6]$ n'est pas de Hall
- $I = [1, 5]$ n'est pas de Hall
- $I = [1, 3]$ n'est pas de Hall
- $I = [4, 5]$ n'est pas de Hall
- $I = [4, 6]$ est de Hall $\Rightarrow$ **on supprime les valeurs 4, 5, 6 des domaines des variables qui ne sont pas dans K_I**

Notes

Contrainte global_cardinality

Présentée la première fois dans [Oplobedu et al., 1989]

$$\mathrm{global\_cardinality}(\underbrace{\{v_1, \ldots, v_n\}}_{\text{Variables}}, \underbrace{\{x_1, \ldots, p\}}_{\text{Valeurs}}, \underbrace{\{nb_1, \ldots, nb_p\}}_{\text{Occurrences}})$$

Retourne **vrai** si parmi les variables $\{v_1, \ldots, v_n\}$, il y a nb_i variables ayant la valeur x_i

Exemple

$\mathrm{global\_cardinality}(\{v_1, v_2, v_3, v_4, v_5, v_6\}, \{0, 1\}, \{2, 4\})$

Dans certains cas, on peut écrire une alldifferent en utilisant une global_cardinality

$\mathrm{alldifferent}(v_1, v_2, v_3) = \mathrm{global\_cardinality}(\{v_1, v_2, v_3\}, \{\bullet, \bullet, \bullet\}, \{1, 1, 1\})$

Notes

Contrainte global_cardinality

- Arc-consistance
  - Développé par [Régin, 1996]
  - Repose sur un algorithme de flot
- Borne-consistance
  - Développé par [Quimper et al., 2003]
  - Repose sur la notion d'intervalle de Hall
  - Développé par [Katriel and Thiel, 2003]
  - Repose sur la convexité pour améliorer l'efficacité de l'algorithme de flot

Notes

Emploi du temps sportif

Description

- n équipes, $n-1$ semaines et $n/2$ périodes
- chaque paire d'équipe joue exactement 1 fois
- chaque équipe joue un match chaque semaine
- chaque équipe joue au plus 2 fois dans la période

Example (Solution possible)

	S1	S2	S 3	S4	S5	S6	S7
P1	1 vs 2	1 vs 3	5 vs 8	4 vs 7	4 vs 8	2 vs 6	3 vs 5
P2	3 vs 4	2 vs 8	1 vs 4	6 vs 8	2 vs 5	1 vs 7	6 vs 7
P3	5 vs 6	4 vs 6	2 vs 7	1 vs 5	3 vs 7	3 vs 8	1 vs 8
P4	7 vs 8	5 vs 7	3 vs 6	2 vs 3	1 vs 6	4 vs 5	2 vs 4

Notes

Séquence magique

Description

Une séquence magique de longueur n est une séquence d'entiers $v_0, \ldots, v_{n-1}$ compris entre 0 et $n-1$ telle que le nombre $i \in \{0, \ldots, n-1\}$ apparaisse exactement v_i fois dans la séquence

Séquence magique ($n = 10$)

	0	1	2	3	4	5	6	7	8	9
v_i	6	2	1	0	0	0	1	0	0	0

Notes

Suite de Langford

Description

Une suite de Langford est une séquence d'entiers $v_1, \ldots, v_{k \times n}$ compris entre 1 et n telle que le nombre $i \in \{1, \ldots, n\}$ apparaisse exactement k fois, et que 2 occurrences successives soient séparées par une distance i

On ne considère ici que pour $k = 2$

Suite de Langford ($n = 7$)

4

7 3 6 2 5 3 2 4 7 6 5 1 4 1

7 1

Notes

Alice et Bob vont au travail

Description

- Alice va au travail en voiture (30 à 40 min) ou par bus (au moins 60 min)
- Bob s'y rend en vélo (40 ou 50 min) ou en moto (20 à 30 min)
- Ce matin :
  - Alice a quitté sa maison entre 7h10 et 7h20
  - Bob est arrivé au travail entre 8h00 et 8h10
  - Alice est arrivée 10 à 20 min après que Bob soit parti

1. Modélisez ce problème
2. L'histoire est-elle cohérente ?
3. Quand Bob est-il parti ? Est-il possible qu'il ait pris son vélo ?
4. L'histoire est-elle cohérente si on ajoute le fait que :
   - la voiture d'Alice est en panne
   - Alice et Bob se sont rencontrés en chemin

Notes

Binairo – Examen 2018

Description

Un jeu belge, basé sur une grille carrée dans laquelle sont inscrits uniquement les chiffres 0 et 1. Sur chaque ligne et chaque colonne :

- il y a autant de 0 que de 1
- il ne peut pas y avoir plus de 2 chiffres identiques côte à côte

Il ne peut y avoir 2 lignes ou 2 colonnes identiques.

Exemple de grille

	0			1	0
	1	0			1
1	0		0	1	
1					
				1	1

Notes

Beldiceanu, N., Carlsson, M., and Rampon, J.-X. (2010).
Global constraint catalog, 2nd edition.
Technical Report T2010:07, The Swedish Institute of Computer Science.

Bessière, C. (1994).
Arc-consistency and arc-consistency again.
Artificial Intelligence, 65(1):179–190.

Bessière, C., Freuder, E. C., and Régin, J.-C. (1999).
Using constraint metaknowledge to reduce arc consistency computation.
Artificial Intelligence, 107(1):125–148.

Bessière, C. and Régin, J.-C. (1996).
Mac and combined heuristics: Two reasons to forsake fc (and cbj?) on hard problems.
In *Proceedings of the Second International Conference on Principles and Practice of Constraint Programming*, volume 1118 of *Lecture Notes in Computer Science*. Springer.

Bessière, C. and Régin, J.-C. (2001).
Refining the basic constraint propagation algorithm.
In *Proceedings of the 17th International Joint Conference on Artificial intelligence (IJCAI'01)*, pages 309–315. Morgan Kaufmann.

Boussemart, F., Hemery, F., Lecoutre, C., and Sais, L. (2004).
Boosting systematic search by weighting constraints.
In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, (ECAI'2004)*, pages 146–150. IOS Press.

Brélaz, D. (1979).
New methods to color the vertices of a graph.
Communications of the ACM, 22(4):251–256.

Costa, M.-C. (1994).
Persistency in maximum cardinality bipartite matchings.
Operations Research Letters, 15(3):143–149.

Haralick, R. M. and Elliott, G. L. (1979).

Notes

Increasing tree search efficiency for constraint satisfaction problems.
In *Proceedings of the 6th International Joint Conference on Artificial intelligence (IJCAI'79)*, pages 356–364. Morgan Kaufmann Publishers Inc.

Hopcroft, J. E. and Karp, R. M. (1973).
An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs.
SIAM Journal on Computing, 2(4):225–231.

Katriel, I. and Thiel, S. (2003).
Fast bound consistency for the global cardinality constraint.
In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP'03)*, volume 2833 of *Lecture Notes in Computer Science*, pages 437–451. Springer Berlin / Heidelberg.

Lauriere, J.-L. (1978).
A language and a program for stating and solving combinatorial problems.
Artificial Intelligence, 10(1):29 – 127.

Lecoutre, C., Boussemart, F., and Hemery, F. (2003).
Exploiting multidirectionality in coarse-grained arc consistency algorithms.
In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP'03)*, volume 2833 of *Lecture Notes in Computer Science*, pages 480–494. Springer.

Lopez-Ortiz, A., Quimper, C.-G., Tromp, J., and Beek, P. V. (2003).
A fast and simple algorithm for bounds consistency of the all different constraint.
In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 245–250.

Mackworth, A. K. (1977a).
Consistency in networks of relations.
Artificial Intelligence, 8(1):99–118.

Mackworth, A. K. (1977b).
On reading sketch maps.
In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 598–606.

Mehlhorn, K. and Thiel, S. (2000).

Notes

Faster algorithms for bound-consistency of the sortedness and the alldifferent constraint.
In *Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming (CP '00)*, volume 1894 of *Lecture Notes in Computer Science*, pages 306–319. Springer.

Mohr, R. and Henderson, T. C. (1986).
Arc and path consistency revisited.
Artificial Intelligence, 28(2):225–233.

Montanari, U. (1974).
Networks of constraints: Fundamental properties and applications to picture processing.
Information Science, 7(2):95–132.

Oplobedu, A., Marcovitch, J., and Tourbier, Y. (1989).
Charme: Un langage industriel de programmation par contraintes, illustré par une application chez renault.
In *Proceedings of the Ninth International Workshop on Expert Systems and their Applications: General Conference*, pages 155–70.

Puget, J.-F. (1998).
A fast algorithm for the bound consistency of alldiff constraints.
In *Proceedings of the 15th National/10th Conference on Artificial Intelligence/Innovative applications of artificial intelligence (AAAI '98/IAAI '98)*, pages 359–366. American Association for Artificial Intelligence.

Quimper, C.-G., van Beek, P., López-Ortiz, A., Golynski, A., and Sadjad, S. (2003).
An efficient bounds consistency algorithm for the global cardinality constraint.
In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP'03)*, volume 2833 of *Lecture Notes in Computer Science*, pages 600–614. Springer Berlin / Heidelberg.

Régin, J.-C. (1994).
A filtering algorithm for constraints of difference in csps.
In *Proceedings of the 12th National Conference on Artificial Intelligence (Vol. 1)*, pages 362–367.

Régin, J.-C. (1996).
Generalized arc consistency for global cardinality constraint.
In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-96)*, pages 209–215.

Notes

Tarjan, R. (1972).
Depth-first search and linear graph algorithms.
SIAM Journal on Computing, 1(2):146–160.

van Hentenryck, P., Deville, Y., and Teng, C.-M. (1992).
A generic arc-consistency algorithm and its specializations.
Artificial Intelligence, 57.

van Hentenryck, P., Saraswat, V. A., and Deville, Y. (1995).
Design, implementation, and evaluation of the constraint language cc(fd).
In *Selected Papers from Constraint Programming: Basics and Trends*, pages 293–316. Springer-Verlag.

Notes