Consistance pour des contraintes avec plus de 2 variables

- Quoi faire avec des contraintes avec plus de 2 variables ?
- Consistance d'hyper-arc: étendre l'arc consistance à un nombre arbitraire de variables
- Déterminer la hyper-arc consistance est NP-difficile

Consistance de bornes

- CSP arithmétique: les contraintes sont sur des entiers
- intervals: $[l..u]$ représente l'ensemble $\{l, l+1, \ldots, u\}$
- Idée: Utiliser la consistance sur les réels et examiner seulement les bornes (inférieurs et supérieurs) du domaine de chaque variable
- Définir $min(D, x)$ comme l'élément minimum dans le domaine de x, pareil $max(D, x)$

Consistance de bornes

- Une contrainte simple c est bornes-consistante avec domaine D, si pour chaque variable x dans $variables(c)$
  - ils existent des réels $d_1, \ldots, d_k$ pour les autres variables $x_1, \ldots, x_k$ tel que
    - $min(D, x_j) \leq d_j \leq max(D, x_j)$ pour tout j et
    - $\{x \leftarrow min(D, x), x_1 \leftarrow d_1, \ldots, x_k \leftarrow d_k\}$ est une solution de c
  - ils existent des réels $d'_1, \ldots, d'_k$ pour les autres variables $x_1, \ldots, x_k$ tel que
    - $min(D, x_j) \leq d'_j \leq max(D, x_j)$ pour tout j et
    - $\{x \leftarrow max(D, x), x_1 \leftarrow d'_1, \ldots, x_k \leftarrow d'_k$ est une solution de c
- Un CSP arithmétique est bornes-consistant, si toutes ses contraintes simples le sont

Comment obtenir un CSP bornes-consistant ?

- Étant donné un domaine D, on doit modifier les bornes, de sorte que le résultat est bornes-consistant
- Utilisation de règles de propagation
- Exemple:
  - $X = Y + Z$ équivalent à $Y = X - Z$ et $Z = X - Y$
  - Raisonner avec max et min
  - $X \geq min(D, Y) + min(D, Z)$, $X \leq max(D, Y) + max(D, Z)$
  - $Y \geq min(D, X) - max(D, Z)$, $Y \leq max(D, X) - min(D, Z)$
  - $Z \geq min(D, X) - max(D, Y)$, $Z \leq max(D, X) - min(D, Y)$
  - cela donne des règles de propagation

Exemple

- $X = Y + Z$, $D(X) = [4..8]$, $D(Y) = [0..3]$, $D(Z) = [2..2]$
- Les règles de propagation donnent:
  - $(0 + 2 =)2 \leq X \leq 5(= 3 + 2)$
  - $(4 - 2 =)2 \leq Y \leq 6(= 8 - 2)$
  - $(4 - 3 =)1 \leq Z \leq 8(= 8 - 0)$
- Les domaines peuvent être réduits:
  $D(X) = [4..5]$, $D(Y) = [2..3]$, $D(Z) = [2..2]$

D'autres règles de propagation

- $4W + 3P + 2C \leq 9$
- $W \leq \frac{9}{4} - \frac{3}{4}min(D, P) - \frac{2}{4}min(D, C)$
- $P \leq \frac{9}{3} - \frac{4}{3}min(D, W) - \frac{2}{3}min(D, C)$
- $C \leq \frac{9}{2} - \frac{4}{2}min(D, W) - \frac{3}{2}min(D, Px)$
- Étant donné un domaine initial $D(W) = [0..9]$,$D(P) = [0..9]$,$D(C) = [0..9]$ on détermine que $W \leq \frac{9}{4}$, $P \leq \frac{9}{3}$,$C \leq \frac{9}{2}$,
- nouveau domaine: $D(W) = [0..2]$,$D(P) = [0..3]$,$D(C) = [0..4]$

Inégalités $Y \neq Z$

- Les inégalités donnent des règles de propagation très faibles
- Seulement si une de deux côtés prend une valeur fixe qui est égale au minimum ou maximum de l'autre il y a propagation
- $D(Y) = [2..4], D(Z) = [2..3]$ pas de propagation
- $D(Y) = [2..4], D(Z) = [3..3]$ pas de propagation
- $D(Y) = [2..4], D(Z) = [2..2]$ propagation $D(Y) = [3..4], D(Z) = [2..2]$

Multiplication $X = Y * Z$

- Si toutes les variables sont positives
  $X \geq min(D, Y) * min(D, Z)$, $X \leq max(D, Y) * max(D, Z)$ etc. pour Y, Z
- sinon $X \geq minimum\{min(D, Y) * min(D, Z), min(D, Y) * max(D, Z),$
  $max(D, Y) * min(D, Z), max(D, Y) * max(D, Z)\}$
- similaire pour borne supérieure pour X en utilisant $maximum$
- mais cela ne marche pas pour Y et Z
- si $min(D, Z) < 0$ et $max(D, Z) > 0$ il n'y a pas de restriction pour Y
- On "attend" jusqu'à-ce que le domaine de Z devienne non-negatif ou non-positif et ensuite en utilise des règles de la forme
  $Y \geq minimum\{min(D, X)/min(D, Z), min(D, X)/max(D, Z),$
  $max(D, X)/min(D, Z), max(D, X)/max(D, Z)\}$
  Attention à la division par 0

Algorithme de bornes consistance

- $bornescons(C, D)$: Appliquer les règles de propagation pour chaque contrainte simple de C, jusqu'a-ce qu'il n'y a plus de changement dans les domaines D.
- On ne réexamine pas une contrainte simple, si les domaines de ses variables n'ont pas changé

Solutionneur bornes consistance

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

Solutionneur par retour en arrière avec bornes consistance

- Appliquer bornes consistance avant de lancer le solutionneur par retour en arrière et à chaque fois qu'une variable est affectée par le solutionneur retour en arrière

Exemple retour en arrière avec bornes consistance

- Problème du sac du contrebandier
- $4W + 3P + 2C \leq 9 \ \wedge \ 15W + 10P + 7C \geq 30$
- Domaines initiaux : $D(W) = [0..9]$, $D(P) = [0..9]$, $D(C) = [0..9]$
- Bornes consistance sur la première contrainte donne:
  $D(W) = [0..2]$, $D(P) = [0..3]$, $D(C) = [0..4]$
- On essaie $W = 0$. Ça donne : $D(W) = [0..0]$, $D(P) = [1..3]$, $D(C) = [0..3]$
- On essaie $P = 1$. Ça donne : $D(W) = [0..0]$, $D(P) = [1..1]$, $D(C) = [3..3]$
  et on a trouvé une solution.
- On peut aussi chercher les autres solutions

Consistance généralisée

- On peut combiner les trois consistances (nœud, arc, bornes) vues jusqu'à présent.
- Toutes ses méthodes utilisent les contraintes simples une par une
- On peut considérer des contraintes simples "complexes" qui sont une conjonction de contraintes simples avec un mécanisme de propagation spécial
- Exemple: $alldifferent(\{V_1, \ldots, V_n\})$
- $alldifferent(\{X, Y, Z\})$ signifie $X \neq Y \ \wedge \ Y \neq Z \ \wedge \ X \neq Z$
- Arc-consistant avec $D(X) = \{1, 2\}$, $D(Y) = \{1, 2\}$, $D(Z) = \{1, 2\}$
- Mais il n'y a pas de solution

Consistance pour alldifferent

- Soit c de la forme $alldifferent(V)$
- Tant qu'il existe $v \in V$ avec $D(V) = \{d\}$
  - $V := V - \{v\}$
  - Pour chaque $v' \in V$
    - $D(v') := D(v') - \{d\}$
- $DV := \bigcup_{v \in V} D(v)$
- Si $|V| > |DV|$ alors retourne $domaine\ faux$
- retourne D

Exemples alldifferent

- $alldifferent(\{X, Y, Z\})$ avec $D(X) = \{1, 2\}$, $D(Y) = \{1, 2\}$, $D(Z) = \{1, 2\}$
- Algorithme retourne $faux$
- $alldifferent(\{X, Y, Z, T\})$ avec $D(X) = \{1, 2\}$, $D(Y) = \{1, 2\}$, $D(Z) = \{1, 2\}$, $D(T) = \{2, 3, 4, 5\}$
- Algorithme ne detecte pas le problème
- On peut utiliser des algorithmes plus compliqués pour cela

Exemple d'utilisation de alldifferent

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

- Le problème du Sudoku consiste à remplir une grille de sorte que chaque ligne, chaque colonne et chaque carré contiennent les chiffres 1 à 9.
- Pour modéliser ce probleme on peut utiliser alldifferent. Comment ?

Optimisation pour CSP sur domaine fini

- Puisque les domaines sont finis, on peut facilement utiliser un solutionneur pour construire un optimiseur
- Soit $intsolv(C, D)$ un solutionneur qui rend un domaine qui est une solution ou $faux$
- $meilleure$ contient la meilleur solution jusqu'a présent
- $reessayeintopt(C, D, f, meilleure)$
  - $D_2 := intsolv(C, D)$
  - Si D_2 est un domaine faux, retourne $meilleure$
  - Soit sol une solution qui correspond à D_2
  - retourne $reessayeintopt(C \wedge f < sol(f), D, f, sol)$

Exemple optimisation en reessayant

- Sac du contrebandier
- $4W+3P+2C \leq 9 \wedge 15W+10P+7C \geq 30$ et $D(W) = [0..9]$, $D(P) = [0..9]$, $D(C) = [0..9]$
- Le contrebandier veut minimiser la perte $-15W - 10P - 7C$
- On utilise la recherche par retour en arrière avec bornes consistances : première solution : $D(W) = \{0\}$, $D(P) = \{1\}$ et $D(C) = \{3\}$. Perte de -31 ou profit de 31
- Nouveau problème en ajoutant $-15W - 10P - 7C < -31$

Exemple suite

- On recommence la recherche par retour en arrière
- D'abord $D(W) = [0..2]$,$D(P) = [0..3]$,$D(C) = [0..4]$
- Mais maintenant le choix de $W = 0$ donne un domaine faux par propagation
- On essaie $W = 1$ et ça donne $D(W) = \{1\}$, $D(P) = \{1\}$ et $D(C) = \{1\}$ avec une perte de -32
- On ajoute $15W - 10P - 7C < -32$ et on recommence: pas de solution

Problème: On refait des calculs plusieurs fois

Optimisation par retour en arrière

- On combine retour en arrière avec optimisation
- À chaque étape, si $meilleure$ est la meilleure solution jusqu'à présent, on ajoute la contrainte $f < meilleure(f)$
- Exemple:
  - La première solution trouvée est : $\{W \leftarrow 0, P \leftarrow 1, C \leftarrow 3\}$ avec perte de -31. On mémorise cette solution comme meilleure jusqu'à présent
  - le retour en arrière revient sur $D(W) = \{0\}$, $D(P) = [1..3]$, $D(C) = [0..4]$ et on ajoute la nouvelle contrainte $-15W - 10P - 7C < -31$.
  - On essaie $P = 2$ et $P = 3$. Cela donne un domaine faux
  - On revient sur $D(W) = [0..2]$, $D(P) = [1..3]$, $D(C) = [0..4]$ et on essaie $W = 1$.
  - Propagation donne $D(W) = \{1\}$, $D(P) = \{1\}$, $D(C) = \{1\}$ avec perte -32. Donc nouvelle meilleure solution.