Décomposition sémantique pour la résolution de systèmes de contraintes de distance

Heikel Batnini Michel Rueher

Université de Nice Sophia-Antipolis - Projet COPRIN INRIA/I3S/CNRS-CERMICS
{Heikel.Batnini,Michel.Rueher}@sophia.inria.fr

Résumé

La plupart des outils pour la résolution de CSP continus s'appuient sur des consistances locales qui encadrent l'espace solution. Les solveurs de contraintes combinent ces techniques avec des techniques de recherche systématiques qui ne tiennent pas compte de la sémantique des contraintes. C'est particulièrement le cas des contraintes géométriques. Nous proposons dans cet article une technique de décomposition de domaines guidée par la sémantique des contraintes de distance. Cette décomposition sémantique, que nous nommerons SDD (Semantic Domain Decomposition), isole les disjonctions dans l'espace de recherche. Les domaines des coordonnées d'un même point sont décomposés et réduits par la SDD en utilisant les propriétés de monotonie et de convexité des contraintes de distance. Nous montrons comment cette technique peut être combinée avec différents filtrage locaux. L'apport de la SDD est mis en évidence sur différents CSP constitués d'équations de distance.

1 Introduction

La satisfaction d'un système de contraintes numériques (ou NCSP [1]) est un problème qui a de nombreuses applications dans les sciences de l'ingénieur. Un NCSP est défini par un ensemble $\mathcal{X}$ de variables, un ensemble $\mathcal{D}$ de domaines associés à ces variables, ainsi qu'un ensemble $\mathcal{C}$ de contraintes constitué d'égalités et d'inégalités entre des fonctions numériques impliquant ces variables. La représentation des domaines par des intervalles a permis une adaptation des outils existants en programmation par contraintes sur des domaines finis. Plus particulièrement, les techniques de consistances locales [Mac77], comme la kB-consistance [Lho93, Lho94] ou la Box-consistance [PVH96] calculent une approximation extérieure de l'ensemble des solutions.

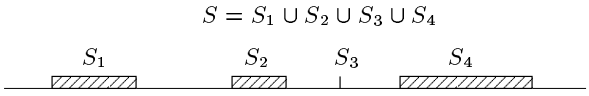


FIG. 1 – *Disjonctions dans l'espace des solutions*

Lorsque l'espace solution contient des disjonctions, une approximation extérieure conduit à conserver dans les domaines des valeurs localement incohérentes. Leur présence peut empêcher des réductions significatives lors de la phase de propagation. Pour isoler les solutions ponctuelles du système, on utilise des méthodes d'énumération, comme la bissection, en combinaison avec

1. Numerical Constraint Solving Problem

ces techniques de filtrage local. Ces méthodes se révèlent inadaptées en présence de continuums de solutions [VSHS02]. Par ailleurs, beaucoup de résolveurs de contraintes mettent en œuvre ces techniques de manière systématique, souvent sans tenir compte de la sémantique du problème. C'est notamment le cas pour la résolution de contraintes géométriques où les points sont représentés par un ensemble de variables traitées de manière indépendante. D'autre part, l'utilisation des propriétés des contraintes, comme la monotonie ou la convexité, améliore l'efficacité des techniques de résolution usuelles [BMB02].

Nous proposons dans cet article une stratégie de recherche qui s'appuie sur la sémantique des contraintes. L'intérêt de cette stratégie est illustrée sur des problèmes de satisfaction de contraintes de distance. Ce type de contraintes intervient dans de nombreux domaines d'application comme la conception optimale de robots [GSR92, Laz92, Die98, Mer03] et plus généralement les problèmes de modélisation géométrique, les problèmes de conformation moléculaire [CH88, KB00], ou encore dans d'autres applications plus théoriques comme le problème de *circle packing* [Mar00, SCCG01, MC04]. Les systèmes d'équations de distance sont des problèmes particulièrement difficile à résoudre par les méthodes classiques. En effet, les domaines peuvent contenir à la fois des continuums et des solutions ponctuelles. De plus, le nombre important d'axes de symétrie et de disjonctions dans l'espace des solutions augmente la combinatoire du problème.

Différentes techniques spécifiques de filtrage ont été proposées pour la résolution de systèmes d'équations de distance, notamment les travaux de Lebbah *et al* qui introduisent une méthode de filtrage global nommée *Quad* et spécifiquement adaptée aux contraintes quadratiques [LRM02, MLR03]. On peut citer également les travaux de Markot et Csendes qui présentent une technique basée sur une représentation des domaines par des polytopes convexes [Mar00, SCCG01, MC04]. Cette technique, nommée *Method of "Active Areas"*, a été réintroduite par Heusch sous la forme d'une contrainte globale [Heu03].

L'approche proposée ici est complémentaire à la fois à ces filtrages dans la mesure où elle guide la stratégie de recherche, mais également aux techniques de décomposition structurelle du graphe de contraintes introduites par Jermann *et al* [JTNR00, JNT02, JNT03]. La section suivante décrit de manière informelle le principe de notre stratégie de recherche guidée par la sémantique des contraintes de distance.

2 Description informelle

Les méthodes de résolution de CSPs numériques sont souvent basées sur une approche de type Branch & Prune. Basiquement, l'idée est de découper le domaine d'une variable afin de diviser le problème en deux sous-problèmes (Branch). Chaque sous-problème est alors réduit ou éliminé en utilisant une méthode de filtrage, par exemple une consistance locale. Puis, un nouveau domaine est choisi pour être découpé et le processus recommence jusqu'à ce que tous les domaines soient de taille inférieure à un certain paramètre de précision.

Notre décomposition se place en amont de la résolution proprement dite dans la mesure où elle divise l'espace de recherche en sous-espaces vérifiant de fortes propriétés (consistance locale et monotonie des contraintes) mais non nécessairement réduits à une solution ponctuelle.

2.1 Décomposition sémantique des domaines

Nous introduisons ici une technique de décomposition sémantique pour les contraintes de distance, nommée SDD (Semantic Domain Decomposition). Cette méthode découpe et réduit les domaines des points mis en jeu dans une contrainte de distance en utilisant ses propriétés de monotonie. La forme canonique des équations de distance [2] $X^2 + Y^2 - D^2 = 0$ est utilisée pour isoler ces parties monotones sur $\mathbb{R}^+ \times \mathbb{R}^+$, $\mathbb{R}^+ \times \mathbb{R}^-$, $\mathbb{R}^- \times \mathbb{R}^+$ et $\mathbb{R}^- \times \mathbb{R}^-$.

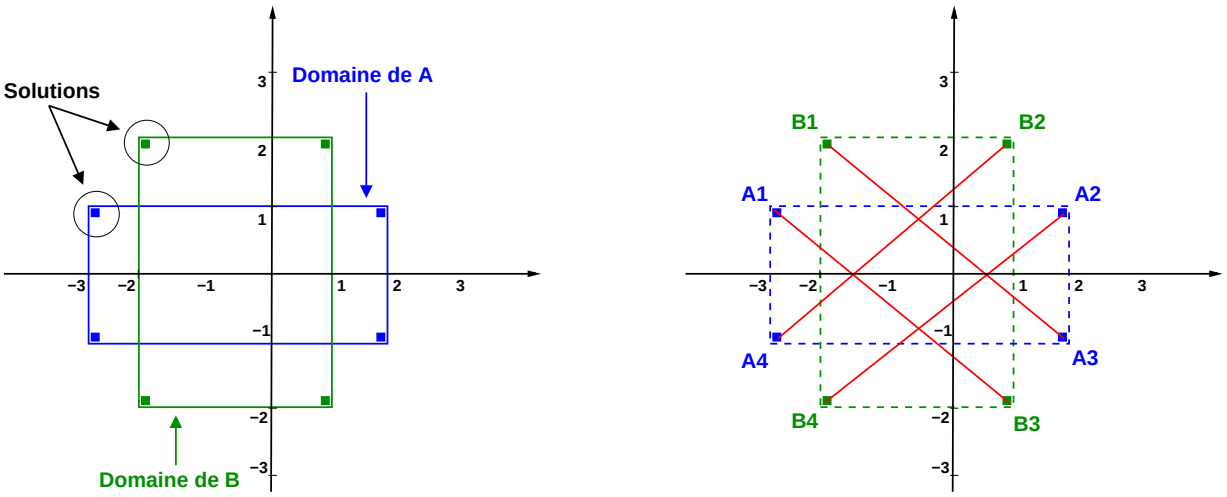


FIG. 2 – *La figure de gauche montre les domaines de A et B après un filtrage par 2B-consistance. Les carrés pleins représentent les continuums de solutions. La figure de droite montre les 4 sous-domaines de A et ceux de B ainsi que leurs liaisons dans la micro-structure calculée par la SDD.*

Un changement de variable linéaire permet de passer de la forme canonique à une contrainte et ainsi d'extraire les sous-domaines des points sur ces parties monotones. L'exemple suivant illustre les apports de SDD relativement aux consistances locales :

Exemple 2.1 *Soient 2 points du plan $A(x_A,y_A)$ et $B(x_B,y_B)$ dont les domaines sont respectivement les boîtes $[-3,2]\times[-1,1]$ et $[-2,1]\times[-2,2]$. On cherche à déterminer les points A et B tels que la longueur du segment $[AB]$, notée D, soit dans l'intervalle $[4.95,5]$. Le CSP correspondant est décrit par la contrainte $(x_A - x_B)^2 + (y_A - y_B)^2 = D^2$, avec $x_A \in [-3,2]$, $x_B \in [-1,1]$, $y_A \in [-2,1]$, $y_B \in [-2,2]$ et $D \in [4.95,5]$.*

2. En dimension 2 pour simplifier la notation. La généralisation en dimension quelconque est triviale.

La 2B-consistance ne permet pas de réduire ces domaines; la figure 2 (à gauche) montre que les continuums de solutions sont proches des bornes des domaines. L'importance du nombre de solutions incohérentes préservées dans les domaines est clairement mise en évidence sur la figure. À droite, la figure 2 montre que la SDD divise le CSP initial en 4 sous-CSP correspondant aux 4 sous-espaces de solutions locales. Ces sous-espaces sont modélisés par un graphe de sous-domaines dont les arêtes sont les segments A_1B_3, A_2B_4, A_3B_1 et A_4B_2.

Notons qu'avec une précision de l'ordre de 0.05 la bissection combinée à la 2B permettrait d'obtenir un résultat similaire mais avec 2 étapes de division alors que la SDD ne nécessite qu'une étape. La combinaison 2B & bissection avec une petite précision conduirait à énumérer un grand nombre de points solutions du système.

Une approche basée sur les unions d'intervalles [BO93] permettrait également d'obtenir une décomposition similaire pour chacun des domaines mais sans la micro-structure. Sans ces liaisons, les 16 combinaisons de sous-domaines doivent être examinées afin de trouver les sous-espaces potentiellement porteurs de solutions.

2.2 Décomposition sémantique d'un CSP

Dans cette section, nous montrons sur un exemple simple comment la SDD est propagée sur l'ensemble du système de contraintes. Le résultat de cette propagation est une décomposition du CSP initial en un ensemble de sous-domaines sur lesquels toutes les contraintes sont consistantes et monotones. Dans cet exemple notre algorithme de décomposition suffit à isoler rapidement les solutions du système.

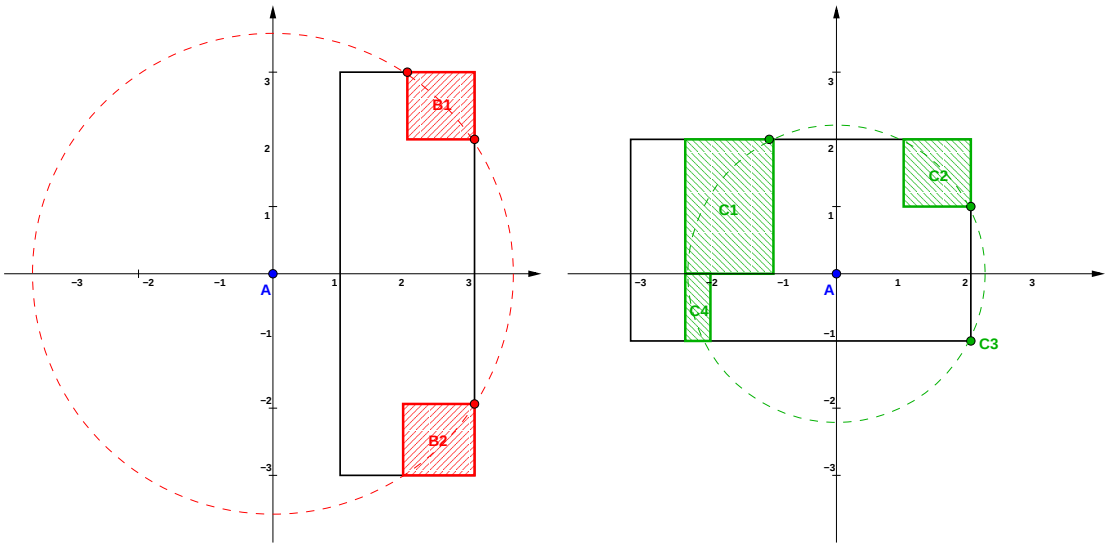
Exemple 2.2 *Considérons dans le plan, 3 points $A(0,0)$, $B(x_B,y_B)$ et $C(x_C,y_C)$ où le point P_1 est fixé à l'origine du repère cartésien. Les domaines initiaux de x_B, y_B, x_C et y_C sont respectivement $[1,3]$, $[-3,3]$, $[-3,2]$ et $[-1,2]$. Les distances entre les points sont données par : $AB^2 = 13$,$AC^2 = 5$ et $BC^2 = 10$. Le CSP correspondant est le suivant :*

- $\mathcal{X} = \{x_B,y_B,x_C,y_C\}$.
- $\mathcal{D} = \{[1,3],[-3,3],[-3,2],[-1,2]\}$.
- $\mathcal{C} = \begin{cases} c_1 : & x_B^2 + y_B^2 = 13 \\ c_2 : & x_C^2 + y_C^2 = 5 \\ c_3 : & (x_B - x_C)^2 + (y_B - y_C)^2 = 10 \end{cases}$

Les 3 solutions de ce CSP sont représentées graphiquement sur la figure 4.

La SDD appliquée à la contrainte c_1 décompose le domaine initial de B en 2 sous-domaines B_1 et B_2. De même, la contrainte c_2 permet à la SDD de décomposer le domaine de C en 4 sous-domaines C_1, C_2, C_3 et C_4. La figure 5 montre le résultat de cette décomposition. Le traitement de la contrainte c_3 nous conduit à examiner les 8 combinaisons de sous-domaines de B et C :

- B_1C_4 et B_2C_4 sont immédiatement éliminées car C_4 n'a de support ni dans B_1, ni dans B_2.
- De même pour B_2C_3 et B_2C_1, car B_2 n'a de support ni dans C_3, ni dans C_1.
- Une consistance locale appliquée à B_1C_1, B_1C_3 et B_2C_2 suffit à isoler les 3 solutions du système.



(a) (b)

FIG. 3 – *La figure 3(a) montre la décomposition du domaine de B en 2 sous-domaines B_1 et B_2, pour la contrainte c_1. La figure 3(b) montre la décomposition du domaine de C en 4 sous-domaines C_1, C_2, C_3 et C_4, pour la contrainte c_2.*

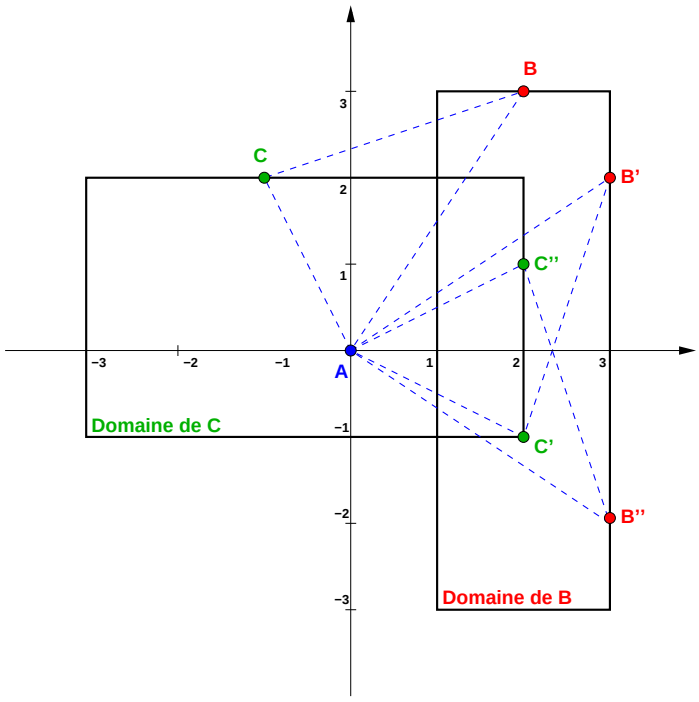


FIG. 4 – *Les 3 solutions du CSP de l'exemple 2.2 sont les triangles ABC, $AB'C'$ et $AB''C''$.*

La suite de cet article est organisé de la manière suivante : dans la section 3, nous donnons les définitions nécessaires à la compréhension de la suite l'article. La section 4 détaille la décomposition sémantique des domaines et l'algorithme de décomposition de CSP. Enfin, la section 5 montre que la SDD améliore les performances des méthodes traditionnelles (consistance locale couplée avec la bissection) pour différentes variantes du classique problème du pentagone, ainsi que pour un problème classique de robotique.

3 Préliminaires

Nous utilisons la définition classique des CSP, telle qu'elle est énoncée par Mackworth :

Définition 3.1 (Constraint Solving Problem (CSP)[Mac77]) *Un $\mathcal{CSP}$ est un triplet $(\mathcal{X},\mathcal{D},\mathcal{C})$ tel que :*

- *$\mathcal{X} = \{x_1,\ldots,x_n\}$ est un ensemble de variables.*
- *$\mathcal{D} = \{D_{x_1},\ldots,D_{x_n}\}$ est un ensemble de domaines. D_{x_i} est le domaine contenant toutes les valeurs potentielles de la variable x_i.*
- *$\mathcal{C} = \{c_1,\ldots,c_m\}$ est un ensemble de contraintes.*

On note $Var(c)$ le sous-ensemble de $\mathcal{X}$ des variables de c.

Dans le cas des CSP continus, les domaines des variables sont représentées par des intervalles, de la forme $D_x = [\underline{x},\overline{x}]$.

Le problème de satisfaction de contraintes de distance consiste à déterminer les positions de n points satisfaisant un ensemble de relations de distance euclidienne, dans un espace borné. Une contrainte de distance est une équation quadratique qui met en jeu les coordonnées de deux points $P_i(x_i^1,x_i^2,\ldots,x_i^p)$ et $P_j(x_j^1,x_j^2,\ldots,x_j^p)$, ainsi qu'une variable réelle positive δ_{ij} :

$$\sum_{k=1}^{p} \left(x_i^k - x_j^k\right)^2 = {\delta_{ij}}^2$$

Les domaines des coordonnées des points et de la distance δ_{ij} sont des intervalles. Le domaine d'un point est une boîte définie par le produit cartésien des domaines de ses coordonnées.

La description la plus générale en dimension p de ce problème sous la forme d'un CSP est la suivante :

- $\mathcal{P} = \{P_i(x_i^1,x_i^2,\ldots,x_i^p)\}_{1\leq i\leq n}$ un ensemble de n points de $\mathbb{R}^p$.
- $\mathcal{D} = \{D_i^1,D_i^2,\ldots,D_i^p\}_{1\leq i\leq n}$ un ensemble de domaines associés aux composantes des points et tels que $D_i^k = [\underline{x_i^k},\overline{x_i^k}]$. Le domaine du point P_i, noté D_i est la boite $D_i^1 \times D_i^2 \times \ldots \times D_i^p$.
- $\Delta = \{\Delta_k\}_{1\leq k\leq m}$ un ensemble de m intervalles associés aux distances euclidiennes δ_k entre les couples de points (P_{i_k},P_{j_k}) définit un ensemble de m contraintes :

$$\forall k \leq m \quad c_k : \quad \textstyle\sum_{l=1}^{p} \left(x_{i_k}^l - x_{j_k}^l\right)^2 = {\delta_k}^2 \quad \text{avec } \delta_k \in \Delta_k$$

En dimension 2, on notera les coordonnées du point P_i par (x_i,y_i), son domaine par $D_i = D_{x_i} \times D_{y_i}$ et les contraintes par :

$$\forall k \leq m \quad c_k : \quad (x_{i_k} - x_{j_k})^2 + (y_{i_k} - y_{j_k})^2 = {\delta_k}^2$$

Les distances sont des caractéristiques du problème : elles ne seront pas représentées par des variables du CSP et on ne cherchera pas à réduire leur domaines. Les intervalles associés aux distances, généralisent les relations de distance au sens large. À un intervalle dont les bornes sont égales correspond une équation. Deux inéquations représentent la relation de distance correspondant à un domaine non ponctuel.

La section suivante présente la technique de décomposition des domaines qui est au cœur de notre démarche.

4 Décomposition sémantique d'un CSP

Les résolveurs de contraintes utilisent habituellement une technique de bissection qui consiste à découper un domaine en deux parties généralement de même taille. Différentes heuristiques (taille des domaines, variables contiguës, . . .) peuvent être utilisées pour sélectionner le domaine de la variable à bissecter. C'est une méthode systématique qui ne prend pas en compte les informations spécifiques aux contraintes considérées.

Nous proposons une stratégie de résolution qui tire profit des informations spécifiques aux contraintes de distance. Les principales caractéristiques de cette méthode sont :

- Un découpage simultané de toutes les variables liées par une contrainte.
- Une stratégie de choix de point de coupe qui exploite les propriétés sémantiques des contraintes (monotonie et convexité).

Cette stratégie améliore les performances des résolveurs de contraintes basées sur des consistances locales (*c.f.* expérimentations dans la dernière section de l'article).

Dans cette section nous décrivons plus formellement notre algorithme de décomposition. La section 4.2 détaille la réécriture du CSP initial avant l'application de notre stratégie de recherche. Dans la section 4.3, nous verrons que le schéma général de cet algorithme repose non sur un découpage systématique des domaines mais guidé par la sémantique des contraintes. Dans la section suivante nous présentons cette technique de décomposition qui est au coeur de notre démarche.

4.1 Décomposition des domaines guidée par la sémantique des contraintes

On considère la forme canonique des équations de distance :

$$c : x^2 + y^2 = \delta^2$$

Cette forme nous permet d'isoler rapidement les sous-domaines des variables x et y sur lesquels la contrainte c possède un support et est monotone.

Définition 4.1 (Semantic Domain Decomposition(SDD)) *Considérons la contrainte* $c(x,y)$: $x^2+y^2=\delta^2$ *avec* $(x,y) \in D$. *On définit l'ensemble SDD*(c,D) *des sous-domaines engendrés par la décomposition sémantique du domaine* D *relativement à la contrainte* c. *Ces sous-domaines sont définis par :*

$$D_+^+ = \square\{(x,y) \in D : x \geq 0 \wedge y \geq 0 \wedge c(x,y)\}$$
$$D_+^- = \square\{(x,y) \in D : x \leq 0 \wedge y \geq 0 \wedge c(x,y)\}$$
$$D_-^- = \square\{(x,y) \in D : x \leq 0 \wedge y \leq 0 \wedge c(x,y)\}$$
$$D_-^+ = \square\{(x,y) \in D : x \geq 0 \wedge y \leq 0 \wedge c(x,y)\}$$

où $\square\{S\}$ *désigne la plus petite boîte contenant tous les éléments de l'ensemble* S *et* $c(\alpha,\beta)$ *signifie que la contrainte* c *est vérifiée pour* $x=\alpha$ *et* $y=\beta$. *Notons que certains de ces sous-domaines peuvent être vides.*

Propriété 4.1 *Soit* $f(x,y) = x^2+y^2-\delta^2$ *et la contrainte définie par* $f(x,y)=0$ *et Soit* $s \in$ *SDD*(c,D), *tel que* $s \neq \emptyset$. *Alors la fonction* f *est monotone sur* s.

Preuve Le vecteur gradient de $f(x,y)$ est défini par :

$$\nabla_f(x,y) = (\frac{\partial f}{\partial x}(x,y), \frac{\partial f}{\partial y}(x,y)) = (2x,2y)$$

Chacunes des composantes de $\nabla_f(x,y)$ est de signe constant sur tout sous-domaine $s \in \text{SDD}(c,D)$, par définition. Donc f est monotone sur s.

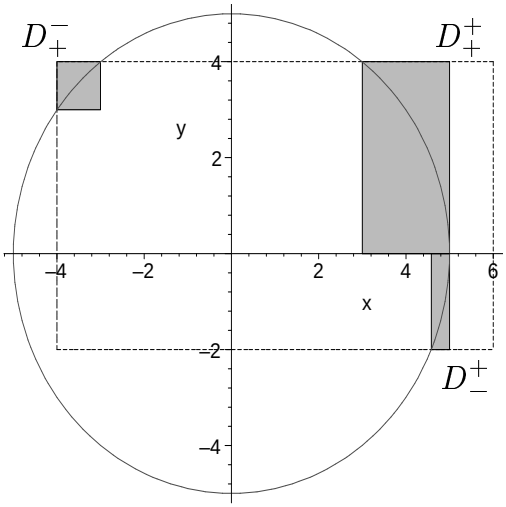


FIG. 5 – *Illustration de la décomposition sémantique sur l'exemple 4.1.*

Comme f est monotone sur chacun des sous-domaines considérés, nous pouvons utiliser l'extension aux intervalles des fonctions de projection[CDR98] associées à $f(x,y)$ pour calculer la plus petite boite qui contient toutes les solutions de $c(x,y)$ sur le dit domaine.

Exemple 4.1 *Considérons les variables x et y de domaines respectifs $[-4,5]$ et $[-2,4]$ et la contrainte c :*

$$c : x^2 + y^2 = 25$$

La figure 5 montre que la décomposition sémantique découpe le domaine du vecteur (x,y) en 3 sous-domaines : $[0,5] \times [0,4]$ réduit à $[3,5] \times [0,4]$, à l'aide de la projection de la contrainte c. De même, $[-4,0] \times [0,4]$ est réduit à $[-4,-3] \times [3,4]$. $[-4,0] \times [-2,0]$ est éliminé puisque la contrainte n'a pas de support dans ce domaine. Enfin, $[0,5] \times [-2,0]$ est réduit à $[4.5825,5] \times [-2,0]$.

La fonction SDD (*c.f.* Fonction 1) réalise cette décomposition sémantique en considérant une contrainte de distance sous forme canonique C et le domaine $D = D_x \times D_y$ du vecteur associé à C. La réduction des sous-domaines (ligne 7) est effectuée par la fonction Narrow(D,C) qui utilise les fonctions de projection suivantes :

$$D_x \leftarrow D_x \cap SQRT(\Delta^2 - Y^2)$$

$$D_y \leftarrow D_y \cap SQRT(\Delta^2 - X^2)$$

où $SQRT(\Delta^2 - X^2)$ est l'extension aux intervalles[CDR98] de la fonction $\sqrt{\delta^2 - x^2}$. Les sous-domaines non réduits à l'ensemble vide sont ajoutés à l'ensemble S, destiné à contenir les sous-domaines résultant de cette décomposition (ligne 9).

Function 1 SDD(C,D)

```
1:  D_1 ← {(x,y) ∈ D : x ≥ 0 ∧ y ≥ 0}
2:  D_2 ← {(x,y) ∈ D : x ≤ 0 ∧ y ≥ 0}
3:  D_3 ← {(x,y) ∈ D : x ≤ 0 ∧ y ≤ 0}
4:  D_4 ← {(x,y) ∈ D : x ≥ 0 ∧ y ≤ 0}
5:  S ← ∅
6:  for all i such that 1 ≤ i ≤ 4 do
7:      D_i ← Narrow(D_i,C)
8:      if D_i ≠ ∅ then
9:          S ← S ∪ {D_i}
10:     end if
11: end for
12: return S
```

Cette fonction dont la complexité temporelle est constante renvoie donc l'ensemble des sous-domaines décrits par la proposition 4.1.

Notons qu'en dimension p le nombre maximal de sous-domaines engendrés par cette décomposition est exponentiel et égal à 2^p. Néanmoins cette méthode est tout à fait adaptée en dimension faible (2 et 3), où les applications faisant intervenir des contraintes de distance sont les plus nombreuses.

Notons également qu'un sous-domaine s issu de la décomposition sémantique ne peut être décomposé à plusieurs reprises, ce qui se traduit par la propriété suivante :

Propriété 4.2 (Point fixe) *Soit* $s \in SDD(c,D)$ *alors* $SDD(c,s) = s$.

4.2 Réécriture du CSP initial

Considérons un CSP $\mathcal{P} = (\mathcal{X},\mathcal{D},\mathcal{C})$ constitué exclusivement de m contraintes de distance de la forme[3] :

$$c_k : (x_{j_k} - x_{i_k})^2 + (y_{j_k} - y_{i_k})^2 = \delta_k^2$$

La réécriture du CSP initial consiste à mettre toutes les contraintes sous forme canonique. Pour chaque contrainte c_k on introduit deux variables additionnelles X_k et Y_k correspondant aux coordonnées du vecteur $\overrightarrow{P_{i_k}P_{j_k}}$. Les domaines V_k des vecteurs (X_k,Y_k) sont calculés en utilisant l'arithmétique des intervalles. La réécriture du système consiste à ajouter les contraintes de changement de repère et à remplacer c_k par sa forme canonique :

$$\begin{cases} c'_k : X_k^2 + Y_k^2 = \delta_k^2 \\ X_k = x_{j_k} - x_{i_k} \\ Y_k = y_{j_k} - y_{i_k} \end{cases}$$

L'ensemble des domaines est donc de la forme :

$$\mathcal{D} = \{V_1, \ldots, V_m, D_1, \ldots, D_n\}$$

où D_i est le domaine du point P_i avec $1 \leq i \leq n$, et V_k est le domaine du vecteur (X_k,Y_k) associé à la contrainte c'_k avec $1 \leq k \leq m$.

Les contraintes c'_k étant sous forme canonique, les domaines V_k des vecteurs (X_k,Y_k) pourront être décomposés en utilisant la technique de décomposition sémantique (*c.f* section 4.1).

4.3 Algorithme de décomposition sémantique

Notre algorithme de Décomposition Sémantique, nommé `DS`, considère un CSP (X,D,C) et calcule une décomposition du domaine initial D et un ensemble de sous-domaines sur lesquels :

- toutes les contraintes de C sont monotones
- une propriété de consistance locale est vérifiée

3. Nous décrivons notre méthode en dimension 2, l'extension en dimension supérieure est triviale

Par exemple, soit le CSP initial $(\{x,y\},\{D_x,D_y\},C)$. DS va générer un ensemble de couples de la forme $\{D_x^i,D_y^j\}$ où $D_x^i \subset D_x$ et $D_y^j \subset D_y$ De plus, pour chacun de ces couples engendrés par DS, les CSP $(\{x,y\},\{D_x^i,D_y^j\},C)$ vérifient une propriété de consistance locale.

DS est donc paramétré par une fonction nommée LC qui étant donné un CSP (X,D,C) renvoie l'ensemble des domaines obtenus par un filtrage utilisant une consistance locale(*e.g.* 2B-consistance[Lho93, Lho94], Box-consistance[PVH96]). Si le CSP ne vérifie pas la consistance locale associée à LC, cette fonction renvoie un ensemble vide.

Le principe de DS (voir fonction 2) est sensiblement le même que celui d'un algorithme de recherche arborescente basé sur la bissection. La principale différence étant le fait que les domaines des (X_k,Y_k) sont décomposés en utilisant la décomposition sémantique décrite précédemment. Contrairement à la bissection dont le processus de décomposition ne s'arrête que lorsque tous les domaines sont suffisamment petits (de taille inférieure à un $\epsilon > 0$ donné), le point fixe est atteint lorsque tous les domaines on été considérés (voir propriété 4.2). Autrement dit, tous les domaines des vecteurs V_k vérifient la propriété $V_k = \mathtt{SDD}(V_k,c_k)$, ce qui signifie que V_k a déjà été decomposé par SDD.

Function 2 $\mathtt{DS}(\mathcal{X},\mathcal{D},\mathcal{C})$

$R \leftarrow \emptyset$
$Q \leftarrow \{\mathcal{D}\}$
while $Q \neq \emptyset$ **do**
  Choose S from Q %% $S = \{V_1,\ldots,V_m,D_1,\ldots,D_n\}$
  if $V_k = \mathtt{SDD}(V_k,c_k)$ for all $V_k \in S$ **then**
    $R \leftarrow R \cup \{S\}$
  else
    Choose V_k from S
    Let $c_k \in C$ the constraint associated to V_k.
    for all $s \in \mathtt{SDD}(V_k,c_k)$ **do**
      $S' \leftarrow S[V_k \leftarrow s]$ %% $S' = \{V_1,\ldots,V_{k-1},s,V_{k+1},\ldots,V_m,D_1,\ldots,D_n\}$
      $S' \leftarrow \mathtt{LC}(\mathcal{X},S',\mathcal{C})$ %% Local filtering of the domain S'
      if $S' \neq \emptyset$ **then**
        $Q \leftarrow Q \cup \{S'\}$
      end if
    end for
  end if
end while
return R

Plus précisément, DS utilise un ensemble Q contenant des sous-domaines du domaine initial et initialisé à $\{D\}$ (ligne 2). S est alors extrait de l'ensemble Q et l'on vérifie si le point fixe est atteint (lignes 4 et 5).

Si c'est le cas, S est ajouté à R, l'ensemble destiné à contenir les sous-domaines engendrés par DS (ligne 6). Sinon, on choisit un vecteur dont le domaine V_k n'a pas encore été décomposé par SDD (ligne 8). Pour tous les sous-domaines engendrés par la décomposition sémantique du domaine du vecteur choisi (ligne 10), on ajoute à Q les sous-domaines de S correspondants s'ils vérifient la propriété de consistance locale (lignes 10 à 15). L'appel à la fonction LC utilise toutes les contraintes de $\mathcal{C}$ pour réduire les domaines de toutes les variables.

L'heuristique utilisée pour choisir le sous-domaine extrait de Q (ligne 4) est de choisir celui dont la décomposition est la plus avancée. Cela correspond à une recherche arborescente en profondeur d'abord, ce choix se fait donc en temps constant.

L'heuristique utilisée pour choisir le prochain domaine à décomposer (ligne 8) est de choisir celui dont la décomposition par SDD engendre un nombre de sous-domaines minimal et strictement supérieur à 1. Une recherche exhaustive du minimum est utilisée avec une complexité temporelle en $\mathcal{O}(m)$ où m est le nombre de vecteurs, c'est-à-dire de contraintes du système.

Nous allons maintenant montrer l'apport de notre algorithme de décomposition sur différentes expérimentations.

5 Résultats expérimentaux

Les sections 5.1 et 5.2 détaillent respectivement les problèmes du pentagone et de la cinématique directe d'un robot plan. Les résultats obtenus par DS sur ces problèmes sont détaillés dans la section 5.3, ainsi que la comparaison de ces résultats avec ceux obtenus avec la bissection.

5.1 Problème classique du pentagone

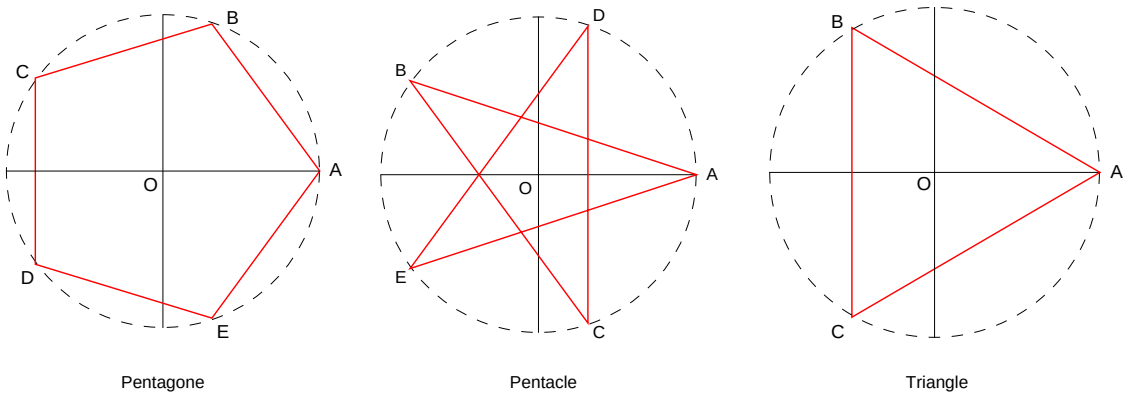


FIG. 6 – *Solutions du problème du pentagone*

Le problème du pentagone est un CSP bien connu dans la communauté de la programmation par contraintes sur les domaines continus. Il est souvent utilisé pour illustrer les problèmes que rencontrent les méthodes de filtrage local, lorsque l'espace solution est fractionné. Il s'agit d'un CSP géométrique en 2D, constitué de 5 points P_1,P_2,P_3,P_4, et P_5 sur le cercle unitaire, tels que les segments P_1P_2, P_2P_3, P_3P_4, P_4P_5, et P_1P_5 soient de longueur égale à d. L'un de ces 5 points est fixé au point de coordonnées (1,0) pour éviter que le nombre de solutions ne soit infini. Selon le choix de la distance d entre deux points consécutifs, on obtient 3 instances que nous nommerons *penta1*, *penta2* et *penta3* donnant lieu à 3 types de solutions (Fig. 6) :

1. *penta1* : Si $d = 2\sin(\frac{\pi}{5})$ il y a 2 solutions pour la combinaison $P_1P_2P_3P_4P_5$ formant un pentagone, $ABCDE$, $AEDCB$.
2. *penta2* : Si $d = 2\sin(\frac{2\pi}{5})$ il y a 2 solutions pour la combinaison $P_1P_2P_3P_4P_5$ formant un pentacle, $ABCDE$, $AEDCB$.
3. *penta3* : Si $d = 2\sin(\frac{2\pi}{3})$ il y a 10 solutions pour la combinaison $P_1P_2P_3P_4P_5$ formant un triangle, $ABCAB$,$ACBAC$, $ABCAC$, $ACBAB$, $ABABC$, $ACACB$, $ABCBC$, $ACBCB$, $ABACB$ et $ACABC$.

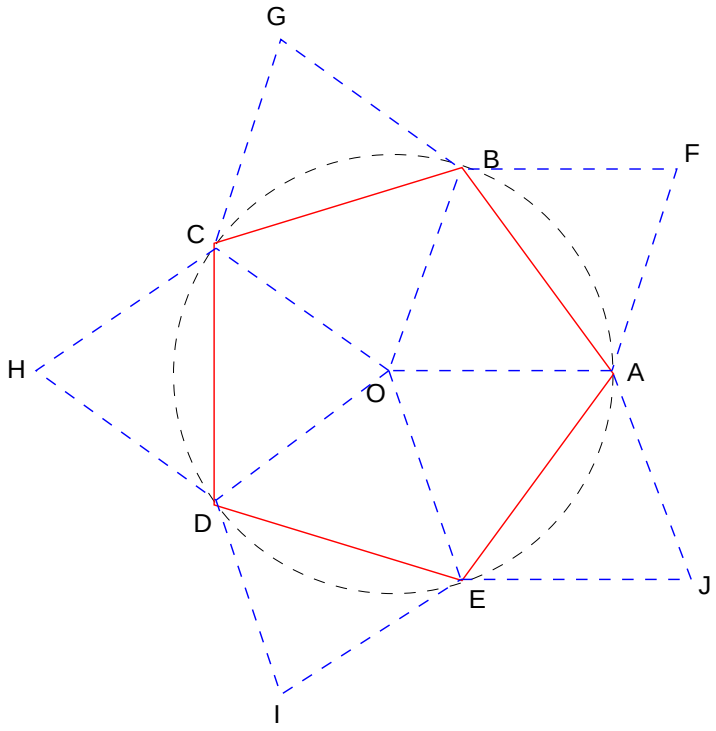


FIG. 7 – *Ajout de contraintes dans le problème du pentagone*

Pour compliquer un peu le problème, cinq points supplémentaires sont ajoutés au problème initial, un entre chaque arête à une distance de 1 de chaque extrémité (Fig. 7). Pour une solution du problème classique, cela engendre $2^5 = 32$ fois plus de solutions, soit 64 pour *penta1* et *penta2*, et 320 pour *penta3*. L'instance correspondant à l'extension de *penta1* (resp. *penta2*, *penta3*) étendue par ce procédé sera nommée *ext-penta1* (resp. *ext-penta2*, *ext-penta3*)

5.2 Cinématique directe d'un robot plan

Le manipulateur de type 3-RPR est formé de deux triangles: la base ABC et la plate-forme DEF reliés par 3 jambes AD, BE et CF (voir figure 8). Le problème de la cinématique directe est de trouver toutes les positions de la plate-forme compatibles avec les longueurs des jambes qui sont les données du problème. Ce problème possède 6 solutions réelles[GSR92].

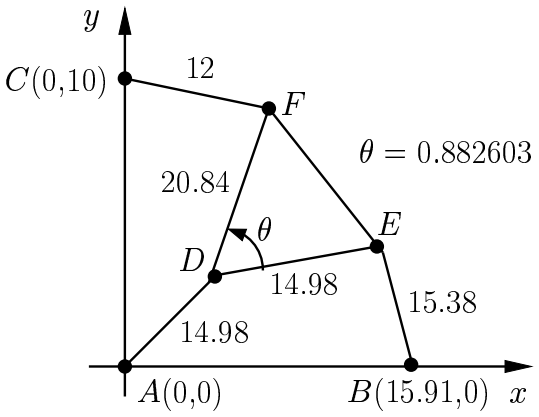


FIG. 8 – *Un manipulateur planaire de type 3-RPR*

5.3 Résultats et analyse

Pb(#sol)	Bissection				DS			
	2B(10^{-10})		Box(10^{-10})		2B(10^{-10})		Box(10^{-10})	
	t(s)	#box	t(s)	#box	t(s)	#box	t(s)	#box
penta1(2)	0.02s	4	0.16s	4	0.03s	2	0.16s	2
penta2(2)	0.03s	100	0.07s	2	0.03s	2	0.03s	2
penta3(10)	0.04s	56	0.63s	192	0.03s	10	0.18s	10
ext-penta1(64)	1.47s	5128	3.02s	64	0.26s	64	2.71s	64
ext-penta2(64)	3457.32s	430798	1.91s	64	0.08s	16	0.52s	16
ext-penta3(320)	21.44s	87642	1705.45s	210398	1.12s	320	9.53s	320
robot2D(6)	7.77s	127	1.7s	12	0.26s	18	0.37s	20

FIG. 9 – *Résultats expérimentaux*

Nous avons comparé les performances de DS avec celles de la bissection en combinant ces méthodes avec la 2B ou la Box avec une précision de 10^{-10}. Les expérimentations ont été réalisés en utilisant le solveur Ilog Solver 5.0 [ILO02] sur un PC équipé d'un microprocesseur pentium IV à 2GhZ et 128Mo de mémoire.

Les résultats (voir tableau 9) montrent que pour les trois premiers exemples (penta1,penta2, penta3), les temps de toutes les approches sont de l'ordre du centième de secondes et la comparaison n'est pas significative. On note toutefois que DS génère une boîte par solution alors que la bissection génère de nombreuses boîtes parasites.

Pour *ext-penta3*, les performances de DS sont bien meilleures que celles d'un algorithme de recherche basé sur la bissection (le facteur de gain est compris entre 20 et 200). Pour *ext-penta2*, le gain de temps est également significatif mais certaines boîtes doivent encore être découpées pour isoler les solutions.

Pour robot2D, le gain de temps est également significatif mais quelques boîtes sans solutions sont générées, ce qui est dû aux faibles capacités de filtrage des consistances locales utilisées. Sur cet ensemble de tests préliminaires, les gains en performances et précision sont très encourageants.

6 Conclusion

Nous avons introduit dans cet article une nouvelle méthode de recherche basée sur une décomposition des domaines des variables, qui exploite directement la sémantique des contraintes de distance entre deux points. Les premiers résultats sur des exemples critiques sont encourageants. La suite des travaux porte d'une part la poursuite des expérimentations sur des problèmes issus de la robotique et de la théorie des mécanismes, d'autre part l'étude des possibilités d'élargir cette approche à d'autres systèmes de contraintes non-linéaires.

Références

[Bat02a] H. Batnini. Contraintes globales pour la résolution de contraintes de distance. Master's thesis, Université de Nice Sophia-Antipolis, Juin 2002.

[Bat02b] H. Batnini. Introduction of redundant constraints for solving distance constraints systems. *Journal of the university of Saärbrück, CALCULEMUS autumn school: Student poster abstracts, Pisa, Italy*, pages 19–23, September 2002.

[BMB02] L. Bordeaux, E. Monfroy, and F. Benhamou. Raisonnement sur les propriétés de contraintes numériques. In M. Rueher, editor, *Programmation en logique par contrainte*, pages 13–26. JFPLC'02, Hermès Science publications, 2002.

[BO93] F. Benhamou and W. Older. Applying interval arithmetic to real, integer and Boolean constraints. *Logic Programming: The ALP Newsletter*, 6(2):13–14, 1993.

[CDR98] H. Collavizza, F. Delobel, and M. Rueher. A note on partial consistencies over continuous domains. In M. Maher and J-F. Puget, editors, *Proc. of CP'98: 4th International Conference on "Principles and Practice of Constraint Programming"*, LNCS 1520, pages 147–162, Pisa, Italy, November 1998. Springer Verlag.

[CH88] G.M. Crippen and T.F Havel. *Distance geometry and molecular conformation*. John Wiley and sons, 1988.

[Die98] P. Dietmaier. The stewart-gough platform of general geometry can have 40 real postures. In J. Lenarcic and M.L. Husty, editors, *Advances in Robot Kinematics: Analysis and Control*, pages 7–16. Kluwer Academic Publisher, 1998.

[GSR92] C. Gosselin, J. Sefrioui, and M. J. Richard. Polynomial solutions to the direct kinematic problem of planar three degree of freedom parallel manipulators. *Mechanism and Machine Theory*, 27(2):107–119, 1992.

[Heu03] M. Heusch. distn: An euclidean distance global constraint. In F. Rossi, editor, *Proc. of CP'03: 9th International Conference on "Principles and Practice of Constraint Programming")*, LNCS 2833, pages 975–975. Springer Verlag, September 2003.

[ILO02] ILOG. *Solver Reference manual http://www.ilog.com/product/jsolver*. 2002.

[JNT02] C. Jermann, B. Neveu, and G. Trombettoni. A new structural rigidity for geometric constraints systems. In *Proc. of Fourth International Workshop on Automated Deduction in Geometry(ADG'02)*, Linz, Austria, 2002.

[JNT03] C. Jermann, B. Neveu, and G. Trombettoni. Inter-block backtracking: Exploiting the structure in continuous csps. In *2nd International Workshop on Global Constrained Optimization and Constraint Satisfaction COCOS'03*, Lausanne, Switzerland, 2003.

[JTNR00] C. Jermann, G. Trombettoni, B. Neveu, and M. Rueher. A constraint programming approach for solving rigid geometric systems. In *Proc. of CP'00: Sixth International Conference on "Principles and Practice of Constraint Programming"*, LNCS 1894, pages 233–248, Singapore, September 2000. Springer Verlag.

[KB00] L. Krippahl and P. Barahona. Psico: Combining constraint programming and optimisation to solve macromolecular structures. In *Proc. of ERCIM/COMPULOG Workshop on Constraints*, Univ. of Padova, Italy, June 2000.

[Laz92] D. Lazard. Stewart platforms and gröbner basis. In *Proceedings of Advances in Robotics Kinematics*, pages 136–142, Septembre 1992.

[Lho93] O. Lhomme. Consistency techniques for numerical csps. In *IJCAI-93*, pages 232–238, 1993.

[Lho94] O. Lhomme. *Contribution à la résolution de contraintes sur les réels par propagation d'intervalles*. Thèse de doctorat, Université de Nice-Sophia Antipolis, 1994.

[LRM02] Y. Lebbah, M. Rueher, and C. Michel. A global filtering algorithm for handling systems of quadratic equations and inequations. In P. Van Hentenryck, editor, *Proc of CP'02: 8th International Conference on "Principles and Practice of Constraint Programming"*, LNCS 2470, Cornell University, Ithaca, NY, USA, September 2002. Springer Verlag.

[Mac77] A.K Macworth. Consistency in networks of relations. *Artificial Intelligence*, pages 99–118, 1977.

[Mar00] M. Cs. Markót. An interval method to validate optimal solutions of the "packing circles in a unit square". *Problems, Central European Journal of Operational Research*, 8:63–78, 2000.

[MC04] M. Cs. Markót and T. Csendes. A new verified optimization technique for the "packing circles in a unit square" problems. *SIAM*, 2004. (submitted).

[Mer03] J-P. Merlet. Solving the forward kinematics of gough-type parallel manipulator with interval analysis. *Research report INRIA 2003*, RR-4013, 2003.

[MLR03] C. Michel, Y. Lebbah, and M. Rueher. Safe embedding of the simplex algorithm in a csp framework. In *5th Int. Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimisation Problems CPAIOR 2003*, pages 210–220, Université de Montréal, 2003. CRT.

[Moo77] R. Moore. *Interval analysis*. Prentice-Hall, 1977.

[PVH96] J.F. Puget and P. Van Hentenryck. A constraint satisfaction approach to a circuit design problem. Technical Report CS-96-34, 1996.

[SCCG01] P.G. Szabó, T. Csendes, L.G. Casado, and I. García. Equal circles packing in a square i. - problem setting and bounds for optimal solutions. *Optimization Theory - Recent Developments from Matrahaza. Kluwer, Dordrecht*, pages 191–206, 2001.

[VSHS02] X.H. VU, D. Sam-Haroud, and M.C. Silaghi. Résolution de problèmes non linéaires avec continuum de solutions. In M. Rueher, editor, *Programmation en logique par contrainte*, pages 13–26. JFPLC'02, Hermès Science publications, 2002.