Filtrage local par décomposition de CSP continus

Heikel Batnini
INRIA-I3S-CERMICS,
2004 route des Lucioles
06902 Sophia-Antipolis
email: hbatnini@sophia.inria.fr

Michel Rueher
INRIA-I3S-CERMICS,
2004 route des Lucioles
06902 Sophia-Antipolis
email: rueher@essi.fr

Résumé

La plupart des outils pour la résolution de CSP continus s'appuient sur des consistances locales qui encadrent l'espace solution. Lorsque celui-ci est fractionné, cela conduit à conserver des valeurs incohérentes dans l'ensemble du système. De ce fait, les techniques de filtrage local sont utilisées en combinaison avec des méthodes d'énumération, comme la *bissection*, afin d'isoler les sous-espaces qui contiennent potentiellement une solution. Nous proposons dans cet article une nouvelle technique, nommée LDF(Local Decomposition Filtering), basée sur un couplage entre une décomposition des domaines utilisant la sémantique des contraintes et une consistance locale. LDF décompose le CSP initial en un ensemble de CSP dont les domaines sont inclus dans les domaines initiaux et permettant d'orienter la recherche de solutions. La réduction globale des domaines est au moins égale à celle de la *2B-consistance*. Nous illustrons l'apport de cette technique sur différents CSP constitués d'équations de distance.

1 Introduction

Le problème de satisfaction de contraintes numériques(ou NCSP [1]) intervient dans de nombreuses applicationsindustrielles(e.g., conception assistée par ordinateur, robotique, ...). 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 consistué 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[8], basées sur l'arithmétique des intervalles[9], comme la kB[6] ou la Box[10], réalisent efficacement une approximation extérieure de l'ensemble des solutions.

Lorsque l'espace solution est divisé en sous-ensembles disjoints, une approximation extérieure conduit à conserver dans certains domaines des valeurs non solutions,

1. Numerical Constraint Solving Problem

dont la 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 ces techniques de filtrage local. Cependant, elles se révèlent souvent inadaptées en présence de continuums de solutions[12].

Nous proposons dans cet article, une nouvelle méthode de filtrage local basée sur une représentation des domaines par union d'intervalles. Cet algorithme, que nous nommerons par la suite LDF(Local Decomposition Filtering), allie décomposition multi-variables et propagation. LDF conserve également une *micro-structure* qui peut directement être exploitée pour guider la recherche de solutions.

Nous avons choisi d'explorer dans un premier temps les apports de cette technique sur des CSP constitués d'équations de distance pour deux raisons. Tout d'abord, c'est un problème qui a un vaste champs d'applications (robotique, biochimie moléculaire ...) et pour lequel l'espace des solutions est souvent constitué de sous-espaces continus. D'autre part, le regroupement des variables est intuitif et correspond à celui des coordonnées d'un même point.

Avant de détailler notre approche, nous allons illustrer ses principes sur un exemple simple.

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

FIG. 1 – *Description du CSP correspondant à l'exemple 1.1*

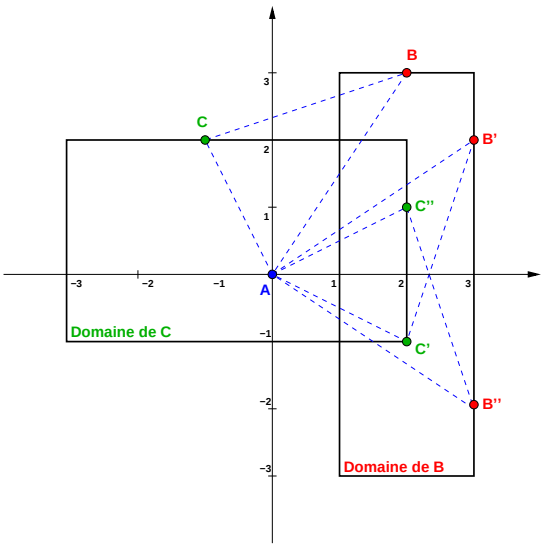

FIG. 2 – *Les 3 solutions de l'exemple 1.1 sont ABC,$AB'C'$ et $AB''C''$.*

Exemple 1.1 *Considérons dans le plan, 3 points* $A(0,0)$, $B(x_B,y_B)$ *et* $C(x_C,y_C)$ *où le point* A *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 connues :* $AB = \sqrt{13}$,$AC = \sqrt{5}$ *et* $BC = \sqrt{10}$. *La figure 1 montre le CSP correspondant à cet exemple, les 3 solutions de ce CSP sont représentées graphiquement sur la figure 2.*

Pour construire la micro-structure de ce CSP, représentée dans la figure 4, LDF commence par découper le domaine de B à l'aide de la contrainte C_{AB}. Le domaine de B est décomposé, selon les axes du repère cartésien, en deux parties réduites à B_1 et B_2 à l'aide de la contrainte C_{AB} (Fig. 3 à gauche). Deux boîtes contenant l'arc de cercle centré en A et de rayon $\sqrt{13}$ sont ainsi obtenues.

De même pour le domaine de C, en utilisant la contrainte C_{AC}, on obtient quatre boîtes C_1, C_2, C_3 et C_4(Fig. 3 à droite). Dans la micro-structure, le domaine de A sera donc lié à B_1 et B_2 d'une part, et à C_1, C_2, C_3 et C_4 d'autre part(Fig. 4).

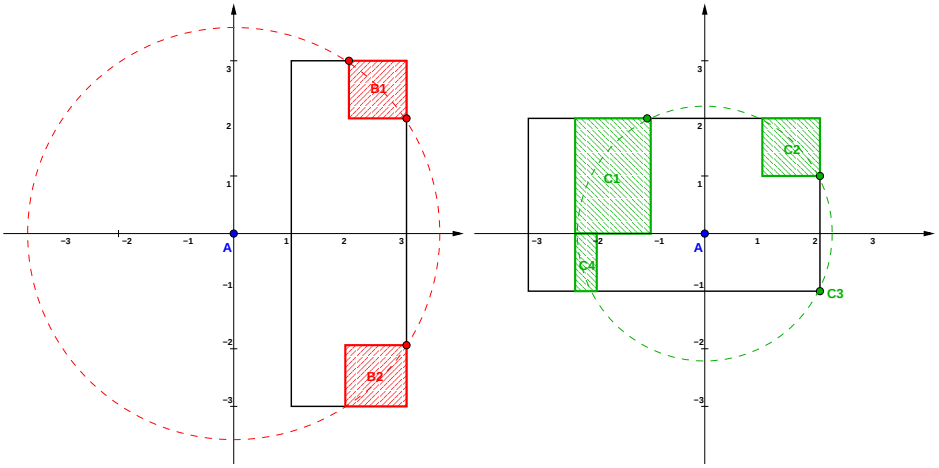

FIG. 3 – *Décomposition des domaines de* B *à gauche et de* C *à droite*

Le traitement de la contrainte C_{BC} nous conduit à examiner les 8 combinaisons des sous-domaines de B et C :

- C_4 est éliminé car il n'a de support ni dans B_1, ni dans B_2.
- C_2 est réduit à $[2,2] \times [1,1]$ et B_2 à $[3,3] \times [-2,-2]$. B_2 et C_2 sont donc reliés par une arête dans la micro-structure(Fig. 4).
- La combinaison B_1C_1 réduit C_1 à $[-1.16,-1] \times [1.91,2]$ et B_1 à $[2,2.16] \times [2,3]$. La combinaison B_1C_3 réduit B_1 à $[2,3] \times [2,2.16]$. Les 2 sous-domaines de B_1 s'intersectent donc on les unie, d'où $B_1 = [2,3] \times [2,3]$. Les arêtes (B_1,C_1) et (B_1,C_3) sont donc ajoutées à la micro-structure(Fig. 4).

La micro-structure générée par LDF sur l'exemple 1.1, décompose le CSP initial en 3 sous-CSP : AB_1C_1, AB_1C_2 etAB_2C_3. Une deuxième itération de ce processus sur chacun de ces 3 sous-CSP suffit pour isoler les 3 solutions du système.

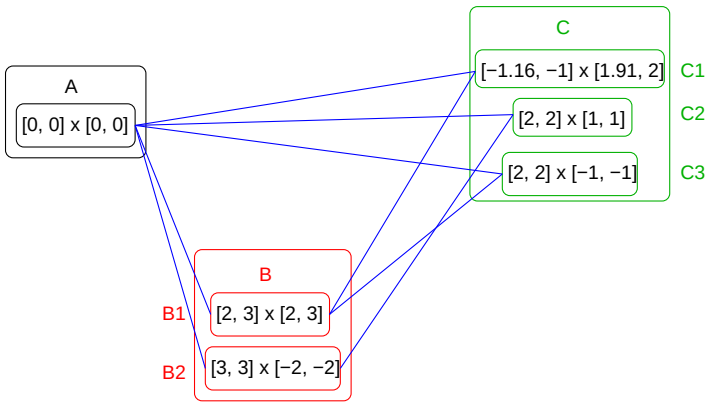

FIG. 4 – *Micro-structure généré par LDF pour l'exemple 1.1*

Les 3 principales caractéristiques de LDF sont les suivantes :

- *Décomposition multi-variables* : Un découpage simultané des domaines de plusieurs variables mises en jeu par une contrainte, en utilisant ses propriétés de monotonie. La forme canonique des équations de distance, $X^2 + Y^2 - D^2 = 0$ est utilisée pour isoler les parties monotones sur $\mathbb{R}^+ \times \mathbb{R}^+$, $\mathbb{R}^+ \times \mathbb{R}^-$, $\mathbb{R}^- \times \mathbb{R}^+$ et $\mathbb{R}^- \times \mathbb{R}^-$. Par changement de repère, on peut alors extraire les sous-domaines des variables sur ces parties monotones.
- *Un algorithme de propagation* : Il s'agit d'un algorithme de point fixe de type 2B-consistance. Chaque réduction du domaine d'une variable est propagée sur l'ensemble des variables qui lui sont liées par une contrainte, jusqu'à ce qu'il n'y ait plus de réduction supérieure à une précision ω donnée. Le filtrage est réalisé par une fonction de projection spécifique à LDF qui réalise la décomposition multi-variable et qui assure la gestion des disjonctions de domaines.
- *Une micro-structure* : LDF met à jour en permanence une *micro-structure* [2], qui est un graphe n-partis ou n est le nombre de groupements de variables. L'ensemble des sommets de la micro-structure correspond à l'ensemble des sous-domaines des regroupements de variables. Deux sous-domaines sont liés par une arête s'ils satisfont l'ensemble des contraintes mettant en jeu les deux regroupements de variables correspondants.

La suite de cet article est organisé de la manière suivante : Dans la section 2, nous rappellons les notions de base qui sont utilisées dans le reste de l'article. Les sections 3 et 4 détaillent respectivement la technique de décomposition des domaines, et l'algorithme de propagation de LDF. La micro-structure générée par LDF ainsi que ses propriétés sont présentées dans la section 5. Enfin, dans la section 6, nous montrons que LDF surpasse les méthodes traditionnelles(kB+bissection) pour différentes variantes du classique problème du pentagone.

2. On peut comparer ce graphe à celui que calcule l'arc-consistance sur des CSP aux domaines finis

2 Préliminaires

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

Définition 2.1 (Constraint Solving Problem(CSP)) *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 $D_x = [\underline{x},\overline{x}]$. La 2B-consistance[6; 7] est définie de la manière suivante :

Définition 2.2 (2B-consistance) *Soit $(\mathcal{X},\mathcal{D},\mathcal{C})$ un $\mathcal{CSP}$ et $c \in C$ une contrainte k-aire sur les variables $(x_1,\ldots,x_k)$. La contrainte c est dite 2B-consistante ssi : $\forall i$, $D_{x_i} = \square\{x_i \in D_{x_i} | \exists v_1 \in D_{x_1} \ldots \exists v_k \in D_{x_k}$ et $c(v_1,\ldots,v_{i-1},x_i,v_{i+1},\ldots,v_k)$ est vérifiée $\}$. La notation $\square\{E\}$ désigne la plus petite boîte contenant tous les éléments de l'ensemble E. Un $\mathcal{CSP}$ est 2B-consistant ssi toutes ses contraintes sont 2B-consistantes.*

L'algorithme de calcul de la 2B-consistance a une complexité temporelle en $\mathcal{O}(nmA)$ dans le pire des cas, où n est le nombre de variables, m est le nombre de contraintes et A est la taille[3] du plus grand domaine de $\mathcal{D}$. C'est un algorihme de point fixe, pouvant converger asymptotiquement en pratique.

Dans la suite, nous nous intéresserons exclusivement aux CSP consistués de contraintes de distances. Nous considérons également que le graphe de contraintes de distances est complété à l'aide de l'algorithme de *bound-smoothing* [4]. Cet algorithme calcule une fermeture transitive du graphe de contraintes, en associant aux distances manquantes un domaine qui vérifie les inégalités triangulaires du systèmes[4].

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

3 Décomposition guidée par les contraintes

Les résolveurs de contraintes utilisent habituellement une technique de bissection qui consiste à décomposer un domaine en deux parties de même taille. Différentes heuristiques (tailles des domaines, variables contiguës, ...) peuvent être utilisées pour sélectionner le domaine de la variable à bissecter.

Nous introduisons ici une technique de décomposition nommée CODD(Constraint Oriented Domain Decomposition), qui considère une contrainte c mettant en jeu un ensemble de variables $\mathcal{X}$. Contrairement à la bissection classique, CODD ne découpe

3. C'est à dire le nombre de flottants compris dans A
4. Étant donnés 3 points A, B, et C, on a $|AC - BC| \leq AB \leq AC + BC$

pas séquentiellement le domaine des variables mais simultanément les domaines d'un sous-ensemble de variables de $\mathcal{X}$. La séléction de cet ensemble de variables est basée sur la sémantique de la contrainte c. Nous allons tout d'abord voir les grands principes de CODD sur un exemple simple en dimension 2.

Dans le cas des équations de distance, on considère deux points de $\mathbb{R}^2$, $P_i(x_i,y_i)$ et $P_j(x_j,x_j)$, reliés par une contrainte de distance $C_{ij} : (x_i - x_j)^2 + (y_i - y_j)^2 = d_{ij}^2$, et dont les domaines sont $D_{P_i} = D_{x_i} \times D_{y_i}$ et $D_{P_j} = D_{x_j} \times D_{y_j}$.
Basiquement, une CODD sur D_{P_i} orientée par la contrainte C_{ij} se fait en trois étapes :

- *Changement de variables* : On pose $X = x_i - x_j$ et $Y = y_i - y_j$ qui sont en fait les composantes du vecteur $\overrightarrow{P_jP_i}$. Les domaines de X et Y sont calculés en utilisant l'arithmétique des intervalles à partir des domaines de x_i, y_i, x_j et y_j.
- *Décomposition* : $D_X \times D_Y$ est décomposé par projection sur les parties monotones de la contrainte $X^2 + Y^2 = d_{ij}^2$. Quatre sous-domaines pour le vecteur $\overrightarrow{P_jP_i}$ sont ainsi obtenus par projection sur chacun des sous-espaces $\mathbb{R}^+ \times \mathbb{R}^+$, $\mathbb{R}^+ \times \mathbb{R}^-$, $\mathbb{R}^- \times \mathbb{R}^+$ et $\mathbb{R}^- \times \mathbb{R}^-$.
- *Filtrage* : Un filtrage par 2B-consistance est effectué sur chacun de ces sous-domaines et l'on répercute ce filtrage sur le domaine de D_{P_i}, par changement de repère($x_i = X + x_j$ et $y_i = Y + y_j$).

D_{Pi} est ainsi décomposé en au plus 4 sous-domaines sur lesquels la contrainte C_{ij} est monotone et convexe. Illustrons cette technique de décomposition sur un exemple simple.

Exemple 3.1 *Soient 2 points $A(x_A,y_A)$ et $A(x_B,y_B)$ tels que $(x_A - x_B)^2 + (y_A - y_B)^2 = d^2$ et dont les domaines sont $x_A,x_B,y_A,y_B \in [-2,3]$ et $d \in [7,8]$.*
Pour réaliser une CODD du domaine de A, $D_A = D_{x_A} \times D_{y_A}$:

- *On pose $X = x_A - x_B$ et $Y = y_A - y_B$ donc le domaine de X et de Y est égal $[-2,3] - [-2,3] = [-5,5]$. Le vecteur $\overrightarrow{AB}$ a donc pour domaine le produit cartésien $[-5,5] \times [-5,5]$.*
- *Ce domaine est découpé en 4 parties sur $\mathbb{R}^+ \times \mathbb{R}^+$, $\mathbb{R}^+ \times \mathbb{R}^-$, $\mathbb{R}^- \times \mathbb{R}^+$ et $\mathbb{R}^- \times \mathbb{R}^-$.*
  Ce qui nous donne 4 boîtes : $[-5,0] \times [-5,0]$, $[-5,0] \times [0,5]$, $[0,5] \times [-5,0]$ et $[0,5] \times [0,5]$.
- *La première boîte est filtrée $[-5,0] \times [-5,0]$ en utilisant les fonctions de projection de la contrainte $X^2 + Y^2 = d^2$ ($X = \sqrt{d^2 - Y^2}$ et $Y = \sqrt{d^2 - X^2}$), ce qui nous donne $[-5, -4.89] \times [-5, -4.89]$. Par changement de repère on obtient :*
  - $D_{x_A} = D_{x_A} \cap ([-5, -4.89] + [-2,3]) = [-2, -1.89]$
  - $D_{y_A} = D_{y_A} \cap ([-5, -4.89] + [-2,3]) = [-2, -1.89]$
- *On procède de la même manière pour $[-5,0] \times [0,5]$, $[0,5] \times [-5,0]$ et $[0,5] \times [0,5]$, et finalement :*

$$D_A = \left\{ \begin{array}{c} [-2, -1.89] \times [-2, -1.89] \\ [-2, -1.89] \times [2.89,3] \\ [2.89,3] \times [-2, -1.89] \\ [2.89,3] \times [2.89,3] \end{array} \right\}.$$

Considérons à présent deux points de $\mathbb{R}^p$, $P_i(x_1^i, \ldots, x_p^i)$ et $P_j(x_1^j, \ldots, x_p^j)$ dont les domaines sont $D_{P_i} = D_{x_1^i} \times \ldots \times D_{x_p^i}$ et $D_{P_j} = D_{x_1^j} \times \ldots \times D_{x_p^j}$. En utilisant la

contrainte $C_{ij} : \sum_{k=1}^{k=p}(x_k^i - x_k^j)^2 = d_{ij}^2$, on réalise une CODD sur D_{P_i} de la manière suivante :

- *Changement de variables* : On pose $X_k = x_k^i - x_k^j$, $\forall k \leq p$, tels que X_k est la k-ème composante du vecteur $\overrightarrow{P_jP_i}$. Les domaines D_{X_k} des X_k sont calculés en utilisant l'arithmétique des intervalles et les domaines de x_k^i et x_k^j.
- *Décomposition* : D_{X_k} est décomposé sur les parties monotones de la contrainte $X_1^2 + \ldots + X_p^2 = d_{ij}^2$.
  2^p sous-domaines pour le vecteur $\overrightarrow{P_jP_i}$ sont ainsi obtenus.
- *Filtrage* : Chacun de ces sous-domaines sont filtrés par une simple projection 2B. Ce filtrage est répercuté sur le domaine de D_{P_i}, par changement de repère($x_k^i = X_k + x_k^j$).

De cette manière, D_{Pi} est découpé en au plus 2^p sous-domaines sur lesquels la contrainte C_{ij} est monotone et convexe.

Dans la suite, nous appellerons **CODD**(d_i, d_j, C_{ij}), la procédure décrite ci-dessus qui renvoie une liste constituée des sous-domaines de d_i, qui vérifient la contrainte C_{ij} pour le domaine d_j de P_j.

Dans la section suivante, nous décrivons la structure de l'algorithme LDF qui filtre les domaines d'un regroupement de variables et gère les disjonctions issues de ce filtrage.

```
PROJECTION(in: D_i, in: D_j, in: C_ij, in-out: L_ij)
  S ← ∅ # Liste de sous-domaines solutions pour D_i
  L_ij ← ∅ # Liste d'adjacence de la micro-structure entre D_i et D_j
  Pour d_i de D_i^1 jusqu'à D_i^{m_i} faire
    Pour d_j de D_j^1 jusqu'à D_j^{m_j} faire
      D ← CODD( d_i, d_j, C_ij)
      d ← head(D)
      Tant que (d ≠ ∅) faire
        s ← head(S)
        ins ← false
        Tant que ((s ≠ ∅) ∧ (¬ins)) faire
          Si ((s ∩ d) ≠ ∅) alors
              s ← s ∪ d
              L_ij ← L_ij.(s,d_j)
              ins ← true
          s ← next(S)
        Si (¬ins) alors
            S ← S.d
            L_ij ← L_ij.(d,d_j)
        d ← next(D)
  retourner S
```

FIG. 5 – *Procédure de projection de la contrainte C_{ij} sur le domaine du point P_i*

4 Algorithme de filtrage par décomposition

L'algorithme standard d'approximation d'un CSP continu utilise une fonction de projection qui filtre le domaine d'une variable x_i, en utilisant une contrainte c et les domaines des autres variables de $Var(c)$. Dans notre cas, il faut prendre en compte les disjonctions de domaines, ainsi que les regroupements des variables. Plus précisément, on regroupe les variables correspondant aux coordonnées d'un même point. Le sous-domaine d'un regroupement est le produit cartésien des domaines des variables qui le composent. Le domaine d'un regroupement de variables est l'union de ses sous-domaines. Les contraintes de distance mettent donc en jeu 2 regroupements de variables P_i et P_j dont les domaines sont $D_i = \{D_i^1, \ldots, D_i^{m_i}\}$ et $D_j = \{D_j^1, \ldots, D_j^{m_j}\}$.

Notre fonction de projection(Fig. 5) filtre le regroupement de variables P_i, suivant la contrainte C_{ij} en utilisant le domaine du regroupement P_j. Pour toutes les combinaisons de sous-domaines $D_i^k \times D_j^{k'}$, on décompose chaque D_i^k par CODD suivant la contrainte C_{ij} et le domaine $D_j^{k'}$. Ceci va engendrer un ensemble de sous-domaines qui vont être insérés dans une liste(S dans la figure 5).

Afin de limiter l'explosion combinatoire, les sous-domaines qui s'intersectent sont fusionnés. On obtient au final une liste de sous-domaines disjoints pour D_i, tels que chaque sous-domaines D_i^k possède un support dans l'un des sous-domaines de D_j pour la contrainte C_{ij}. Chaque nouveau sous-domaine inséré dans S est lié par une arête à un sous-domaine de D_j dans la micro-structure. L'opération de mise-à-jour de la micro-structure consiste à réinitialiser la liste d'adjacence L_{ij} du graphe bi-parti entre les domaines D_i et D_j, puis d'insérer l'arête (d,d_j), pour chaque sous-domaine d de D_i qui a un support dans d_j.

Comme l'algorithme de propagation standard, LDF dépend d'un paramètre de précision ω qui permet d'approcher l'espace solution à ω près. L'algorithme LDF(ω) est donné par la figure 6. Cet algorithme produit une micro-structure d'un CSP décomposé(Def. 4.1). Dans le cas des équations de distance, toutes les contraintes sont binaires sur l'ensemble des partitions de $\mathcal{X}$, puisqu'elles lient deux points de $\mathbb{R}^p$.

Définition 4.1 (CSP décomposé) *Un $\mathcal{CSP}$ décomposé est un quadruplet $(\mathcal{X},\mathcal{P},\mathcal{D},\mathcal{C})$ tel que :*

- *$\mathcal{X} = \{x_1, \ldots, x_N\}$ est un ensemble de N variables.*
- *$\mathcal{P} = \{P_1, \ldots, P_n\}$ une partition de $\mathcal{X}$ d'ordre n.*
- *$\mathcal{D} = \{D_1, \ldots, D_n\}$ est un ensemble d'ensembles de sous-domaines associés à $\mathcal{P}$ et tels que $D_i = \{D_i^1, \ldots, D_i^{m_i}\}$, avec D_i^k sous-domaine du regroupement P_i.*
- *$\mathcal{C} = \{c_1, \ldots, c_m\}$ est un ensemble de contraintes telles que $\forall j \leq m$, $\exists k$ tel que $Var(c_j) = P_{i_1} \cup \ldots \cup P_{i_k}$. On dit que c_j est k-aire sur $\mathcal{P}$.*

Dans la section suivante, nous décrivons formellement la micro-structure générée par LDF ainsi que quelques-unes de ses propriétés.

LDF(**in**: ω, **in**: $\mathcal{P} = (\mathcal{X},\mathcal{D},\mathcal{C})$, **out**: $\mathcal{M}$)
$Q \leftarrow \{< D_i,D_j,C_{ij} > | C_{ij} \in \mathcal{C}\}$
$\mathcal{M} \leftarrow$ { L'ensemble des listes d'adjacence $L_{ij} | C_{ij} \in \mathcal{C}\}$
Tant que $Q \neq \emptyset$ **faire**
  extraire $< D_i,D_j,C_{ij} >$ de Q
  $D'_i \leftarrow$ **PROJECTION**(D_i, D_j, C_{ij}, L_{ij})
  Si il existe un sous-domaine de D_i réduit de plus de ω **alors**
    $D_i \leftarrow D'_i$
    $Q \leftarrow Q \cup \{< D_j,D_i,C_{ij} > | C_{ij} \in \mathcal{C}\}$
retourner $\mathcal{M}$

FIG. 6 – *LDF : Algorithme de filtrage local par décomposition*

5 Micro-structure générée par LDF

5.1 Définition et propriétés

Commencons par définir la notion de micro-structure pour un un CSP décomposé :

Définition 5.1 (Micro-structure d'un CSP décomposé) *Soit $\mathcal{P}$ un CSP décomposé. On appelle micro-structure $\mathcal{M}(\mathcal{P})$, un graphe $(V(\mathcal{M}),E(\mathcal{M}))$dont :*

- *l'ensemble $V(\mathcal{M})$ de ses sommets est en bijection avec l'ensemble des sous-domaines des regroupements de $\mathcal{P}$.*
- *l'ensemble $E(\mathcal{M})$ de ses arêtes est un ensemble de couples $(D_i^k,D_j^{k'})$ tels que $i,j \leq n$, $k \leq m_i$ et $k' \leq m_j$.*

Quatre propriétés directement issues de la construction de la micro-structure par l'algorithme de projection de LDF sont identifiées :

- Étant donnés deux regroupements de variables P_i et P_j, mis en jeu par une contrainte de distance C_{ij}, tout sous-domaine D_i^k de P_i est lié par une arête à au moins un des sous-domaines de P_j.
- S'il existe une arête entre D_i^k et plusieurs sous-domaines de P_j, alors D_i^k est la plus petite boîte contenant l'ensemble des projections de ces sous-domaines de P_j par la contrainte C_{ij}.
- Tout sous-domaine de P_i contient une boîte d_i, telle qu'il existe une boîte d_j dans un des sous-domaines de P_j, avec C_{ij} 2B-consistante sur les domaines $d_i \times d_j$.
- Si l'arête $(D_i^k,D_j^{k'})$ est l'unique arête issue de D_i^k et l'unique arête issue de $D_j^{k'}$, alors C_{ij} est 2B-consistante sur les domaines $D_i^k \times D_j^{k'}$.

Une conséquence directe de ces propriétés est que, pour 2 points P_i et P_j, la contrainte C_{ij} est 2B-consistante sur les domaines $\Box\{D_i\}$ et $\Box\{D_j\}$.

Intuitivement, si chacune des bornes des D_i^k possède un support dans un des sous-domaines de P_j, alors les bornes de la boîte englobant ces sous-domaines possède également un support dans un D_j^k et donc également dans $\Box\{D_j\}$.

Considérons le CSP P', équivalent au CSP décomposé par LDF, mais dont l'ensemble des domaines est $\mathcal{D}' = \{\Box\{D_1\},\ldots,\Box\{D_n\}\}$. Alors, toutes les contraintes C_{ij} sont 2B-consistantes sur les domaines $\Box\{D_i\}$ et $\Box\{D_j\}$ et on en déduit à l'aide de la définition 2.2 que P' est 2B-consistant.

La réduction des domaines effectuée par LDF est donc au moins équivalente à celle effectuée par la 2B-consistance.

5.2 Résolution par LDF

La micro-structure génerée par LDF est comparable à celle d'un CSP fini arc-consistant. En effet, les variables correspondent aux points et les valeurs à leurs sous-domaines. La recherche des sous-CSP contenant potentiellement une solution, est donc similaire à une recherche de solution dans un CSP fini arc-consistant. Cette recherche peut donc se faire en appliquant des algorithmes du type backtrack. L'algorithme de recherche de sous-CSP que nous avons implémenté effectue un parcours DFS de la micro-structure pour trouver l'ensemble des chemins passant par un des sous-domaines de chacun des points. Les sous-CSP qui ne forment pas une clique maximale sont éliminés [5]. La complexité de cet algorithme est exponentielle, mais reste utilisable dans les cas où le nombre de sous-domaines par point est suffisemment petit.

Des travaux complémentaires nous permettront d'améliorer le calcul des sous-CSP, notemment en choisissant une structure de données plus astucieuse et un algorithme inspiré par les méthodes de recherche de solution des CSP finis. Une stratégie de bissection et filtrage par 2B sur chacun des sous-CSP isolés de cette manière peut être appliquée. C'est ce que l'on appellera par la suite, *résolution par LDF*.

Nous allons maintenant montrer l'apport de notre algorihtme LDF, sur un exemple symptomatique des difficultés que rencontrent les méthodes de consistances locales : le problème du pentagone.

6 Applications : Le problème du pentagone

6.1 La version classique

- $\mathcal{X} = \{x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4, x_5, y_5\}$.
- $\mathcal{D} = \{[0,0],[0,0],[1,1],[0,0],[-1,1],\ldots,[-1,1]\}$
- $\mathcal{C} = \begin{cases} C_0 : & (x_1 - x_0)^2 + (y_1 - y_0)^2 & = & 1 \\ C_1 : & (x_2 - x_0)^2 + (y_2 - y_0)^2 & = & 1 \\ C_2 : & (x_3 - x_0)^2 + (y_3 - y_0)^2 & = & 1 \\ C_3 : & (x_4 - x_0)^2 + (y_4 - y_0)^2 & = & 1 \\ C_4 : & (x_5 - x_0)^2 + (y_5 - y_0)^2 & = & 1 \\ C_5 : & (x_2 - x_1)^2 + (y_2 - y_1)^2 & = & d^2 \\ C_6 : & (x_2 - x_3)^2 + (y_2 - y_3)^2 & = & d^2 \\ C_7 : & (x_3 - x_4)^2 + (y_3 - y_4)^2 & = & d^2 \\ C_8 : & (x_4 - x_5)^2 + (y_4 - y_5)^2 & = & d^2 \\ C_9 : & (x_5 - x_1)^2 + (y_5 - y_1)^2 & = & d^2 \end{cases}$

FIG. 7 – *Description classique du CSP pentagone*

5. C'est-à-dire dont tous les domaines ne sont pas inter-connectés

Le problème du pentagone est un CSP(Fig. 7) 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, consitué de 5 points équidistants sur le cercle unitaire. L'un de ces 5 points est fixé au point de coordonnées (1,0) pour éviter que le nombre de solutions ne soit infini.

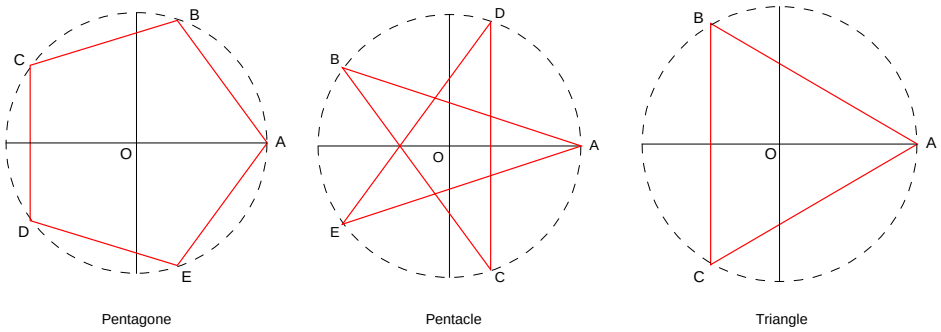

FIG. 8 – *Solutions du problème du pentagone*

Selon le choix de la distance d entre deux points consécutifs, on obtient 3 instances donnant lieu à 3 types de solutions(Fig. 8) :

- *Pentagone* : Si $d = 2\sin(\frac{\pi}{5})$ il y a 2 solutions pour la combinaison $P_1P_2P_3P_4P_5$ formant un pentagone, $ABCDE$, $AEDCB$.
- *Pentacle* : 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, $ABCDE$, $AEDCB$.
- *Triangle* : 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 $ACABC$.

Csp	LDF(10^{-9})	Résolution par LDF(10^{-9})	2B(10^{-9})+bissection
Pentagone	0.05s	0.07s	0.03s
Pentacle	0.05s	0.06s	0.05 s
Triangle	0.07s	0.08s	0.04s

FIG. 9 – *Résultats expérimentaux pour le problème du pentagone*

Chacune de ces 3 instances est résolue efficacement par ILOG Solver[11] avec une combinaison 2B-bissection. Les temps de calcul[6] sont du même ordre que ceux que l'on obtient en utilisant LDF, indiqués sur le tableau 9.

La figure 10, montre une partie de la micro-structure produite par LDF sur l'instance *"Pentagone"*, dans laquelle ne figure que la solution $ABCDE$(Fig. 8). Dans cet exemple, tous les sous-domaines sont réduits à des solutions ponctuelles par LDF, ce qui réduit la résolution à une simple recherche de sous-CSP, le filtrage n'étant plus nécessaire puisque déjà réalisé. Pour des questions de présentation, nous avons reportés les résultats dans la figure 8 avec une précision de 10^{-2}. Les résultats ont toujours été obtenus avec une précision de 10^{-9}.

6. Réalisés sur un bi-Pentium IV 2GHz avec 512Mo de RAM

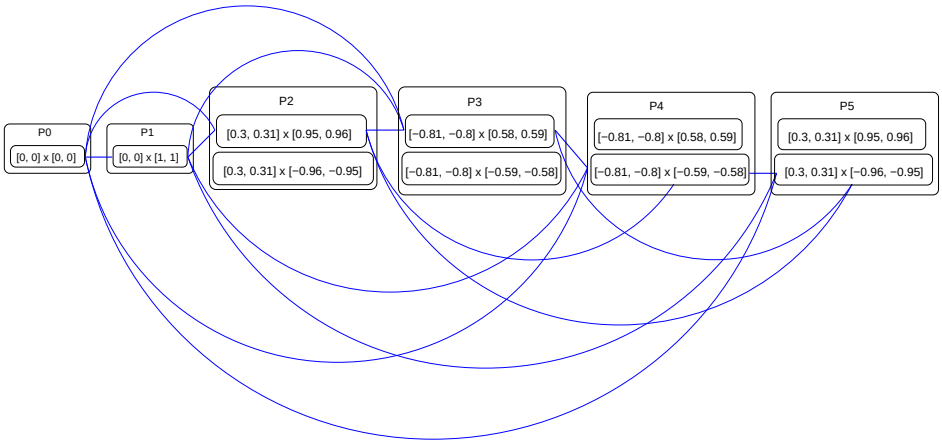

FIG. 10 – *Micro-structure générée par LDF pour l'instance "pentagone"*

6.2 Extension de la version classique

Cinq points supplémentaires sont ajoutés au problème initial, un entre chaque arête à distance de 1 de chaque extrémité(Fig. 11). Pour une solution du problème classique, nous obtenons ainsi 2^5 fois plus de solutions, soit 64 pour *pentagone* et *pentacle*, et 320 pour *triangle*. La résolution de ce problème par la combinaison 2B+bissection est moins efficace dans la plupart des cas et produit une multitude de solutions dupliquées. En utilisant LDF, toutes les solutions de ces 3 instances sont isolées en quelques secondes(Fig. 12); c'est-à-dire que la décomposition conduit sur cet exemple à générer tous les intervalles avec la précision requise. Ceci n'est toutefois pas une propriété de LDF.

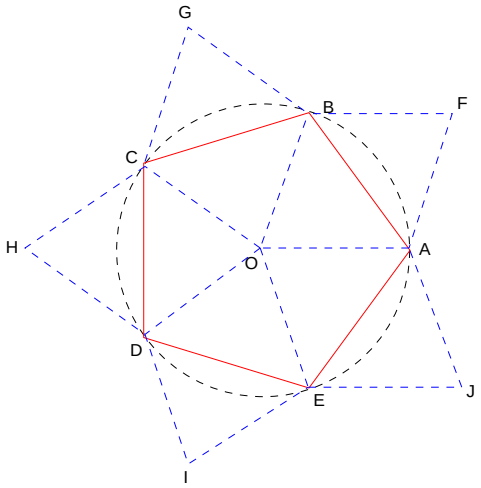

FIG. 11 – *Ajout de contraintes dans le problème du pentagone*

Csp	LDF(10^{-9})	Résolution par LDF(10^{-9})	2B(10^{-9})+bissection
Pentagone	1.40s	2.14s	2.04s
Pentacle	1.53s	2.27s	162.85s
Triangle	0.7s	6.44s	12.69s

FIG. 12 – *Résultats expérimentaux pour le problème du pentagone étendu*

7 Conclusion

Nous avons introduit dans cet article une nouvelle méthode de filtrage 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 cliniques 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

[1] Frédéric Benhamou, David McAllester, and Pascal Van Hentenryck. CLP(intervals) revisited. In Maurice Bruynooghe, editor, *Proceedings of the 1994 International Symposium*, pages 124–138. MIT Press, 1994.

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

[3] H. Collavizza, F. Delobel, and M. Rueher. A note on partial consistencies over continuous domains. *Lecture Notes in Computer Science*, 1520:147–??, 1998.

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

[5] Y. Lebbah, M. Rueher, and C. Michel. A global filtering algorithm for handling systems of quadratic equations and inequations. In *CP'2002, Eighth International Conference on Principles and Practice of Constraint Programming*, Cornell University, Ithaca, NY, USA, Sept. 7-13 2002.

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

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

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

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

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

[11] ILOG Solver. *Reference manual.* 2002.

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