QuadDist: Filtrage global pour les contraintes de distance euclidienne

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

Cet article présente un algorithme de filtrage global pour résoudre les systèmes d'équations de distance. Cette nouvelle technique nommée *QuadDist* est dérivée de *Quad*, une technique de filtrage global pour les contraintes quadratiques [LRM02]. *Quad* calcule une relaxation linéaire des termes quadratiques des équations et utilise l'algorithme du simplexe pour réduire les domaines des variables. Nous présentons dans ce papier une approximation linéaire dédiée aux équations de distance euclidienne. Le point clé de cette nouvelle méthode réside dans le fait que ces approximations ne sont pas générées pour chaque terme des équations mais globalement pour chaque contrainte de distance. *QuadDist* définit donc une approximation plus précise que *Quad*, ce qui augmente la vitesse de convergence la méthode. D'autre part, contrairement à la *Quad*, notre linéarisation des contraintes ne nécessite pas l'ajout de variables supplémentaires ce qui réduit la taille des systèmes linéaires résolus par le simplexe. Les résultats expérimentaux de *QuadDist* pour la résolution de CSP constitués d'équations de distance sont prometteurs.

1 Introduction

Cet article introduit un nouvel algorithme de filtrage global pour résoudre les systèmes d'équations et d'inéquations de distance. Ces contraintes de distance sont définies par :

$$\sum_{k=1}^{k=p} (x_{ik} - x_{jk})^2 = \delta_{ij}^2$$

où x_{ik} est la k-ème coordonnée du point P_i dans un espace euclidien de dimension p et δ_{ij} est un nombre réel positif. De manière générale, la distance peut être donnée par un intervalle : $\delta_{ij} \in [m,M]$, où m (resp. M) est la distance euclidienne minimale (resp. maximale) entre les points P_i et P_j. Le problème de satisfaction de contraintes de distance est un problème NP-complet qui apparaît dans de nombreux domaines d'application comme en biochimie moléculaire, en conception assistée par ordinateur ou encore en robotique.

Les outils classiques pour résoudre les contraintes numériques sont basées sur des consistances locales comme la 2B-consistance [Lho93] ou la Box-consistance [BMV94; VHMK97]. L'inconvénient de ces méthodes vient du fait que les contraintes sont traitées indépendamment et sans exploiter leurs propriétés sémantiques.

Lebbah *et al* ont introduit un algorithme de filtrage global, nommé *Quad*, pour traiter les contraintes quadratiques [LRM02]. *Quad* génère une relaxation linéaire des termes quadratiques des équations puis d'utiliser l'algorithme du simplexe pour réduire les domaines des variables.

Nous présentons dans ce papier une approximation linéaire dédiée aux équations de distance euclidienne. Le point clé de cette nouvelle méthode réside dans le fait que ces approximations ne sont pas générées pour chaque terme des équations mais globalement pour chaque contrainte de distance. *QuadDist* définit donc une approximation plus précise que *Quad*. Contrairement à la *Quad* notre linéarisation des contraintes ne nécessite pas l'ajout de variables additionnelles, ce qui réduit la taille des systèmes linéaires engendrés. Les performances de *QuadDist* sur des CSP constitués d'équations de distance sont tout à fait encourageantes.

Heusch [Heu03] a introduit une contrainte globale pour traiter les relations de distance entre n points mais sans fournir de résultats expérimentaux. Son approche est basée sur la méthode "Active Areas" utilisées par Markót et Csendes [Mar00]. L'avantage de *QuadDist* est qu'il est s'agit d'un schéma plus général et plus facile à mettre en œuvre.

La section suivante présente les principes de *QuadDist* sur un exemple très simple.

1.1 Présentation informelle

Considérons le système de contraintes suivant :

$$\mathcal{C} = \begin{cases} c_1 : & x^2 + y^2 = & 10 \\ c_2 : & (x-4)^2 + y^2 = & 10 \end{cases}$$

et supposons que les domaines des variables x et y soient $D_x = [1,3]$ et $D_y = [1,4]$. La solution de $\mathcal{C}$ (voir figure 1) est le point d'intersection de deux cercles de rayon $\sqrt{10}$: $\mathcal{C}_1$ centré en $(0,0)$ et $\mathcal{C}_2$ centré en $(4,0)$.

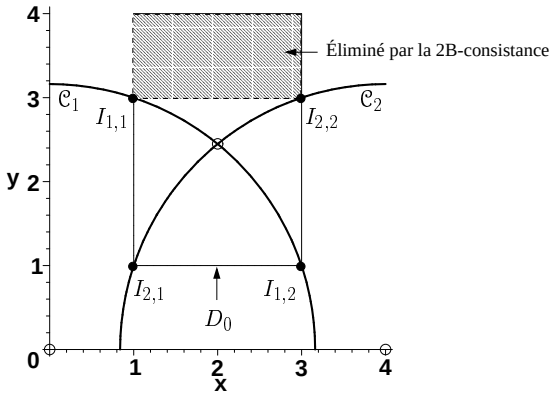

FIG. 1 – *Réduction du domaine par 2B-consistance*

La 2B-consistance réduit le domaine initial $D_0 = D_x \times D_y$ à [1,3] × [1,3] puisque les bornes des domaines des variables satisfont les contraintes c_1 et c_2. La figure 1 montre en effet que les points $I_{1,1}$ et $I_{1,2}$ satisfont $c1$ et les points $I_{2,1}$ et $I_{2,2}$ satisfont $c2$. Pour réduire davantage les domaines, il faut utiliser une consistance plus forte ou une méthode d'énumération comme la bissection.

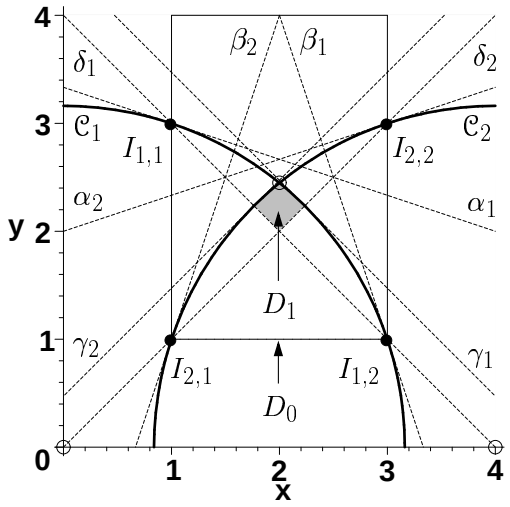

FIG. 2 – *Première approximation des solutions de* $\mathcal{L}(\mathcal{C})$.

Le processus de relaxation de *QuadDist* (détaillé dans la section 3) calcule l'ensemble $\mathcal{L}(\mathcal{C})$ des approximations linéaires des contraintes de c_1 et c_2:

$$\mathcal{L}(\mathcal{C}) = \underline{\mathcal{L}}(c_1) \cup \underline{\mathcal{L}}(c_2) \cup \overline{\mathcal{L}}(c_1) \cup \overline{\mathcal{L}}(c_2)$$

où $\underline{\mathcal{L}}(c_i)$ est l'ensemble des sous-estimateurs de la contrainte c_i et $\overline{\mathcal{L}}(c_i)$ est l'ensemble des surestimateurs de c_i. Plus précisément :

- $\overline{\mathcal{L}}(c_i)$ est défini par trois tangentes au cercle $\mathcal{C}_i$:
  - α_i et β_i, les tangentes à $\mathcal{C}_i$ aux points $I_{i,1}$ et $I_{i,2}$, les points d'intersection entre $\mathcal{C}_i$ et la boîte définie par les domaines initiaux de x et y.
  - γ_i, la tangente au milieu de l'arc de cercle $\widehat{I_{i,1}I_{i,2}}$.
- $\underline{\mathcal{L}}(c_i)$ est la corde δ_i entre les points $I_{i,1}$ et $I_{i,2}$.

Plus formellement, le système linéaire $\mathcal{L}(\mathcal{C})$ généré par *QuadDist* est le suivant:

$$\mathcal{L}(\mathcal{C}) = \begin{cases} \alpha_1 & : & 3y + x - 10 \leq 0 \\ \beta_1 & : & y + 3x - 10 \leq 0 \\ \gamma_1 & : & 2y + 2x - \sqrt{80} \leq 0 \\ \delta_1 & : & y + 2x - 8 \geq 0 \\ & & \\ \alpha_2 & : & 3y - (x-4) - 10 \leq 0 \\ \beta_2 & : & y - 3(x-4) - 10 \leq 0 \\ \delta_2 & : & 2y - 2(x-4) - 8 \geq 0 \\ \gamma_2 & : & 2y - 2(x-4) - \sqrt{80} \leq 0 \end{cases}$$

Le simplexe est utilisé pour calculer les valeurs minimales et maximales de x et y satisfaisant les contraintes linéaires du système $\mathcal{L}(\mathcal{C})$. Après cette étape de filtrage, le domaine initial D_0 est réduit à la boîte englobant D_1 (voir figure2), puis le processus de relaxation recommence. Quatre itérations supplémentaires permettent d'isoler la solution du système avec une précision de 6 chiffres décimaux.

1.2 Plan de l'article

Tout d'abord, nous introduirons les notations et rappellerons les grandes lignes de l'algorithme *Quad*. La section 3 détaille le principe général de notre approche ainsi que le processus de linéarisation en dimension 2. Dans la section 4 la taille des programmes linéaires générés par *QuadDist* est comparé à ceux que génère la *Quad*. Enfin, la section 4 présente quelques résultats expérimentaux sur des problèmes académiques.

2 Définitions et notations

2.1 Notations

Nous nous intéressons aux $\mathcal{CSP}$s qui contiennent exclusivement des contraintes de distance. Les domaines des variables de tels $\mathcal{CSP}$s sont des intervalles. L'intervalle $[\underline{x},\overline{x}]$ représente l'ensemble des nombres réels x tels que $\underline{x} \leq x \leq \overline{x}$.

Un $\mathcal{CSP}$s constitué d'équations de distance est défini par :

- Un ensemble de n points $\mathcal{P} = \{P_i\}_{1 \leq i \leq n}$ dans un espace euclidien de dimension p. On dénote par x_{ik}, la k-ème coordonnée du point P_i.
- Un ensemble de domaines $\mathcal{D} = \{D_i\}_{1 \leq i \leq n}$, où le vecteur d'intervalles $D_i = D_{i1} \times \ldots \times D_{ip}$ est le domaine du point P_i, avec $D_{ik} = [\underline{x_{ik}},\overline{x_{ik}}]$ le domaine associé à x_{ik}.
- Un ensemble de contraintes de distance :

$$\mathcal{C} = \{c_{ij} : \sum_{k=1}^{k=p} (x_{ik} - x_{jk})^2 = \delta_{ij}^2\},$$

où $1 \leq i,j \leq n$. La distance δ_{ij} est un nombre réel strictement positif ou un intervalle $[\underline{\delta_{ij}},\overline{\delta_{ij}}]$.

Pour simplifier la notation en dimension 2, les coordonnées du point P_i seront notées par x_i et y_i et le domaine du point P_i par $D_i = D_{x_i} \times D_{y_i}$, où $D_{x_i} = [\underline{x_i},\overline{x_i}]$, $D_{y_i} = [\underline{y_i},\overline{y_i}]$. Dans ce cas, les contraintes du système sont de la forme :

$$\mathcal{C} = \{c_{ij} : (x_i - x_j)^2 + (y_i - y_j)^2 = \delta_{ij}^2\}$$

2.2 Quad :un filtrage global pour les contraintes quadratiques

Quad est un algorithme de filtrage global pour les systèmes de contraintes quadratiques (*c.f* [LRM02; MLR03] pour une description plus détaillée). Cet algorithme repose sur une approximation des contraintes quadratiques du système initial par un ensemble de contraintes linéaires. Le simplexe est utilisé pour calculer une borne inférieure et supérieure des variables.

La relaxation des contraintes quadratiques consiste d'abord à introduire une nouvelle variable pour chaque terme non linéaire des contraintes quadratiques. Les domaines des variables additionnelles sont calculés en utilisant l'arithmétique de base sur les intervalles. Les termes quadratiques sont ensuite approximés par deux classes de relaxations linéaires. Ces

relaxations génèrent des surestimateurs et des sous-estimateurs linéaires qui définissent une enveloppe convexe des termes quadratiques. Ces approximations apportent une meilleure estimation de l'ensemble des solutions que les fonctions de projection basées sur l'arithmétique des intervalles.

Nous montrons dans ce papier que ces relaxations peuvent être améliorées de manière significative en construisant une approximation linéaire globalement pour chaque contrainte de distance plutôt qu'en combinant les approximations de ses termes quadratiques.

Dans la section suivante, nous montrons comment le processus de relaxation de *Quad-Dist* exploite la sémantique des contraintes de distance pour calculer une approximation plus précise que la *Quad* sans introduire de variables additionnelles.

3 *QuadDist*: Une approche globale

QuadDist est une contrainte globale qui a été conçue pour résoudre efficacement les problèmes mettant en jeu des contraintes de distance dans le plan et dans l'espace tridimensionnel. Pour simplifier la description de notre méthode, nous détaillerons notre approche en dimension 2 où les illustrations sont plus intuitives.

3.1 Schéma général

QuadDist calcule une réduction globale des domaines en itérant les deux étapes suivantes jusqu'à atteindre un point fixe :

1. **Relaxation des contraintes** : Génération de $\mathcal{L}(\mathcal{C})$ l'ensemble des approximations linéaires des contrainte $c_{ij} \in \mathcal{C}$ de la forme :

   $$c_{ij} : (x_j - x_i)^2 + (y_j - y_i)^2 = \delta_{ij}^2$$

   où x_i et y_i (respectivement x_j et y_j) sont les coordonnées du point P_i (respectivement du point P_j). Ces approximations linéaires sont de la forme :

   $$a(x_j - x_i) + b(y_j - y_i) + c \leq 0$$

   où a, b et c sont des nombres réels.
2. **Filtrage des domaines** : Utilisation du simplexe pour minimiser et maximiser chaque x_i et y_i soumis aux contraintes linéaires du système $\mathcal{L}(\mathcal{C})$.

Ce processus itératif s'arrête lorsque toutes les réductions de domaines sont plus petites qu'un certain réel positif ϵ.

Pour calculer les relaxations linéaires, les contrainte c_{ij} sont mises sous forme canonique en effectuant la substitution suivante :

$$x = x_j - x_i$$
$$y = y_j - y_i$$

où x et y sont des variables temporaires dont les domaines D_x et D_y sont calculés en utilisant l'arithmétique de base des intervalles. La contrainte c_{ij} s'écrit donc :

$$c_{ij} : x^2 + y^2 = \delta_{ij}^2$$

Les domaines de x et de y et la distance δ_{ij} sont utilisés pour déterminer les coefficients a, b et c des relaxations linéaires de c_{ij} ; ces relaxations sont de la forme :

$$ax + by + c \leq 0$$

où a, b et c sont des nombres réels. Enfin, on génère les relaxations de la contrainte c_{ij} qui sont effectivement ajoutées au système de contraintes; à savoir :

$$a(x_j - x_i) + b(y_j - y_i) + c \leq 0$$

Le problème se ramène donc à calculer l'ensemble $\mathcal{L}(c)$ des approximations linéaires d'une contrainte sous forme canonique $c : x^2 + y^2 = \delta^2$. Ce problème est loin d'être trivial car l'ensemble des solutions de c n'est en général ni convexe ni monotone. La section suivante montre comment décomposer les domaines de x et y de manière à ce que ces propriétés soient vérifiées, tandis que la section 3.3 montre comment calculer les relaxations de c.

3.2 Décomposition sémantique des domaines

Considérons la contrainte $c : x^2 + y^2 = \delta^2$ avec $(x,y) \in D = D_x \times D_y$ et $\delta > 0$. D est décomposée en au plus quatre sous-domaines sur lesquels la contrainte c est monotone et convexe :

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

où $\square E$ dénote la plus petite boîte contenant l'ensemble des valeurs de l'ensemble E. Notons que chacun de ces sous-domaines peut être réduit à l'ensemble vide.

Exemple 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 3 montre que la décomposition sémantique découpe le domaine du point (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 $[4.5825,5] \times [-2,0]$.

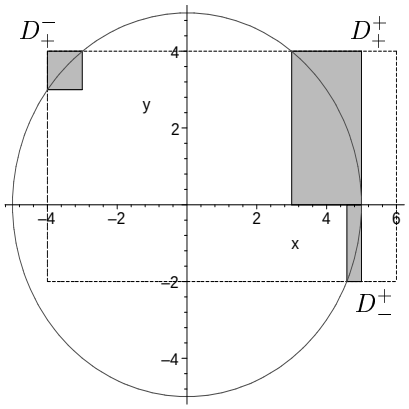

FIG. 3 – *Illustration de la décomposition sémantique sur l'exemple 1.*

La décomposition sémantique de D satisfait la propriété suivante :

Propriété 1 *Soit $f(x,y) = x^2 + y^2 - \delta^2$ et la contrainte définie par $f(x,y) = 0$ et Soit $s \in \{D_+^+, D_-^+, D_+^-, D_-^-\}$, tel que $s \neq \emptyset$. Alors la fonction f est monotone sur s.*

Preuve La monotonie est triviale puisque le vecteur gradient de f est $(2x,2y)$ et que ses composantes sont de signe constant sur D_+^+, D_-^+, D_+^- et D_-^-.

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 boîte qui contient toutes les solutions de $c(x,y)$ sur le dit domaine.

Pour calculer une approximation linéaire des relations de distance, nous générons un ensemble d'inéquations qui encadrent chacun des espace solutions des sous-domaines monotones. Ces ensembles d'inéquations sont ensuite combinés pour obtenir une approximation du domaine complet. Nous présentons dans la suite les relaxations linéaires de D_+^+, les autres se dérivant aisément par symétrie.

La section 3.3 montre comment obtenir une relaxation de D_+^+ tandis que la section 3.4 montre comment combiner les approximations des sous-domaines monotones pour obtenir une relaxation du domaine D.

3.3 Relaxation des contraintes de distance

Pour définir une approximation linéaire de l'espace des solutions, on distingue les relations de distance suivantes :

Distance exacte	c :	$x^2 + y^2 = \delta^2$
Distance minimale	$\underline{c}$:	$x^2 + y^2 \geq \delta^2$
Distance maximale	$\overline{c}$:	$x^2 + y^2 \leq \delta^2$

La contrainte c étant la conjonction des contraintes $\underline{c}$ et $\overline{c}$, le problème se ramène à trouver les relaxations de $\underline{c}$ et $\overline{c}$. Les relaxations de c seront définies par l'union de ces estimations.

Les relaxations de $\underline{c}$ et $\overline{c}$ sont calculées en considérant les points d'intersection entre le cercle défini par c et la boîte D_+^+. La figure 4 montre que ces points, que l'on définit comme étant les *points d'intersection* de D_+^+, sont $I_1(\overline{x},\underline{y})$ et $I_2(\underline{x},\overline{y})$.

De manière triviale, les tangentes en un point de l'arc de cercle $\widehat{I_1 I_2}$ définissent une surestimation des solutions de $\overline{c}$. De même, la corde $[I_1 I_2]$ définit une sous-estimation des solutions de $\underline{c}$.

Plus formellement, les propositions 1 et 2 définissent l'ensemble de sous-estimations de $\underline{c}$ et l'ensemble des surestimations de $\overline{c}$ sur le sous-domaine D_+^+.

L'ensemble $\underline{\mathcal{L}}(\underline{c},D_+^+)$ des sous-estimations de $\underline{c}$ correspondent au demi-plan délimité par le segment $[I_1 I_2]$ et qui ne contient pas le point $(0,0)$.

Proposition 1 (Sous-estimations) *Considérons la contrainte $\underline{c}$: $x^2+y^2 \geq \delta^2$, avec $(x,y) \in D_+^+ = [\underline{x},\overline{x}] \times [\underline{y},\overline{y}]$.*

Soit $\underline{\mathcal{L}}(\underline{c},D_+^+)$ l'ensemble défini par la relation suivante :

$$\beta : \quad (\overline{x} - \underline{x})y + (\underline{y} - \overline{y})x + \underline{x}\underline{y} - \overline{x}\overline{y} \quad \geq 0$$

Alors, $\underline{\mathcal{L}}(\underline{c},D_+^+)$ est une sous-estimation linéaire de $\underline{c}$.

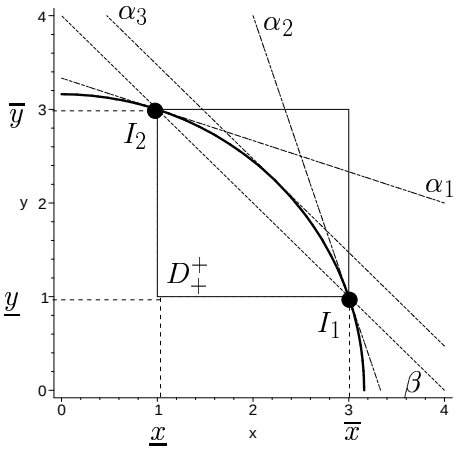

FIG. 4 – *Relaxation de c sur* D_+^+.

L'ensemble $\overline{\mathcal{L}}(\overline{c},D_+^+)$ des surestimations de $\overline{c}$ correspond à l'intersections des demi-plans contenant le point $(0,0)$ et délimités par trois tangentes en I_1, en I_2 et au milieu de l'arc de cercle $\widehat{I_1 I_2}$.

Proposition 2 (Surestimations) *Considérons la contrainte* $\overline{c} : x^2 + y^2 \leq \delta^2$, *avec* $(x,y) \in D_+^+ = [\underline{x},\overline{x}] \times [\underline{y},\overline{y}]$.

Soit $\overline{\mathcal{L}}(\overline{c},D_+^+)$, *l'ensemble défini par les relations suivantes :*

$$\left\{ \begin{array}{lrl} \alpha_1 : & \overline{y}y + \underline{x}x - \delta^2 & \leq 0 \\ \alpha_2 : & \underline{y}y + \overline{x}x - \delta^2 & \leq 0 \\ \alpha_3 : & (\overline{x} - \underline{x})y + (\underline{y} - \overline{y})x - \gamma & \leq 0 \end{array} \right.$$

où $\gamma = \delta\sqrt{(\overline{x} - \underline{x})^2 + (\overline{y} - \underline{y})^2}$.

Alors, $\overline{\mathcal{L}}(\overline{c},D_+^+)$ *est une surestimation linéaire de* $\overline{c}$.

Les surestimations et les sous-estimations de D_+^-, D_-^+ et D_-^- sont dérivées de D_+^+ par symétrie. Les relaxations complètes de c, $\underline{c}$ et $\overline{c}$ sur les sous-domaine S sont alors définies par :

$$\left\{ \begin{array}{rl} \overline{\mathcal{L}}(c,S) = & \overline{\mathcal{L}}(\overline{c},S) \\ \underline{\mathcal{L}}(c,S) & \underline{\mathcal{L}}(\underline{c},S) \\ \mathcal{L}(c,S) = & \underline{\mathcal{L}}(c,S) \cup \overline{\mathcal{L}}(c,S) \end{array} \right.$$

La section suivante montre comment combiner ces approximations linéaires pour obtenir $\mathcal{L}(c,D)$, une approximation de c sur le domaine initial D.

3.4 Combinaison des approximations

Les tangentes au cercle défini par c surestiment les sous-ensembles de solutions contenus par chacun les sous-domaines. Par conséquent, l'ensemble des surestimations de D est défini par l'union des surestimations de ses sous-domaines :

$$\overline{\mathcal{L}}(c,D) = \overline{\mathcal{L}}(c,D_+^+) \cup \overline{\mathcal{L}}(c,D_-^+) \cup \overline{\mathcal{L}}(c,D_+^-) \cup \overline{\mathcal{L}}(c,D_-^-)$$

La combinaison des sous-estimations des sous-domaines est plus compliquée; l'union de ces approximations n'étant pas convexe en général.

Soit k le nombre de sous-domaines monotones non vides de D. Il est évident que pour $k = 4$ il n'existe aucune sous-estimation convexe. Le problème se ramène donc à combiner les sous-estimations de c sur k sous-domaines monotones avec $k = 2$ ou $k = 3$. Il est facile de montrer que pour $k = 2$, les 2 sous-domaines sont contenus dans deux quadrants contigus.

Dans tous les cas, $\underline{\mathcal{L}}(c,D)$ doit être une sous-estimation de $\underline{\mathcal{L}}(c,S)$ pour tout sous-domaine S de D. Considérons tous les demi-plans définis par les segments entre chaque couples de *points d'intersection*. La sous-estimation globale est délimitée par le segment qui sous-estime l'ensemble de ces demi-plans, dans le domaine considéré.

Ce segment (ou ce plan en dimension 3) peut être déterminé en calculant l'enveloppe convexe de l'ensemble des *points d'intersection*. Dans le plan, on peut toutefois déterminer plus simplement ce segment en triant l'ensemble des points d'intersections par angle croissant par rapport à un point choisi dans un quadrant ne contenant pas de solutions. $\underline{\mathcal{L}}(c,D)$ est alors défini par le premier et le dernier point de cet ensemble trié.

La figure 5 montre un exemple d'une telle combinaison pour $k = 3$.

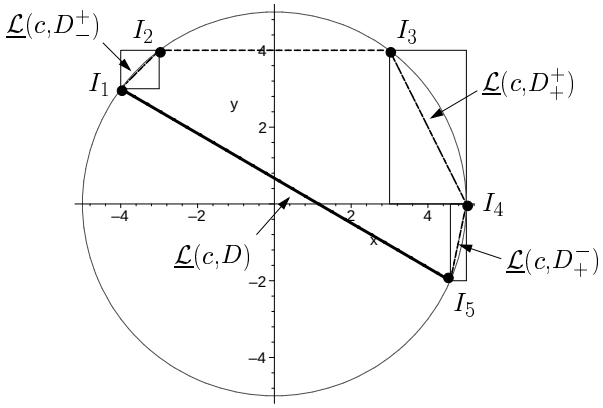

FIG. 5 – *Sous-estimations de trois sous-domaines.*

Dans la section suivante, nous comparons les systèmes linéaires engendrés par la *Quad* avec ceux générés par *QuadDist*.

4 Taille des programmes linéaires générés

Dans le cas général, c'est-à-dire pour des contraintes de distance dans un espace de dimension p, le nombre de contraintes générées par *QuadDist* est exponentiel. En effet, le nombre de contraintes linéaires générées est :

$$(p+1)2^p m$$

où $(p+1)$ est le nombre de contraintes linéaires engendrées sur un sous-domaine monotone, 2^p est le nombre maximal de sous-domaines engendrés par la décomposition sémantique

et m est le nombre de contraintes de distance du système initial. Toutefois, rappelons que *QuadDist* a été conçu pour résoudre des systèmes de contraintes de distance dans le plan et dans l'espace. Dans ce cas, le nombre de contraintes générées par *QuadDist* est tout à fait raisonnable.

Nous comparons dans ce chapitre la taille des programmes linéaires engendrés par *QuadDist* à ceux générés par *Quad* en dimension 2 et 3.

Considérons un CSP constitué de m contraintes de distance mettant en jeu n points.

Nombre de variables *Quad* utilise la forme développée des équations de distance en dimension 2 :

$$x_i^2 + x_j^2 + y_i^2 + y_j^2 + 2x_ix_j + 2y_iy_j = \delta_{ij}^2$$

et ajoute une variable supplémentaire pour chaque terme carré et pour chaque terme produit, soit $2n + 2m$ variables ajoutées aux $2n$ variables initiales et donc un total de $(4n + 2m)$ variables.

QuadDist n'introduit pas de variables additionnelles, donc le nombre de variables des systèmes linéaires engendrés est égal à $2n$.

Le nombre de variables des programmes linéaires engendrés par *QuadDist* est donc toujours plus petit.

Nombre de contraintes linéaires Au pire des cas, *Quad* engendre 3 inéquations linéaires pour chaque terme carré et 4 inéquations pour chaque terme produit, soit au total $C_Q = 4 \times 2m + 3 \times 2n = 8m + 6n$.

QuadDist calcule au pire des cas 3 surestimations sur chacun des 4 sous-domaines monotones et aucune sous-estimation. Le nombre de contraintes linéaires engendrées est donc 12 par contraintes, soit $C_{QD} = 12m$ contraintes linéaires.

Si le graphe de contraintes ne contient pas de noeuds isolés, c'est-à-dire si $n > m$, on a $C_Q - C_{QD} = 6n - 4m > 2m$, alors le nombre de contraintes engendrées par *QuadDist* est inférieur au nombre de contraintes générées par *Quad*. En dimension 3, on peut montrer de manière analogue que $C_Q - C_{QD} > 0$ si et seulement si $m < 9/20n$. Autrement dit, *QuadDist* génère toujours moins de contraintes linéaires que *Quad* pour des graphes de contraintes ayant une densité moyenne inférieure à $9/20$.

5 Résultats expérimentaux

Nous avons comparé les temps de résolution de *QuadDist* avec ceux du solveur *Realpaver* [1] et ceux de la *Quad* sur le problème du pentagone(penta1, penta2 et penta3) ainsi que sur une extension de ce problème (*c.f.* section 5.1).

Le tableau 6 récapitule les résultats obtenus sur un PC muni d'un processeur Pentium IV à 2Ghz avec 256Mo de RAM. Une case vide signifie que le solveur n'a pas résolu complètement le problème en moins d'une demi-heure. Pour chacun des solveur on affiche les temps de calcul en secondes et le nombre de boîtes calculées.

On peut remarquer que pour l'ensemble de ces problèmes, *QuadDist* améliore significativement les performances de la *Quad*. *Realpaver* qui combine la Box-consistance et la Hull-consistance, deux filtrages peu coûteux, est plus efficace que *QuadDist* sur les instances les plus simples. Ceci s'explique aisément du fait du coût initial de l'appel au simplexe.

On peut aussi remarquer que pour un problème comme ext-penta1, *Realpaver* génère de nombreuses boîtes parasites alors que *QuadDist* isole toutes les solutions pour ces différents problèmes.

1. Realpaver[Gra03] est un solveur qui combine avantageusement des versions optimisées des consistances locales comme la 2B-consistance et la Box-consistance.

Problème(# solutions)	Realpaver		Quad		QuadDist	
	t(s)	#box	t(s)	#box	t(s)	#box
penta1(2)	0.02s	2	2s	2	0.23s	2
penta2(2)	0.02s	2	0.51s	2	0.41s	2
penta3(10)	0.03s	10	3.71s	10	0.52s	10
ext-penta1(64)	7.66s	1066	-	-	3.10s	64
ext-penta2(64)	-	-	-	-	8.15s	64
ext-penta3(320)	-	-	-	-	15.25s	320

FIG. 6 – *Résultats expérimentaux*

La section suivante décrit plus en détail les problèmes sur lesquels nous avons réalisé nos expérimentations.

5.1 Problème classique 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

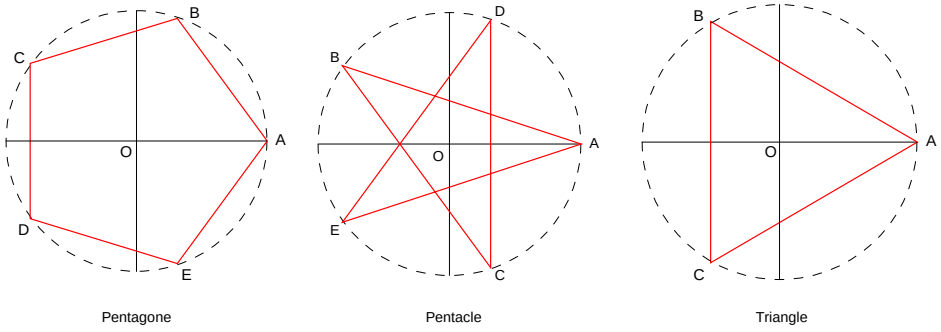

FIG. 7 – *Solutions du problème du pentagone*

obtient 3 instances que nous nommerons *penta1*, *penta2* et *penta3* donnant lieu à 3 types de solutions (Fig. 7) :

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 $ACABC$.

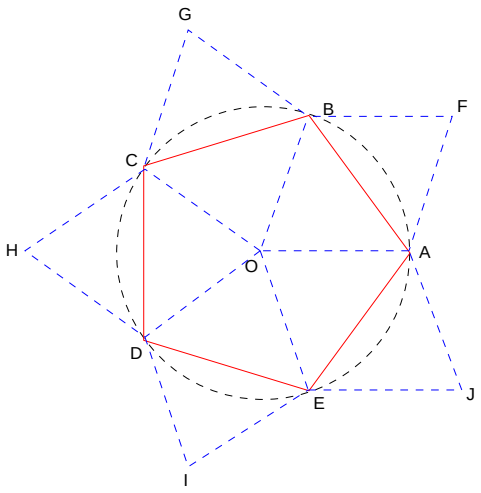

FIG. 8 – *Ajout de contraintes dans le problème du pentagone*

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. 8). 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*)

6 Conclusion

Cet article présente un algorithme de filtrage global pour résoudre les systèmes d'équations de distance. Cet algorithme réalise un filtrage global basé sur des approximations linéaires plus précises que celles qui sont générées par la *Quad*. Ces relaxations dédiées n'introduisent pas de variables additionnelles et permettent à la *Quad* de gagner en efficacité sur la résolution d'équations de distance.

Contrairement à la *Quad*, *QuadDist* ne garantit pas la correction des approximations linéaires calculées. L'adaptation des principes proposés dans [MLR03] ne devrait toutefois pas poser de difficultés majeures.

Références

[BMV94] F. Benhamou, D. McAllister, and P. Van Hentenryck. CLP(intervals) revisited. In Maurice Bruynooghe, editor, *Proceedings of the 1994 International Symposium*, pages 124–138. MIT Press, 1994.

[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.

[BR03] H. Batnini and M. Rueher. Semantic decomposition for solving distance constraint. In F. Rossi, editor, *Proc. of CP'03: 9th International Conference on "Principles and Practice of Constraint Programming")*, LNCS 2833, pages 964–964. Springer Verlag, September 2003.

[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.

[Gra03] L. Granvilliers. *Realpaver v0.3 user's manual. Solving non-linear constraint by interval computations*. IRIN, http://www.sciences.univ-nantes.fr/info/perso/permanents/granvil/realpaver/, July 2003.

[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.

[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.

[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).

[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.

[VHMK97] P. Van Hentenryck, D. McAllister, and D. Kapur. Solving polynomial systems using a branch and prune approach. *SIAM, Journal of Numerical Analysis*, 34(2):797–827, April 1997.