# Contrainte de non-chevauchement entre objets décrits par des inégalités non-linéaires

Ignacio Salas, Gilles Chabert, Alexandre Goldsztejn

▶ **To cite this version:**

Ignacio Salas, Gilles Chabert, Alexandre Goldsztejn. Contrainte de non-chevauchement entre objets décrits par des inégalités non-linéaires. Journées Francophones de Programmation par Contraintes (JFPC), Jun 2014, Angers, France. hal-01079579

**HAL Id: hal-01079579**
**https://hal.science/hal-01079579**

Submitted on 3 Nov 2014

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Contrainte de non-chevauchement entre objets décrits par des inégalités non-linéaires

**Ignacio Salas**[1] **Gilles Chabert**[1] **Alexandre Goldsztejn**[2]

[1] Mines de Nantes - LINA (UMR 6241)
[2] CNRS - LINA (UMR 6241)
{ignacio.salas,gilles.chabert}@mines-nantes.com alexandre.goldsztejn@univ-nantes.fr

**Résumé**

Le placement d'objets 2D dans un espace limité est un problème omniprésent aussi bien sur le plan académique qu'industriel. Quel que soit le contexte, la résolution de ce problème exige la capacité de pouvoir déterminer où un premier objet peut être placé de telle façon qu'il ne chevauche pas un second objet, précédemment placé. Ce sous-problème s'appelle la contrainte de non-chevauchement. La complexité de cette contrainte de non-chevauchement dépend du type d'objets considérés. Elle est simple dans le cas de rectangles. Elle a également été étudiée dans le cas de polygones. Cet article propose une approche numérique pour la classe générale des objets décrits par des inégalités non-linéaires. Notre objectif ici est de *calculer* la contrainte de non-chevauchement, c'est à dire, de décrire l'ensemble de toutes les positions et orientations qui peuvent être attribuées au premier objet de telle sorte que l'intersection avec le second soit vide. Nous nous basons sur un algorithme de branch & prune dédié. Nous montrons d'abord que la contrainte de non-chevauchement équivaut à une somme de Minkowski, même lorsque l'orientation est prise en compte. Nous en déduisons un *contracteur intérieur*, c'est à dire, un opérateur qui élimine du domaine courant un sous-ensemble de positions et orientations qui violent nécessairement la contrainte de non-chevauchement. Ce contracteur intérieur est intégré dans une boucle de *sweep*, une technique utilisée jusqu'ici uniquement pour les domaines discrets. Nous aboutissons ainsi à un algorithme de branch & prune présentant de bien meilleures performances que Rsolver, outil de référence pour la résolution de contraintes quantifiées en domaines continus.

## 1 Introduction

L'objectif de cet article est de calculer l'ensemble de toutes les positions et orientations qui peuvent être données à un objet de telle sorte qu'il ne chevauche pas un second objet (voir la figure 1, qui montre le cas plus simple où l'orientation n'est pas considérée). Nous abordons le cas général d'objets décrits par des inégalités non-linéaires. Calculer cet ensemble est une tâche centrale pour la résolution de problèmes de placement, qui consistent à placer un ensemble d'objets dans un espace limité de telle sorte qu'ils ne se chevauchent pas deux à deux.

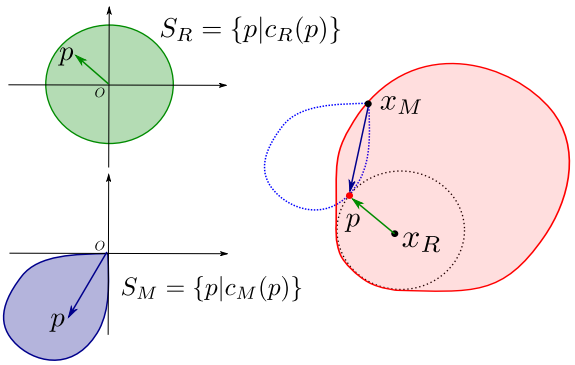


FIGURE 1 – **Contrainte de non-chevauchement**. *Gauche :* deux objets $S_R$ et $S_M$. *Droite :* la région rouge représente l'ensemble de toutes les positions $x_M$ pour $S_M$ où la contrainte de non-chevauchement est violée.

Dans cette introduction, nous définissons d'abord ce qu'est un *objet* dans notre contexte et formalisons la contrainte de non-chevauchement. Nous explicitons ensuite le type d'objets considéré puis expliquons ce que signifie ici *calculer* un ensemble. Nous mentionnons enfin d'autres travaux liés à cette problématique.

Dans la section 2, nous montrons que notre prob-

lème équivaut au calcul d'une somme de Minkowski. Sur la base de cette observation, nous proposons un algorithme de branch & prune dans la section 3. Des résultats expérimentaux sont donnés dans la section 4, avant de conclure.

### 1.1 Définition des objets

Par simplicité, supposons d'abord que l'orientation est fixée, c'est à dire, que les objets peuvent seulement être translatés.

Déplacer un objet revient à fixer la position d'un point particulier que nous appelons *l'origine*. Cette origine peut être choisie arbitrairement. Par exemple, dans le placement de rectangles, l'origine d'un rectangle peut être un sommet ou le milieu. Une fois cette convention faite pour l'origine, la forme de l'objet est simplement une contrainte usuelle. Pour l'illustrer, considérons de nouveau le placement de rectangles. Si son origine est le coin inférieur-gauche, le rectangle de dimensions $l_1$ et $l_2$ est l'ensemble de tous les $p \in \mathbb{R}^2$ tels que

$$c(p) \iff 0 \leq p_1 \leq l_1 \wedge 0 \leq p_2 \leq l_2. \tag{1}$$

La contrainte

$$c(p) \iff -\frac{l_1}{2} \leq p_1 \leq \frac{l_1}{2} \wedge -\frac{l_2}{2} \leq p_2 \leq \frac{l_2}{2}. \tag{2}$$

correspond, elle, à un rectangle dont l'origine est le milieu. Cette contrainte, bien sûr, n'est qu'un simple décalage de la contrainte précédente. Ainsi, la forme d'un objet peut être exprimée par une simple contrainte, moyennant une convention implicite pour l'origine. Prenons un dernier exemple : un cercle de rayon $r$ est l'ensemble de tous les points $p \in \mathbb{R}^2$ de tels que

$$c(p) \iff \|p\| \leq r, \tag{3}$$

l'origine étant, dans ce cas, le centre du cercle.

La contrainte $c(p)$ définit un objet sans déplacement ni rotation. La contrainte plus générale définissant un objet placé en $x$ et tournée d'un angle $\alpha$ est facilement obtenue comme suit. Tout d'abord, dans le cas d'une simple translation, la partie du plan occupée par l'objet placé en $x$ est l'ensemble de tous les points $p$ tels que $c(p - x)$ est satisfait. Ajoutons maintenant l'orientation. Avec un argument géométrique classique, un objet placé en $x$ et tournée d'un angle $\alpha$ est la contrainte

$$c\big(R_{-\alpha}(p - x)\big) \tag{4}$$

où $R_\alpha$ est la matrice de rotation d'angle $\alpha$ :

$$R_\alpha = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}. \tag{5}$$

De façon équivalente, nous pouvons dire que $c(p) \equiv c(R_0(p - 0))$ représente l'objet placé en 0 et tournée d'un angle 0. Pour conclure :

- *un objet* est une contrainte dans le plan (c.a.d., avec deux variables),
- *placer* un objet signifie fixer les coordonnées de son origine implicite,
- *orienter* un objet signifie fixer l'angle de rotation de son origine implicite.

Dans cet article nous considérons des objets décrits par des inégalités non-linéaires $c(p) \iff f(p) \leq 0$. Ex., un cercle de rayon 1 dont l'origine est le milieu est l'ensemble de tous les $p \in \mathbb{R}^2$ tels que $f(p) \leq 0$ avec $f : p \mapsto \|p\| - 1$. Pour clarifier la présentation, nous supposons que chaque objet est décrit par une seule inégalité, mais nos résultats peuvent être étendus facilement au cas plus général des objets décrits par des formules logiques du premier ordre (disjonctions de conjonctions d'inégalités). Il n'est également émis aucune hypothèse sur les fonctions impliquées, hormis le fait qu'elles soient définies par des expressions mathématiques basés sur les opérateurs standards ($+$, $\times$, $\sqrt{\cdot}$, exp, etc.). En particulier, il n'y a aucune hypothèse de convexité sur les objets en entrée.

### 1.2 Contrainte de non-chevauchement

Nous pouvons nous concentrer maintenant sur la contrainte de non-chevauchement. La contrainte de non-chevauchement implique deux objets, l'un étant fixé, l'autre représentant les inconnues du problème. Pour cette raison, nous appellerons "objet de référence" le premier objet et noterons $c_R$ la contrainte le décrivant. Le deuxième objet sera appelé "objet mobile" et sa contrainte notée $c_M$.

Nous allons montrer à la fin de cette section que le cas général où les deux objets sont translatés et/ou tournés peut en fait être obtenu à partir du cas plus simple où l'objet de référence n'a pas subi de transformation. Intuitivement, le repère dans lequel la contrainte de non-chevauchement est établie peut être centré sur l'objet de référence et aligné avec son orientation, quoique la formule exacte ne soit pas si triviale.

Nous allons donc introduire dans la définition ci-dessous uniquement la position $x_M$ et l'angle de rotation $\alpha_M$ de l'objet mobile. La contrainte de non-chevauchement est la négation de la contrainte de chevauchement qui peut être déclarée ainsi :

**Définition 1 (Contrainte de chevauchement)** *Étant donné deux contraintes $c_R$ et $c_M$, un vecteur $x_R \in \mathbb{R}^2$ et $\alpha_R \in [0, 2\pi]$ :*

$$\begin{gathered}\mathrm{overlap}_{(c_R, c_M)}(x_M, \alpha_M) \iff \\ \exists p \in \mathbb{R}^2, c_R(p) \wedge c_M\big(R_{-\alpha_M}(p - x_M)\big).\end{gathered} \tag{6}$$

Dans le cas d'une translation uniquement, cela ce simplifie en

$$\begin{array}{c}\text{overlap}_{(c_R,c_M)}(x_M) \iff \\ \exists p \in \mathbb{R}^2 c_R(p) \wedge c_M(p - x_M).\end{array} \quad (7)$$

Notre objectif est de *calculer* la contrainte de chevauchement. *Calculer* signifie qu'une représentation (numérique mais garantie) explicite de l'ensemble solution $\mathcal{S}' := \{(x_M, \alpha_M),\ \text{overlap}_{(c_R,c_M}(x_M, \alpha_M)\}$ (ou $\mathcal{S} := \{x_M,\ \text{overlap}_{(c_R,c_M)}(x_M)\}$ dans le cas du déplacement) doit être renvoyée par notre algorithme.

Nous montrons maintenant que la satisfiabilité de la contrainte de chevauchement dans le cas où l'objet de référence possède une position $x_R$ et une orientation $\alpha_R$ peut être testée en utilisant $\mathcal{S}'$ (et notamment sa représentation explicite). Plus précisément :

**Proposition 1** *L'objet mobile de position $x_M$ et d'orientation $\alpha_M$ chevauche l'objet de référence de position $x_R$ et d'orientation $\alpha_R$ ssi*

$$(R_{-\alpha_R}(x_M - x_R), \alpha_M - \alpha_R) \in S' \quad (8)$$

**Preuve 1** *Par définition, $(x_M, \alpha_M)$ satisfait la contrainte de chevauchement avec l'objet de référence placé en $x_R$ et tourné de $\alpha_R$ ssi il existe $\exists p \in \mathbb{R}^2$ tel que $c_R(R_{-\alpha_R}(p - x_R))$ et $c_M(R_{-\alpha_M}(p - x_M))$. Cela équivaut à ce qu'il existe $q \in \mathbb{R}^2$ obtenu ainsi :*

$$q = R_{-\alpha_R}(p - x_R) \iff p = R_{\alpha_R} q + x_R, \quad (9)$$

*tel que $c_R(q)$ et*

$$\begin{array}{c}c_M(R_{-\alpha_M}(R_{\alpha_R} q + x_R - x_M)) \iff \\ c_M(R_{-\alpha_M+\alpha_R}(q + R_{-\alpha_R}(x_R - x_M)));\end{array} \quad (10)$$

*ce qui équivaut à (8).* ▲

**Remarque 1** *Les applications de placement considèrent la contrainte de non-chevauchement (plutôt que la contrainte de chevauchement). La caractérisation de cette dernière est obtenue en échangeant simplement les rôles des deux ensembles $\mathcal{I}$ et $\mathcal{O}$ dans la définition 2 ci-dessousb. La contrainte de chevauchement est préférée ici car elle simplifie les expressions des contraintes et leur lien avec la somme de Minkowski, présentée dans la section 2.*

### 1.3 Intervalles, Boîtes et Pavage

La représentation que nous utilisons est un *pavage*. Cette représentation est un choix naturel dans le contexte de la programmation par contraintes où la grande majorité des algorithmes sur les variables continues, pour ne pas dire tous, supposent des domaines sous forme d'intervalles (voir, e.g., [3, 8]). Dans la définition ci-après, nous appelons *boîte* un produit cartésien de $d$ intervalles où $d$ est 2 dans le cas de $\mathcal{S}$ et 3 dans le cas de $\mathcal{S}'$.

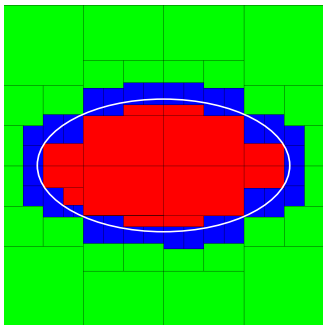

FIGURE 2 – **Pavage d'une ellipse**. Les boîtes rouges à l'intérieur appartiennent à $\mathcal{I}$, les boîtes bleues à la frontière appartiennent à $\mathcal{B}$ et les boîtes vertes à l'extérieur appartiennent à $\mathcal{O}$.

**Définition 2 (Pavage)** *Un pavage d'un ensemble $\mathcal{S} \subset \mathbb{R}^d$ est un triplet $(\mathcal{I}, \mathcal{B}, \mathcal{O})$ où $\mathcal{I}$ (pour "intérieure'), $\mathcal{O}$ (pour "extérieur") et $\mathcal{B}$ (pour "bord") sont trois ensembles de boîtes vérifiant*

$$\cup\mathcal{I} \subset \mathcal{S}, \quad (\cup\mathcal{O}) \cap \mathcal{S} = \emptyset \quad et \quad \cup(\mathcal{B} \cup \mathcal{I} \cup \mathcal{O}) = \mathbb{R}^d. \quad (11)$$

Un exemple de pavage apparaît dans la figure 2.

### 1.4 Contribution et Travaux Liées

Cet article propose une algorithme calculant un pavage de la contrainte de chevauchement. Une première contribution est sur la modélisation du problème : Nous montrons que la contrainte de chevauchement peut être exprimée comme une somme de Minkowski. D'un côté, cela généralise l'approche et simplifie la description de l'algorithme. De l'autre côté, cela permet de traiter de manière homogène le cas simple de la translation et le cas plus complexe combinant translation et rotation, en remplaçant l'angle de rotation par une cordonnée de translation supplémentaire dans un espace "augmenté" (voir proposition 2). L'autre contribution est algorithmique. Nous proposons un contracteur *intérieur* original pour ce problème, c'est à dire, un opérateur identifiant une partie de l'ensemble solution. Cet opérateur met en œuvre une *boucle de sweep* et exploite les propriétés de la somme de Minkowski. Le deuxième opérateur est un test de rejet extérieur basé sur une propagation de contraintes classique. Les deux opérateurs sont entrelacés dans un algorithme de branch & prune, qui calcule le pavage souhaité.

D'un autre point de vue, la définition 1 signifie que notre problème appartient à la catégorie des contraintes existentiellement quantifiées. Un algorithme

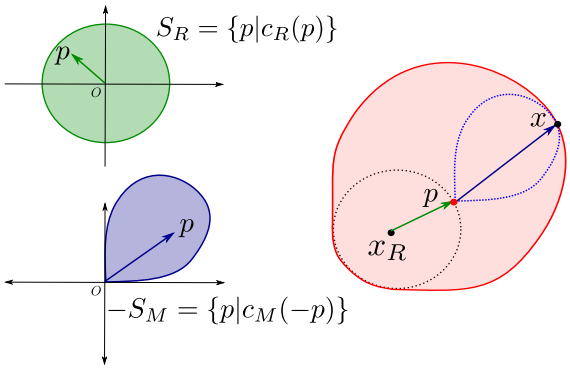


FIGURE 3 – **Somme de Minkowski.** La somme de Minkowski de $S_R$ (objet de référence) et $-S_M$ (symétrique de l'objet mobile) coïncide avec la description de la contrainte de non-chevauchement dans la figure 1.

de l'état de l'art pour calculer le pavage d'inégalités existentiellement quantifiées est décrit dans [11] et mis en œuvre dans l'outil RSOLVER [10] (Rsolver est un algorithme générique, sorte de variante numérique de CAD, pour résoudre des contraintes quantifiées). Notons que des techniques générales et efficaces pour les contraintes d'égalité existent aussi, cf. [6, 7].

Finalement, mentionnons qu'une formule exacte pour la contrainte de chevauchement $\mathcal{S}$ a été donnée dans [2] pour le cas où les objets sont des polytopes. L'ensemble, dans ce cas, est l'enveloppe convexe des points obtenus en sommant un sommet du premier polytope avec un sommet du second.

## 2 Contrainte de chevauchement et somme de Minkowski

Dans cette section, nous montrons que la contrainte de chevauchement peut être reformulée comme une somme de Minkowski. Cette reformulation sous-tend le solveur branch & prune que nous présentons plus loin. Rappelons d'abord la définition de la somme de Minkowski de deux ensembles :

**Définition 3 (Somme de Minkowski)**
*Étant donné deux ensembles équi-dimensionnels $S_1, S_2 \subseteq \mathbb{R}^d$, la somme de Minkowski est*

$$S_1 + S_2 = \{x_1 + x_2 \in \mathbb{R}^d : \ x_1 \in S_1, \ x_2 \in S_2\}. \quad (12)$$

*La différence de Minkowski est définie en conséquence par :*

$$S_1 - S_2 = \{x_1 - x_2 \in \mathbb{R}^d : \ x_1 \in S_1, \ x_2 \in S_2\}. \quad (13)$$

La figure 3 montre un exemple de deux ensembles et leur somme de Minkowski.

En considérant $S_1$ comme un contrainte $c_1$ (i.e., $x \in S_1 \iff c_1(x)$) et de façon similaire, $S_2$ comme la contrainte $c_2$, nous avons, de façon équivalente :

$$S_1 + S_2 = \{x \in \mathbb{R}^d : \ \exists p \in \mathbb{R}^d, \ c_1(p) \wedge c_2(x-p)\}, \quad (14)$$

où $d$ est le nombre de variables dans la contrainte. En comparant (14) et (7), on voit immédiatement que $\mathcal{S} = S_R - S_M$, c.a.d., la contrainte de chevauchement peut être représentée comme une somme de Minkowski dans le cas d'une translation seulement.

Nous montrons maintenant que $\mathcal{S}'$ est aussi une somme de Minkowski, un résultat moins trivial. Pour cela, nous plongeons l'objet mobile $S_M \subseteq \mathbb{R}^2$ dans $\mathbb{R}^3$ en encodant la rotation dans la dimension supplémentaire :

$$\begin{aligned} \mathcal{S}'_M &:= \{(v, \beta) : c_M(R_\beta \, v)\} \\ &= \{(v, \beta) : R_\beta \, v \in \mathcal{S}_M\}. \end{aligned} \quad (15)$$

La proposition suivante énonce alors que la contrainte de chevauchement avec rotation $\mathcal{S}'$ peut être réécrite comme une différence de Minkowski entre deux tels ensembles "augmentés" (voir figure 4).

**Proposition 2**

$$\mathcal{S}' = S_R \times \{0\} - S'_M. \quad (16)$$

**Preuve 2** *Par définition, $(x_M, \alpha_M) \in \mathcal{S}'$ est vrai si et seulement si $\exists p \in \mathbb{R}^2$ tel que $c_R(p)$ et $c_M(R_{-\alpha_M}(p - x_M))$.*

*De façon équivalente, il existe $u_R \in \mathcal{S}_R$ et $u_M \in \mathcal{S}_M$ tels que $u_R = p$ et $u_M = R_{-\alpha_M}(u_R - x_M) \iff x_M = u_R - R_{\alpha_M} u_M$. Finalement le vecteur $(x_M, \alpha_M)$ est prouvé comme étant la somme de $(u_R, 0) \in \mathcal{S}_R \times \{0\}$ et $(R_{\alpha_M} u_M, \alpha_M) \in \mathcal{S}'_M$.* ▲

## 3 Algorithme

Notre objectif est désormais de calculer un pavage $(\mathcal{I}, \mathcal{B}, \mathcal{O})$ de la somme $\mathcal{S}$ de deux ensembles $\mathcal{S}_1$ et $\mathcal{S}_2$. D'après la section précédente, le lien avec la contrainte de non-chevauchement se fait en remplaçant $\mathcal{S}_1$ par $S_R$ ou $S'_R$ et $\mathcal{S}_2$ par $-S_M$ ou $-S'_M$.

Notre algorithme est basé sur un branch & bound récursif classique de type SIVIA [9, 5]. L'opération centrale effectuée sur chaque boîte $[x]$ se divise en trois parties. D'abord, $[x]$ est contracté en une boîte $[x]'$ par un *contracteur intérieur*, c'est à dire, un opérateur qui garantit :

$$[x]' \subseteq [x] \ \wedge \ [x] \backslash [x]' \subseteq \mathcal{I}. \quad (17)$$

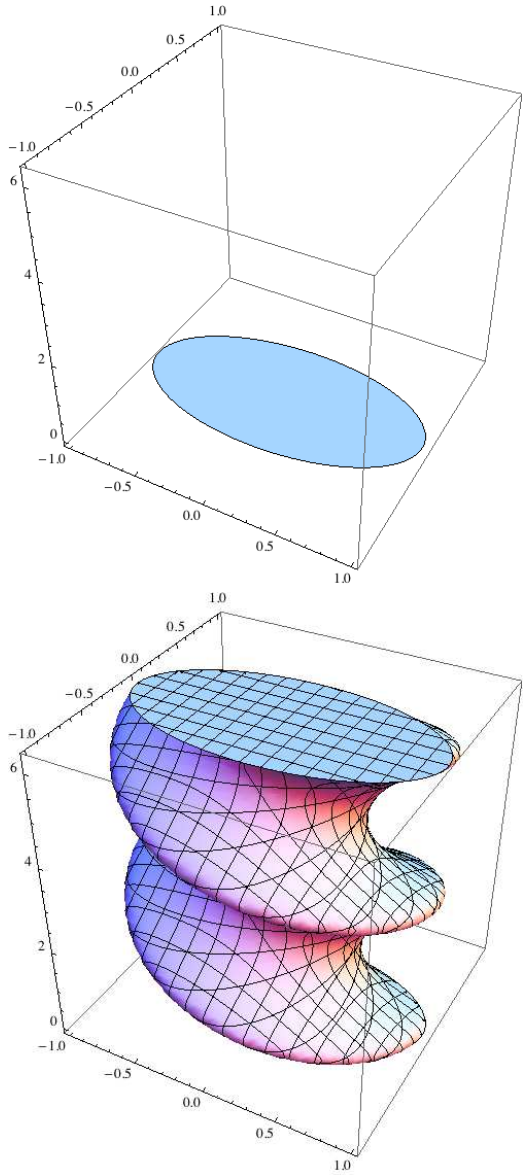

FIGURE 4 – Ensembles dont la différence de Minkowski donne la contrainte de chevauchement de deux ellipses, lorsque la rotation est prise en compte. *Gauche :* l'ellipse de référence augmentée, avec la coordonnée de rotation mise à 0. *Droite :* l'ellipse mobile augmentée $S'_M$.

Si $[x]' \neq \emptyset$ alors $[x]'$ est contracté en une boîte $[x]''$ par un *contracteur extérieur* qui garantit :

$$[x]'' \subseteq [x]' \ \wedge \ [x]' \backslash [x]'' \subseteq \mathcal{O}. \tag{18}$$

Finalement, si $[x]'' \neq \emptyset$ alors $[x]''$ est découpé en deux nouvelles boîtes qui sont ajoutées dans la liste des boîtes frontières $\mathcal{B}$. La récursivité s'arrête quand la surface totale des boîtes dans $\mathcal{B}$ est en dessous de $\varepsilon\%$ de la surface de la boîte initiale, où $\varepsilon$ est un paramètre défini par l'utilisateur.

L'originalité de notre approche réside dans le contracteur intérieur, que nous allons décrire maintenant.

### 3.1 Contracteur Intérieur

Notre contracteur intérieur est basé sur *l'arithmétique intérieure*, une variante de l'arithmétique d'intervalles classique qui permet de construire une sous-boîte d'une boîte $[x]$, à l'intérieur d'un ensemble donné $\mathcal{S}$ décrit par des inégalités. Cette technique a été introduite pour la première fois dans §3 de [4] puis utilisée dans [1] dans le cadre de l'optimisation globale. Cette arithmétique intérieure peut aussi être utilisée avec un point initial (ou une boîte initiale) qui est alors *étendu*. Plus précisément, étant donné une boîte $[x]$ et $\tilde{x} \in [x]$, elle produit une boîte $[\tilde{x}]$ vérifiant

$$\tilde{x} \in [\tilde{x}] \subseteq [x] \ \wedge \ [\tilde{x}] \subseteq \mathcal{S}, \tag{19}$$

ou une boîte vide si $\tilde{x} \notin \mathcal{S}$. Cette arithmétique partage des propriétés similaires avec son pendant classique : la complexité temporelle est en la longueur de l'expression de la contrainte et elle produit une boîte optimale $[\tilde{x}]$ (c.a.d., de taille maximale sur chaque dimension) si aucune variable n'apparaît deux fois dans l'expression.

Avant de décrire la façon dont une boîte $[x]$ peut être contractée avec cette nouvelle arithmétique, adressons d'abord une question plus simple : comment trouver une sous-boîte de $[x]$ qui soit à l'intérieure de $\mathcal{S}$ ?

Une réponse possible consiste à chercher deux boîtes $[x]_1$ et $[x]_2$ telles que

$$[x]_1 \subseteq \mathcal{S}_1, \quad [x]_2 \subseteq \mathcal{S}_2 \ \text{ et } \ ([x]_1 + [x]_2) \cap [x] \neq \emptyset \tag{20}$$

car, dans ce cas, $([x]_1 + [x]_2) \subseteq [x] \cap \mathcal{S}$. Pour trouver de telles boîtes, nous pouvons calculer en parallèle deux pavages, le premier pour $\mathcal{S}_1$, l'autre pour $\mathcal{S}_2$, et arrêter le processus lorsque deux boîtes qui satisfont (20) sont identifiées. En combinant les boîtes du premier pavage avec celles du deuxième, cette approche revient à exécuter un branch & bound dans un espace $(2 \times d)$-dimensionnel. Notons que ce branch & bound est un sous-solveur embarqué dans le solveur principal, celui pour la variable $x$. Notre objectif est de réduire le sous-solveur à seulement $d$ dimensions, ce qui, en passant, est le coût incompressible à payer pour manipuler $d$ paramètres existentiellement quantifiés.

À cette fin, nous utilisons la même idée que ci-dessus, mais en se basant cette fois sur la relation (14) (voir figure 5). Pour construire une boîte intérieure dans $[x]$, nous supposons tout d'abord qu'un point quelconque $\tilde{x}$ a été choisi à l'intérieur de $[x]$ (ce point est, en fait, produit automatiquement par la boucle

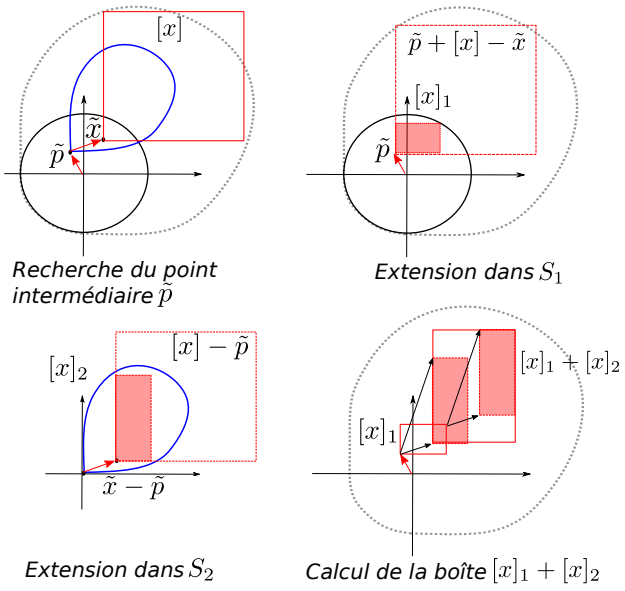


FIGURE 5 – **Étapes du contracteur intérieur.**

de *sweep*, comme cela sera expliqué dans la figure 6). Nous cherchons alors un autre point $\tilde{p}$ tel que

$$c_1(\tilde{p}) \wedge c_2(\tilde{x}-\tilde{p}). \tag{21}$$

La recherche de ce point est la tâche du sous-solveur [1].

Lorsque que $\tilde{p}$ est trouvé, il est "étendu" à une sous-boîte $[x]_1$ de $(\tilde{p}+[x]-\tilde{x})$ à l'intérieur de $\mathcal{S}_1$ grâce à l'arithmétique intérieure. Le point $(\tilde{x}-\tilde{p})$ est également étendu à une sous-boîte $[x]_2$ de $([x]-\tilde{p})$ a l'intérieur de $\mathcal{S}_2$. Cependant, si $\tilde{x}$ s'avère être à l'extérieur de $\mathcal{S}$, cette dernière inflation échoue et le processus, dans ce cas, est arrêté. Sinon, la boîte qui résulte satisfait $([x]_1+[x]_2) \subseteq \mathcal{S}$ et $([x]_1+[x]_2) \cap [x] \neq \emptyset$.

Notons que $([x]_1+[x]_2) \cap [x] \neq \emptyset$ est seulement une conséquence de $\tilde{x} \in [x]$. Donc les boîtes initiales utilisées pour les deux extensions sont en quelque sorte arbitraires, mais la façon dont nous les avons fixées est un heuristique qui tend à maximiser la surface de la boîte finale $([x]_1+[x]_2) \cap [x]$.

Maintenant que nous possédons une technique pour construire une boîte intérieure dans $[x]$ contenant un point spécifique $\tilde{x}$, nous pouvons utiliser ce service dans une boucle de *sweep*. La boucle de sweep peut être vue simplement comme une façon de contracter une boîte en "empilant" des sous-boîtes jusqu'à ce qu'une face soit entièrement couverte. Ce procédé est brièvement résumé dans la figure 6. Le lecteur intéressé peut se référer à [4] pour plus détails.

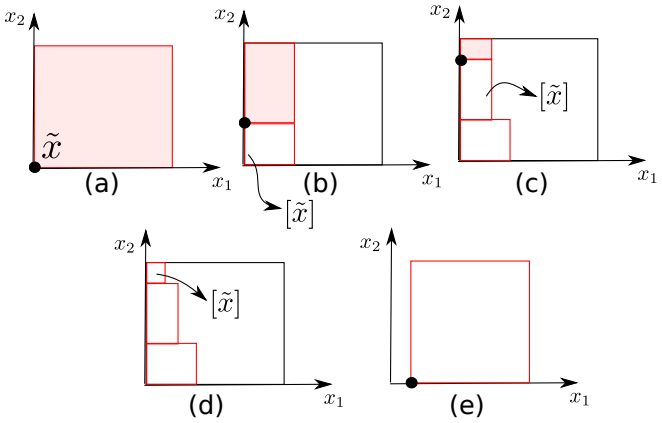


FIGURE 6 – **Boucle de sweep**. La séquence d'images illustre une contraction pour la borne inférieure de $x_1$. À chaque étape, le point $\tilde{x}$ étendu est le coin inférieur-gauche de la boîte grisée. La boîte intérieure $[\tilde{\mathbf{x}}]$ est en blanc. La borne inférieure de $x_1$ peut être réduite aussitôt que la projection des boîtes blanches sur $x_2$ recouvre entièrement la face $[x_2]$, ce qui est le cas à l'étape **e)**.

### 3.2 Contracteur Extérieur

Le contracteur extérieur est moins sophistiqué que le contracteur intérieur et agit comme un simple *test de rejet* : la boîte est entièrement éliminée ou conservée intacte.

Rejecter une boîte $[x]$ signifie prouver $[x] \not\subseteq S_1+S_2$, c'est à dire

$$\forall x \in [x] \quad \forall p \in \mathbb{R}^d,\ \neg(c_1(p) \wedge c_2(x-p)). \tag{22}$$

Cette assertion peut être vérifiée en exécutant le même sous-solveur que pour le contracteur intérieur, à ceci près que le point $\tilde{x}$ est cette fois remplacé par la boîte courante $[x]$. Si le sous-solveur ne trouve pas de solution, l'assertion précédente est prouvée. Notons que seules les coordonnés de $p$ sont bisectées, le sous-solveur prouve donc une assertion plus forte si la contraction par rapport à $c_2$ n'est pas optimale (l'assertion exacte dépend du niveau de cohérence effectuée par la contraction par rapport à $c_2$). Notons aussi que la précision utilisée dans le sous-solveur est dynamiquement réglée à le largeur de $[x]$ afin de maintenir une certaine uniformité dans le temps passé par le sous-solveur tout

1. Ce sous-solveur est mis en œuvre avec un branch & bound standard basé sur HC4 [3]. Puisqu'une seule solution suffit, à chaque nœud de la recherche, les inégalités sont vérifiées avec un point $\tilde{p}$ tiré aléatoirement dans le domaine courant. Si les deux sont satisfaites, la recherche est interrompue et $\tilde{p}$ est retourné. La profondeur de la recherche est également contrôlée par une précision sur la largeur du domaine, afin d'assurer une terminaison du sous-solveur en temps limité. En cas de terminaison normale, il n'a pu être trouvé de $\tilde{p}$, et donc de boîte intérieure.

au long de la recherche globale (sur $x$). Cette précision dynamique assure également que le contracteur extérieur est *convergent*, c'est à dire, qu'il rejette toute boîte suffisamment petite extérieure à $\mathcal{S}$.

On peut être surpris par la simplicité de ce test de rejet et s'attendre à un contracteur plus élaboré pour la région extérieure, à l'instar de celui que nous avons proposé pour la région intérieure. Mais la situation peut être interprétée dans l'autre sens. Du fait que la contrainte de chevauchement soit en deux dimensions seulement, un test de satisfiabilité intérieur serait probablement suffisant, si tant est qu'il soit assez rapide et convergent. Cependant, un tel test revient à prouver pour $[x] \subseteq S_1 + S_2$ l'assertion suivante

$$\forall x \in [x] \quad \exists p \in \mathbb{R}^d,\ (c_1(p) \wedge c_2(x-p)).$$

Or, contrairement à (22), les deux quantificateurs $\forall$ et $\exists$ sont cette fois impliqués, ce qui veut dire que le problème est beaucoup plus difficile. Donc, le contracteur intérieur peut être vu ici comme un moyen de compenser l'absence de test intérieur.

Notre argument précédent est basé seulement sur le temps d'exécution. Il est clair qu'un contracteur extérieur peut aussi conduire à une pavage plus compact, mais la taille de ce pavage est de toute façon conditionnée par la représentation de la frontière, de telle sorte qu'un gain drastique sur cet aspect ne serait de toute façon pas réellement envisageable.

## 4 Résultats expérimentaux

### Protocole

L'algorithme que nous proposons dans cet article calcule un pavage $(\mathcal{I}, \mathcal{B}, \mathcal{O})$ de la contrainte de non-chevauchement. La difficulté de cette tache dépend principalement de trois critères :

- *occurrences des variables* : le nombre de fois où chaque variable apparaît dans les expressions des inégalités impacte directement l'efficacité du contracteur ; plus une variable apparaît, moins les contracteurs sont efficaces. C'est une limitation bien connue de l'arithmétique d'intervalles classique qui s'applique également à l'arithmétique intérieure (utilisée par le contracteur intérieur).
- *convexité* : si les objets ne sont pas convexes, la frontière de la contrainte de non-chevauchement sera moins lisse. Le pavage sera donc plus complexe (et donc plus lent à calculer), particulièrement à proximité de la frontière.
- *degrés de liberté* : c'est à dire, si nous prenons en compte la rotation ou pas. Autoriser la rotation donne lieu à un problème d'une difficulté supérieur pour plusieurs raisons. Tout d'abord, la taille du pavage étant exponentielle en le nombre de degrés de liberté, un pavage en 3D ne s'obtient pas en général dans l'échelle de temps d'un pavage 2D. De plus, les angles dans les inégalités engendrent de nombreuses multi-occurrences (voir les équations (4) et (5)) et augmentent considérablement les non-convexités par l'introduction de fonctions trigonométriques.

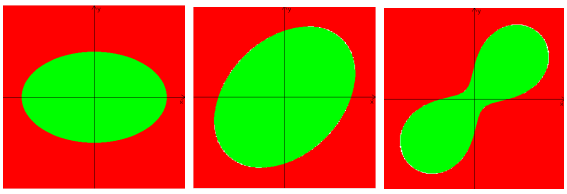

FIGURE 7 – **Objets de complexité croissante**. *De gauche à droite* : objets №1, 2 et 3.

Notre campagne de tests est basée sur ces critères. Nous avons fait deux types d'expériences. La première est avec translation seulement. Nous avons considéré trois objets de difficulté croissante. L'objet №1 est une ellipse simple :

$$\begin{array}{c}\text{Object №1 :}\\ (p_1/2)^2 + p_2^2 \leq 1.\end{array} \tag{23}$$

L'objet №2 est une ellipse "oblique" (tournée d'un angle quelconque). Les objets №1 et №2 ont évidemment la même complexité si la rotation est un degré de liberté, mais pas si nous nous limitons à translater. Cela tient au fait que l'objet tourné introduit plusieurs occurrences de $p_1$ et $p_2$ :

$$\begin{array}{c}\text{Object №2 :}\\ 1.5 \times p_1^2 + 1.5 \times p_2^2 - p_1 \times p_2 - 0.2 \leq 0.\end{array} \tag{24}$$

Finalement, le troisième objet a une forme de "cacahuète". Il cumule la multi-occurrences de variables et la non-convexité (comme on peut l'observer sur la figure 7) :

$$\begin{array}{c}\text{Object №3 :}\\ (p_1^2 + p_2^2)^2 - 2 \times (p_1 \times p_2) - 0.02 \leq 0.\end{array} \tag{25}$$

La contrainte de non-chevauchement implique deux objets : l'objet de "référence" (dont les coordonnées sont fixées à l'origine du repère) et l'objet "mobile" qui représente les inconnues. Nous avons considéré toutes les combinaisons possibles avec les trois types d'objets ci-dessus, c'est à dire, les 6 premières cases de la table 1.

Dans notre seconde série d'expériences, nous avons introduit la rotation et testé avec, d'une part, deux

| Cas | Objet de ref. | Objet mobile | Rotation |
|---|---|---|---|
| 1 | 1 | 1 | no |
| 2 | 1 | 2 | no |
| 3 | 1 | 3 | no |
| 4 | 2 | 2 | no |
| 5 | 2 | 3 | no |
| 6 | 3 | 3 | no |
| 7 | 1 | 1 | yes |
| 8 | 3 | 3 | yes |

TABLE 1 – Cas d'étude.

ellipses et, d'autre part, deux "cacahuètes" (cas 7 et 8).

Dans chaque expérience, le processus de pavage est interrompu quand la surface totale de la frontière $\mathcal{B}$ est en dessous de $\varepsilon$% de la surface de la boîte initiale (domaine initial pour les variables), où $\varepsilon$ est un paramètre de précision défini par l'utilisateur. Nous avons appliqué la même politique à RSOLVER, l'outil avec lequel nous nous comparons.

Puisque que la précision est en proportion de la surface du domaine initial, il convient de noter que la qualité du pavage dépend aussi de qualité de l'enveloppe initiale. Plus le domaine initial est grand, moins le résultat final est précis. Pour cette raison, et afin donner à $\varepsilon$ une valeur significative, nous avons initialisé dans les expériences la boîte initiale à une enveloppe assez précise de l'ensemble $\mathcal{S}$ ou $\mathcal{S}'$ (comme cela peut être vu sur les figures 8 et suivantes).

### Résultats (sans rotation)

D'abord, nous comparons dans la table 2 les temps d'exécution obtenus par RSOLVER et notre algorithme pour les 6 premiers cas, avec une precision $\varepsilon$ mise à 3.25%. Ce choix pour $\varepsilon$ correspond à la valeur minimale avec laquelle RSOLVER produit un pavage dans les limites de temps.

Les pavages obtenus sont représentés sur la figure 8.

Nous donnons sur la figure 9 une analyse plus détaillée des temps de calcul pour les deux cas extrêmes (cas 1 et 6), où $\varepsilon$ varie cette fois de 10% à 1%. Elles montrent des gains significatifs en performance absolue, ainsi qu'un meilleur comportement asymptotique, par rapport à RSolver.

Les résultats confirment d'abord les niveaux de complexité présumés des différent cas, puisque les instances plus "dures" exigent en effet plus de temps pour être résolues. Ils montrent aussi que notre algorithme est plus compétitif que RSOLVER. Il faut néanmoins garder à l'esprit que RSOLVER est un solveur générique, qui n'exploite pas la structure spécifique du problème que l'on traite ici. Les graphiques indiquent également que l'écart entre notre approche et RSolver,

| Cas | Rsolver | Notre algorithme |
|---|---|---|
| 1 | 4,07 | 0,37 |
| 2 | 51,55 | 2,67 |
| 3 | 611,85 | 7,90 |
| 4 | 132,46 | 5,58 |
| 5 | 656,11 | 12,00 |
| 6 | 771,00 | 26,82 |

TABLE 2 – Temps d'exécution (en s) pour les 6 premiers cas (la précision est à 3.25%).

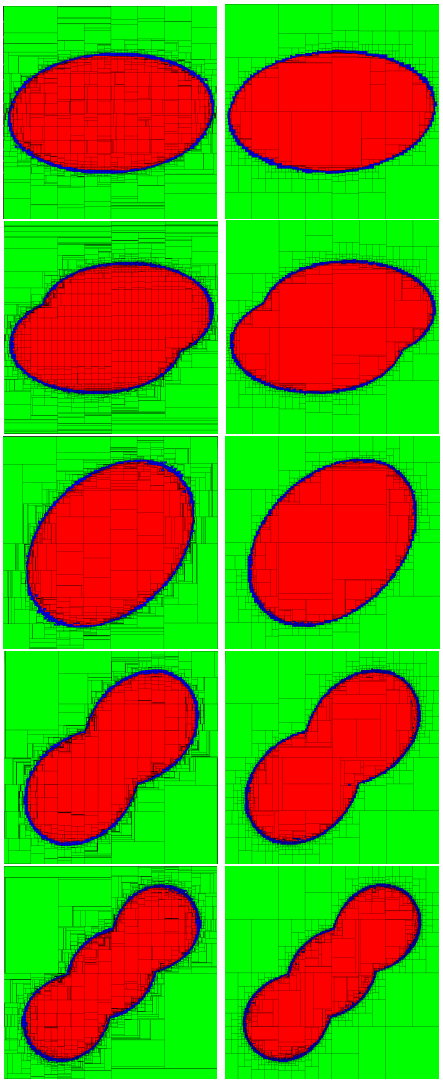

FIGURE 8 – **Pavage obtenu pour les cas 2-6**. La précision est à 3.25%. *Gauche :* avec RSolver. *Droite :* avec notre algorithme.

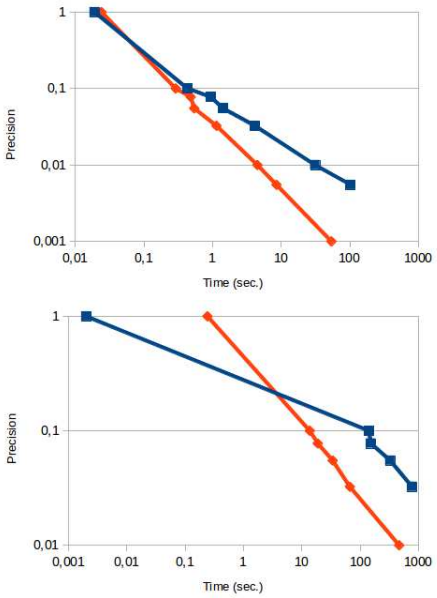


FIGURE 9 – **Temps en fonction de la précision (gauche : cas 1 ; droite : cas 6)**. Chaque courbe représente le temps de pavage (axe vertical) en fonction de la précision $\varepsilon$ (axe horizontal). Les deux axes sont en échelle logarithmique.

pour un cas donné, augmente lorsque la valeur de la précision diminue.

### Résultat (avec rotation)

Nous présentons ici seulement des résultats préliminaires avec la rotation.

Les figures 10 et 11 montrent des sections 2D des pavages 3D que nous avons obtenus dans les cas 7 et 8, avec une précision de 3.25%. Une section 2D est obtenue en fixant l'angle à une certaine valeur et en sélectionnant les boîtes dans le pavage 3D pour lesquelles l'intervalle de l'angle contient cette valeur. Les deux autres dimensions sont alors tracées.

Nous avons attribué au domaine de l'angle l'intervalle $[0, 0.7]$ pour le cas 7, et l'intervalle $[0, 0.3]$ pour le cas 8. Le seul pavage qu'il a été possible d'obtenir dans le délai imparti était avec notre algorithme, pour le cas 7. Ce pavage a été calculé en 9 minutes tandis que RSOLVER n'a pas donné de résultat après 80 minutes. Les deux algorithmes ne terminent pas au bout de 100 minutes dans le cas 8.

Quand un programme est arrêté manuellement, un résultat partiel est affiché. Un résultat partiel signifie que la surface de $\mathcal{B}$ dépasse $\varepsilon$% la largeur initiale.

L'objectif principal de cette expérience est de montrer que notre approche reste valide dans le cas où

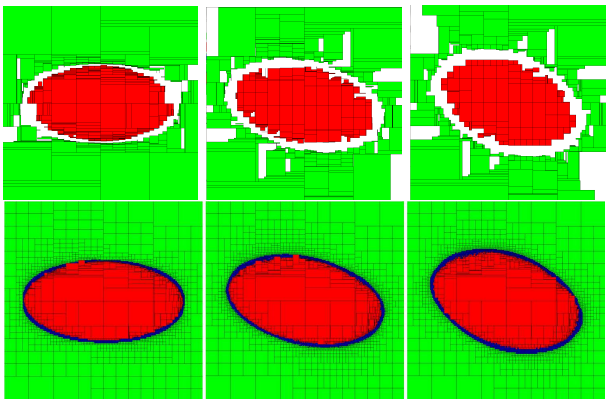

FIGURE 10 – **Section 2D d'un pavage 3D obtenu pour le cas 7.** *De gauche à droite :* les angles de rotation sont 0, 0.4 et 0.7. *Haut :* avec RSOLVER. *Bas :* avec notre algorithme.

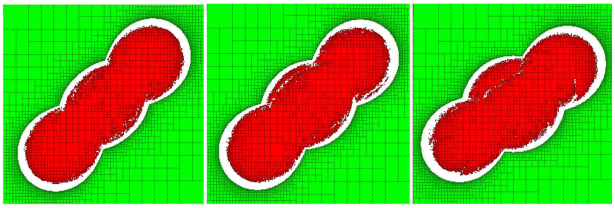

FIGURE 11 – **Section 2D d'un pavage 3D obtenu pour le cas 8.** De gauche à droite, les angles de rotation sont 0, 0.1 et 0.3. Ce pavage est calculé avec notre algorithme.

la rotation est prise en compte. La rotation implique simplement une transformation des expressions qui décrivent les objets. Elle n'exige aucun changement dans l'algorithme lui-même. Il est clair néanmoins que faire le pavage exhaustif d'un ensemble en 3D est une tâche très lourde, quel que soit l'algorithme.

## 5 Conclusion

Dans le cas d'objets définis par des inégalités non-linéaires, la contrainte de non-chevauchement ne peut être traitée que numériquement. Dans cet article, nous avons donné un moyen efficace pour générer une approximation garantie de cette contrainte, sous forme de pavage. Ce pavage représente explicitement la contrainte, c'est à dire, l'ensemble de toutes les positions et orientations acceptables pour un objet par rapport à un autre, déjà placé. Nos premières expériences ont montré des gains importants en efficacité par rapport à RSolver, le solveur de l'état de l'art pour les contraintes numériques quantifiées, et notamment dans

le cas où l'orientation de l'objet est prise en compte. Néanmoins, notre algorithme doit encore être amélioré dans ce cas, où les temps de calculs restent malgré tout importants. Les travaux futurs visent à améliorer encore l'efficacité de la méthode, en particulier par l'introduction d'un contracteur extérieur à la place du test de rejet extérieur actuel. Ces pavage pré-calculés seront également intégrés dans un algorithme de placement exploitant les descriptions explicites des contraintes de chevauchement.

## Références


[1] I. Araya, G. Trombettoni, B. Neveu, and G. Chabert. Upper Bounding in Inner Regions for Global Optimization under Inequality Constraints. *Journal of Global Optimization*, page (to appear), 2014.

[2] N. Beldiceanu, Q. Guo, and S. Thiel. Non-Overlapping Constraints between Convex Polytopes. In *7th International Conference on Principles and Practice of Constraint Programming (CP'01)*, volume 2239 of *Lecture Notes in Computer Science*, pages 392–407. Springer-Verlag, 2001.

[3] F. Benhamou and L. Granvilliers. Continuous and interval constraints. In *Handbook of Constraint Programming*, chapter 16, pages 571–604. Elsevier, 2006.

[4] G. Chabert and N. Beldiceanu. Sweeping with Continous Domains. In D. Cohen, editor, *16th International Conference on Principles and Practice of Constraint Programming (CP'10)*, volume 6308 of *Lecture Notes in Computer Science*, pages 137–151, St Andrews, Scotland, 2010. Springer-Verlag.

[5] G. Chabert and L. Jaulin. Contractor Programming. *Artificial Intelligence*, 173(11) :1079–1100, 2009.

[6] A. Goldsztejn and L. Jaulin. Inner and Outer Approximations of Existentially Quantified Equality Constraints. In *CP*, pages 198–212. Springer, 2006.

[7] D. Ishii, A. Goldsztejn, and C. Jermann. Interval-Based Projection Method for Under-Constrained Numerical Systems. *Constraints*, 17(4) :432–460, 2012.

[8] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis*. Springer, 2001.

[9] L. Jaulin and E. Walter. Set Inversion via Interval Analysis for Nonlinear Bounded-Error Estimation. *Automatica*, 29(4) :1053–1064, 1993.

[10] S. Ratschan. RSolver.

[11] S. Ratschan. Efficient Solving of Quantified Inequality Constraints over the Real Numbers. *ACM Transactions on Computational Logic*, 7(4) :723–748, 2006.