HAL
open science

Extraction de régions intérieures pour améliorer le majorant en optimisation globale sous contraintes

Ignacio Araya, Gilles Trombettoni, Bertrand Neveu, Gilles Chabert

To cite this version:

Ignacio Araya, Gilles Trombettoni, Bertrand Neveu, Gilles Chabert. Extraction de régions intérieures pour améliorer le majorant en optimisation globale sous contraintes. 9èmes Journées Francophones de Programmation par Contraintes (JFPC 2013), Jun 2013, Aix-en-Provence, France. hal-00879489

HAL Id: hal-00879489

https://inria.hal.science/hal-00879489v1

Submitted on 4 Nov 2013

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.

Extraction de régions intérieures pour améliorer le majorant en optimisation globale sous contraintes

Ignacio Araya[1] **Gilles Trombettoni**[2] **Bertrand Neveu**[3] **Gilles Chabert**[4]

[1] UTFSM, Universitad Federico Santa Maria, Valparaiso, Chile
[2] LIRMM, Université Montpellier 2, France
[3] Imagine LIGM Université Paris–Est, France
[4]LINA, Ecole de Mines de Nantes, France

iaraya@inf.utfsm.cl,Gilles.Trombettoni@lirmm.fr,Bertrand.Neveu@enpc.fr,Gilles.Chabert@emn.fr

Résumé

En optimisation globale continue sous contraintes, la recherche d'un meilleur point réalisable utilise généralement des méthodes d'optimisation locale en chaque nœud de l'arbre de recherche développé par une méthode de type *Branch & Bound*. Nous proposons une approche alternative quand les contraintes sont des inégalités et que l'espace réalisable a un volume non nul. Tout d'abord, on extrait une région intérieure, c.-à-d. un polyèdre convexe ou une boîte entièrement réalisable. Ensuite, on sélectionne un point dans la région intérieure extraite et on met à jour le majorant de la fonction objectif si ce point l'améliore.

Nous décrivons dans cet article deux algorithmes originaux d'extraction de régions intérieures implantés dans `IbexOpt`. Ils s'appliquent à des contraintes continues non convexes contenant des opérateurs comme $+$, $\cdot$, $/$, $power$, $sqrt$, exp, log, sin. Cette approche produit de très bons résultats sur des systèmes de taille moyenne du banc d'essai `COCONUT`.

1 Introduction

En optimisation globale sous contraintes[1], la principale méthode exacte utilisée est basée sur l'algorithme du *Branch and Bound* (B&B). La recherche d'un majorant de l'objectif consiste à trouver un point réalisable qui améliore le meilleur coût trouvé jusqu'à présent. La plupart des optimiseurs globaux ont recours à la minimisation locale utilisant une relaxation lagrangienne. La relaxation considérée peut être complexe et rendre l'optimisation locale coûteuse en temps de calcul. Cet article décrit une approche alternative qui évite l'évaluation répétée de cette relaxation. L'approche cherche à construire un espace réalisable de volume non nul formé par les contraintes d'inégalité.

Plus précisément, le problème d'optimisation globale que nous traitons est défini par :

$$\min_{x\in[x]} f(x) \text{ soumis à } g(x) \leq 0,$$

où $f : \mathbb{R}^n \rightarrow \mathbb{R}$ est la fonction objectif et $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$ la fonction représentant les contraintes. $x = (x_1, ..., x_i, ...x_n)$ est un vecteur de variables ayant pour domaine une boîte $[x] = [x_1] \times ... \times [x_i] \times ... \times [x_n]$. Un point x est **réalisable** s'il satisfait les contraintes.

L'idée principale de cet article est l'exploitation de *régions intérieures*, c.-à-d. des sous-ensembles de l'espace de recherche dont tous les points sont réalisables.

Deux communautés étudiant les méthodes à intervalles ont proposé plusieurs heuristiques pour extraire des boîtes intérieures à partir d'un domaine donné (boîte extérieure). La communauté d'analyse numérique par intervalles a étudié les systèmes linéaires ayant pour coefficients des intervalles [20, 21]. La communauté de programmation par contraintes a proposé plusieurs heuristiques générales pour des systèmes d'inégalités non convexes [8, 3, 9].

Cet article décrit deux algorithmes originaux d'extraction de région intérieure. Ces heuristiques sont appliquées pour la première fois en optimisation globale sous contraintes.

A chaque nœud de notre B&B appelé `IbexOpt` [23], un majorant est calculé en utilisant deux pro-

[1]Cet article considère le problème de minimisation, sans perte de généralité.

cédures d'extraction de région intérieure appelées `InnerPolytope` et `InHC4`. L'algorithme `InnerPolytope`, décrit dans la partie 3, construit un hyperplan pour chaque contrainte d'inégalité en utilisant une formule de Taylor sur intervalles avec pour point d'expansion un coin de la boîte courante. Si l'on réussit à construire un polytope intérieur non vide, le point minimisant la linéarisation de l'objectif est utilisé pour mettre à jour le majorant.

`InHC4` est décrit dans la partie 4. Il essaie tout d'abord d'extraire une boîte intérieure contrainte par contrainte. En cas d'échec, il tire aléatoirement un point dans la boîte courante et teste s'il est réalisable. En cas de succès, une analyse de monotonie de f remplace les intervalles des variables monotones par leur borne adéquate dans la boîte intérieure trouvée par `InHC4`. Aucun test des contraintes n'est alors nécessaire car tous les points de cette boîte sont réalisables. Le point réalisable finalement obtenu est utilisé pour mettre à jour le majorant.

Contrairement aux approches existantes, les algorithmes de mise à jour du majorant que nous proposons séparent la partie réalisable (traitée en premier par l'extraction de région intérieure) et le calcul de l'objectif (traité ensuite dans la région intérieure).

La partie 5 met en évidence le fait que, comme les autres algorithmes d'extraction de région intérieure, nos algorithmes sont des heuristiques. En d'autres termes, il peuvent parfois ne pas trouver de région intérieure, même si une telle région existe. Cependant, nous soulignons qu'ils sont en général peu coûteux.

Les expérimentations de la partie 6 montrent que la recherche de majorant utilisant les deux procédures `InnerPolytope` et `InHC4` apporte d'importants gains de performance à `IbexOpt`.

2 Prérequis et problème traité

Les intervalles offrent des calculs rigoureux sur des ordinateurs en gérant des arrondis extérieurs sur des bornes flottantes.

Un **intervalle** $[x_i] = [\underline{x_i}, \overline{x_i}]$ définit l'ensemble des réels x_i t.q. $\underline{x_i} \leq x_i \leq \overline{x_i}$, où $\underline{x_i}$ et $\overline{x_i}$ sont des nombres flottants. L'ensemble de tous les intervalles est noté $\mathbb{IR}$. La taille ou **largeur** de $[x_i]$ est $w([x_i]) = \overline{x_i} - \underline{x_i}$. Une **boîte** $[x]$ est le produit cartésien d'intervalles $[x_1] \times ... \times [x_i] \times ... \times [x_n]$. Sa largeur est définie par $\max_i w([x_i])$. $m([x])$ désigne le milieu de $[x]$. L'**enveloppe** (ou *hull*) d'un sous-ensemble S de $\mathbb{R}^n$ est la plus petite boîte n-dimensionnelle contenant S.

L'*arithmétique d'intervalles* [16] a été définie pour étendre à $\mathbb{IR}$ les fonctions élémentaires sur $\mathbb{R}$. Par exemple, la somme sur les intervalles est définie par $[x_1] + [x_2] = [\underline{x_1} + \underline{x_2}, \overline{x_1} + \overline{x_2}]$. Quand une fonction f est une composition de fonctions élémentaires, on doit définir une *extension* de f aux intervalles pour assurer un calcul de l'image conservatif.

Définition 1 (Extension d'une fonction à $\mathbb{IR}$; fonction d'inclusion)
Considérons une fonction $f : \mathbb{R}^n \to \mathbb{R}$.
$[f] : \mathbb{IR}^n \to \mathbb{IR}$ est une **extension** *de f aux intervalles ssi :*

$$\begin{array}{ll} \forall [x] \in \mathbb{IR}^n & [f]([x]) \supseteq \{f(x),\ x \in [x]\} \\ \forall x \ \in \mathbb{R}^n & f(x) = [f]([x,x]) \end{array}$$

L'**extension naturelle** $[f]_N$ d'une fonction sur les réels f fait la correspondance de f aux intervalles en utilisant l'arithmétique d'intervalles. Les linéarisations *intérieures* proposées dans cet article sont liées à l'**extension de Taylor sur intervalles** [16], définie comme suit :

$$[f]_T([x]) = f(\tilde{x}) + \sum_i [a_i] \,.\, ([x_i] - \tilde{x_i})$$

où $\tilde{x}$ désigne un point quelconque de $[x]$, comme $m([x])$; $[a_i]$ désigne $\left[\frac{\partial f}{\partial x_i}\right]_N([x])$.

Exemple Etudions $f(x_1, x_2) = 3{x_1}^2 + {x_2}^2 + x_1 x_2$ dans la boîte $[x] = [-1,3] \times [-1,5]$. L'évaluation naturelle donne : $[f]_N([x_1],[x_2]) = 3[-1,3]^2 + [-1,5]^2 + [-1,3][-1,5] = [0,27] + [0,25] + [-5,15] = [-5,67]$. Les dérivées partielles sont : $\frac{\partial f}{\partial x_1}(x_1,x_2) = 6x_1 + x_2$, $[\frac{\partial f}{\partial x_1}]_N([-1,3],[-1,5]) = [-7,23]$, $\frac{\partial f}{\partial x_2}(x_1,x_2) = x_1 + 2x_2$, $[\frac{\partial f}{\partial x_2}]_N([x_1],[x_2]) = [-3,13]$. L'évaluation de Taylor sur intervalles avec $\tilde{x} = m([x]) = (1,2)$ donne : $[f]_T([x_1],[x_2]) = 9 + [-7,23][-2,2] + [-3,13][-3,3] = [-76,94]$.

Problème traité

Un problème d'optimisation globale continue sous contraintes est défini de la manière suivante :

Définition 2 (Optimisation globale sous contraintes)
Soit $x = (x_1, ..., x_i, ...x_n)$ un vecteur de variables de domaine $[x]$, une fonction $f : \mathbb{R}^n \to \mathbb{R}$, des fonctions $g : \mathbb{R}^n \to \mathbb{R}^m$ et $h : \mathbb{R}^n \to \mathbb{R}^p$.
Etant donné le système $S = (f, g, h, [x])$, le problème d'optimisation globale sous contraintes consiste à trouver :

$$\min_{x \in [x]} f(x) \ \ \textit{soumis à} \ \ g(x) \leq 0 \wedge h(x) = 0.$$

f est appelée **fonction objectif** *; g et h sont des* **contraintes** *d'inégalité et d'égalité respectivement. x est dit* **réalisable** *s'il satisfait les contraintes.*

Notre optimiseur à intervalles extrait des *boîtes intérieures* et des *polytopes intérieurs* à partir de boîtes (extérieures) classiques.

Définition 3 *Soit* $(f, g, [x])$ *un système comprenant seulement des contraintes d'inégalité. Une* **région intérieure** r^{in} *est un sous-ensemble de* $[x]$ *entièrement réalisable, c.-à-d.,* $r^{in} \subset [x]$ *et tous les points* $x \in r^{in}$ *vérifient* $g(x) \leq 0$.

Seuls quelques résolveurs traitent rigoureusement l'optimisation globale telle que définie ci-dessus. Des optimiseurs globaux utilisant un B&B à intervalles, comme GlobSol [10] et Icos [12], renvoient une petite boîte garantie de contenir un vecteur réel x ϵ-minimisant : $f(x)$ t.q. $g(x) \leq 0 \wedge h(x) = 0$.[2]

Inspirés par le résolveur IBBA [18], nous proposons de relâcher les égalités $h_j(x) = 0$ par des équations "épaisses" $h_j(x) \in [-\epsilon_{eq}, +\epsilon_{eq}]$, soit 2 inégalités : $-\epsilon_{eq} \leq h_j(x) \leq +\epsilon_{eq}$. Ainsi, notre optimiseur global `IbexOpt` [23] calcule un vecteur flottant x ϵ-minimisant :

$$f(x) \ s.t. \ g(x) \leq 0 \wedge (-\epsilon_{eq} \leq h(x) \leq +\epsilon_{eq}).$$

Soulignons que `IBBA` et `IbexOpt` peuvent seulement garantir l'optimum global du système relâché, quoique ϵ_{eq} puisse souvent être choisi arbitrairement petit. Notons aussi que la plupart des optimiseurs globaux déterministes (généralement basés sur un B&B), comme Baron [22], ne peuvent garantir la solution, ni du problème d'optimisation globale (Définition 2), ni du problème relâché. Notons enfin qu'en pratique la plupart des équations sont initialement "épaisses" et ne nécessitent donc pas d'être encore plus relaxées avec un ϵ_{eq}. Les équations modélisant les systèmes physiques ont en effet souvent au moins un coefficient pouvant être représenté par une constante intervalle. Ce paramètre correspond à une incertitude bornée, comme une imprécision sur une mesure, ou une constante irrationnelle, comme π (approximée par un intervalle comme $[3.14159265, 3.14159266]$).

Par souci de simplicité, puisque notre optimiseur accepte des inégalités et des égalités relaxées qui sont aussi des inégalités, nous considérons par la suite le problème d'optimisation globale suivant :

$$\min_{x \in [x]} f(x) \ \ s.c. \ \ g(x) \leq 0,$$

c'est-à-dire que les fonctions $h(x) - \epsilon_{eq}$ (≤ 0) et $-h(x) - \epsilon_{eq}$ (≤ 0) appartiennent au vecteur de fonctions $g : \mathbb{R}^n \to \mathbb{R}^m$.

De nombreux opérateurs mathématiques convexes et non-convexes peuvent être pris en compte, comme $+, \cdot, /, x^n, exp, log, sqrt, sinus$, etc.

[2] ϵ-minimiser $f(x)$ signifie minimiser $f(x)$ avec une précision ϵ, i.e., trouver x tel que pour tout y, on a $f(y) \geq f(x) - \epsilon$.

3 Algorithme du polytope intérieur

L'idée principale est de construire un demi-espace pour chaque inégalité $g_j(x) \leq 0$, tel que *tous* les points de ce demi-espace satisfassent la contrainte. Pour cela, nous approximons une fonction non linéaire par une formule de Taylor au premier ordre sur intervalles. La formule usuelle, définie dans la partie 2, peut sélectionner n'importe quel point d'expansion $\tilde{x}$ dans la boîte pour réaliser la linéarisation. Au lieu du point milieu, nous choisissons un *coin* de la boîte, ici $\underline{x}$. Considérons une fonction $g_j : \mathbb{R}^n \to \mathbb{R}$ et un domaine $[x]$. Pour toute variable $x_i \in x$, soit $[a_i^j] = \left[\frac{\partial g_j}{\partial x_i}\right]_N([x])$. L'idée est de borner $g_j(x)$ par au-dessus avec une fonction affine $g_j^l(x)$ obtenue par une formule de Taylor sur un coin. Pour tout vecteur réel $x \in [x]$, nous avons :

$$g_j(x) \leq g_j^l(x) = g_j(\underline{x}) + \sum_i \overline{a_i^j} \cdot (x_i - \underline{x_i}). \qquad (1)$$

Si nous considérons une inégalité $g_j(x) \leq 0$, la relation (1) nous permet de construire un hyperplan $g_j^l(x) = 0$ bornant l'espace réalisable par au-dessus : $g_j(x) \leq g_j^l(x) \leq 0$. La fonction linéaire $g_j^l(x)$ peut donc être utilisée pour définir une région intérieure de $[x]$.

Proposition 1 *La linéarisation sur intervalles (1) est correcte et rigoureuse, c.-à-d. qu'elle est robuste aux erreurs dues aux calculs sur les nombres à virgule flottante.*

La rigueur est assurée par Taylor sur intervalles [17]. La correction de la relation (1) tient au fait que chaque "variable" $(x_i - \underline{x_i})$ est positive puisque son domaine est $[0, d_i]$, avec $d_i = w([x_i]) = \overline{x_i} - \underline{x_i}$. Donc, le maximum de chaque terme $[a_i^j].(x_i - \underline{x_i})$ pour tout point $(x_i - \underline{x_i}) \in [0, d_i]$ est obtenu avec $\overline{a_i^j}$.

Exemple. Considérons la contrainte

$$g_1(x_1, x_2) = x_1^3 + cos(x_1) - sin(x_2) - 0.15 \leq 0$$

dans une boîte $[x_1] \times [x_2] = [-0.32, 0.52] \times [0.90, 1.06]$. Le gradient de g_1 est :

- $\frac{\partial g_1}{\partial x_1} = 3.[-0.32, 0.52]^2 - sin([-0.32, 0.52]) = 3[0, 0.2704] - [-0.314, 0.497] = [-0.497, 1.1252]$
- $\frac{\partial g_1}{\partial x_2} = -cos([0.90, 1.06]) = [-0.6216, -0.4888]$

La fonction g_1 est illustrée par la figure 1. L'espace réalisable apparaît en gris.

Appliquer (1) à g_1 donne : $g_1(-0.32, 0.90) + 1.1252\,(x1 + 0.32) - 0.4889\,(x2 - 0.90) \leq 0$. Le demi-espace intérieur correspondant est en gris foncé sur la partie gauche de la figure. La partie droite montre un espace calculé par une procédure de Taylor sur intervalles standard avec pour point d'expansion le milieu de la boîte :

$g_1(0.10, 0.98) + [-0.497, 1.1252]\,(x1 - 0.10) + [-0.6216, -0.4889]\,(x2 - 0.98)$ Cette formule donne quatre hyperplans sur les quatre quadrants (séparés par des pointillés sur la figure) :

1. $0.0155067948 - 0.497\,(x1 - 0.1) - 0.6216\,(x_2 - 0.98) \leq 0$ ($x_1 < m[x_1]$ et $x_2 < m[x2]$)
2. $0.0155067948 - 0.497\,(x1 - 0.1) - 0.4889\,(x_2 - 0.98) \leq 0$ ($x_1 < m[x_1]$ et $x_2 > m[x2]$)
3. $0.0155067948 + 1.1252\,(x1 - 0.1) - 0.4889\,(x_2 - 0.98) \leq 0$ ($x_1 > m[x_1]$ et $x_2 > m[x2]$)
4. $0.0155067948 + 1.1252\,(x1 - 0.0) - 0.6216\,(x2 - 0.98) \leq 0$ ($x_1 > m[x_1]$ et $x_2 < m[x2]$)

On remarque que seulement les hyperplans 2 et 3 apparaissent sur la figure ; les autres ne coupent pas la boîte.

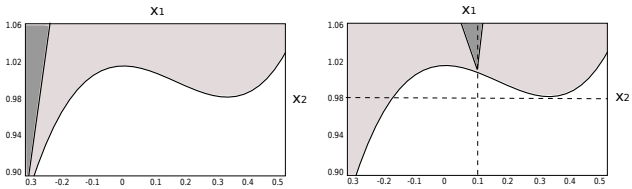

FIG. 1 – A gauche : demi-espace intérieur créé par une formule de Taylor par intervalles sur un coin $(\underline{x_1}, \underline{x_2})$. A droite : deux demi-espaces intérieurs créés à partir du point milieu dans deux des quatre quadrants.

Un programme linéaire pour améliorer le majorant

En appliquant cette idée à la fonction objectif $f(x)$ et aux inégalités $g_j(x) \leq 0$, on peut construire le programme linéaire LP^{ub} :

$$\begin{array}{rl} LP^{ub} = \min & f(\underline{x}) + \sum_i \overline{a_i} \bullet (x_i - \underline{x_i}) \\ \text{soumis à :} & \forall j \;\; g_j(\underline{x}) + \sum_i \overline{a_i^j} \bullet (x_i - \underline{x_i}) \leq 0 \\ & \forall i \;\; \underline{x_i} \leq x_i \wedge x_i \leq \overline{x_i} \end{array}$$

Un algorithme du simplexe résout LP^{ub} et renvoie l'infaisabilité ou une solution optimale x^l (cf. l'algorithme 1). L'infaisibilité ne prouve rien car le système linéarisé est plus contraint que l'original, qui pourrait avoir des solutions. Si l'algorithme du simplexe calcule une solution x^l optimale de l'approximation intérieure, alors x^l est aussi une solution du système original, mais peut-être pas l'optimale. Le processus précédent est correct sur les nombres réels, mais n'est pas nécessairement correct sur un ordinateur à cause des erreurs d'arrondi. En effet, nous utilisons un algorithme du simplexe standard sur les flottants et il est possible que le meilleur point retourné par le résolveur de programmation linéaire soit légèrement

Algorithm 1 `InnerPolytopeUB` (**in :** S, $[x]^{out}$; **in-out :** ub, x_{ub})

$LP^{ub} \leftarrow$ `InnerLinearization` $(S, [x]^{out})$
$x^l \leftarrow$ `Simplex`(LP^{ub})
if $x^l \neq \perp$ **and** `FeasibilityCheck` (x^l, S) **then**
  $cost \leftarrow \overline{[f]_N([x^l, x^l])}$
  if $cost < ub$ **then** $ub \leftarrow cost$; $x_{ub} \leftarrow x^l$ **end if**
end if

en dehors du polytope intérieur. C'est pourquoi nous rendons le processus rigoureux en testant le caractère réalisable de x avec une évaluation par intervalles (voir la ligne 3 de l'algorithme 1).

Le pseudo-code montre finalement que l'on évalue la fonction objectif (l'originale, pas la linéarisée) au point x^l ce qui peut baisser le majorant. Dans ce cas, on met à jour le point x_{ub} et son coût, le nouveau majorant ub.

Travaux connexes, discussion

La formule de Taylor sur intervalles a souvent été utilisée pour produire une approximation linéaire de l'espace réalisable ou de la fonction objectif. Cependant, quand le point d'expansion est choisi strictement à l'intérieur du domaine, le système obtenu n'est pas convexe. Il est formé d'une intersection de sous-espaces non convexes [10]. Contracter de manière optimale une boîte contenant cette relaxation non convexe a été prouvé comme étant NP-difficile [11]. Cela explique pourquoi la communauté d'analyse par intervalles a beaucoup étudié ce problème pendant des décennies [19, 5].

Plusieurs chercheurs ont proposé de choisir comme point d'expansion de la formule de Taylor un coin de la boîte courante, au lieu du point milieu [2, 13, 14, 17]. Le principal inconvénient est de construire une relaxation de plus grand volume, l'avantage principal étant que cette approximation est formée d'un unique quadrant et est un polytope convexe.

La formulation duale que nous avons définie dans (1) n'a jamais été utilisée pour extraire un polytope intérieur et améliorer le majorant en optimisation globale sous contraintes.

4 L'algorithme InHC4

La boucle principale d'`InHC4` suit le schéma général proposé dans [3, 8]. Elle consiste à traiter chaque contrainte à tour de rôle et à intersecter à la volée les différentes boîtes produites.[3] Plus précisément :

[3] Ce schéma diffère cependant d'une propagation du type `HC4`, où une même contrainte peut être traitée plusieurs fois.

– Le traitement d'une contrainte $g_j(x) \leq 0$ prend une boîte en entrée et construit à l'intérieur de celle-ci une sous-boîte qui est intérieure pour cette seule contrainte.
– La boîte en entrée pour la $j^{\text{ème}}$ contrainte est la boîte intérieure produite par le traitement de la contrainte précédente $g_{j-1}(x) \leq 0$. Par récurrence, la boîte produite est donc intérieure pour les j premières contraintes.
– Le traitement de la première contrainte $g_1(x) \leq 0$ est effectué avec la boîte extérieure courante du B&B.

Ainsi, la boîte (éventuellement vide) produite par le traitement de la dernière contrainte est intérieure pour toutes les contraintes.

Le traitement d'une contrainte dans `InHC4` est en revanche radicalement différent des procédés de réfutation de [8, 3]. Notre procédure `InHC4-Revise` tente d'extraire une boîte intérieure pour chaque opérateur de la contrainte. De façon similaire à la procédure interne `HC4-Revise` (en abrégé, `HC4R`) de l'algorithme de propagation `HC4` [4, 15], notre procédure `InHC4-Revise` (en abrégé, `InHC4R`) est basée sur une représentation arborescente (et plus généralement, sous forme de DAG) de l'expression de la contrainte, comme illustré à la figure 2.

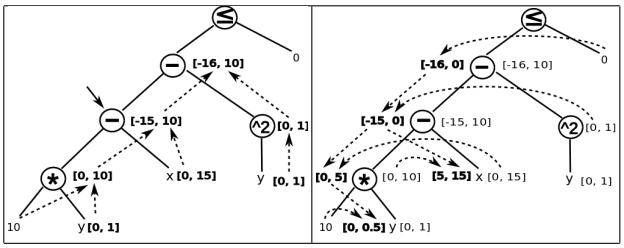

FIG. 2 – Représentation arborescente de la contrainte $10y - x - y^2 \leq 0$. **Gauche :** Phase montante (évaluation). **Droite :** Phase descendante (projection intérieure).

Notons $[x]$ la boîte initiale et $g_j(x) \leq 0$ la contrainte. Chaque nœud de l'arbre est associé à un intervalle, les intervalles associés aux feuilles étant les composantes de $[x]$. Les deux phases suivantes sont ensuite effectuées.

Phase montante d'évaluation (cf. fig. 2–gauche). L'arbre est parcouru des feuilles à la racine et les intervalles associés à chaque opérateur sont calculés avec l'arithmétique d'intervalles. Par exemple, l'intervalle du nœud marqué par une flèche est initialisé à $[0, 10] - [0, 15] = [-15, 10]$. A chaque nœud, l'intervalle est le résultat d'une évaluation naturelle sur intervalles de la sous-expression qu'il représente. Cette phase est la même que dans l'algorithme de propagation `HC4` [4, 15].

Phase descendante de projection (cf. fig. 2–droite). Des racines aux feuilles, les intervalles de chaque nœud sont contractés en utilisant des opérateurs spécifiques de *projection intérieure*.

Pour chaque nœud $(z = x_1 \; op \; x_2)$ correspondant à un opérateur binaire op et associé à un intervalle $[z]$, le rectangle formé par les intervalles des nœuds-fils x_1 et x_2 est contracté en une boîte $[x_1]^{in} \times [x_2]^{in}$ telle que :

$$\forall (x_1, x_2) \in [x_1]^{in} \times [x_2]^{in} : \quad x_1 \; op \; x_2 \in [z]. \qquad (2)$$

Si op est un opérateur unaire (c.-à-d., $z = op(x)$), son unique nœud-fils est contracté en un intervalle $[x]^{in}$, tel que :

$$\forall x \in [x]^{in} : \quad op(x) \in [z]. \qquad (3)$$

Si la projection intérieure produit une boîte vide (aucune boîte satisfaisant (2) ou (3) n'a pu être construite), alors la phase descendante est interrompue. Cela signifie qu'`InHC4R` n'a pu parvenir à construire une boîte intérieure pour $g_j(x) \leq 0$ donc, a fortiori pour les n inégalités. Dans ce cas, la boucle principale d'`InHC4` est interrompue.

Considérons par exemple la phase descendante appliquée à l'opérateur de multiplication de la figure 2–droite et ses deux nœuds-fils. Les intervalles contractés apparaissent en gras sur la gauche de chaque nœud (par ex., $[z]$ a été contracté à $[0, 5]$). Avant la contraction, les nœuds-fils ont pour intervalles associés $[10, 10]$ et $[0, 1]$. Ils sont alors réduits à $[10, 10]$ et $[0, 0.5]$ respectivement. Cette contraction est conforme à (2), c.-à-d., $\forall y \in [0, 0.5]$, $10 \bullet y \in [0, 5]$. Le paragraphe suivant détaille comment une telle opération élémentaire est effectuée par `InHC4R`.

4.1 Opérateurs élémentaires de projection intérieure

Nous allons décrire maintenant la façon dont les projections intérieures peuvent être implantées pour chaque opérateur élémentaire. Cette étude étend et simplifie l'étude au cas par cas faite dans le §3 de [6]. Nous proposons ici une projection générique basée uniquement sur des propriétés de monotonie et sur laquelle s'appuieront les projections spécifiques aux opérateurs.[4] Le premier cas est trivial mais il sert de point de départ pour comprendre les autres cas.

Cas 1 : opérateurs unaires monotones

Le premier cas est celui des opérateurs unaires monotones, comme log et exp. Plus précisément, il rassemble les opérateurs $z = op(x)$ continus et monotones pour x variant dans $[x]$.

[4] Par ailleurs, Chabert & Beldiceanu traitent en fait le problème dual consistant à trouver des boîtes *interdites* (c.-à-d., contenant uniquement des points qui violent au moins une contrainte). De plus, leur approche nécessite un point initial autour duquel les boîtes sont construites suivant le principe d'*inflation*.

Dans ce cas, une projection intérieure maximale (c.-à-d, non extensible, aux erreurs d'arrondi près) est évidente à calculer. Il suffit d'appliquer l'opérateur inverse de op (cf. fig. 3), comme le fait la projection extérieure dans HC4R. Cependant, pour prendre en compte rigoureusement les erreurs d'arrondi, le mode d'arrondi doit être *intérieur* (et non *extérieur*, comme dans HC4R).[5]

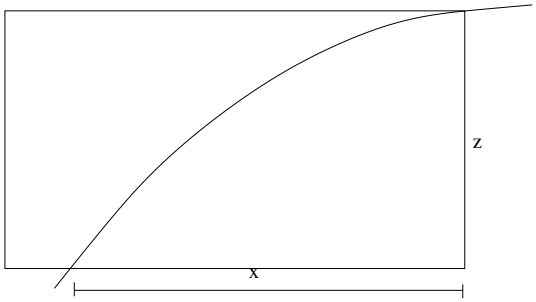

FIG. 3 – Cas 1 : Projection intérieure d'un opérateur monotone. Le segment horizontal en bas de figure est le résultat de la projection sur x.

Cas 2 : opérateurs unaires non-monotones

Le deuxième cas est celui des opérateurs $z = op(x)$ non-monotones pour $x \in [x]$, tels que x^2 ou $\sin$.

Ces opérateurs sont en fait tous monotones par morceaux. On peut donc calculer une projection intérieure pour chaque branche monotone et retourner aléatoirement l'un des intervalles formés. Bien sûr, lorsque des intervalles se touchent, on considère plutôt leur union.

Notons que pour les opérateurs unaires non-monotones, HC4R calcule l'enveloppe des intervalles obtenus pour chaque branche. Mais dans le cas d'une projection intérieure, un seul intervalle est conservé car les espaces entre les intervalles représentent des points qui ne sont pas intérieurs.

Cas 3 : opérateurs binaires monotones

Pour les opérateurs binaires (ou n-aire) monotones vis-à-vis de chacune de leurs variables, une procédure générique, appelée MonoMaxInnerBox, calcule aléatoirement une boîte intérieure maximale (si une telle boîte existe), suivant le principe représenté à la figure 5.

[5]Pour qu'un calcul soit conservatif en arithmétique flottante, c.-à-d., pour que l'image d'un intervalle contienne toutes les images réelles possibles, la borne inférieure (resp. supérieure) de l'image doit être arrondie vers $-\infty$ (resp. $+\infty$). Ces deux opérations consituent un arrondi *extérieur*.

Avec l'arrondi *intérieur*, les arrondis sont inversés : vers $+\infty$ et $-\infty$ respectivement. La propriété vérifiée par l'arrondi extérieur (resp. intérieur) pour un opérateur op est (1) et (2) respectivement :

1. $\forall x \in [x] \ \exists z \in op([x]) \ \ z = op(x)$
2. $\forall z \in op([x]) \ \exists x \in [x] \ \ z = op(x)$

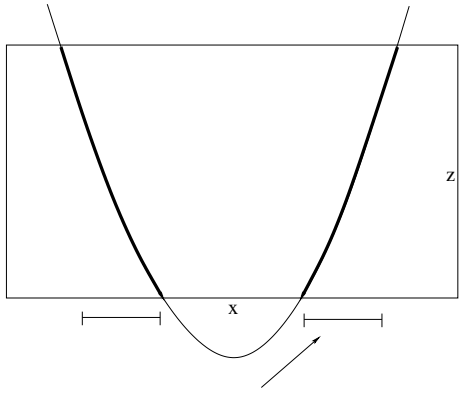

FIG. 4 – Cas 2 : projection intérieure d'un opérateur non-monotone : sqr. Deux intervalles (les segments horizontaux) sont obtenus en projetant sur x (c.-à-d., en utilisant l'inverse de sqr), et l'un d'entre eux est choisi aléatoirement.

Un opérateur binaire monotone revient à considérer deux inégalités $\underline{z} \leq (x_1 \ op \ x_2) \leq \overline{z}$. Chaque inégalité est traitée en séquence, la boîte intérieure calculée avec la première servant d'entrée pour la seconde. Comme décrit sur la fig. 5, il existe en général une infinité de boîtes maximales si bien que l'une d'entre elles est sélectionnée aléatoirement.

Cette procédure est utilisée, bien entendu, pour la projection intérieure de l'addition et de la soustraction. Elle est également utilisée pour les différents cas (monotones) des opérateurs non-monotones : la multiplication et la division, décrits ci-dessous.

Cas 4 : opérateurs binaires non-monotones

La fig. 6 illustre les deux cas à considérer pour la multiplication $x_1 \cdot x_2 \in [z]$, suivant que 0 appartienne ou non à $[z]$. Pour le cas $x_1 \cdot x_2 \in [z] \ni 0$, notons qu'une procédure directe (un peu plus complexe) pourrait être appliquée sans avoir recours à MonoMaxInnerBox.

Différentes implantations de la division sont possibles. Une approche consiste à réécrire $x_1/x_2 = x_1 \cdot \frac{1}{x_2} \in [z]$. IbexOpt propose aussi une procédure directe. Des tests (non exhaustifs) montrent que les deux versions présentent des performances similaires en pratique.

4.2 Amélioration du majorant avec InHC4

L'algorithme 2 montre comment InHC4 est utilisé par IbexOpt pour améliorer la borne supérieure.

Si une boîte intérieure $[x]^{in}$ est trouvée par InHC4, on applique une analyse de monotonie (MonotonicityAnalysis) sur la fonction objectif par rapport à chaque variable x_i. Si la dérivée partielle

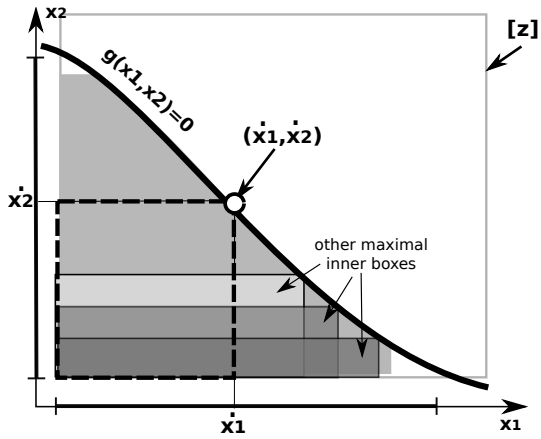

FIG. 5 – Cas 3, procédure `MonoMaxInnerBox` : La boîte en pointillés correspond à une boîte intérieure maximale de $[z]$ pour la contrainte monotone $g(x_1, x_2) \leq 0$. Un point $\dot{x_1}$ est tiré aléatoirement dans le segment horizontal des valeurs admissibles. Une seule valeur $\dot{x_2}$ peut alors rendre la boîte intérieure maximale.

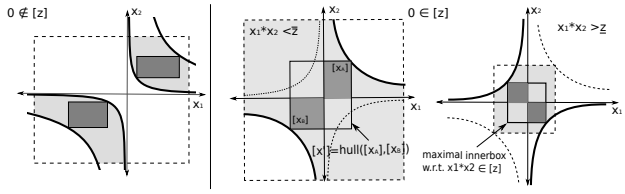

FIG. 6 – Projection intérieure pour la multiplication. **Gauche :** Deux boîtes maximales sont calculées indépendamment par `MonoMaxInnerBox` dans les deux régions intérieures disjointes (quadrants) définies par l'opérateur $x_1.x_2 \in [z] \geq 0$. **Milieu et droite :** Boîte maximale calculée pour $x_1.x_2 \in [z] \ni 0$ ($\overline{z} \geq -\underline{z}$) via deux appels à `MonoMaxInnerBox` (boîtes en gris).

$[a_i] = \left[\frac{\partial f}{\partial x_i}\right]_N([x]^{in}) \geq 0$ alors f est croissante par rapport à $x_i \in [x]$ si bien que $[x_i]$ est remplacé par l'intervalle dégénéré $[\underline{x_i}, \underline{x_i}]$ puisqu'on minimise $f(x)$ sur $[x]^{in}$. Si $[a_i] \leq 0$, f est décroissante et $[x_i]$ est remplacé par $[\overline{x_i}, \overline{x_i}]$.

Enfin, nous tirons aléatoirement un point x dans la boîte résultante [6], et remplaçons x_{ub} par x si x satisfait les contraintes et améliore le coût ub. Deux cas peuvent en fait se produire. Si une boîte intérieure a été extraite par `InHC4`, un point est sélectionné à l'intérieur de $[x]^{in}$. La satisfiabilité de x n'a plus besoin d'être testée puisque $[x]^{in}$ ne contient que des points admissibles. Si aucune boîte intérieure n'est disponible, un point aléatoire est tiré dans la boîte extérieure $[x]^{out}$ et les contraintes doivent cette fois être testées.

[6] Un échantillon constitué de plusieurs points (au lieu d'un seul) s'avère être contre-productif en pratique.

Algorithm 2 `Inhc4UB` (**in :** S, $[x]^{out}$; **in-out :** ub, x_{ub})

$[x]^{in} \leftarrow$ `InHC4` (S, $[x]^{out}$) /* Inner box extraction */
if $[x]^{in} \neq \emptyset$ **then**
  $[x]^{in} \leftarrow$ `MonotonicityAnalysis` (f, $[x]^{in}$)
  $x \leftarrow$ `RandomProbing`($[x]^{in}$) // or gradient descent
else
  $x \leftarrow$ `RandomProbing`($[x]^{out}$)
end if
$cost \leftarrow \overline{[f]_N([x,x])}$ /* Cost evaluation */
if $cost < ub$ **and** ($[x]^{in} \neq \emptyset$ **or** $\overline{[g]_N([x,x])} \leq 0$)
then
  $ub \leftarrow cost$; $x_{ub} \leftarrow x$
end if

Remplacer ce simple échantillonage par une recherche locale simple (de type descente de gradient) n'a pas amélioré notre stratégie. Une comparaison avec des algorithmes plus sophistiqués (quasi-Newton, etc) serait cependant nécessaire pour émettre un avis définitif sur l'intérêt de la recherche locale dans ce contexte.

5 Propriétés

On peut déduire quelques propriétés à partir des descriptions des deux sections précédentes. La première est négative.

Observation 1 *Les deux algorithmes* `InnerPolytope` *et* `InHC4` *sont corrects mais incomplets : ce sont des heuristiques qui peuvent parfois ne rien trouver alors qu'il existe une région intérieure.*

L'algorithme `InnerPolytope` est incomplet (même en négligeant les erreurs d'arrondis) car le polytope intérieur (intersecté avec la boîte) possède un volume plus petit que celui de l'espace réalisable non-convexe et peut ainsi parfois être vide.

L'algorithme `InHC4` est aussi incomplet (en négligeant les erreurs d'arrondis) car tous les cas de figure, sauf le premier, ignorent une partie de l'espace réalisable. Puisque en chaque nœud du parcours de l'arbre de haut en bas une partie de l'espace réalisable n'est pas extraite, le processus peut passer à côté d'une boîte intérieure non vide obtenue par composition des opérateurs de base.

Dans le cas 2, on sélectionne seulement un intervalle parmi ceux de l'union d'intervalles possibles. Dans le cas 3, une seule boîte intérieure maximale est calculée (en faisant un choix aléatoire sur $[x_1]$) parmi un choix infini de boîtes possibles. De plus, une boîte (seule) ne peut pas contenir ou définir une région intérieure. Le dernier cas rassemble les deux défauts (des cas 2 et 3). Considérons par exemple le cas de

la multiplication $x_1.x_2 \in [z]$ où z est positif (cf. fig. 6–gauche).

En revanche, la proposition suivante souligne un aspect intéressant de InHC4.

Proposition 2 *Soit g_j une fonction contenant au plus une occurrence de chaque variable.*

Si InHC4-Revise *réussit à extraire une boîte intérieure $[x]^{in}$ vis à vis de la contrainte $g_j(x) \leq 0$ dans le domaine $[x]$, alors $[x]^{in}$ est maximal (en négligeant les erreurs d'arrondi dus à l'arithmétique flottante), c'est-à-dire qu'il n'existe pas de boîte $[x]^{in'} \supset [x]^{in}$ qui soit une boîte intérieure de $[x]$ vis à vis de g_j.*

Démonstration (esquisse)

Nous pouvons démontrer que chaque opérateur unaire et binaire calcule une boîte intérieure maximale, en négligeant la perte entraînée par les arrondis intérieurs. La preuve est directe pour les cas 1 et 2. Dans le cas 3, on peut l'obtenir par construction à partir de la procédure MonoMaxInnerBox. La seule difficulté réside dans le cas $x_1 \,.\, x_2 \in [z]$ quand $0 \in [z]$.[7]

Dans le cas où une contrainte contient seulement une occurrence de chaque variable, InHC4R calcule ainsi une boîte intérieure maximale, quand une telle boîte existe. La condition de simple occurrence rejoint celle du HC4R standard [4]. Elle assure que l'expression forme une structure d'arbre et non un graphe acyclique orienté (une variable apparaissant plusieurs fois dans l'expression a plusieurs parents dans le graphe) qui ne préserve pas la propriété par récurrence. Autrement dit, la condition assure que, pendant le parcours de haut en bas de l'expression, la composition des boîtes intérieures produit des intervalles intérieurs maximaux sur toutes les dimensions. □

Observons cependant que la propriété de maximalité ne tient pas pour un *système* d'inégalités traité par InHC4.

Une deuxième proposition donne la complexité en temps en pire cas de nos algorithmes d'extraction de régions intérieures.

Proposition 3 *Soit $(f, g, [x])$ un système de n variables et m inégalités. Soit k le nombre maximum d'opérateurs unaires et binaires dans une fonction g_j. Soit t le temps maximum requis pour évaluer un opérateur mathématique primitif.*

La complexité en temps au pire cas de la procédure d'extraction InnerPolytope *est $O(m(k.t+n))$. La complexité en temps au pire de la procédure d'extraction* InHC4 *est $O(m.k.t)$.*

[7] L'implantation directe est plus facile à vérifier car un coin de la boîte cherche à maximiser la boîte dans une dimension tandis que l'autre coin maximise la boîte dans l'autre dimension...

Démonstration

Pour InnerPolytope, un hyperplan (ou une forme linéaire de la fonction objectif) se calcule en temps $O(k.t+n)$. En effet, calculer le gradient d'une fonction g_j s'obtient en temps $O(k.t+n)$ en utilisant la différenciation automatique ; évaluer $g_j(\underline{x})$ requiert un temps en $O(k.t)$; la somme des termes $\overline{a_i^j}.(x_i - \underline{x_i})$ est en $O(n)$ pour toutes les variables. Finalement, générer le programme linéaire LP^{ub} demande $m+1$ appels à la procédure précédente (pour les m contraintes plus la fonction objectif). Notons qu'il faut compter en plus la complexité d'un appel à un résolveur de PL dans celle de la procédure InnerPolytopeUB.

La procédure InHC4 traite au plus m contraintes une par une. La procédure InHC4R lance au plus k évaluations sur intervalles en $O(t)$ chacune pendant la phase d'évaluation (de bas en haut), et k fois un nombre constant (entre 1 et 4) d'appels à la procédure MonoMaxInnerBox pendant la phase de projection (de haut en bas). Pour le traitement d'un opérateur binaire, la complexité de MonoMaxInnerBox est en $O(t)$ car il effectue deux itérations (sur les deux variables), chacune étant dominée par une évaluation sur intervalles de l'opérateur primitif. □

Soulignons que contrairement à InnerPolytope, la complexité en pire cas de InHC4 n'est pas atteinte quand InHC4R échoue dans le traitement d'une contrainte, puisque la boucle sur toutes les contraintes est alors interrompue. C'est pourquoi moins InHC4 a de chance d'extraire une boîte intérieure (parce que la boîte initiale est grande comparée à l'espace réalisable), moins de contraintes sont traitées par InHC4 et la complexité en temps reste modérée en pratique.

6 Expérimentations

Nous avons implanté ces deux algorithmes d'extraction de régions intérieures dans IbexOpt [23], qui étend la bibliothèque en C++ Ibex (Interval Based EXplorer) [7] de résolution sur intervalles à l'optimisation globale.

A chaque nœud du B&B, IbexOpt est appelé avec nos meilleurs opérateurs de réduction de l'espace de recherche et d'amélioration du minorant de l'objctif :

- L'opérateur ACID(Mohc) est une version adaptative de CID [24] utilisant Mohc [1] comme contracteur de base. Mohc exploite la monotonie des contraintes pour mieux contracter la boîte courante.
- L'opérateur X-Newton utilise la forme duale de (1) pour contracter l'espace de recherche et améliorer le minorant [2].

La plupart des problèmes sont résolus en utilisant comme heuristique de bissection une variante

de la fonction *Smear* de Kearfott décrite dans [23]. Quelques problèmes, indiqués avec `rr` dans le tableau 1 ont utilisé l'heuristique du tour de rôle.

Pour trouver un majorant, `IbexOpt` appelle les procédures `InnerPolytopeUB` et `InHC4UB` à chaque nœud du B&B.

Nous avons testé ces procédures sur un ensemble d'instances provenant de la série 1 du banc d'essai d'optimisation globale Coconut. Les équations $h_k(x) = 0$ sont remplacées par les inégalités $-\epsilon_{eq} \leq h_k(x) \leq \epsilon_{eq}$, avec $\epsilon_{eq} =$ `1.e-8`. Nous avons sélectionné les 59 systèmes résolus avec un temps de calcul allant de 1 seconde à 1 heure par `IbexOpt` ou `IbexOpt0` sur un ordinateur standard avec un processeur Pentium à 3 GHz. `IbexOpt0` désigne une version précédente de notre optimiseur global où la recherche du majorant était effectuée en tirant un point au hasard dans la boîte (extérieure) courante à chaque nœud du B&B et en testant les contraintes sur ce point (comme l'optimiseur global IBBA [18] qui choisit le milieu de la boîte). Le tableau 1 donne le nom et le nombre de variables de chaque système testé.

nom	n	nom	n	nom	n
alkyl (rr)	14	ex6_1_4	6	ex9_2_6	16
bearing	14	ex6_2_6	3	ex14_1_2	6
ex2_1_3	13	ex6_2_8	3	ex14_1_6	9
ex2_1_5	10	ex6_2_9	4	ex14_1_7	10
ex2_1_6	10	ex6_2_10	6	ex14_2_1	5
ex2_1_7	20	ex6_2_11	3	ex14_2_3	6
ex2_1_8	24	ex6_2_12	4	ex14_2_4	6
ex2_1_9	10	ex6_2_14	4	ex14_2_6	5
ex2_1_10	20	ex7_2_1	7	ex14_2_7	6
ex3_1_1	8	ex7_2_3	9	haverly	12
ex3_1_3	6	ex7_2_7 (rr)	4	hhfair	28
ex5_2_2_c1	9	ex7_2_8 (rr)	8	himmel11	9
ex5_2_2_c2	9	ex7_2_9 (rr)	10	himmel16	18
ex5_2_2_c3	9	ex7_3_4	12	house	8
ex5_2_4	7	ex7_3_5	13	hydro	30
ex5_3_2	22	ex8_1_8	6	immun (rr)	21
ex5_4_2	8	ex8_5_1	6	launch	38
ex5_4_3	16	ex8_5_2	6	meanvar	7
ex6_1_1	8	ex8_5_3	5	process	10
ex6_1_3	12	ex8_5_6	6		

TAB. 1 – Ensemble des 59 systèmes sélectionnés dans la 1ère série du banc d'essais Coconut.

La figure 7 et le tableau 2 montrent une comparaison des performances entre `IbexOpt` et `IbexOpt0`. IN représente `IbexOpt` et RAND `IbexOpt0`. `IP` représente une variante de l'optimiseur appelant uniquement `InnerPolytopeUB` à chaque nœud et INHC4 une variante n'appelant que `InHC4UB` à chaque nœud.

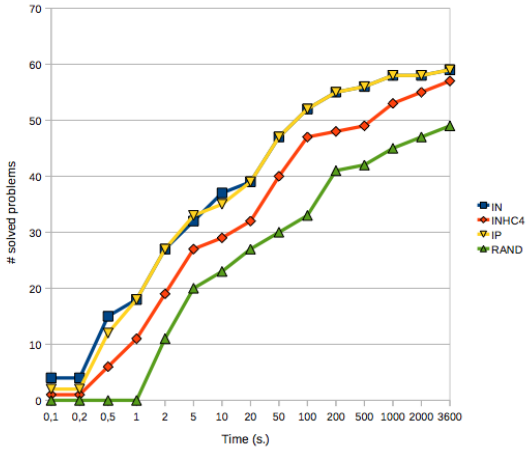

FIG. 7 – Profil de performance. Un point sur une courbe indique le nombre de systèmes résolus dans le temps CPU en abscisse par l'optimiseur correspondant.

Gain	IN	InHC4	IP
< 0.69	0	0	0
0.69 – 0.9	1	4	2
0.9 – 1	2	2	2
1 – 1.1	0	11	0
1.1 – 2	14	13	13
2 – 10	19	15	22
10 – 100	11	3	8
> 100	3	2	3
Explosion mémoire avec `IbexOpt0`	9	7	9
Résolus dans le temps imparti	59	57	59

TAB. 2 – Gains obtenus par les différents optimiseurs X (IN, InHC4, IP) par rapport à `IbexOpt0`. Le gain est défini par $\frac{time(IbexOpt0)}{time(X)}$. Chaque ligne (intervalle de gains), indique le nombre de problèmes obtenant ce gain pour les différents optimiseurs. L'avant dernière ligne indique le nombre de systèmes résolus par X mais qui explosent en mémoire avec `IbexOpt0`.

Résultats

Ces expérimentations montrent les apports significatifs de notre recherche de majorant, par rapport au test des contraintes sur un point de la boîte extérieure courante. Une perte de performance de 31% n'a été observée que sur une seule instance alors qu'un gain d'un facteur au moins 2 a été obtenu sur 33 systèmes. De plus, `IbexOpt0` n'a pas pu résoudre 10 des 59 systèmes : il a atteint la limite de temps pour 1 système et a causé une explosion mémoire pour 9 systèmes.

InnerPolytope apparaît plus utile que `InHC4`, mais l'ajout des 2 algorithmes rend le B&B plus robuste et donne de meilleures performances que l'utilisation de `IP` et `InHC4` seuls.

Etude qualitative

Plusieurs analyses qualitatives ont été menées pour mieux comprendre les tendances générales de nos algorithmes.

Nous avons d'abord cherché à déterminer quelle procédure d'extraction de région intérieure était la plus utile pour chaque système. Pour cela, nous avons compté le nombre de fois où `InHC4UB` et `InnerPolytopeUB` amélioraient le majorant. Aucune conclusion générale ne s'est dégagée, le résultat dépendant de chaque instance. Nous avons ensuite mesuré la taille moyenne des boîtes extérieures pour lesquelles les algorithmes arrivaient à extraire une région intérieure. Il est apparu qu'`InHC4UB` améliorait souvent le majorant dans des boîtes plus grandes qu'`InnerPolytopeUB`. Ceci confirme le fait qu'une forme de Taylor donne une meilleure approximation d'une fonction non convexe dans des petites boîtes. Nous avons aussi observé une plus grande variabilité dans le temps de résolution quand `IbexOpt` est appelé avec seulement `InHC4UB` plutôt qu'avec `InnerPolytopeUB` seul. Ceci est sans doute dû aux choix aléatoires faits dans les cas 2, 3 et 4 de `MonoMaxInnerBox`.

En conclusion, la recherche d'un majorant dans des régions intérieures semble prometteuse. La prochaine étape consistera à étudier si cette approche peut être étendue à de plus gros systèmes et, dans le cas contraire, comment l'améliorer.

Références

[1] I. Araya, G. Trombettoni, and B. Neveu. Exploiting Monotonicity in Interval Constraint Propagation. In *Proc. AAAI*, pages 9–14, 2010.

[2] I. Araya, G. Trombettoni, and B. Neveu. A Contractor Based on Convex Interval Taylor. In *CPAIOR*, pages 1–16. LNCS 7298, 2012.

[3] F. Benhamou and F. Goualard. Universally Quantified Interval Constraints. In *Proc. CP, LNCS 1894*, pages 67–82, 2004.

[4] F. Benhamou, F. Goualard, L. Granvilliers, and J.-F. Puget. Revising Hull and Box Consistency. In *Proc. ICLP*, pages 230–244, 1999.

[5] C. Bliek. *Computer Methods for Design Automation*. PhD thesis, MIT, 1992.

[6] G. Chabert and N. Beldiceanu. Sweeping with Continuous Domains. In *Proc. CP, LNCS 6308*, pages 137–151, 2010.

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

[8] H. Collavizza, F. Delobel, and M. Rueher. Extending Consistent Domains of Numeric CSP. In *Proc. IJCAI*, pages 406–413, 1999.

[9] A. Goldsztejn. *Définition et Applications des Extensions des Fonctions Réelles aux Intervalles Généralisés : Nouvelle Formulation de la Théorie des Intervalles Modaux et Nouveaux Résultats*. PhD thesis, University of Nice Sophia Antipolis, 2005.

[10] R. B. Kearfott. *Rigorous Global Search : Continuous Problems*. Kluwer Academic Publishers, 1996.

[11] V. Kreinovich, A.V. Lakeyev, J. Rohn, and P.T. Kahl. *Computational Complexity and Feasibility of Data Processing and Interval Computations*. Kluwer, 1997.

[12] Y. Lebbah, C. Michel, M. Rueher, D. Daney, and J.P. Merlet. Efficient and safe global constraints for handling numerical constraint systems. *SIAM Journal on Numerical Analysis*, 42(5) :2076–2097, 2005.

[13] Y. Lin and M. Stadtherr. LP Strategy for the Interval-Newton Method in Deterministic Global Optimization. *Industrial & engineering chemistry research*, 43 :3741–3749, 2004.

[14] D. McAllester, P. Van Hentenryck, and D. Kapur. Three Cuts for Accelerated Interval Propagation. Technical Report AI Memo 1542, MIT, 1995.

[15] F. Messine. *Méthodes d'Optimisation Globale basées sur l'Analyse d'Intervalle pour la Résolution des Problèmes avec Contraintes*. PhD thesis, LIMA-IRIT-ENSEEIHT-INPT, Toulouse, 1997.

[16] R.E. Moore. *Interval Analysis*. Prentice-Hall, 1966.

[17] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge Univ. Press, 1990.

[18] J. Ninin, F. Messine, and P. Hansen. A Reliable Affine Relaxation Method for Global Optimization. Technical Report RT-APO-10-05, IRIT, 2010.

[19] W. Oettli. On the Solution Set of a Linear System with Inaccurate Coefficients. *SIAM J. Numerical Analysis*, 2(1) :115–118, 1965.

[20] J. Rohn. Inner Solutions of Linear Interval Systems. In *Proc. Interval Mathematics 1985, LNCS 212*, pages 157–158, 1986.

[21] S.P. Shary. Solving the Linear Interval Tolerance Problem. *Mathematics and Computers in Simulation*, 39 :53–85, 1995.

[22] M. Tawarmalani and N. V. Sahinidis. A Polyhedral Branch-and-Cut Approach to Global Optimization. *Mathematical Programming*, 103(2) :225–249, 2005.

[23] G. Trombettoni, I. Araya, B. Neveu, and G. Chabert. Inner Regions and Interval Linearizations for Global Optim. In *AAAI*, pages 99–104, 2011.

[24] G. Trombettoni and G. Chabert. Constructive Interval Disjunction. In *Proc. CP, LNCS 4741*, pages 635–650, 2007.