`IbexOpt`, un Branch & Bound à intervalles pour l'optimisation non linéaire sous contraintes : dix ans après

Gilles Chabert[1], Bertrand Neveu[2], Ignacio Araya[3], Gilles Trombettoni[4], Jordan Ninin[5]

[1] IRT, LS2N, Nantes
[2] LIGM, Ecole des Ponts, Marne-la-Vallée
[3] U. Catolica, Valparaiso
[4] LIRMM, U. de Montpellier, CNRS
[5] LabSTICC, ENSTA Bretagne

Mots-clés : *optimisation non linéaire, optimisation globale exacte, contraintes, intervalles*

1 Introduction

`IbexOpt` est un outil d'optimisation non linéaire sous contraintes utilisant des opérateurs algorithmiques à intervalles rigoureux, dont la première version est sortie en 2011 [Trombettoni et al. 11] [1]. Les méthodes à intervalles confèrent à `IbexOpt` deux avantages principaux : d'abord, la garantie de la solution obtenue malgré les problèmes d'arrondi sur les nombres flottants ; ensuite, la possibilité de définir les contraintes et la fonction-objectif à base d'opérateurs mathématiques très variés incluant opérateurs arithmétiques, trigonométriques, exponentiel, logarithme, voire des opérateurs non dérivables comme la valeur absolue.

2 Problème résolu

`IbexOpt` traite le problème d'optimisation non linéaire continu suivant : $\min_{x\in[x]\subset\mathbb{R}^n} f(x)$ s.c. $g(x) \leq 0 \wedge h(x) = 0$, où $f : \mathbb{R}^n \to \mathbb{R}$ est la fonction-objectif, $g : \mathbb{R}^n \to \mathbb{R}^{m_1}$ et $h : \mathbb{R}^n \to \mathbb{R}^{m_2}$ définissent les contraintes d'inégalité et d'égalité resp. $x = \{x_1...x_n\}$ est un vecteur de variables à valeurs dans le domaine/boîte $[x]$.

Si le mode *rigoureux* est activé, une "petite" boîte $[x]^*$ est calculée contenant un point réel réalisable x^* ε-optimisant f. Sans le mode rigoureux, `IbexOpt` calcule un vecteur flottant x^* qui satisfait seulement une approximation $-\varepsilon_h \leq h(x) \leq +\varepsilon_h$ des équations.

3 Composants algorithmiques d'`IbexOpt`

Contrairement aux outils classiques de programmation mathématique non convexe, `IbexOpt` ne détecte pas la convexité du problème, n'utilise pas les dérivées secondes des fonctions traitées et ne fait pas appel à de l'optimisation locale pour la recherche de point réalisable.

`IbexOpt` applique une méthode complète de recherche arborescente par séparation - évaluation (*Branch & Bound*) utilisant des composants issus de la programmation par contraintes sur intervalles, de l'analyse par intervalles et de la programmation mathématique. A chaque itération, des opérateurs de contraction sont appliqués au système de contraintes pour réduire la boîte courante sans perte de solution. On recherche aussi un minorant et un meilleur point réalisable dans cette boîte réduite. L'algorithme termine quand on atteint la précision demandée sur l'objectif. Mentionnons les principaux composants algorithmiques utilisés dans la stratégie par défaut :

— Le B&B effectue un parcours des nœuds en meilleur d'abord, qui lance à chaque nœud une procédure appelée `FeasibleDiving` [Neveu et al. 16]. A partir de chaque nœud sélectionné (ayant le plus petit minorant de l'objectif), `FeasibleDiving` poursuit une

1. Les références citées se trouvent sur le site web d'Ibex : `http://www.ibex-lib.org/doc/reference.html`.

branche en profondeur en sélectionnant à chaque bissection le nœud avec le plus petit minorant (l'autre étant placé dans la liste des nœuds ouverts) jusqu'à parvenir à une feuille. L'arbre de recherche se poursuit alors avec le meilleur nœud ouvert.

— La stratégie de choix de domaine de variable à bissecter dans l'arbre de recherche étend la stratégie *Smear* qui a le défaut de considérer que toutes les contraintes ont le même impact sur les variables. La variante `lsmear` proposée pondère les contraintes par les multiplicateurs de Lagrange optimaux obtenus sur une approximation linéaire du problème [Araya et al. 18].

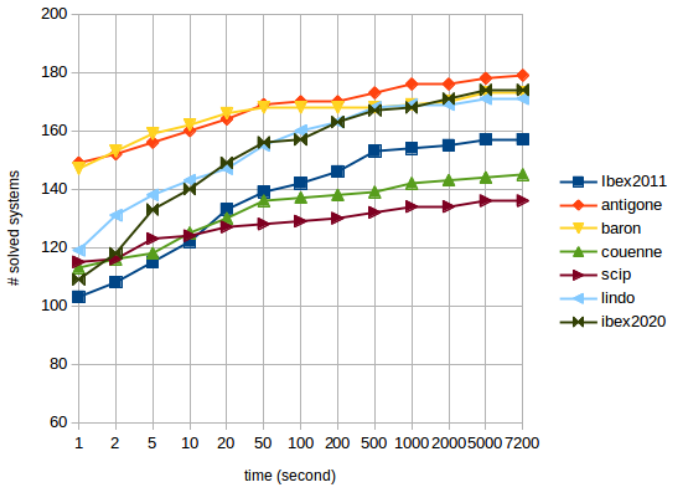
— Pour réduire le domaine des variables et améliorer le minorant, la stratégie de contraction par défaut est `ACID(HC4)` [Neveu et al. 15]. HC4 est l'algorithme de propagation de contraintes sur intervalles de l'état de l'art (cf. [Benhamou et al. 99] et thèse de F. Messine). ACID effectue un travail spécifique sur quelques variables du système choisies de manière adaptative. Pour chacune de ces variables, ACID partitionne leur domaine en sous-boîtes sur lesquelles il applique HC4 avant de renvoyer l'enveloppe des sous-boîtes contractées [Trombettoni, Chabert 07].

— L'autre méthode de contraction est basée sur une approximation convexe polyédrale du système de contraintes. Chaque inégalité non linéaire est approximée par trois demi-espaces : deux hyperplans sont calculés par X-Taylor, une variante de Taylor intervalles, dont le point d'expansion est pris sur un sommet de la boîte étudiée et sur le sommet opposé [Araya et al. 12]. Le troisième hyperplan est obtenu par un calcul basé sur l'arithmétique affine [Ninin, Messine 09]. La solution trouvée par l'algorithme du simplexe est corrigée par la méthode de Neumaier-Shcherbina pour rester rigoureux.

— La recherche d'un point réalisable est basée sur l'extraction de *régions intérieures*, c'est-à-dire des boîtes ou des polytopes entièrement réalisables [Araya et al. 14].

4 Comparaison avec les solveurs NLP

La figure montre un profil de performance portant sur 198 problèmes issus des séries 1 et 2 de la base Coconut (toutes les instances de moins 50 variables – et de plus de 6 variables pour la série 2 – et résolus par au moins un solveur en moins de 2h).



5 Aspects logiciels

`IbexOpt` fait partie de la bibliothèque à intervalles IBEX (cf. `ibex-lib.org`, [Chabert, Jaulin 09]). Cette bibliothèque en C++ sous licence LGPL sera bientôt disponible en tant que paquet Debian. Ses dépendances sont, pour le solveur linéaire, CLP, Soplex ou CPLEX ; pour l'arithmétique par intervalles, GAOL ou Filib.

Le format d'entrée des problèmes est un format propriétaire (minibex) ou AMPL (via NL). Nous envisageons d'offrir à moyen terme une interface Python de `IbexOpt`.