Utilisation des techniques de programmation par contraintes pour une implémentation rigoureuse et efficace de la réduction basée sur l'optimalité

Yahia Lebbah, Claude Michel, Michel Rueher

Utilisation des techniques de programmation par contraintes pour une implémentation rigoureuse et efficace de la réduction basée sur l'optimalité*

Yahia Lebbah[1,2] **Claude Michel**[1] **Michel Rueher**[1]

[1] Université de Nice Sophia Antipolis, I3S-CNRS - Projet CeP
Polytech'Nice-Sophia - Département Sciences Informatiques
930, Route des Colles - BP 145
06903 Sophia Antipolis Cedex

[2] Département d'Informatique
Université d'Oran Es-Senia
B.P. 1524 EL-M'Naouar
31000 Oran, Algeria

{ylebbah, cpjm, rueher}@polytech.unice.fr

Résumé

La réduction basé sur l'optimalité (ou RBO) est une technique qui a été proposée pour améliorer les algorithmes d'optimisation globale. Elle cherche à profiter des bornes connues du domaine de la fonction objectif pour tenter de réduire les bornes des domaines des variables et, ainsi, accélérer le processus de recherche d'un optimum global. Toutefois, l'algorithme de base de la RBO n'est pas sûr et peut donc rendre le processus de recherche de l'optimum global incomplet et l'empêcher d'atteindre le véritable optimum global. Récemment, Kearfott a proposé une implémentation sûre de la RBO. Malheureusement, son approche souffre de certaines limitations et est relativement lente. Dans cet article, nous montrons comment les techniques de filtrage des CSPs peuvent être avantageusement utilisées pour implémenter la RBO de manière sûre et efficace.

Abstract

Optimality-based reduction s a new technique which tries to improve the standard Branch and Bound algorithm. It attempts to take advantage of the known bounds of the objective function to reduce the domain of the variables, and thus to speed up the search of a global optimum. However, the basic algorithm is unsafe, and thus, the overall process may no longer be complete and may not reach the actual global optimum. Recently, Kearfott has proposed a safe implementation of optimality-based reduction. Unfortunately, his method suffers from some limitations and is rather slow. In this paper, we show how constraint programming filtering techniques can be used to implement optimality-based reduction in a safe and efficient way.

*Cet article a été publié à SAC'07 (track CSP).

1 Introduction

L'optimisation globale est utilisée dans de nombreuses applications allant de la conception de bras de robots [17] aux problèmes de contrôle [2] en passant par les problèmes d'équilibre chimique [9] ou de problèmes d'ordonnancement [2]. Plusieurs problèmes durs d'optimisation, comme le problème du voyageur de commerce ou les problèmes de conformation moléculaire, sont des problèmes d'optimisation globale. Dans ce papier, nous considérons le problème générique d'optimisation globale $\mathcal{P}$ qui consiste à minimiser une fonction objectif contrainte par un système d'équations et d'inéquations non-linéaires,

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) = 0,\ i = 1..k \\ & g_j(x) \leq 0,\ j = k+1..m \end{array} \quad (1)$$

où $x \in \mathbf{x}$, $f : \mathbb{R}^n \rightarrow \mathbb{R}$ et $g_j : \mathbb{R}^n \rightarrow \mathbb{R}$; Les fonctions f, g_i et g_j sont continuement différentiables sur un vecteur $\mathbf{x}$ d'intervalles de $\mathbb{R}$.

Les principales difficultés dans la résolution de ce type de problèmes d'optimisation globale proviennent de l'existence de multiples optimum locaux, dont seulement quelques uns sont des optimum globaux, mais aussi de la possible présence de discontinuités dans l'espace faisable [23]. C'est pourquoi les problèmes d'optimisation globale sont souvent difficiles à résoudre. Aussi, plusieurs techniques ont été proposées afin d'améliorer l'approche traditionnelle basée sur la méthode de résolution par séparation et évaluation. La réduction basée sur l'optimalité est probablement l'une de ces techniques les plus intéressantes.

Introduite par Ryoo et Sahinidis dans [26], la réduction basée sur l'optimalité (ou RBO) cherche à exploiter les bornes connues de la fonction objectif afin de réduire la taille des domaines des variables. Cette technique utilise une propriété bien connue du "point col" afin de calculer de nouvelles bornes du domaine d'une variable en fonction des bornes connues du domaine de la fonction objectif[1]. Toutefois, l'algorithme de base de la RBO n'est pas rigoureux et peut donc rendre le processus global de résolution incomplet et incapable d'atteindre effectivement l'optimum global.

Avant d'entrer dans les détails, illustrons le fonctionnement de la RBO à l'aide d'un petit exemple d'optimisation globale[2] :

$$\begin{array}{ll} \text{minimize} & -x_1 + x_1x_2 - x_2 \\ \text{subject to} & \\ & c_1 : -2x_1 + 2x_2 \leq 1 \\ & c_2 : 3x_1 - x_2 \leq 3; \\ & x_1, x_2 \in [0, 5] \end{array} \quad (2)$$

L'unique terme non linéaire, x_1x_2, est linéarisé grâce à la relaxation bilinéaire suivante :

$$\begin{array}{ll} \text{minimize} & -x_1 - x_2 + x_3 \\ \text{subject to} & \\ & c_1 : -2x_1 + 2x_2 \leq 1 \\ & c_2 : 3x_1 - x_2 \leq 3; \\ & c_3 : \overline{x}_2x_1 + \overline{x}_1x_2 - x_3 \leq \overline{x}_1\overline{x}_2 \\ & c_4 : \underline{x}_2x_1 + \underline{x}_1x_2 - x_3 \leq \underline{x}_1\underline{x}_2 \\ & x_1, x_2 \in [0, 5]; x_3 \in [0, 25] \end{array} \quad (3)$$

où x_3 est une variable de linéarisation qui représente le terme bilinéaire x_1x_2 ; $\overline{x}_i$ dénote la borne supérieure de la variable x_i alors que $\underline{x}_i$ dénote sa borne inférieure.

Pour le problème (3), l'algorithme du simplexe fournit la solution duale $\lambda_1^* = 0$ et $\lambda_2^* = 0.1666$, et $L = -1.08333$ comme borne inférieure. Le solveur local Minos [22] appliqué au problème initial (2) donne une borne supérieure $U = -1.005$ correspondant au point faisable $x^* = (0.917, 1.062)$.

La méthode de la RBO (cf. section 4.1) applique la relation $-2x_1 + 2x_2 - 1 \geq -(U - L)/\lambda_1^*$ à la contrainte active c_1 pour générer la relation $x_1 \leq 4.57$ et permet ainsi de retirer l'intervalle $[4.57, 5]$ du domaine de x_1. Le point critique est ici le calcul rigoureux de la solution duale de c_1. Récemment, Kearfott [13, 12] a proposé une implémentation rigoureuse de la RBO, implémentation basée sur un encadrement rigoureux de la solution duale. Malheureusement, cette méthode souffre de plusieurs restrictions et est lente.

Nous montrons dans ce papier que les techniques de programmation par contraintes peuvent être utilisées de manière simple pour réfuter les boites identifiées par la RBO et potentiellement exemptes de solution. Concrètement, les techniques de filtrage sont utilisées pour réduire ces boites à des ensembles vides, et donc, pour prouver qu'elles ne contiennent aucun point faisable. Cette approche basée sur les techniques de programmation par contraintes ne souffre pas des mêmes restrictions que la méthode de Kearfott. Les premières expérimentations montrent aussi qu'elle est plus performante.

Le reste de cet article est organisé comme suit. La première section rappelle les bases des techniques de filtrage sur le continu nécessaires à la compréhension du reste du papier. Nous y rappelons aussi les principes de la `Quad` qui joue un rôle essentiel dans notre schéma algorithmique. La section 2 introduit le schéma global d'un algorithme rigoureux de séparation/évaluation. La section suivante décrit la méthode de la RBO et introduit notre mise en oeuvre rigoureuse de la RBO basée sur les techniques de PPC. Finalement, les résultats de nos premières expérimentations seront décrits.

[1] Plus précisément, Ryoo et Sahinidis ont introduit dans [26] quatre tests qui reposent sur deux techniques différentes : la réduction basée sur l'optimalité et une autre technique dite de "probing". Alors que la première a montré de réelles capacités, la seconde technique appairait, selon leur auteurs, comme moins efficace pour réduire les bornes du domaine d'une variable.

[2] Ce problème est une légère modification (seule la contrainte c_1 diffère) d'un problème décrit dans [26].

2 Techniques de filtrage sur le continu

La première sous-section rappelle les notions de base sur les techniques de filtrage sur le continu. La seconde sous-section rappelle l'intuition sur laquelle repose les techniques de filtrage de la `Quad` qui sont utilisées dans notre algorithme. Une présentation plus détaillée de ces concepts et techniques est disponible dans [4, 14, 16].

2.1 Filtrages par $2B$-consistance et Box-consistance

La consistance d'arc joue un rôle clef dans la programmation par contraintes. Une contrainte c_j est arc-consistante [19] si pour chaque variable x_i de c_j, chaque valeur du domaine $\mathbf{x}_i$ de x_i a un support dans les domaines de toutes les autre variables de c_j. Toutefois, il n'est généralement pas possible d'atteindre la consistance d'arc pour les systèmes de contraintes sur les réels. C'est pourquoi, différentes relaxations de la consistance d'arc ont été proposées. Les relaxations les plus connues de la consistance d'arc utilisées sur le continu sont la $2B$–consistance et la box–consistance.

La $2B$–consistance (cf. [7, 27, 5, 18]) ne nécessite de vérifier la consistance d'arc que sur chacune des bornes de l'intervalle. De manière informelle, une variable x est $2B$–consistante pour la contrainte "$f(x, x_1, \ldots, x_n) = 0$" si la borne supérieure (resp. inférieure) du domaine de x est la plus petite (resp. la plus grande) solution de l'équation $f(x, x_1, \ldots, x_n) = 0$.

La $2B$–consistance définit une propriété locale sur les bornes du domaine d'une variable pour une seule contrainte. Une contrainte c est $2B$–consistante si, pour chaque variable x_i de c, il existe une valeur dans le domaine de toutes les autres variables qui satisfasse c lorsque x_i est fixée à $\underline{x}_i$ et à $\overline{x}_i$.

La box–consistance [4] est une relaxation plus forte (i.e., elle propose un filtrage moins strict) de la consistance d'arc que la $2B$–consistance. La variable x est box–consistante pour la contrainte "$f(x, x_1, \ldots, x_n) = 0$" si les bornes du domaine de x correspondent au zéro le plus à gauche et au zéro le plus à droite de l'extension optimale aux intervalles de $f(x, x_1, \ldots, x_n)$. Cependant, les algorithmes de $2B$–consistance effectuent un filtrage plus faible (i.e., un filtrage qui produit de plus grands intervalles) que les algorithmes de box–consistance lorsque au moins une des variables apparait plus d'une fois dans une même contrainte [8]. Ce comportement résulte de la nécessaire décomposition des contraintes à occurrences multiples effectuée par les algorithmes de $2B$–consistance. La box–consistance génère un système de fonctions d'intervalles univariées qui peut être résolu par des méthodes numériques telles que le Newton par intervalles. Á l'opposé de la $2B$–consistance, la box–consistance ne nécessite aucune décomposition de contrainte et n'amplifie donc pas le problème de localité.

2.2 Le filtrage de la `Quad`

L'algorithme de filtrage de la `Quad` [16] repose sur les techniques de relaxations linéaires. Un solveur linéaire est utilisé pour calculer finement les bornes des domaines des variables. Le processus de filtrage de la `Quad` consiste en trois étapes principales : la reformulation, la linéarisation et la réduction.

La reformulation génère un ensemble de contraintes. Plus précisément, cet ensemble contient des inégalités linéaires qui approximent la sémantique des termes non linéaires des contraintes initiales.

Le processus de linéarisation décompose d'abord chaque terme non linéaire en sommes et produits de termes univariés ; il remplace ensuite chaque terme non linéaire par une nouvelle variable associée. Les termes non linéaires sont approximés par de fines relaxations linéaires convexes.

L'étape de réduction correspond à un algorithme de point fixe qui appelle de manière itérative un solveur linéaire pour réduire les bornes supérieures et inférieures de chacune des variables du problème initial. Ce processus s'arrête lorsque la différence entre l'itération courante et l'itération précédente sur les bornes inférieures et supérieures est inférieure à un ϵ donné. L'algorithme converge et termine si ϵ est supérieur à zéro.

La rigueur des calculs est particulièrement importante dans un processus qui repose sur des relaxations linéaires. Grâce aux travaux de Shcherbina et Neumaier [24], il est possible de calculer une sous-estimation rigoureuse du minimum d'un programme linéaire. Cependant, ce minimum n'est rigoureux que si le problème linéaire est lui même une approximation rigoureuse du problème initial. Michel et al. [20] ont introduit des corrections des coefficients des relaxations linéaires qui garantissent que le programme linéaire est une approximation conservative de l'ensemble des solutions du problème initial. Ce travail a été étendu par Borradaile et Van Hentenryck [6] aux relaxations multivariées.

Ainsi, l'algorithme de filtrage de la `Quad` offre un cadre rigoureux et efficace pour réduire les domaines des variables qui permet d'effectuer des réductions plus efficaces que la $2B$–consistance ou la box–consistance. Par ailleurs, la `Quad` fourni tout les outils de base nécessaire à une implémentation efficace et rigoureuse d'un solveur d'optimisation globale basé sur les relaxations linéaires.

Rappelons maintenant notre algorithme de résolution par séparation et évaluation.

Algorithm 1 Algorithme de résolution par séparation et évaluation

Function BB(IN $\mathbf{x}$, ϵ; OUT $\mathcal{S}$, $[L,U]$)

% $\mathcal{S}$: set of proved feasible points
% $\mathbf{f}_{\mathbf{x}}$ denotes the set of possible values for f in $\mathbf{x}$
$\mathcal{L} \leftarrow \{\mathbf{x}\}$; $\mathcal{S} \leftarrow \emptyset$; $(L,U) \leftarrow (-\infty, +\infty)$;
while $w([L,U]) > \epsilon$ **do**
  $\mathbf{x}' \leftarrow \mathbf{x}''$ such that $\underline{\mathbf{f}}_{\mathbf{x}''} = min\{\underline{\mathbf{f}}_{\mathbf{x}''} : \mathbf{x}'' \in \mathcal{L}\}$; $\mathcal{L} \leftarrow \mathcal{L} \backslash \mathbf{x}'$;
  $\overline{\mathbf{f}}_{\mathbf{x}'} \leftarrow min(\overline{\mathbf{f}}_{\mathbf{x}'}, U)$;
  $\mathbf{x}' \leftarrow Prune(\mathbf{x}')$;
  $\underline{\mathbf{f}}_{\mathbf{x}'} \leftarrow LowerBound(\mathbf{x}')$;
  $(\overline{\mathbf{f}}_{\mathbf{x}'}, \mathbf{x}_p, Proved) \leftarrow UpperBox(\mathbf{x}')$;
  if $Proved$ **then** $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathbf{x}_p\}$; **endif**
  if $\mathbf{x}' \neq \emptyset$ **then**
    $(\mathbf{x}'_1, \mathbf{x}'_2) \leftarrow Split(\mathbf{x}')$; $\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathbf{x}'_1, \mathbf{x}'_2\}$;
  endif
  if $\mathcal{L} = \emptyset$ **then**
    $(L,U) \leftarrow (+\infty, -\infty)$;
  else
    $(L,U) \leftarrow (min\{\underline{\mathbf{f}}_{\mathbf{x}''} : \mathbf{x}'' \in \mathcal{L}\}, min\{\overline{\mathbf{f}}_{\mathbf{x}''} : \mathbf{x}'' \in \mathcal{S}\})$;
  endif
endwhile

3 Algorithme de résolution par séparation et évaluation

L'approche bien connue de résolution par séparation et évaluation fourni un cadre très général à la recherche d'un minimum global. C'est un processus complet qui constitue le cœur de l'implémentation d'une plateforme sûre et rigoureuse [15]. Nous décrivons ici cette dernière approche qui reste au centre de cet article.

L'algorithme de résolution par séparation et évaluation utilisé ici combine des techniques issues de l'analyse par intervalles avec des techniques issues de la programmation par contraintes au sein du schéma classique de résolution par séparation et évaluation décrit par Horst et Tuy dans [11]. Les techniques d'analyse par intervalles fournissent les outils qui garantissent la rigueur des calculs alors que les techniques de programmation par contraintes améliorent la réduction de l'espace faisable.

Le solveur (cf. algorithme 1) calcule un encadrement des minimiseurs globaux ainsi qu'un encadrement rigoureux de la valeur du minimum global à l'intérieur d'une boîte initiale $\mathbf{x}$. L'algorithme maintient deux listes : une liste $\mathcal{L}$ de boîtes à traiter et une liste $\mathcal{S}$ de boîtes contenant un point faisable dont l'existence a été prouvé. Il fourni un encadrement rigoureux $[L,U]$ de l'optimum global en fonction d'une tolérance donnée ϵ.

L'algorithme sélectionne la boîte $\mathbf{x}'$ qui contient la plus petite borne inférieure de la valeur de la fonction objectif. La fonction *Prune* réduit la taille de la boîte $\mathbf{x}'$ à l'aide des techniques de filtrage. Notre implémentation de la fonction *Prune* utilise un simple algorithme de filtrage par 2b–consistance. $LowerBound(\mathbf{x}')$ calcule ensuite rigoureusement une borne inférieure de la valeur de la fonction objectif au sein de la boîte $\mathbf{x}'$ en utilisant une relaxation linéaire du problème initial. De fait, la fonction *LowerBound* est basée sur les techniques de linéarisations de la `Quad`. Une valeur rigoureuse de $\underline{\mathbf{f}}_{\mathbf{x}'}$ est obtenue à l'aide d'un solveur de programmes linéaires.

$UpperBox(\mathbf{x})$ construit une boîte qui contient un point faisable dont l'existence est prouvée. Une méthode de recherche locale permet de trouver rapidement un point faisable. Les techniques d'analyse par intervalles sont utilisées pour vérifier l'existence d'un point faisable au sein de la boîte[3]. Lorsque *UpperBox* réussi à prouver l'existence d'un point faisable, la boîte $\mathbf{x}_p$ qui contient ce point faisable est ajoutée à la liste $\mathcal{S}$. À ce stade, si la boîte $\mathbf{x}'$ est vide alors, soit elle ne contient pas de point faisable, soit sa borne inférieure $\underline{\mathbf{f}}_{\mathbf{x}'}$ est plus grande que la la borne supérieure courante U. Dans ces deux cas, la boîte est dite "élaguée". Si $\mathbf{x}'$ n'est pas vide, la boîte est coupée en deux par rapport à l'une des variables du problème[4]. À chaque étape de sélection d'une boîte, l'algorithme calcule la plus petite borne inférieure L des boîtes restant à calculer et la plus petite borne supérieure U des boîtes contenant un point faisable. L'algorithme termine lorsque la distance entre U et L devient plus petite qu'une tolérance donnée ϵ. Un optimum prouvé ne peut évidement pas toujours être trouvé et, dans certains cas, l'algorithme 1 doit être stoppé afin de récupérer les boîtes contenant un point faisable.

La section suivante est consacrée à la réduction basée sur l'optimalité. Nous y rappelons d'abords les définitions de base avant de décrire l'approche proposée par Kearfott. Pour finir, nous introduisons notre algorithme rigoureux de calcul des réductions offertes par la RBO.

4 La réduction basée sur l'optimalité

4.1 Notions de base

La réduction basée sur l'optimalité a été introduite par Ryoo et Sahinidis dans [26]. Elle exploite une propriété du "point col" (cf., par exemple, le chapitre 5 de [21]) afin de réduire les domaines des variables du problème d'optimisation. La réduction basée sur l'op-

[3] Nous nous appuyons sur les techniques introduites par Hansen [10] pour traiter les systèmes sous-contraints.

[4] Différentes heuristiques sont utilisées pour sélectionner la variable dont le domaine sera découpé.

timalité se fonde sur les deux théorèmes suivants pour améliorer les bornes du domaine d'une variable :

Theorem 1 *Soit U, une borne supérieure connue du problème initial P. Soit L, une borne inférieure d'une relaxation convexe R de P. Si la contrainte $x_i - \overline{x}_i \leq 0$ est active au niveau de la solution optimale de R et si son multiplicateur λ_i^* est strictement positif, alors*

$$x_i \geq x_i' \text{ avec } x_i' = \overline{x}_i - \frac{U-L}{\lambda_i^*}. \quad (4)$$

Par conséquent, si $x_i' > \underline{x}_i$, le domaine de x_i peut être fixé à $[x_i', \overline{x}_i]$ sans qu'aucun optimum global ne soit perdu.

λ_i^* dénote la solution duale de R. Informellement, une contrainte $A_i x_i \leq b_i$ est active si $A_i x_i = b_i$. Notons qu'une telle égalité est difficile à vérifier en présence de calculs sur les nombres à virgule flottante.

Theorem 2 *Soit U, une borne supérieure connue du problème initial P. Soit L, une borne inférieure d'une relaxation convexe R de P. Si la contrainte $\underline{x}_i - x_i \leq 0$ est active au niveau de la solution optimale de R et si son multiplicateur λ_i^* est strictement positif, alors*

$$x_i \leq x_i'' \text{ avec } x_i'' = \underline{x}_i + \frac{U-L}{\lambda_i^*}. \quad (5)$$

Par conséquent, si $x_i'' < \overline{x}_i$, le domaine de x_i peut être fixé à $[\underline{x}_i, x_i'']$ sans qu'aucun optimum global ne soit perdu.

Le premier théorème fourni un test pour améliorer la borne supérieure du domaine d'une variable alors que le second théorème fourni un test pour améliorer sa borne inférieure.

Ces résultats ont été généralisés aux autres types de contraintes grâce au théorème suivant :

Theorem 3 *Soit U, une borne supérieure connue du problème initial P. Soit L, une borne inférieure d'une relaxation convexe R de P. Si la contrainte $g_i(x) \leq 0$ est active au niveau de la solution optimale de R et si son multiplicateur λ_i^* est strictement positif, alors*

$$g_i(x) \geq -\frac{U-L}{\lambda_i^*}. \quad (6)$$

Ce dernier théorème permet d'exploiter toute contrainte active pour tenter de réduire le domaine des variables.

Tous ces théorèmes sont détaillés et prouvés dans [26].

4.2 Approche de Kearfott : une RBO rigoureuse basée sur un encadrement rigoureux de la solution duale

Le point critique dans les formules (4)(5)(6) de réduction basée sur l'optimalité réside dans les solution duales approximatives fournies par la méthode du Simplexe[5]. Notons que les techniques suggérées par Neumaier et al [24] et utilisées par la `Quad` ne permettent pas de calculer une solution rigoureuse du problème dual. Pour obtenir un résultat valide, les solutions duales doivent donc être prouvées afin de conserver la rigueur du processus de résolution par séparation et évaluation. Kearfott [12] a exploré cette approche dans le cas particulier des relaxations linéaires. Considérons le cas des relaxations linéaires composées uniquement d'inégalités :

$$\begin{array}{ll} min & d^T x \\ s.t. & Ax \leq b \end{array} \quad (7)$$

Le dual de (7) est le programme linéaire suivant :

$$\begin{array}{ll} max & b^T y \\ s.t. & A^T \lambda = d \end{array} \quad (8)$$

où λ dénote les variables duales utilisées dans les formules de la RBO. Le système de Kuhn-Tucker (KT)[6] permet d'obtenir les bornes inférieures et supérieures rigoureuses des systèmes (7) et (8).

$$(KT) \left\{ \begin{array}{ll} A^T\lambda - d & = 0 \\ \lambda_i(A_{i,:}x - b_i) & = 0, 1 \leq i \leq m \end{array} \right. \quad (9)$$

où $A_{i,:}$ est la i-ème ligne de A.

Un résultat est bien connu est que si la i-ème contrainte $A_i : x \leq b_i$ est inactive pour une solution –i.e., l'inégalité stricte $x^* < b_i$ est vraie pour x^*, la solution du primal– alors la variable duale correspondante (appelée aussi multiplicateur de Lagrange) y_i est égale à zéro. Par conséquent, les contraintes inactives doivent être identifiées afin de rendre le système (9) linéaire. Cependant, cette approche soulève deux problèmes majeurs :

1. Puisque les contraintes d'égalités sont remplacées par deux contraintes d'inégalités, certaines des contraintes de (9) sont redondantes. La méthode de Newton ne permet donc plus de valider les solutions.
2. Les contraintes inactives sont identifiées à l'aide des solutions duales approximatives fournies par

[5] Nous utilisons ici une implémentation efficace de la méthode du Simplexe i.e. une implémentation qui s'appuie sur des calculs avec des nombres en virgule flottante.

[6] Pour une introduction détaillée au système de Kuhn-Tucker, voir [3].

la méthode du Simplexe (Les contraintes inactives sont les contraintes dont la solution duale est nulle).

Kearfott traite ces problèmes en relâchant la relaxation. Par conséquent, la méthode finale de validation basée sur la méthode de Newton appliquée à (9) ne réussie pas toujours, et quand elle réussi, les bornes trouvées sont plutôt larges en raison du relâchement de la relaxation.

4.3 Une implémentation rigoureuse de la RBO basée sur les techniques de la programmation par contraintes

Comme nous l'avons déjà mentionné, le point critique dans la méthode de la RBO réside dans l'approximation de la solution duale calculée par la méthode du Simplexe. Autrement dit, à cause des erreurs d'arrondi, l'optimum global peut être perdu lorsque la formule (4) est appliquée pour réduire le domaine d'une variable x_i.

Une observation essentielle est que les techniques de filtrage peuvent être utilisées pour prouver qu'aucun point faisable n'est contenu dans le domaine de x_i lorsque celui ci est réduit à $[\underline{x}_i, x'_i]$. De fait, si le système de contraintes

$$\begin{array}{l} f(x) \leq U \\ g_i(x) = 0,\ i = 1..k \\ g_j(x) \leq 0,\ j = k+1..m \end{array} \qquad (10)$$

n'a aucune solution lorsque le domaine de x est fixé à $[\underline{x}_i, x'_i]$, alors la réduction calculée par la méthode de la RBO est valide. Si le filtrage ne peut prouver l'absence de solution au sein de la boîte considérée, cette boîte doit juste être ajoutée à $\mathcal{L}$, la liste des boîtes qui doivent encore être traitées (cf. les algorithmes 1 et 2).

Le même raisonnement s'applique à la réduction du domaine de $\mathbf{f}$, i.e., lorsque l'algorithme 2 tente d'améliorer la borne inférieure de la fonction objectif $\mathbf{f}$ grâce à la formule (6).

L'algorithme 2 détaille le nouveau processus de calcul de la borne inférieure. Notez que l'algorithme 1 reste presque inchangé : seul l'appel $\underline{\mathbf{f}}_{\mathbf{x}'} \leftarrow LowerBound(\mathbf{x}')$ y est remplacé par $(\underline{\mathbf{f}}_{\mathbf{x}'}, x', \mathcal{L}) \leftarrow LowerBound(\mathbf{x}', L, U, \mathcal{L})$.

Algorithm 2 Calcul d'une borne inférieure rigoureuse avec la RBO

Function `LowerBound`(`IN` **x**, $L, U, \mathcal{L}$; `OUT` ($\underline{\mathbf{f}}_{\mathbf{x}'}, \mathcal{L}$))

$\mathcal{L}_r \leftarrow \emptyset$ % $\mathcal{L}_r$: set of potential non-solution boxes
Compute $\underline{\mathbf{f}}$ with `Quad` in **x**
for each variable **x** **do**
  Apply formula 4 of OBR and
    add the generated potential non-solution boxes to $\mathcal{L}_r$
for each box $\mathbf{B_i}$ in $\mathcal{L}_r$ **do**
  $\mathbf{B'_i} \leftarrow 2B$-filtering($\mathbf{B_i}$)
  if $\mathbf{B'_i} = \emptyset$ **then** reduce the domain of x_i
  else $\mathbf{B''_i} \leftarrow$ `Quad`-filtering($\mathbf{B'_i}$)
    if $\mathbf{B''_i} = \emptyset$ **then** reduce the domain of x_i
    else add $\mathbf{B_i}$ to $\mathcal{L}$; **endif**
  endif
Apply formula 6 of OBR to reduce the
    lower bound of the objective function **f**
Use $2B$-filtering and `Quad`-filtering to validate the reduction

5 Résultats expérimentaux

Afin d'évaluer la contribution de l'approche basée sur la PPC, nous avons comparé ses performances avec celles de la méthode proposée par Kearfott [12] sur un ensemble de 78 problèmes d'optimisation globale. 17 d'entre eux, les problèmes 'c-chem', sont décris dans [25]. 6 d'entre eux proviennent des pages 140 à 147 de la thèse d'Audet [1]. Les 55 problèmes restant proviennent de la bibliothèque de tests 1 de COCONUT[7]. Tous ces problèmes ont moins de 100 variables et de 100 contraintes.

L'algorithme de séparation et évaluation est implémenté au sein du solveur ICOS[8]. Les relaxations linéaires utilisées pour le calcul de la borne inférieure sont résolues par le solveur de programmes linéaires Coin/CLP[9]. L'étape de recherche d'une borne supérieure repose sur le solveur de programmes non-linéaires Coin/IPOPT[10] basé sur la méthode du point intérieur. L'ensemble des expérimentations a été effectué sur un ordinateur portable équipé d'un Pentium III à 1Ghz.

Nous avons procédé aux expérimentations suivantes :

- un calcul de l'optimum global sans réduction basée sur l'optimalité (intitulée "no OBR" dans les tables) ;
- un calcul de l'optimum global avec une réduction non-rigoureuse basée sur l'optimalité (inti-

[7] http ://www.mat.univie.ac.at/~neum/glopt/coconut/Benchmark/Library1_new_v1.html Les problèmes testés sont : alkyl, chance, circle, dispatch, ex14_1_1, ex14_1_2, ex14_1_3, ex14_1_4, ex14_1_5, ex14_1_6, ex14_1_8, ex14_1_9, ex14_2_2, ex14_2_5, ex2_1_1, ex2_1_10, ex2_1_2, ex2_1_4, ex2_1_5, ex2_1_6, ex3_1_2, ex3_1_3, ex3_1_4, ex4_1_1, ex4_1_3, ex4_1_4, ex4_1_5, ex4_1_6, ex4_1_7, ex4_1_8, ex4_1_9, ex5_2_2_case1, ex5_2_2_case3, ex5_4_2, ex6_1_2, ex6_1_4, ex7_2_2, ex7_2_5, ex7_2_6, ex7_2_10, ex7_3_1, ex7_3_2, ex7_3_3, ex8_1_1, ex8_1_5, ex8_1_6, ex8_1_8, ex9_1_6, ex9_1_7, ex9_1_9, ex9_2_8, himmel11, house, nemhaus, rbrock, sample, wall

[8] http ://www.essi.fr/~lebbah/icos/index.html

[9] http ://www.coin-or.org/cgi-bin/cvsweb.cgi/COIN/Clp/

[10] http ://list.coin-or.org/mailman/listinfo/coin-ipopt

	$\Sigma_t(s)$	$\%saving$
no OBR	2384.36	-
unsafe OBR	881.51	63.03%
safe OBR Kearfott	1975.95	17.13%
safe OBR CP	454.73	80.93%

TAB. 1 – Synthèse des résultats (avec un timeout de 500s)

tulée "unsafe OBR" dans les tables) ;
- un calcul de l'optimum global avec une réduction basée sur l'optimalité utilisant la méthode rigoureuse de Kearfott (intitulée "safe OBR Kearfott" dans les tables) ;
- un calcul de l'optimum global avec une réduction basée sur l'optimalité utilisant notre approche rigoureuse (intitulée "safe OBR CP" dans les tables) ;

Pour réfuter les boites rejetées par la RBO, une combinaison entre un filtrage par $2B$–consistance et le filtrage de la *Quad* a été utilisée. Plus précisément, le filtrage de la *Quad* est seulement utilisé quand le filtrage par $2B$–consistance n'est pas capable de rejeter les boites identifiées par la RBO.

Le tableau 1 offre une synthèse des résultats obtenus par les différentes méthodes avec un timeout de 500s. La seconde colonne donne le temps global (en secondes) nécessaire à la résolution de l'ensemble des problèmes. La dernière colonne donne le pourcentage de gain de temps obtenu par la réduction basée sur l'optimalité par rapport à une simple méthode de séparation et évaluation.

La seconde ligne du tableau synthétise les résultats obtenus par une RBO non rigoureuse. Les résultats fournis par l'algorithme de séparation et évaluation sont donc susceptibles d'être faux. Toutefois, ces expérimentations soulignent le bénéfice potentiel de la RBO dans un processus de séparation et évaluation. De fait, la RBO peut améliorer significativement le temps de recherche d'un optimum global.

Malheureusement, la quasi totalité de l'apport de la RBO est perdu avec l'approche prônée par Kearfott. A contrario, l'approche basée sur la PPC préserve et améliore même les performances. Une observation essentielle est que l'approche basée sur la PPC est capable de résoudre tous ces problèmes en moins de $56s$ alors que l'approche de Kearfott est stoppée avant d'avoir achevé ses calculs par le timeout de $500s$ sur 3 des problèmes.

Le lecteur peut être surpris par le fait que les performances de l'approche basée sur la PPC soient supérieures à celle de la RBO non-rigoureuse. Un analyse minutieuse des résultats permet d'expliquer ce point :

name	safe OBR CP	safe OBR Kearfott	% saving
himmel11	2.4	-	-
ex5_2_2_case3	4.49	-	-
c-chem7	10.5	-	-
ex2_1_5	3.52	15.64	77.49%
ex7_2_5	2.19	7.14	69.32%
ex14_2_5	2.48	7.19	65.50%
ex7_2_10	0.16	0.41	60.97%
ex8_1_6	0.73	1.54	52.59%
ex14_2_2	3.05	6.38	52.19%
ex7_3_1	3.38	5.95	43.19%
ex2_1_1	0.25	0.21	-19.04%
ex9_1_9	0.11	0.09	-22.22%
ex2_1_4	1.25	1.01	-23.76%
ex9_2_8	0.05	0.04	-25%
c-chem18	4.83	3.57	-35.29%
ex3_1_2	0.19	0.14	-35.71%
c-audet147	0.61	0.44	-38.63%
c-audet140b	0.13	0.09	-44.44%
alkyl	32.45	20.55	-57.90%
ex5_2_2_case1	7.75	2.88	-169.09%

TAB. 2 – Safe OBR CP vs safe OBR Kearfott

il s'avère que les réductions erronées de domaines effectuées par l'approche non-rigoureuse pénalise le calcul de la borne supérieure courante de la fonction objectif.

L'approche basée sur la PPC introduite ici est environ cinq fois plus rapide que l'approche de Kearfott. De fait, le surcoût dû à notre approche reste négligeable puisque le mécanisme de preuve s'appuie sur la $2B$–consistance, une technique de filtrage ici peu coûteuse[11]. Ceci est mis en évidence dans le tableau 2 qui compare les résultats obtenus par l'approche à base de PPC avec ceux de l'approche de Kearfott. Les 10 premiers résultats correspondent à ceux où l'approche à base de PPC se comporte le mieux par rapport à l'approche de Kearfott. Les 10 derniers résultats correspondent à ceux où l'approche à base de PPC se comporte le moins bien par rapport à l'approche de Kearfott. La première colonne donne les noms des problèmes. La seconde colonne donne le temps total (en secondes) nécessaire à la résolution du problème en utilisant l'approche à base de PPC. La troisième colonne donne le temps total (en secondes) nécessaire à la résolution du problème en utilisant l'approche de Kearfott. La dernière colonne donne le pourcentage de gain de temps de l'approche à base de PPC par rapport à l'approche de Kearfott.

[11] C'est la raison pour laquelle la technique plus coûteuse de la `Quad` n'est presque jamais mise en œuvre dans ces exemples.

6 Conclusion

Nous avons montré ici comment les techniques de filtrage issues de la programmation par contraintes peuvent permettre une implémentation rigoureuse et efficace de la réduction basée sur l'optimalité. Grâce à la programmation par contraintes, l'algorithme de séparation et évaluation peut bénéficier de l'apport de la RBO à l'aide d'un simple mais efficace processus de réfutation.

Les premières expérimentation ont montré que notre approche se comporte avantageusement par rapport à celle de Kearfott. En utilisant un processus de réfutation basé sur la programmation par contraintes, la RBO est en moyenne 5 fois plus rapide qu'avec la procédure de Kearfott. L'approche basée sur la PPC améliore donc significativement le processus de séparation et évaluation.

Une procédure de réfutation basée sur la programmation par contraintes a permis une intégration simple de la RBO. Cette approche semble assez générale pour être appliquée à d'autres méthodes non-rigoureuses. La prochaine étape consiste donc à tester cette approche sur d'autres méthodes non-rigoureuses dans le but d'améliorer le processus de séparation et évaluation.

Références

[1] Charles Audet. *Optimisation Globale Structurée : Propriétés, Équivalences et Résolution*. thèse de doctorat, École Polytechnique de Montréal, Quebec, 1997.

[2] Venkataramanan Balakrishnan et Stephen P. Boyd. Global optimization in control system analysis and design. Dans C.T. Leondes, editeur, *Control and Dynamic Systems : Advances in Theory and Applications*, volume 53. Academic Press, New York, New York, 1992.

[3] Mokhtar S. Bazaraa, Hanif D. Sherali, et C. M. Shetty. *Nonlinear Programming : Theory and Algorithms*. John Wiley & Sons, 1993.

[4] Frédéric Benhamou, David McAllester, et Pascal Van Hentenryck. Clp(intervals) revisited. Dans *Proc. of the ISLP'94*, pages 124–138, 1994.

[5] Frédéric Benhamou et William Older. Applying interval arithmetic to real, integer and boolean constraints. *Journal of Logic Programming*, pages 1–24, 1997.

[6] Glencora Borradaile et Pascal Van Hentenryck. Safe and tight linear estimators for global optimization. *Mathematical Programming*, 2005.

[7] John G. Cleary. Logical arithmetic. *Future Computing Systems*, pages 125–149, 1987.

[8] Hélène Collavizza, Fran cois Delobel, et Michel Rueher. Comparing partial consistencies. *Reliable Computing*, pages 213–228, 1999.

[9] Christodoulos A. Floudas. Deterministic global optimization in design, control, and computational chemistry. Dans *IMA Proceedings : Large Scale Optimization with Applications. Part II : Optimal Design and Control*, pages 129–184, 1997.

[10] Eldon R. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker, New York, 2004.

[11] Rainer Horst et Hoang Tuy. *Global Optimization : Deterministic Approches*. Springer-Verlag, 1993.

[12] R. Baker Kearfott. Validated probing with linear relaxations. *submitted to Journal of Global Optimization*, 2005.

[13] R. Baker Kearfott. Discussion and empirical comparisons of linear relaxations and alternate techniques in validated deterministic global optimization. *journal of Optimization Methods and Software*, pages 715–731, Octobre 2006.

[14] Yahia Lebbah et Olivier Lhomme. Accelerating filtering techniques for numeric CSPs. *Artificial Intelligence*, 139(1) :109–132, 2002.

[15] Yahia Lebbah, Claude Michel, et Michel Rueher. An efficient and safe framework for solving optimization problems. *JCAM (accepted for publication)*, 2005.

[16] Yahia Lebbah, Claude Michel, Michel Rueher, David Daney, et Jean-Pierre Merlet. Efficient and safe global constraints for handling numerical constraint systems. *SIAM Journal on Numerical Analysis*, 42(5) :2076–2097, 2004.

[17] Eric Lee et Constantinos Mavroidis. Solving the geometric design problem of spatial 3R robot manipulators using polynomial homotopy continuation. *Journal of Mechanical Design*, 124(4) :652–661, Décembre 2002.

[18] Olivier Lhomme. Consistency techniques for numeric CSPs. Dans *Proceedings of IJCAI'93*, pages 232–238, Chambéry(France), 1993.

[19] Alan K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, pages 99–118, 1977.

[20] Claude Michel, Yahia Lebbah, et Michel Rueher. Safe embedding of the simplex algorithm in a CSP framework. Dans *Proc. of CPAIOR 2003, Montréal*, 2003.

[21] Michel Minoux. *Mathematical Programming. Theory, Algorithms and Applications.* Wiley, 1986.

[22] Bruce A. Murtagh et Michael A. Saunders. Minos 5.5 user's guide. Technical Report SOL 83-20R, Systems Optimization Laboratory, Dep. of Operations Research, Stanford University, Juillet 1998.

[23] Arnold Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 2004.

[24] Arnold Neumaier et Oleg Shcherbina. Safe bounds in linear and mixed-integer programming. *Mathematical Programming*, pages 283–296, 2004.

[25] Hong S. Ryoo et Nikolaos V. Sahinidis. Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers Chemical Engineering*, 19(5) :551–566, 1995.

[26] Hong S. Ryoo et Nikolaos V. Sahinidis. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, pages 107–138, 1996.

[27] Djamila Sam-Haroud et Boi Faltings. Consistency techniques for continuous constraints. *Constraints*, 1(1/2) :85–118, 1996.