Hybridation de programmation par contraintes sur intervalles et d'algorithmes évolutionnaires pour l'optimisation globale - Cooperation of evolutionary algorithms and interval constraint programming

Charlie Vanaret, Jean-Baptiste Gotteland, Nicolas Durand

Hybridation de programmation par contraintes sur intervalles et d'algorithmes évolutionnaires pour l'optimisation globale

Charlie Vanaret Jean-Baptiste Gotteland Nicolas Durand

Laboratoire de Mathématiques Appliquées, Informatique, Automatique pour l'Aérien
Ecole Nationale de l'Aviation Civile
Institut de Recherche en Informatique de Toulouse, France
{vanaret, gottelan, durand}@recherche.enac.fr

Mots-clés : *optimisation globale, programmation par contraintes sur intervalles, algorithmes évolutionnaires, hybridation, optimisation déterministe et stochastique*

1 Introduction

Le caractère combinatoire de nombreux problèmes d'optimisation non linéaires handicape fortement la recherche d'optima globaux. Les méthodes déterministes garantissant l'optimalité de la solution, telles que les algorithmes de Branch and Bound par Intervalles, ne convergent pas dans un délai raisonnable. A l'inverse, les Algorithmes Evolutionnaires opérant dans le domaine continu fournissent rapidement des solutions satisfaisantes, sans toutefois garantir leur optimalité. [ADGG12] ont posé les bases d'un nouveau type d'algorithme hybride combinant l'efficacité des Algorithmes Evolutionnaires et l'exactitude de l'Arithmétique d'Intervalles dans le but de *prouver l'optimalité* des solutions de problèmes difficiles sans contraintes. Une preuve d'optimalité inédite a été donnée pour plusieurs fonctions de benchmark, ce qui met en évidence la validité de cette approche. Cependant, son efficacité peut être grandement améliorée en tirant parti de la forme analytique de la fonction : la formulation originale de l'algorithme hybride n'exploite pas la monotonie locale de la fonction et les techniques de propagation de contraintes.

Dans la présente communication, nous proposons une approche avancée dans laquelle Algorithmes Evolutionnaires coopèrent avec Programmation par Contraintes sur Intervalles. Les performances sont comparées sur deux fonctions de benchmark et de nouveaux résultats optimaux sont obtenus. Sans perte de généralité, on se place par la suite dans le cas d'un problème de minimisation.

2 Algorithmes évolutionnaires

Les Algorithmes Evolutionnaires (EA) sont des algorithmes stochastiques d'optimisation globale qui s'inspirent de la théorie de l'évolution [Hol75]. Une population d'individus, solutions candidates au problème, est modifiée itérativement via des opérateurs de sélection, reproduction et mutation selon un critère d'adaptation – la fonction objectif à optimiser – dans le but d'obtenir des solutions satisfaisantes. Les EA fournissent généralement de bonnes solutions à des problèmes difficiles sans hypothèses de continuité ou de dérivabilité, *sans toutefois garantir leur optimalité.*

3 Arithmétique d'intervalles

L'Arithmétique d'Intervalles (IA) est une méthode d'analyse numérique introduite par Moore [Moo66] pour borner les erreurs d'arrondis dues à l'arithmétique flottante. Un intervalle compact à bornes flottantes est noté $X = [\underline{X}, \overline{X}]$, et un vecteur d'intervalles est appelé *boîte*. L'IA

étend aux intervalles les opérateurs $(+, -, \times, /)$ et les fonctions élémentaires $(\exp, \cos, \dots)$ en arrondissant les calculs vers l'extérieur. Une **extension aux intervalles** F d'une fonction réelle f guarantit un encadrement rigoureux de l'image de f. Cependant, l'IA produit généralement une large surestimation de l'image lorsque des variables apparaissent plusieurs fois dans l'expression de F.

L'algorithme de **Branch and Bound par Intervalles** (IBBA) exploite les propriétés conservatives des extensions aux intervalles pour borner de manière rigoureuse les solutions de problèmes d'optimisation [Han92]. Il consiste à diviser l'espace de recherche en sous-domaines sur lesquels F est évaluée. En maintenant à jour une borne supérieure $\tilde{f}$ du minimum global, les parties de l'espace de recherche qui ne peuvent pas contenir une solution optimale sont éliminées. Les boîtes restantes sont stockées dans une file de priorité pour être traitées récursivement, jusqu'à atteindre une précision désirée sur la taille de la boîte (ϵ_x) et de son image (ϵ_f). Le processus est répété jusqu'à exploration complète du domaine. Du fait de sa complexité exponentielle, l'algorithme reste peu efficace sur des problèmes de grande dimension.

4 Propagation de contraintes

La Programmation par Contraintes sur Intervalles (ICP) permet de résoudre des systèmes d'équations non linéaires. Les algorithmes de filtrage (contraction), issus des communautés d'ICP et d'AI, réduisent (contractent) le domaine des variables sans perte de solutions [CJ09]. L'algorithme standard `HC4` [BGGP99] renforce la consistance d'enveloppe et appelle itérativement une procédure `HC4Revise` qui travaille sur une seule contrainte : un double parcours de l'arbre syntaxique d'une contrainte contracte chaque occurrence d'une variable. Il consiste en une phase d'évaluation (bottom-up) en chaque nœud de l'arbre et une phase inverse (top-down) de propagation de la contrainte en utilisant des fonctions de projection.

Propager les contraintes $f \leq \tilde{f}$ et $\nabla f = 0$ permet de contracter les domaines des variables ou prouver qu'une partie de l'espace de recherche ne peut pas contenir une solution optimale. Notre algorithme est implémenté selon un schéma de Branch and Contract (IBCA), avec une étape de filtrage basée sur `HC4Revise`.

5 Algorithme hybride coopératif

L'algorithme coopératif combine un EA et un IBCA qui s'exécutent indépendamment et communiquent via mémoire partagée. L'EA recherche des solutions satisfaisantes et améliore la borne de l'IBCA afin d'éliminer plus rapidement des parties de l'espace de recherche. Lorsqu'un point faisable trouvé par l'IBCA améliore la borne supérieure du minimum global, il est intégré à la population de l'EA. Un troisième processus projette périodiquement dans le domaine de l'IBCA les individus piégés dans des minima locaux.

Dans [ADGG12], les individus de l'EA sont évalués en arrondissant au flottant le plus proche. Néanmoins, l'évaluation par intervalles du meilleur individu $\mathbf{x_{best}}$ est *nécessaire* lorsque la borne de l'IBCA est mise à jour, de manière à conserver un encadrement rigoureux du minimum global. Une borne supérieure rigoureuse du minimum global est alors $\overline{F(\mathbf{x_{best}})}$.

6 Résultats optimaux

6.1 Fonction de Michalewicz

La fonction de Michalewicz est une fonction multimodale ($n!$ optima locaux) définie sur $[0, \pi]^n$.

$$f_n(x) = -\sum_{i=1}^{n} \sin(x_i) \left[\sin\left(\frac{i x_i^2}{\pi} \right) \right]^{20}$$

Les meilleurs solutions connues jusqu'à $n = 50$ sont répertoriées dans [Mis06]. Leur optimalité n'est pas prouvée. Très peu de résultats existent en ce qui concerne les méthodes déterministes. L'optimum pour $n = 12$ est prouvé dans [ADGG12] pour $\epsilon_x = 10^{-3}$ et $\epsilon_f = 10^{-4}$ au bout de **6000s** : $f_{12}^* = -11.64957$. Les *minima globaux* de la fonction de Michalewicz obtenus avec notre algorithme sont présentés dans le tableau 1 pour $n = 12$ à 50 et $\epsilon_x = \epsilon_f = 10^{-10}$.

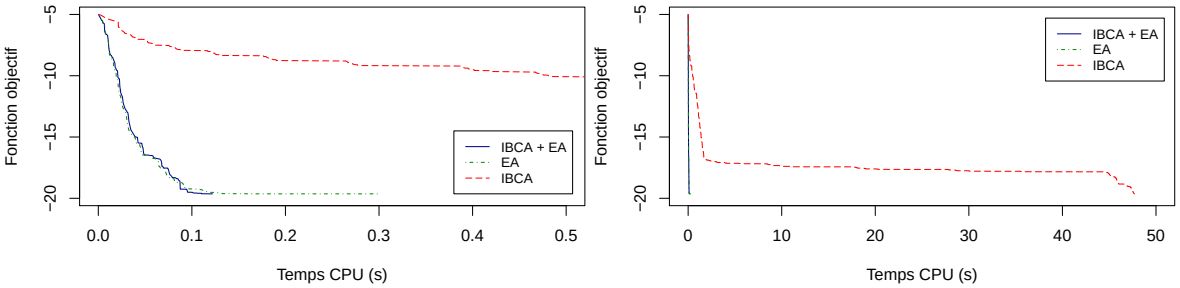
n	f_n^*
12	-11.64957499871
20	-19.63701359935
30	-29.63088385032
40	-39.62674886468
50	-49.62483231828

TAB. 1 – Minima globaux

n	Temps (s)	Eval f	Eval F	Eval ∇F
12	0.042	10401	678	4445
20	0.14	25236	1608	17916
30	0.43	57781	3641	59969
40	1.28	132795	8165	177598
50	3.51	302326	17123	473168

TAB. 2 – Valeurs moyennes pour 20 exécutions successives

Le tableau 2 présente le temps de calcul moyen et le nombre moyen d'évaluations de la fonction objectif, de son extension aux intervalles et de celle de son gradient pour 20 exécutions successives de notre algorithme hybride. Pour $n = 12$, nous obtenons un gain moyen en temps CPU d'un facteur 140000 par rapport à [ADGG12]. Les performances des différents algorithmes pour $n = 20$ sont comparées sur la figure 1. Sur cette instance, notre algorithme hybride converge 390 fois plus rapidement que l'IBCA seul. La coopération entre EA et IBCA accélère également la convergence de l'EA.



(a) Début de convergence

(b) Fin de convergence

FIG. 1 – Comparaison de performance ($n = 20$)

6.2 Fonction Egg Holder

La fonction Egg Holder est une fonction multimodale définie sur $[-512, 512]^n$, particulièrement difficile à minimiser.

$$f_n(x) = -\sum_{i=1}^{n-1}[(x_{i+1} + 47)\sin(\sqrt{|x_{i+1} + 47 + \frac{x_i}{2}|}) + x_i \sin(\sqrt{|x_i - (x_{i+1} + 47)|})]$$

La meilleure solution connue pour $n = 2$ est $f_2^* = f_2(512, 404.2319) = -959.64$ [Mis06]. A notre connaissance, aucune preuve d'optimalité n'existe dans la littérature. Les *minima globaux* de la fonction Egg Holder pour $n = 2$ à 5 et $\epsilon_x = \epsilon_f = 10^{-6}$ sont présentés dans le tableau 3. Les solutions correspondantes sont données dans le tableau 5. Il s'agit de résultats inédits en ce qui concerne les méthodes déterministes. Le tableau 4 présente les résultats moyens obtenus par notre algorithme hybride. On notera la croissance exponentielle du nombre d'évaluations de la fonction objectif et de son extension aux intervalles lorsque n croît.

n	f_n^*
2	-959.6406627
3	-1888.3213909
4	-2808.1847922
5	-3719.7248363

TAB. 3 – Minima globaux

n	Temps (s)	Eval f	Eval F	Eval ∇F
2	0.006	4068	239	189
3	0.03	42878	1857	3286
4	0.11	120292	4550	11167
5	2.12	2302362	73332	266461

TAB. 4 – Valeurs moyennes pour 20 exécutions successives

n	x_n^*
2	(512, 404.2318127)
3	(481.4628940, 436.9295405, 451.7697125)
4	(482.4274331, 432.9533118, 446.9596239, 460.4887620)
5	(485.5898346, 436.1237076, 451.0831990, 466.4312184, 421.9585188)

TAB. 5 – Coordonnées des solutions optimales

7 Conclusion

La recherche d'optima globaux d'un problème d'optimisation numérique constitue un problème difficile à résoudre lorsque la fonction objectif est non linéaire. Notre algorithme coopératif combinant la fiabilité de la Programmation par Contraintes sur Intervalles et l'efficacité des Algorithmes Evolutionnaires *accélère la convergence* et *prouve l'optimalité de la solution*. Au sein de l'hybride, la borne supérieure du minimum globale mise à jour par l'Algorithme Evolutionnaire (EA) permet à l'Algorithme de Séparation et Contraction sur Intervalles (IBCA) d'éliminer plus efficacement les parties de l'espace de recherche qui ne peuvent pas contenir la solution optimale. La propagation des contraintes permet d'éliminer de nombreuses boîtes de faible diamètre en prouvant que l'optimum ne peut pas y être atteint ou qu'elles ne contiennent pas de point stationnaire. Le calcul inédit des minima globaux de la fonction de Michalewicz ($n = 50$) et de la fonction Egg Holder ($n = 5$), difficiles à optimiser en raison de leurs nombreux minima locaux, illustre le gain de performance obtenu.

Références

[ADGG12] J.-M. Alliot, N. Durand, D. Gianazza, and J.-B. Gotteland. Finding and proving the optimum : Cooperative stochastic and deterministic search. In *20th European Conference on Artificial Intelligence (ECAI 2012), August 27-31, 2012, Montpellier, France*, 2012.

[BGGP99] F. Benhamou, F. Goualard, L. Granvilliers, and J-F. Puget. Revising hull and box consistency. In *International Conference on Logic Programming*, pages 230–244. MIT press, 1999.

[CJ09] G. Chabert and L. Jaulin. Contractor programming. *Artificial Intelligence*, 173 :1079–1100, 2009.

[Han92] E. Hansen. *Global optimization using interval analysis*. Dekker, 1992.

[Hol75] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[Mis06] S. K. Mishra. Global optimization by differential evolution and particle swarm methods : Evaluation on some benchmark functions. Technical report, University Library of Munich, Germany, 2006.

[Moo66] R. E. Moore. *Interval Analysis*. Prentice-Hall, 1966.