La recherche unidimensionnelle pour la résolution de problèmes d'optimisation à grande dimensionnalité

Vincent Gardeux[1,2], Rachid Chelouah[1], Patrick Siarry[2]

[1] EISTI ; Avenue du Parc, 95011 Cergy, France
[2] LiSSi ; Université de Paris 12 ; 61, avenue du Général de Gaulle ; 94010 Créteil, France
vincent.gardeux@eisti.fr

1 Introduction

De nombreuses métaheuristiques permettent de résoudre des problèmes d'optimisation non linéraires à variables continues. Cependant, la plupart des méthodes actuelles ne sont pas conçues pour résoudre efficacement des problèmes dimensionnellement grands, et leurs performances se détériorent rapidement lorsque la dimensionnalité de l'espace de recherche augmente. Ces problèmes particuliers apparaîssent pourtant de plus en plus souvent dans des applications telles que le *data* ou le *web mining*. Ainsi, des méthodes de résolution dimensionnellement robustes sont de plus en plus sollicitées. Nous proposons dans cet article deux méthodes issues de l'optimisation unidimensionnelle permettant de résoudre rapidement des problèmes unimodaux et multimodaux de grande dimensionnalité.

2 Méthodes proposées

La résolution de problèmes d'optimisation à une dimension est classique. La méthode de Newton est l'un des premiers algorithmes permettant de les résoudre (1669). On appelle *Line Search*, une famille de ces méthodes à une dimension qui peuvent être utilisées comme *routine* afin de résoudre des problèmes multidimensionnels. Soit $f : \mathbb{R}^n \to \mathbb{R}$ la fonction objectif à optimiser. L'optimisation revient alors à trouver la valeur de α qui minimise f dans la direction d :

$$h(\alpha) = f(x + \alpha * d) \tag{1}$$

où $x, d \in \mathbb{R}^n$ et $\alpha \in \mathbb{R}$.
Ce type d'optimisation est notamment utilisé par la méthode du Gradient [2]. Les méthodes que nous proposons [1] n'utilisent pas de calcul de direction. En effet, sur des problèmes à grande dimensionnalité, ce calcul peut s'avérer fastidieux ou coûteux. A la place, nos méthodes divisent un problème à N dimensions en N problèmes unidimensionnels qu'elles traitent séparément. Le problème revient alors à optimiser n fonctions unidimensionnelles notées h_i telles que $1 \leqslant i \leqslant n$.
De plus, les fonctions objectifs peuvent être multimodales. Pour éviter de rester bloqués dans des optima locaux, nous utilisons un algorithme de dispersion, qui nous permet de réinitialiser l'optimisation. Cet algorithme calcule la dispersion par rapport à toutes les solutions trouvées, afin de ne pas réinitialiser la solution proche d'un chemin déjà parcouru.
Le premier algorithme est baptisé *Classic Unidirectional Search* (CUS), il parcourt chaque dimension

consécutivement. Sur chaque dimension i, il incrémente ou décrémente α_i d'un pas p_i fixé jusqu'à ce que $f(x_1, x_2, ..., x_i + \alpha_i, ..., x_n)$ ne s'améliore plus. Lorsque c'est le cas, il optimise la dimension suivante et ainsi de suite. Lorsque toutes les dimensions sont parcourues, l'algorithme recommence à partir de la première. Si, après le parcours de toutes les dimensions, aucune amélioration de la solution n'est possible, alors le pas p est modifié ainsi : $p = p * 0.5$, afin de "plonger" vers la solution. Pour réduire le risque de convergence prématurée vers un minimum local, nous proposons un autre algorithme, nommé *Enhanced Unidirectional Search* (EUS), qui permet de réduire la taille des pas à chaque étape. Il parcourt chaque dimension successivement, tout comme le premier, cependant, à chaque nouvelle dimension, il n'incrémente ou ne décrémente α_i qu'une seule fois.

3 Exemple de résultats

Nous nous proposons de confronter les résultats obtenus par nos algorithmes sur le *benchmark* créé pour la conférence CEC'08. Les algorithmes sont classés en fonction de leur performance, c'est-à-dire de l'erreur moyenne obtenue après un nombre fixé d'évaluations de la fonction objectif. Les trois qui ont été classés en premier lors de ce concours sont MTS, LSEDA-gl et jDEdynNP-F. La Figure 1 mentionne ces résultats avec ceux obtenus par CUS et EUS pour 1000 dimensions.

Fonction	CUS	EUS	MTS	LSEDA-gl	JDEdynNP-F
Shifted Sphere	9,67E-12	9,66E-12	0,00E+0	3,23E-13	1,14E-13
Shifted Schwefel Problem 2,21	1,81E+2	8,01E-1	4,72E-2	1,04E-5	1,95E+1
Shifted Rosenbrock	2,52E+3	8,24E+2	3,41E-4	1,73E+3	1,31E+3
Shifted Rastrigin	3,41E+2	9,62E-12	0,00E+0	5,45E+2	2,17E-4
Shifted Griewank	5,99E-12	6,39E-12	0,00E+0	1,71E-13	3,98E-14
Shifted Ackley	1,98E+1	1,98E+1	1,24E-11	4,26E-13	1,47E-11

FIGURE 1 – Erreurs moyennes atteintes pour D = 1000

4 Conclusion

Dans cet article, nous avons proposé deux méthodes de résolution unidimensionnelle pour aider à la résolution de problèmes continus à grande dimensionnalité. Ces algorithmes ont été confrontés à des algorithmes récents ayant obtenu de bons résultats sur des *benchmarks* destinés à l'optimisation en grande dimensionnalité. Après comparaison, nos algorithmes, et plus particulièrement l'algorithme EUS, montrent des performances comparables aux 5 autres algorithmes, sur 5 fonctions sur 6. Ces résultats sont encourageants, pour un algorithme très simple à implémenter et sans paramètres. Dans de futures recherches, nous tenterons de les hybrider, afin d'améliorer leur efficacité.

Références

[1] V. Gardeux, R. Chelouah, P. Siarry, and F. Glover. Unidimensional search for solving continuous high-dimensional optimization problems. In *proceedings of the International Conference on Intelligent Systems Design and Applications*, November 2009. To appear.

[2] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4) :308–313, 1965.