EasyChair Preprint

№ 2525

How sensitivity analysis can improve metaheuristic convergence

Peio Loubiel̀ re, Astrid Jourdan, Patrick Siarry and Rachid Chelouah

February 1, 2020

Comment l'analyse de sensibilité peut aider à la convergence des métaheuristiques

Peio Loubière[1], Astrid Jourdan[1], Patrick Siarry[2], Rachid Chelouah[1]

[1] EISTI - Laboratoire Quartz, Cergy-Pontoise - Pau
{plo, aj, rc}@eisti.eu

[2] Université Paris-Est Créteil, LISSI (EA 3956) 122, rue Paul Armangot, 94400 Vitry sur Seine
siarry@u-pec.fr

Mots-clés : *analyse de sensibilité, optimisation, métaheuristiques.*

1 Introduction

Dans le contexte de l'optimisation difficile, les métaheuristiques guident l'optimisation globale de fonctions non-linéaires. Elles procèdent par stratégies de recherche dont le but est d'explorer efficacement un vaste espace de recherche. Partant d'un ensemble de solutions candidates, les techniques à base de population explorent aléatoirement le voisinage de chaque point. Un voisin est créé par modification (décalage) d'une ou plusieurs variables (composantes) d'une solution considérée. Les variables et le décalage sont choisis de manière aléatoire et le voisin ainsi créé est évalué à l'aide de la fonction objectif.

Nous nous proposons d'utiliser les informations récoltées durant l'exploration afin de caractériser le comportement des variables (influence et comportement) grâce à une méthode d'Analyse de Sensibilité (AS), puis de guider la recherche vers des zones prometteuses de l'espace de recherche.

Les méthodes d'AS sont majoritairement utilisées pour éliminer d'un problème les variables non influentes, avant optimisation. Dans notre cas, le but est de coupler une méthode d'AS et une métaheuristique afin d'avoir suffisamment d'informations pertinentes pour guider le processus d'optimisation et accélérer la convergence.

2 Algorithme proposé et résultats

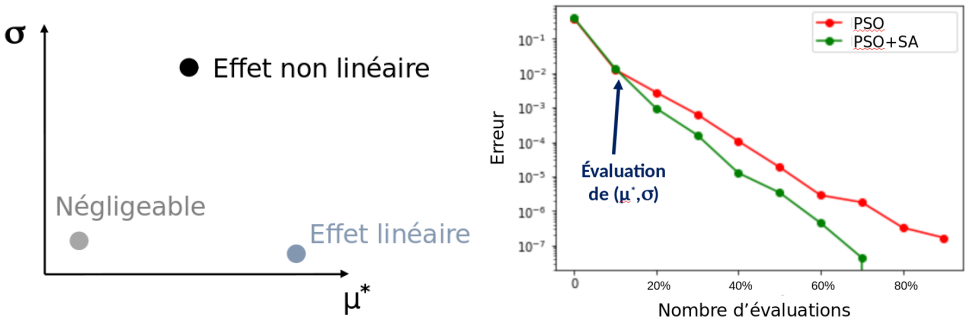
Dans un précédent travail [1], nous avons proposé une méthode naïve d'AS, basée sur les coefficients de la méthode de Morris [2], nous permettant de :

— caractériser et classer les variables selon leur influence ;
— déterminer leur comportement (linéaire ou non).

Nous avons intégré cette méthode à un algorithme d'Évolution Différentielle, afin de sélectionner en priorité les variables les plus influentes. L'influence est proportionnelle à la distance du couple (μ^*, σ) à l'origine (cf Fig. 1a) ; où μ^* et σ représentent la moyenne absolue et l'écart-type des coefficients de corrélation calculés sur un ensemble de voisinages restreints.

Néanmoins certaines métaheuristiques, telles que Particle Swarm Optimization (PSO), génèrent un voisin en décalant toutes les variables, ne tenant pas compte de leur influence. Il serait donc intéressant de pouvoir injecter la connaissance du comportement de chaque variable afin d'adapter la longueur du pas de décalage en fonction du comportement de la variable : un pas plus long si la variable a un comportement linéaire et plus court dans le cas contraire.

La figure 1a rappelle qu'une valeur importante de μ^* indique un comportement linéaire de la variable et qu'une valeur importante de σ indique un comportement non-linéaire. Nous avons alors défini un coefficient δ afin de modifier la longueur du pas de décalage (cf eq. (1)) tel que pour chaque variable j :



(a) Influences (b) Convergence

FIG. 1 – Rappels et résultats

— $\delta_j > 1$: effet linéaire (on augmente le décalage) ;
— $\delta_j < 1$: effet non-linéaire ou négligeable (on diminue le décalage).

$$x_j^{t+1} = x_j^t + \delta_j . v_j^{t+1}, \text{ avec } \delta_j = \frac{\mu_j^{*2}}{\sigma_j}. \tag{1}$$

Nous avons appliqué notre modification à l'algorithme PSO et nous l'avons testée sur une fonction exemple (cf eq. (2)). Cette fonction a été choisie en dimension 5 afin d'avoir la première variable à comportement linéaire, les deux suivantes non-linéaires et les deux dernières négligeables.

$$f(x) = 3.5 * (w_1 - 1) + \sum_{i=2}^{3} [\sin^2(\pi * w_i) + (w_{i-1} - 1)^2 * (1 + 20 * \sin^2(\pi * w_i + 1)/10], \tag{2}$$
avec $w_i = 1 + (x_i - 1)/4$.

La figure 1b illustre l'évolution de la valeur médiane de l'optimum sur 30 exécutions de 3500 évaluations chacune, pour PSO d'origine et intégrant notre méthode d'AS. Les deux comportements sont similaires, puis dès que l'on a récolté suffisamment d'information, on applique la méthode d'AS et on calcule les couples (μ^*, σ) pour chaque variable, ainsi que leur coefficient δ associé. Nous observons alors une convergence plus rapide pour l'algorithme (PSO+SA) intégrant notre méthode.

3 Conclusions et perspectives

Dans notre approche, nous collectons les informations durant un nombre fixe d'itérations, puis nous évaluons les δ_j, utilisés ensuite lors des itérations suivantes, jusqu'à convergence. Néanmoins, pour certaines fonctions du *benchmark* CEC 2013, cela conduit à une convergence prématurée, alors que l'algorithme d'origine continue à converger.

Il serait alors intéressant de relancer une analyse, de manière périodique, afin de déterminer si les coefficients δ_j n'ont pas évolué et pouvoir ainsi modifier le comportement de la recherche.

Références

[1] Peio Loubière, Astrid Jourdan, Patrick Siarry and Rachid Chelouah. *A sensitivity analysis method aimed at enhancing the metaheuristics for continuous optimization.* Artificial Intelligence Review, vol. 50, p. 625-647, 2018.

[2] Max D. Morris. *Factorial sampling plans for preliminary computational experiments.* Technometrics, vol. 33, n 2, p. 161-174, 1991.