Sous-estimateurs Quadratiques convexes par morceaux pour les problèmes d'optimisation quadratique

Amélie Lambert[1], Daniel Porumbel[1]

Cnam, Cédric, France

{amelie.lambert,daniel.porumbel}@cnam.fr

Mots-clés : *Quadratic Programming, Convex Quadratic Piece-wise under-estimator, Branch-and-Bound, Experiences*

Dans ce papier, nous introduisons une famille paramétrée de bornes quadratiques convexes par morceaux pour le problème $(P) : \left\{ \min\limits_{x\in[\ell,u]} \langle Q, xx^\top \rangle + c^\top x \right\}$, où $(Q, c, \ell, u) \in (\mathcal{S}_n, \mathbb{R}^n, \mathbb{R}^n_+, \mathbb{R}^n_+)$, avec $\mathcal{S}_n$ l'ensemble des matrices d'ordre n. Nous proposons ensuite un algorithme itératif original pour calculer les paramètres qui maximisent la valeur de la borne qui peut être utilisé dans un algorithme de spatial branch-and-bound pour résoudre (P) à l'optimalité globale.

1 Une famille de bornes quadratique convexes par morceaux

Notre idée est de construire une famille de relaxations quadratique convexes de (P). Pour cela, nous introduisons n^2 variables Y_{ij} modélisant les produits $x_i x_j$, puis nous relâchons classiquement les égalités $Y_{ij} = x_i x_j$ par les enveloppes de McCormick (1)-(2).Soit $\mathcal{K}$, un ensemble de p matrices semi-définies positives $\mathcal{K} = \{S_k \in \mathcal{S}_n^+, k = 1, \dots, p\}$, nous introduisons la famille de relaxations quadratique convexes par morceaux de (P) paramétrée par l'ensemble $\mathcal{K}$:

$$(P_{\mathcal{K}}) \begin{cases} \min\limits_{x\in[\ell,u]} \ t \\ \text{s.t.}\ Y_{ij} \leq u_j x_i + \ell_i x_j - \ell_i u_j,\ Y_{ij} \leq \ell_j x_i + u_i x_j - u_i \ell_j & (1) \\ \quad Y_{ij} \geq u_j x_i + u_i x_j - u_i u_j, Y_{ij} \geq \ell_j x_i + \ell_i x_j - \ell_i \ell_j & (2) \\ \quad t \geq \langle S_k, xx^\top \rangle + c^\top x + \langle Q - S_k, Y \rangle \qquad k \in \mathcal{K} & (3) \end{cases}$$

Clairement, pour tout ensemble $\mathcal{K}$, le problème $(P_{\mathcal{K}})$ est une relaxation de (P), puisque pour toute solution $\bar{x}$ de (P) de valeur τ, la solution $(\bar{x}, \bar{x}\bar{x}^\top)$ est réalisable pour $(P_{\mathcal{K}})$ et a pour valeur τ. De plus, comme les matrices $S_k \in \mathcal{S}_n^+$, les Contraintes (3) forment un espace réalisable quadratique convexe par morceaux et $(P_{\mathcal{K}})$ est un problème convexe. Nous posons maintenant le problème de déterminer le meilleur ensemble de matrices $\mathcal{K}$ dans le sens où il maximise la valeur de la borne. Plus formellement, nous voulons résoudre le problème suivant :

$$(LB_{\mathcal{K}}) \left\{ \max\limits_{\mathcal{K} = \cup_{k=1,\dots,p} S_k \in \mathcal{S}_n^+} v(P_{\mathcal{K}}) \right.$$

Remarquons ici que pour $p = 1$, la valeur optimale de $(LB_{\mathcal{K}})$ est au moins égale à la valeur optimale de la relaxation SDP "Shor plus RLT" [1] de (P). Pour cela il suffit de construire S_1 à partir des variables optimales duales de cette relaxation SDP, comme cela est décrit dans [2].

2 Un algorithme itératif pour résoudre $(LB_{\mathcal{K}})$

Le principe est de démarrer d'un ensemble initial de matrices semi-définies positives $\mathcal{K}_0$. Ensuite, à chaque itération j, nous évaluons $(P_{\mathcal{K}_j})$ de valeur τ^j et de solution optimale (x^j, Y^j).

Puis, nous ajoutons la matrice $S_{j+1} \in \mathcal{S}_n^+$ à l'ensemble courant $\mathcal{K}_j$, qui est calculée à partir de (x^j, Y^j). Plus précisément, nous prenons S_{j+1} comme combinaison linéaire des matrices $v_i v_i^T$, où les v_i sont les vecteurs propres de la matrice $(x^j x^{j\top} - Y^j)$ associés à ses valeurs propres strictement positives. Un tel choix nous assure que l'évaluation de $(P_{\mathcal{K}_{j+1}})$ au point (x^j, Y^j) sera toujours strictement supérieure à τ^j. L'algorithme est présenté dans l'Algorithme 1. Nous présentons en Figure 1 une illustration graphique de l'algorithme.

Algorithm *Sous-estimateur Quadratique Convexe par morceau* $(Q, c, r, p, \mathcal{K}_0)$

- $\mathcal{K}_1 = \mathcal{K}_0$ /* Ensemble initial $\mathcal{K}_0$ de matrices semi-definies positives */
- $(x^1, Y^1, \tau^1) \leftarrow$ Résolution$(\overline{P}_{\mathcal{K}_1})$ /*On note τ^1 la valeur optimale de $(\overline{P}_{\mathcal{K}_1})$ */
- $j \leftarrow 1$ /*le numéro de l'itération courante*/
- **while** $((x^j x^{j\top} - Y^j \not\preceq 0)$ *or* $(j < p))$ **do**
  - $\lambda^j \leftarrow$ l'ensemble des s valeurs propres positives de $(x^j x^{j\top} - Y^j)$
  - $v^j \leftarrow$ l'ensemble des s vecteurs propres associés aux éléments de λ^j
  - $S_{j+1} \leftarrow r \sum_{i=1}^{s} v_i v_i^\top$
  - $\mathcal{K}_{j+1} = \mathcal{K}_j \cup S_{j+1}$ /* On ajoute une coupe quadratique */
  - $(x^{j+1}, Y^{j+1}, \tau^{\mathcal{K}_{j+1}}) \leftarrow$ Résolution$(\overline{P}_{\mathcal{K}_{j+1}})$
  - $j \leftarrow j+1$ /*Nombre de coupes quadratiques ajoutées*/
  - **if** *(rank* $(Y^j) = 1)$ *(i.e.* $Y^j = x^j x^{j\top}$ *)* **then**
    - (x^j, Y^j, τ^j) est la solution optimale de (P).
- **return** (x^j, Y^j, τ^j)

Algorithm 1: Un algorithme itératif pour résoudre $(LB_{\mathcal{K}})$

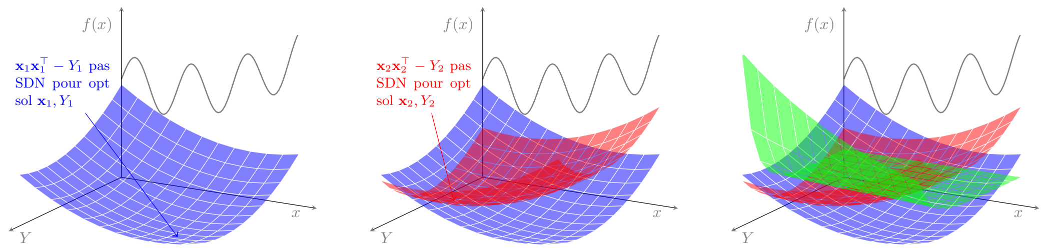

FIG. 1 – Illustration : Initialisation suivie de 2 itérations de l'Algorithme 1

3 Conclusions et perspectives

Nous avons présenté une nouvelle famille de bornes quadratique convexes par morceaux et proposé un algorithme itératif qui ajoute successivement des coupes quadratiques jusqu'à construire une relaxation qui atteint la valeur de la relaxation SDP "Shor plus RLT". De plus, lorsque notre algorithme génère une matrice Y^j de rang 1, l'optimalité de la solution retournée est certifiée. Lorsque ce n'est pas le cas, nous proposons un algorithme de spatial branch-and-bound basée sur cette borne. Les premiers résultats numériques sont encourageants.

Références

[1] K. M. Anstreicher Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming *Journal of Global Optimization*, 43(2) :471–484, 2009.

[2] S. Elloumi and A. Lambert Global solution of non-convex quadratically constrained quadratic programs *Optimization Methods and Software*, 31(1) :98–114, 2019.