Algorithme branch-and-bound pour l'optimisation exacte en norme l0

Ramzi Ben Mhenni, Sébastien Bourguignon, Marcel Mongeau, Jordan Ninin, Hervé Carfantan

Algorithme *branch-and-bound* pour l'optimisation exacte en norme ℓ_0

Ramzi BEN MHENNI[1], Sébastien BOURGUIGNON[1], Marcel MONGEAU[2], Jordan NININ[3], Hervé CARFANTAN[4]

[1]École Centrale de Nantes, LS2N, 1 rue de la Noë, 44321 Nantes, France

[2]ENAC, Université de Toulouse, ENAC, 7 avenue Edouard Belin, 31055 Toulouse, France

[3]ENSTA-Bretagne, Lab-STICC, 2 rue François Verny, 29806 Brest cedex 9, France

[4]Université de Toulouse, IRAP, 14 avenue Édouard Belin, 31400 Toulouse, France

ramzi.ben-mhenni@ec-nantes.fr; sebastien.bourguignon@ec-nantes.fr;
marcel.mongeau@enac.fr; jordan.ninin@ensta-bretagne.fr; herve.carfantan@irap.omp.eu

Résumé – Nous proposons une approche d'optimisation globale de critères de moindres carrés pénalisés par la « norme » ℓ_0, avec un algorithme de séparation et évaluation progressive (*branch-and-bound*). Nous construisons une procédure dédiée d'exploration de l'arbre de recherche combinatoire. Nous montrons que chaque nœud parcouru peut être évalué par l'optimisation de problèmes en norme ℓ_1 avec des contraintes de borne. Nous construisons alors un algorithme de type *active-set* pour les résoudre. Notre procédure permet de résoudre des problèmes de taille modérée mais difficiles, et s'avère notamment plus efficace que le solveur générique CPLEX s'appuyant sur une reformulation MIQP (*Mixed-integer Quadratic Programming*) du problème.

Abstract – We propose a global optimization approach to solve ℓ_0-norm-penalized least-squares optimization problems, using a dedicated branch-and-bound implementation. A specific tree search strategy is built. Then we show that each subproblem involved at a given node can be evaluated *via* ℓ_1-norm-based optimization problems with box constraints, for which an active-set algorithm is built. Our method is able to solve moderate-size, yet difficult, problems. In particular, it is more efficient than the generic MIQP (*Mixed-integer Quadratic Programming*) solver CPLEX.

1 Introduction

L'approximation parcimonieuse est un thème de recherche très actif en traitement du signal, consistant à rechercher une solution $\boldsymbol{x} \in \mathbb{R}^N$ parcimonieuse (avec un grand nombre de composantes nulles), approchant des données $\boldsymbol{y} \in \mathbb{R}^D$, sous un modèle linéaire $\boldsymbol{y} \simeq \mathbf{H}\boldsymbol{x}$. Ce problème peut être décrit par la minimisation de l'erreur d'approximation pénalisée par le nombre de composantes non nulles de $\boldsymbol{x}$, la « norme » ℓ_0 [1, 2] :

$$\mathcal{P} : \min_{\boldsymbol{x}} \tfrac{1}{2}\|\boldsymbol{y} - \mathbf{H}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_0, \text{ où } \|\boldsymbol{x}\|_0 := \text{Card}\{q | x_q \neq 0\}.$$

La résolution de $\mathcal{P}$ n'est en général pas envisagée de manière globale, sous prétexte qu'il s'agit d'un problème NP-difficile [3]. Aussi, de nombreux travaux en traitement du signal et en statistique ont proposé des méthodes d'optimisation locale, basées sur la *relaxation* de la « norme » ℓ_0 par la norme ℓ_1 ou sur des algorithmes gloutons [4].

Or, il a été montré dans [5] qu'on pouvait résoudre $\mathcal{P}$ exactement pour des problèmes de taille modérée mais difficiles, où ces approches échouent à trouver la solution globale. Sous la contrainte supplémentaire $\|\boldsymbol{x}\|_\infty := \max_q |x_q| \leq M$ (où M est une grande constante), le problème :

$$\mathcal{P}^{(0)} : \quad \min_{\boldsymbol{x}} \tfrac{1}{2}\|\boldsymbol{y} - \mathbf{H}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_0 \text{ s.c. } \|\boldsymbol{x}\|_\infty \leq M \quad (1)$$

a été reformulé, en ajoutant de variables de décision binaires, en MIQP (*Mixed-Integer Quadratic Program*), puis résolu avec le solveur générique MIQP CPLEX, s'appuyant sur un schéma de type *branch-and-bound*.

Dans cet article, nous construisons un algorithme *branch-and-bound* spécifique au problème $\mathcal{P}^{(0)}$. Nous montrons qu'une telle stratégie peut être développée sans recourir à des variables binaires et que la résolution de chaque sous-problème mis en jeu s'apparente à un problème en norme ℓ_1, que nous résolvons avec un algorithme dédié de type *active set*.

2 Méthode *branch-and-bound* spécifique

La méthode de *branch-and-bound* [6], repose sur l'alternance d'une étape de *séparation* et d'une étape d'*évaluation*. La première consiste à diviser un problème en sous-problèmes disjoints, plus simples à résoudre, constituant un arbre binaire de recherche. Dans notre cas, chaque séparation correspond à la décision : $x_q \neq 0$ ou $x_q = 0$? (voir Figure 1). Aussi, en un

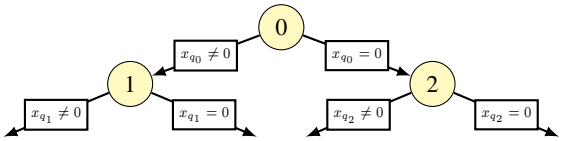

FIGURE 1 – Étape de séparation dans un arbre binaire de décision : chaque nœud est divisé en deux nœuds fils obtenus en contraignant une variable à être nulle ou non-nulle.

nœud i de l'arbre, certaines décisions sont prises sur la nullité des variables : les variables indexées par S^1 sont non nulles, celles indexées par S^0 sont nulles (et donc enlevées du critère) et il reste à prendre des décisions sur les variables indéterminées indexées par $\bar{\mathrm{S}}$.

Nous notons $\mathbf{H}_{\mathrm{S}}$ la sous-matrice formée par les colonnes de $\mathbf{H}$ indexées par S. De même, $\boldsymbol{x}_{\mathrm{S}}$ désigne le sous-vecteur correspondant. Le nœud i correspond alors au problème $\mathcal{P}^{(i)}$:

$$\mathcal{P}^{(i)} : \min_{\boldsymbol{x}_{\mathrm{S}^1}, \boldsymbol{x}_{\bar{\mathrm{S}}}} \tfrac{1}{2}\|\boldsymbol{y} - \mathbf{H}_{\mathrm{S}^1}\boldsymbol{x}_{\mathrm{S}^1} - \mathbf{H}_{\bar{\mathrm{S}}}\boldsymbol{x}_{\bar{\mathrm{S}}}\|_2^2 + \lambda\|\boldsymbol{x}_{\bar{\mathrm{S}}}\|_0 (+\lambda\,|\mathrm{S}^1|) \quad \text{s.c. } \|\boldsymbol{x}\|_\infty \leq M.$$

Le terme entre parenthèses (où $|\mathrm{S}^1|$ est la taille de S^1) étant constant, il ne sera plus considéré dans les expressions à venir.

L'évaluation du nœud i est basée sur le calcul d'une borne inférieure $z_\ell^{(i)}$ pour $\mathcal{P}^{(i)}$, qui indiquera si ce dernier peut contenir une solution optimale. Plus précisément, soit z_U la meilleure valeur connue du critère (une borne supérieure sur la valeur optimale du problème initial $\mathcal{P}^{(0)}$). Alors, si $z_\ell^{(i)} \geq z_U$, le nœud est élagué. Sinon, ce nœud est séparé en deux sous-problèmes selon une nouvelle décision : $x_{q_i} \neq 0$ ou $x_{q_i} = 0$? L'algorithme terminera avec un minimum global en un nombre fini d'itérations. Sa complexité dans le pire cas est celle d'une recherche exhaustive. Son efficacité dépend de la qualité des bornes calculées (évaluation), ainsi que des stratégies de branchement et d'exploration (séparation), que nous construisons maintenant.

2.1 Étape d'évaluation

Borne inférieure. En un nœud i quelconque de l'arbre, une borne inférieure de $\mathcal{P}^{(i)}$ peut être obtenue comme suit :

$$\mathcal{R}^{(i)} : z_\ell^{(i)} := \min_{\boldsymbol{x}_{\mathrm{S}^1}, \boldsymbol{x}_{\bar{\mathrm{S}}}} \tfrac{1}{2}\|\boldsymbol{y} - \mathbf{H}_{\mathrm{S}^1}\boldsymbol{x}_{\mathrm{S}^1} - \mathbf{H}_{\bar{\mathrm{S}}}\boldsymbol{x}_{\bar{\mathrm{S}}}\|_2^2 + \frac{\lambda}{M}\|\boldsymbol{x}_{\bar{\mathrm{S}}}\|_1 \quad \text{s.c. } \|\boldsymbol{x}\|_\infty \leq M. \tag{2}$$

En effet, en raison des contraintes de borne, on a

$$\|\boldsymbol{x}_{\bar{\mathrm{S}}}\|_0 = \sum_{q \in \bar{\mathrm{S}} | x_q \neq 0} 1 \geq \sum_{q \in \bar{\mathrm{S}} | x_q \neq 0} \frac{|x_q|}{M} = \frac{1}{M}\|\boldsymbol{x}_{\bar{\mathrm{S}}}\|_1,$$

donc la valeur de la fonction objectif de $\mathcal{R}^{(i)}$ est inférieure à celle de $\mathcal{P}^{(i)}$. Les deux problèmes étant définis sur le même domaine réalisable, $\mathcal{R}^{(i)}$ est une *relaxation* de $\mathcal{P}^{(i)}$. Nous retrouvons la propriété souvent énoncée que la norme ℓ_1 fournit une relaxation continue convexe de la norme ℓ_0. Ceci n'est cependant vrai que sous des hypothèses supplémentaires bornant l'espace des solutions : dans le cas contraire, le domaine $\{\boldsymbol{x} |\ \|\boldsymbol{x}\|_0 \leq K\}$ n'est pas borné (la « norme » ℓ_0 n'est pas une norme !) et ne peut être inclus dans $\{\boldsymbol{x} |\ \|\boldsymbol{x}\|_1 \leq \tau\}$, quelque soit τ. Nous voyons en particulier que le réglage de M est un point sensible ayant un impact direct sur la qualité de la relaxation : plus sa valeur est grande, plus la relaxation est mauvaise.

Le problème $\mathcal{R}^{(i)}$ est un problème d'optimisation en norme ℓ_1 spécifique, avec des contraintes de borne et où la norme ℓ_1 n'affecte qu'une partie des variables. Nous proposons en section 3 un algorithme de type *active-set* exploitant la structure particulière de $\mathcal{R}^{(i)}$.

Borne supérieure. Une fois la borne inférieure calculée pour un nœud i, ce dernier est élagué si $z_\ell^{(i)} \geq z_U$. Si $z_\ell^{(i)} < z_U$, remarquons que si à l'optimum de $\mathcal{R}^{(i)}$, $\boldsymbol{x}_{\bar{\mathrm{S}}} = \mathbf{0}$ (rendu possible par la pénalisation ℓ_1), alors la relaxation est exacte : le sous-problème $\mathcal{P}^{(i)}$ est résolu (plus besoin de séparer) et on met à jour la borne supérieure. Sinon, on résout le problème obtenu en annulant $\boldsymbol{x}_{\bar{\mathrm{S}}}$:

$$\mathcal{Q}^{(i)} : z_u^{(i)} := \min_{\boldsymbol{x}_{\mathrm{S}^1}} \tfrac{1}{2}\|\boldsymbol{y} - \mathbf{H}_{\mathrm{S}^1}\boldsymbol{x}_{\mathrm{S}^1}\|_2^2 \text{ s.c. } \|\boldsymbol{x}\|_\infty \leq M \tag{3}$$

ce qui peut permettre de raffiner la borne supérieure si $z_u^{(i)} < z_U$. En particulier, si S^1 correspond au support de l'optimum global, alors $z_u^{(i)}$ correspond à l'optimum global, ce qui favorisera l'élagage de nœuds sous-optimaux. Cette analyse renforce l'intérêt d'une stratégie d'exploration priorisant des « bonnes » combinaisons de variables non nulles, détaillée ci-après.

2.2 Branchement et stratégie d'exploration

Afin de sélectionner l'indice q_i de la variable sur laquelle « brancher » pour subdiviser le problème $\mathcal{P}^{(i)}$, nous exploitons la solution $\boldsymbol{x}^{(i)}$ fournie par $\mathcal{R}^{(i)}$ en choisissant la variable de valeur absolue maximale :

$$q_i = \arg\max_{q \in \bar{\mathrm{S}}} |x_q^{(i)}|\,, \text{ où } \boldsymbol{x}^{(i)} := \arg\min \mathcal{R}^{(i)}. \tag{4}$$

L'idée sous-jacente est de sélectionner d'abord les variables susceptibles d'être non nulles dans la solution optimale. Nous choisissons alors une stratégie de parcours de l'arbre *en profondeur d'abord* [6], qui consiste à subdiviser et descendre le plus profondément avant de parcourir les nœuds en attente, et en explorant d'abord la branche correspondant à la décision $x_{q_i} \neq 0$. Cette stratégie s'apparente aux méthodes dites *forward selection* utilisées par la plupart des algorithmes gloutons conçus pour l'approximation parcimonieuse.

L'algorithme 1 résume la méthode proposée, où L contient la liste des sous-problèmes actifs et $\widehat{\boldsymbol{x}}$ est la meilleure solution connue le long du parcours.

0. **Initialisation :** $L \leftarrow \{\mathcal{P}^{(0)}\}$; $z_U = +\infty$; $\widehat{\boldsymbol{x}} := 0$
1. **Optimalité :** si $L = \emptyset$, retourner la solution optimale $\hat{\boldsymbol{x}}$.
2. **Sélection du nœud :** choisir un sous-problème $i \in L$ et l'éliminer de L.
3. **Évaluation du nœud :** calculer $z_\ell^{(i)}$ par (2).
4. **Élagage :**
   - Si $z_\ell^{(i)} \geq z_U$, élaguer i et retourner à l'étape 1.
   - Si $z_\ell^{(i)} < z_U$
     - Si $\boldsymbol{x}_{\bar{\mathrm{S}}}^{(i)} = \mathbf{0}$, alors $z_U \leftarrow z_\ell^{(i)}$ et $\widehat{\boldsymbol{x}} \leftarrow \boldsymbol{x}^{(i)}$. Élaguer i et retourner à l'étape 1.
     - Sinon, calculer $z_u^{(i)}$ par (3). Si $z_u^{(i)} < z_U$, alors $z_U \leftarrow z_u^{(i)}$ et $\widehat{\boldsymbol{x}} \leftarrow \arg\min \mathcal{Q}^{(i)}$.
5. **Branchement :** subdiviser le nœud i par (4) et ajouter les nouveaux sous-problèmes à L.

Algorithme 1 : Algorithme *branch-and-bound* pour $\mathcal{P}^{(0)}$.

3 Algorithme *active-set* pour l'optimisation ℓ_1 avec contraintes de borne

Nous proposons de résoudre le problème $\mathcal{R}^{(i)}$ par une généralisation de l'algorithme de type *active-set*, appelé *feature-sign search* dans [7] pour le cas standard des moindres carrés pénalisés par la norme ℓ_1. Cette approche semble appropriée pour plusieurs raisons. Tout d'abord, il s'agit d'une méthode exacte et la solution est obtenue en un nombre fini d'itérations. Ce point permet de garantir les bornes inférieures calculées dans la procédure de *branch-and-bound*. Ensuite, elle peut naturellement incorporer les spécificités du problème $\mathcal{R}^{(i)}$, à savoir une norme ℓ_1 opérant seulement sur une partie des variables et des contraintes de borne sur toutes les variables. Enfin, l'algorithme permet d'exploiter les calculs effectués au nœud considéré juste avant le nœud courant (démarrage à chaud) lors de l'exploration de l'arbre de recherche.

3.1 Conditions d'optimalité

Notre algorithme *active set* exploite des conditions d'optimalité établies pour $\mathcal{R}^{(i)}$ (conditions classiques de Karush-Kuhn-Tucker étendues au cas convexe non-différentiable [8]). Les détails peuvent être trouvés dans [9]. Nous nous intéressons aux composantes de $\boldsymbol{x}$ qui atteignent la contrainte de borne ($x_q = \pm M$ pour $q \in \bar{\mathrm{S}} \cup \mathrm{S}^1$) et aux points de non-différentiabilité de la norme ℓ_1 ($x_q = 0$ pour $q \in \bar{\mathrm{S}}$). Définissons les ensembles d'indices suivants (que nous appelons le *support*) correspondant à un point $\boldsymbol{x}^\star$ donné :

$$\left\{ \begin{array}{lll} \bar{\mathrm{S}}_{in} & := \{q \in \bar{\mathrm{S}} & \text{t.q. } 0 < |x_q^\star| < M\} \\ \mathrm{S}^1_{in} & := \{q \in \mathrm{S}^1 & \text{t.q. } |x_q^\star| < M\} \\ \bar{\mathrm{S}}_0 & := \{q \in \bar{\mathrm{S}} & \text{t.q. } |x_q^\star| = 0\} \\ \bar{\mathrm{S}}_\square & := \{q \in \bar{\mathrm{S}} & \text{t.q. } |x_q^\star| = M\} \\ \mathrm{S}^1_\square & := \{q \in \mathrm{S}^1 & \text{t.q. } |x_q^\star| = M\} \end{array} \right. . \quad (5)$$

Les conditions d'optimalité de $\mathcal{R}^{(i)}$ se résument comme suit : $\boldsymbol{x}^\star$ est solution de $\mathcal{R}^{(i)}$ si et seulement si :

$$\left\{ \begin{array}{l} \boldsymbol{c}_{\bar{\mathrm{S}}_{in}} = \lambda \mathrm{sgn}(\boldsymbol{x}^\star_{\bar{\mathrm{S}}_{in}}) \\ \boldsymbol{c}_{\mathrm{S}^1_{in}} = 0 \end{array} \right. \quad (6) \qquad \left\{ \begin{array}{r} \left|\boldsymbol{c}_{\bar{\mathrm{S}}_0}\right| < \lambda \\ \boldsymbol{c}_{\bar{\mathrm{S}}_\square} \odot \mathrm{sgn}(\boldsymbol{x}^\star_{\bar{\mathrm{S}}_\square}) \geq \lambda \\ \boldsymbol{c}_{\mathrm{S}^1_\square} \odot \mathrm{sgn}(\boldsymbol{x}^\star_{\mathrm{S}^1_\square}) \geq 0 \end{array} \right. \quad (7)$$

où $\odot$ dénote le produit de Hadamard (produit terme à terme), $\mathrm{sgn}(\alpha) = 1$ si $\alpha > 0$ et -1 si $\alpha < 0$ et

$$\boldsymbol{c} := \mathbf{H}^T(\boldsymbol{y} - \mathbf{H}_{\mathrm{S}^1}\boldsymbol{x}^\star_{\mathrm{S}^1} - \mathbf{H}_{\bar{\mathrm{S}}}\boldsymbol{x}^\star_{\bar{\mathrm{S}}}) . \quad (8)$$

3.2 Algorithme de type active-set

L'algorithme *active set* considère séparément les variables *fixées* ($\boldsymbol{x}_{\bar{\mathrm{S}}_\square}$ et $\boldsymbol{x}_{\mathrm{S}^1_\square}$ à la borne $\pm M$ et $\boldsymbol{x}_{\bar{\mathrm{S}}_0}$ à 0) et l'ensemble dit *actif* composé des variables pénalisées par la norme ℓ_1 (*i.e.* $\boldsymbol{x}_{\bar{\mathrm{S}}_{in}}$) et leurs signes associés $\boldsymbol{\theta}_{\bar{\mathrm{S}}_{in}}$, et des variables non pénalisées $\boldsymbol{x}_{\mathrm{S}^1_{in}}$, pour lesquelles le système d'équations (6) donne :

$$\left\{ \begin{array}{rcl} \boldsymbol{x}_{\bar{\mathrm{S}}_{in}} & = & \left(\mathbf{H}^T_{\bar{\mathrm{S}}_{in}} \boldsymbol{B} \mathbf{H}_{\bar{\mathrm{S}}_{in}}\right)^{-1} \left(\mathbf{H}^T_{\bar{\mathrm{S}}_{in}} \boldsymbol{B}\boldsymbol{r} - \lambda\, \boldsymbol{\theta}_{\bar{\mathrm{S}}_{in}}\right) \\ \boldsymbol{x}_{\mathrm{S}^1_{in}} & = & (\mathbf{H}^T_{\mathrm{S}^1_{in}} \mathbf{H}_{\mathrm{S}^1_{in}})^{-1} (\mathbf{H}^T_{\mathrm{S}^1_{in}} \boldsymbol{r} - \mathbf{H}^T_{\mathrm{S}^1_{in}} \mathbf{H}_{\bar{\mathrm{S}}_{in}} \boldsymbol{x}_{\bar{\mathrm{S}}_{in}}) \end{array} \right. \quad (9)$$

où $\boldsymbol{B} := \boldsymbol{I} - \mathbf{H}_{\mathrm{S}^1_{in}} (\mathbf{H}^T_{\mathrm{S}^1_{in}} \mathbf{H}_{\mathrm{S}^1_{in}})^{-1} \mathbf{H}^T_{\mathrm{S}^1_{in}}$, $\boldsymbol{I}$ est la matrice identité de la taille appropriée et $\boldsymbol{r} := \boldsymbol{y} - \mathbf{H}_{\bar{\mathrm{S}}_\square} \boldsymbol{x}_{\bar{\mathrm{S}}_\square} - \mathbf{H}_{\mathrm{S}^1_\square} \boldsymbol{x}_{\mathrm{S}^1_\square}$.

L'algorithme procède de la façon suivante. Les variables actives sont d'abord calculées par (9) et la variable fixée dont la coordonnée s'éloigne le plus des conditions d'optimalité restantes (7) est intégrée à l'ensemble actif. Une nouvelle solution $\boldsymbol{x}^{\text{new}}$ est alors calculée en réestimant les variables actives par (9), les autres variables n'ayant pas changé. Ensuite, une recherche scalaire est effectuée sur le chemin de $\boldsymbol{x}$ à $\boldsymbol{x}^{\text{new}}$. Notons $f(t)$ la valeur de la fonction objectif de $\mathcal{R}^{(i)}$ en $\boldsymbol{x}^t := \boldsymbol{x} + t(\boldsymbol{x}^{\text{new}} - \boldsymbol{x})$, pour $t \in [0, t^{\max}]$, où $t^{\max} \leq 1$ définit la valeur maximale de t telle que $\|\boldsymbol{x}^t\|_\infty \leq M$. Par construction, f est continue et quadratique par morceaux sur $[0, t^{\max}]$ et sa forme change lorsqu'une composante de $\boldsymbol{x}^t$ change de signe. Il y a donc un nombre fini de points d'intérêt où évaluer f.

La fonctionnement est résumé dans l'algorithme 2. Chaque itération réduit la valeur de la fonction objectif et le processus converge vers un minimum global en un nombre fini d'itérations. Une preuve complète peut être trouvée dans [7] dans le cas standard de la pénalisation ℓ_1, qui s'étend sans problème aux spécificités de $\mathcal{R}^{(i)}$.

0. **Initialisation :** $\boldsymbol{x} \leftarrow \boldsymbol{x}^0$; $\boldsymbol{\theta} \leftarrow \mathrm{sgn}(\boldsymbol{x}^0)$; initialiser $\{\bar{\mathrm{S}}_{in}, \mathrm{S}^1_{in}, \bar{\mathrm{S}}_\square, \mathrm{S}^1_\square, \bar{\mathrm{S}}_0\}$ par (5).
1. **Répéter**
   — calculer $\boldsymbol{x}^{\text{new}}_{\bar{\mathrm{S}}_{in}}$ et $\boldsymbol{x}^{\text{new}}_{\mathrm{S}^1_{in}}$ par (9) ;
   — calculer $t^\star = \arg\min_{t \in [0, t^{\max}]} f(t)$;
   — $\boldsymbol{x} \leftarrow \boldsymbol{x} + t^\star(\boldsymbol{x}^{\text{new}} - \boldsymbol{x})$ et $\boldsymbol{\theta} \leftarrow \mathrm{sgn}(\boldsymbol{x})$;
   — mettre à jour le support par (5) ;
   jusqu'à $\mathrm{sgn}(\boldsymbol{x}) = \mathrm{sgn}(\boldsymbol{x}^{\text{new}})$ et $\|\boldsymbol{x}^{\text{new}}\|_\infty \leq M$.
2. **Si** les conditions d'optimalité (7) sont satisfaites, **alors** retourner $\boldsymbol{x}$ la solution optimale.
   Sinon,
   — choisir un indice $j \in \bar{\mathrm{S}}_0 \cup \bar{\mathrm{S}}_\square \cup \mathrm{S}^1_\square$ correspondant à une condition d'optimalité (7) violée ;
   — $\bar{\mathrm{S}}_{in} \leftarrow \bar{\mathrm{S}}_{in} \cup \{j\}$;
   — $\theta_j \leftarrow \mathrm{sgn}\,(-c_j)$ si $j \in \bar{\mathrm{S}}_0$,
   $\theta_j \leftarrow \mathrm{sgn}\,(x_j)$ si $j \notin \bar{\mathrm{S}}_0$; retourner à l'étape 1.

Algorithme 2 : Méthode *active set* pour résoudre $\mathcal{R}^{(i)}$.

Chaque itération consiste principalement à résoudre des systèmes linéaires dans (9), de taille $|\bar{\mathrm{S}}_{in}|$ et $|\mathrm{S}^1_{in}|$. Le support changeant peu entre deux itérations, on peut résoudre ces systèmes par des mises à jour récursives. Nous utilisons ici le lemme d'inversion des matrices partitionnées. Enfin, pour résoudre un sous-problème donné, nous initialisons l'algorithme à la solution du nœud parent, les deux problèmes ne différant que par une variable.

4 Résultats expérimentaux

Pour résoudre $\mathcal{P}^{(0)}$, nous insérons notre algorithme active-set de calcul de la relaxation $\mathcal{R}^{(i)}$ dans le schéma de *branch-and-bound* décrit en section 2, que nous appelons B&B$_{\text{R-Act}}$. Nous comparons les performances de cet algorithme à :

- ▷ la même stratégie d'exploration *branch-and-bound*, où la relaxation continue est résolue avec le solveur de programmation quadratique CPLEX 12.8 (noté B&B$_{\text{R-Cplex}}$) ;
- ▷ la résolution du problème MIQP équivalent [5, Table I] par le solveur MIQP CPLEX 12.8 (noté MIQP$_{\text{Cplex}}$).

Toutes les méthodes sont implémentées en C++. Les calculs sont limités à un seul cœur afin de se concentrer sur la performance de l'algorithme (capacités de parallélisation désactivées) ; le temps CPU maximum est fixé à 1 000 secondes.

Nous considérons des problèmes simulés de déconvolution parcimonieuse [5] [1], où $\boldsymbol{y} \in \mathbb{R}^{120}$, $\boldsymbol{x} \in \mathbb{R}^{100}$, et le nombre de composantes non nulles K varie de 5 à 9. Malgré leur faible taille, ces problèmes sont difficiles à résoudre et les méthodes parcimonieuses classiques fournissent de moins bonnes solutions que l'optimum global ℓ_0 [5]. Nous considérons également des problèmes simulés de sélection de variables avec des données aléatoires, où $\mathbf{H} \in \mathbb{R}^{500\times1000}$. Les coefficients de $\mathbf{H}$ sont gaussiens, le choix des coefficients non nuls est aléatoire et les amplitudes associées sont générées selon $u + \text{sgn}(u)$, où u est gaussien centré de variance unitaire, afin d'éviter des valeurs arbitrairement faibles. En raison de la nature aléatoire de $\mathbf{H}$, ce type de problème est plus facile à résoudre, c'est pourquoi nous pouvons monter en dimension. Pour les deux classes de problèmes, les données sont contaminées par un bruit blanc gaussien de rapport signal sur bruit 10 dB. Le paramètre λ est réglé de façon statistique en fonction de l'écart-type σ du bruit et du degré de parcimonie : $\lambda = 2\sigma^2 \log(\frac{N}{K} - 1)$ (voir par exemple [2]), et on fixe $M = 1.1\|\mathbf{H}^T\boldsymbol{y}\|_\infty$, comme dans [5].

Les résultats moyennés sur 50 instances sont consignés dans la Table 1. Par construction, B&B$_{\text{R-Act}}$ et B&B$_{\text{R-Cplex}}$ explorent le même nombre de nœuds, mais B&B$_{\text{R-Act}}$ est beaucoup plus rapide et résout plus d'instances en moins de 1 000s, démontrant l'efficacité de l'algorithme active-set. Sur les problèmes de déconvolution, le nombre de nœuds explorés est légèrement favorable à MIQP$_{\text{Cplex}}$. Rappelons que CPLEX, un des solveurs les plus efficaces, bénéficie de décennies de savoir-faire dans la résolution de MIQP, mettant en œuvre de nombreuses techniques non utilisées par B&B$_{\text{R-Act}}$. Malgré cela, le temps d'exécution est en faveur de B&B$_{\text{R-Act}}$, le temps de calcul par nœud étant bien inférieur. Sur les problèmes de sélection de variables, B&B$_{\text{R-Act}}$ est jusqu'à 80 fois plus rapide que MIQP$_{\text{Cplex}}$.

	K	Branch-and-bound dédié						Solveur MIQP		
		B&B$_{\text{R-Act}}$			B&B$_{\text{R-Cplex}}$			MIQP$_{\text{Cplex}}$		
		T (s)	Nds (10^3)	F	T (s)	Nds (10^3)	F	T (s)	Nds (10^3)	F
Déconv.	5	**0.7**	2.02	0	32.6	2.02	0	3.2	1.98	0
	7	**4.4**	10.22	0	187.3	10.22	7	7.4	9.61	0
	9	**17.1**	31.88	4	470.7	31.87	28	17.3	23.74	2
Sél. var.	5	**1.4**	0.06	0	275.5	0.06	10	118.1	0.11	0
	10	**3.9**	0.13	0	508.2	0.13	32	189.4	0.42	3
	15	**45.4**	0.51	8	-	-	50	665.3	2.15	40

TABLE 1 – Efficacité algorithmique pour des problèmes de déconvolution et de sélection de variables, en fonction du nombre K de variables non nulles. Temps de calcul (T), nombre de nœuds explorés (Nds) et nombre de problèmes non résolus en 1 000 s (F).

1. Données disponibles sur `pagesperso.ls2n.fr/~bourguignon-s/`

5 Conclusion

La reformulation sous la forme de *Mixed-Integer Quadratic Program* et l'usage d'un solveur permettent de résoudre de façon exacte certains problèmes d'optimisation en norme ℓ_0 [5]. Nous avons démontré que ces problèmes pouvaient bénéficier d'approches de résolution dédiées, permettant d'aborder des problèmes de plus grande taille. Notre méthode dépasse les performances d'un solveur MIQP pourtant réputé puissant, grâce à l'exploitation des structures mathématiques du problème, non prises en compte par les logiciels génériques. D'une part, nous avons proposé une stratégie d'exploration exploitant la parcimonie de la solution, privilégiant l'activation de variables non nulles dans le parcours de l'arbre de décision. Cette stratégie ne peut être mise en œuvre en utilisant un solveur comme CPLEX. D'autre part, en interprétant l'évaluation de chaque nœud comme un problème en norme ℓ_1, nous avons pu proposer une résolution dédiée, exploitant le savoir-faire développé en optimisation ℓ_1. Diverses perspectives se dégagent dans la poursuite de ces travaux. La construction de relaxations plus efficaces que la relaxation ℓ_1, le développement de stratégies d'exploration plus raffinées basées par exemple sur des algorithmes gloutons sur mesure [2] ou la recherches de méthode de coupe [6] dédiées à ce problème en sont quelques exemples.

Remerciements. Ce travail a été partiellement financé par l'ANR, projet JCJC MIMOSA ANR-16-CE33-0005.

Références

[1] C. Zala, "High-resolution inversion of ultrasonic traces," *IEEE Trans Ultrason Ferroelectr Freq Control*, vol. 39, no. 4, July 1992.

[2] C. Soussen, J. Idier, D. Brie, and J. Duan, "From Bernoulli Gaussian deconvolution to sparse signal restoration," *IEEE Trans Signal Process*, vol. 59, no. 10, 2011.

[3] B.K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM J Sci Comput*, vol. 24, no. 2, 1995.

[4] J. A. Tropp and S. J. Wright, "Computational methods for sparse solution of linear inverse problems," *Proc IEEE*, vol. 98, no. 6, June 2010.

[5] S. Bourguignon, J. Ninin, H. Carfantan, and M. Mongeau, "Exact sparse approximation problems via mixed-integer programming : Formulations and computational performance," *IEEE Trans Signal Process*, vol. 64, no. 6, March 2016.

[6] L. A. Wolsey, *Integer Programming*, Wiley, New York, NY, USA, 1998.

[7] H. Lee, A. Battle, R. Raina, and A.Y. Ng, "Efficient sparse coding algorithms," in *Adv Neural Inf Process Syst*, 2007, pp. 801–808.

[8] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, 1970.

[9] R. Ben Mhenni, S. Bourguignon, and J. Ninin, "Global optimization for sparse solution of least squares problems," Tech. Rep., École Centrale de Nantes, 2019.