Node-screening pour le problème des moindres carrés avec pénalité ℓ_0

Théo GUYARD[1], Cédric HERZET[2], Clément ELVIRA[3], Ayşe N. ARSLAN[1]

[1]Univ Rennes, INSA Rennes, CNRS, IRMAR-UMR 6625, F-35000 Rennes, France

[2]INRIA Rennes-Bretagne Atlantique, Campus de Beaulieu, 35000 Rennes, France

[3]IETR UMR CNRS 6164, CentraleSupélec Rennes Campus, 35576 Cesson Sévigné, France

prénom.nom@{insa-rennes,inria,centralesupelec}.fr

Résumé – Dans cet article, nous présentons une nouvelle méthode de *node-screening* permettant d'accélérer un algorithme de *Branch and Bound (BnB)* résolvant le problème des moindres carrés avec pénalité ℓ_0. Notre contribution est un ensemble de tests permettant de détecter des solutions réalisables qui ne peuvent pas être optimales. Cela permet d'élaguer des nœuds au cours du BnB, réduisant ainsi le temps de résolution.

Abstract – We present a novel node-screening methodology to safely discard irrelevant nodes within a generic Branch-and-Bound (BnB) algorithm solving the ℓ_0-penalized least-squares problem. Our contribution is a pair of simple tests to detect sets of feasible vectors that cannot yield optimal solutions. This allows to prune nodes of the BnB search tree, thus reducing the overall optimization time.

1 Introduction

Trouver une représentation parcimonieuse est un problème fondamental dans le domaine des statistiques, de l'apprentissage automatique ou des problèmes inverses. Cela consiste à décomposer un vecteur de données $\mathbf{y} \in \mathbb{R}^m$ comme une combinaison linéaire de colonnes (ou *atomes*) d'un dictionnaire $\mathbf{A} \in \mathbb{R}^{m\times n}$. Cette tâche peut être effectuée en résolvant

$$\min_{\mathbf{x}\in\mathbb{R}^n} \ \tfrac{1}{2}\|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_0 \qquad (1)$$

où $\|\mathbf{x}\|_0$ compte le nombre d'entrées non nulles dans $\mathbf{x}$ et où $\lambda > 0$ est un paramètre à calibrer. Malheureusement, le problème (1) est NP-difficile dans le cas général [1]. Cela a conduit les chercheurs à développer des procédures sous-optimales pour approcher sa solution afin de traiter des problèmes en grande dimension. Parmi les plus populaires, on peut citer les algorithmes dits "gloutons" et les méthodologies basées sur la relaxation convexe de la norme ℓ_0 [2, Section 3].

Ces procédures sous-optimales ne sont toutefois garanties de résoudre (1) que sous des conditions restrictives qui sont rarement satisfaites en pratique. Cependant, il y a eu un récent regain d'intérêt pour les méthodes résolvant (1) exactement, comme par exemple dans [3, 4]. De nombreuses approches utilisent le fait que le problème

$$p^\star = \min_{\mathbf{x}\in\mathbb{R}^n} \ \tfrac{1}{2}\|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_0 \ \text{ t.q. } \ \|\mathbf{x}\|_\infty \leq M \qquad (P)$$

est équivalent à (1) à condition que M soit choisi suffisamment grand, et où $\|\cdot\|_\infty$ correspond à la norme ℓ_∞. Le problème (P) peut alors être reformulé comme un *"Mixed-Integer program"* (MIP) en introduisant des variables binaires codant la nullité des entrées de $\mathbf{x}$ [4]. Il a été montré numériquement que (P) pouvait être résolu pour des problèmes de taille modérée.

Dans un article récent, Atamtürk et Gómez ont étendu la notion de *screening*, introduite par El Ghaoui *et al.* dans [5], aux problèmes non convexes favorisant des solutions parcimonieuses. Plus précisément, les auteurs ont décrit une nouvelle méthodologie permettant de détecter (certaines) entrées nulles et non nulles dans les minimiseurs d'un problème parcimonieux non convexe particulier [6]. Leur méthodologie est ensuite utilisée comme une étape de pré-traitement et permet de réduire la dimensionnalité du problème.

Dans cet article, nous améliorons la méthode proposée dans [6] en introduisant de nouveaux tests de "*node-screening*" qui permettent d'élaguer des *nœuds* dans l'arbre de BnB. Contrairement à [6], une propriété *d'imbrication* entre les tests de *node-screening* à différents nœuds permet potentiellement d'élaguer de *multiples* branches de l'arbre à *n'importe quelle étape* du processus d'optimisation, et ce à un coût marginal.

Notre article est organisé comme suit. Dans la Section 2, nous décrivons un algorithme de BnB adapté à (P). La Section 3 présente nos nouveaux tests de *node-screening* et explique comment les implémenter efficacement dans le BnB. Dans la Section 4, nous évaluons les performances de notre méthode. Les détails techniques et les preuves de nos résultats sont accessibles dans le document suivant [7].

Notations. Dans cet article, les matrices et vecteurs sont respectivement représentées par des lettres grasses majuscules et minuscules. $\mathbf{0}$ désigne le vecteur nul. La i-ème colonne d'une matrice $\mathbf{A}$ est notée $\mathbf{a}_i$. De même, la i-ème entrée d'un vecteur $\mathbf{x}$ est notée x_i. Les lettres calligraphiques sont utilisées pour désigner les ensembles et la notation $|\cdot|$ fait référence à leur cardinalité. De plus, $\mathbf{x}_\mathcal{S}$ désigne la restriction de $\mathbf{x}$ à ses éléments indexés par $\mathcal{S}$. De même, $\mathbf{A}_\mathcal{S}$ correspond à la restriction de $\mathbf{A}$ à ses colonnes indexées par $\mathcal{S}$.

2 Algorithmes de *Branch and Bound*

Les algorithmes de BnB désignent une famille de méthodes pour résoudre (entres autres) des problèmes MIPs. Ces approches consistent à énumérer (implicitement) toutes les solutions réalisables et à appliquer des règles pour détecter les candidats non pertinents. Particularisé au problème (P), un BnB revient à parcourir un arbre de décision binaire où chaque nœud correspond à une nouvelle décision quant à la nullité (ou non) d'une entrée de la variable $\mathbf{x}$. Ce principe est illustré par la Figure 1a. Formellement, nous définissons un nœud comme un triplet $\nu = (\mathcal{S}_0, \mathcal{S}_1, \bar{\mathcal{S}})$ où $\mathcal{S}_0$ et $\mathcal{S}_1$ contiennent les indices des entrées de $\mathbf{x}$ qui sont forcées à être nulles et non nulles et où $\bar{\mathcal{S}}$ rassemble toutes les entrées de $\mathbf{x}$ pour lesquelles aucune décision n'a encore été prise. Dans ce formalisme, le principe d'un algorithme BnB est le suivant : à partir du nœud $\nu = (\emptyset, \emptyset, \{1, \dots, n\})$, la méthode alterne entre le traitement du nœud actuel (évaluation ou *bounding*) et la sélection d'un nouveau nœud (branchement ou *branching*). L'algorithme identifie le minimum global du problème en un nombre fini d'étapes avec une complexité au pire cas égale à celle d'une recherche exhaustive dans l'arbre. En pratique, son efficacité dépend à la fois du nombre de nœuds explorés et de la capacité à les traiter rapidement.

2.1 Étape d'*évaluation*

Lors du traitement du nœud $\nu = (\mathcal{S}_0, \mathcal{S}_1, \bar{\mathcal{S}})$, on teste si un vecteur $\mathbf{x}$ avec des zéros sur $\mathcal{S}_0$ et des valeurs non nulles sur $\mathcal{S}_1$ peut être un minimiseur global de (P). Plus précisément, on cherche la plus petite valeur de la fonction objectif atteinte par les candidats vérifiant ces contraintes, *i.e.*, la valeur de

$$\begin{aligned} p^\nu \;=\; \min_{\mathbf{x}\in\mathbb{R}^n} \tfrac{1}{2}\|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}_{\bar{\mathcal{S}}}\|_0 + \lambda|\mathcal{S}_1| \\ \text{t.q. } \|\mathbf{x}\|_\infty \le M \;\text{ et }\; \mathbf{x}_{\mathcal{S}_0} = \mathbf{0}. \end{aligned} \tag{P^ν}$$

On a $p^\nu \ge p^\star$ puisque tous les minimiseurs de (P^ν) sont également réalisables pour (P). De plus, l'égalité est vérifiée dès lors que les contraintes induites par $\mathcal{S}_0$ et $\mathcal{S}_1$ correspondent à la configuration de l'un des minimiseurs de (P).

Si $p^\nu > p^\star$, le nœud ν peut être élagué de l'arbre puisqu'aucun vecteur $\mathbf{x}$ vérifiant les contraintes définies à ce nœud (et donc à tout descendant) ne peut aboutir à un minimum global. Malheureusement, cette règle d'élagage présente un faible intérêt pratique puisque $p^\star$ n'est pas connu à l'avance. De plus, évaluer p^ν lorsque $\bar{\mathcal{S}} \neq \emptyset$ est également un problème NP-difficile. Il est toutefois possible de concevoir une version relâchée de ce test : notons respectivement p_l^ν et p_u des bornes inférieure et supérieure de p^ν et $p^\star$. Alors, une condition suffisante pour élaguer le nœud ν est de tester si $p_l^\nu > p_u$.

Pour obtenir p_l^ν, une approche standard consiste à résoudre

$$\begin{aligned} p_l^\nu \;=\; \min_{\mathbf{x}\in\mathbb{R}^n} \tfrac{1}{2}\|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2 + \tfrac{\lambda}{M}\|\mathbf{x}_{\bar{\mathcal{S}}}\|_1 + \lambda|\mathcal{S}_1| \\ \text{t.q. } \|\mathbf{x}\|_\infty \le M \;\text{ et }\; \mathbf{x}_{\mathcal{S}_0} = \mathbf{0} \end{aligned} \tag{P_l^ν}$$

où la norme ℓ_0 a été remplacée par une norme ℓ_1. On a bien $p_l^\nu \le p^\nu$ puisque $M^{-1}\|\mathbf{x}_{\bar{\mathcal{S}}}\|_1 \le \|\mathbf{x}_{\bar{\mathcal{S}}}\|_0$ pour tout vecteur réalisable $\mathbf{x}$. Remarquons que (P_l^ν) est un problème LASSO contraint qui peut être résolu en temps polynomial en utilisant l'une des nombreuses méthodes adaptées à cette classe de problèmes [2, Chap. 15].

La résolution des problèmes relâchés (P_l^ν) permet d'obtenir à faible coût des valeurs pertinentes de p_u. Plus précisément, chaque évaluation de la fonction objectif de (P^ν) fournit une borne supérieure p_u^ν de p^ν. La valeur de p_u est alors mise à jour selon $p_u \leftarrow \min(p_u, p_u^\nu)$. Au cours de la procédure de BnB, p_u converge vers $p^\star$ et les relaxations sont renforcées à mesure que de nouvelles variables sont fixées, ce qui permet d'élaguer les nœuds de manière plus efficace.

2.2 Étape de *branchement*

Si le nœud $\nu = (\mathcal{S}_0, \mathcal{S}_1, \bar{\mathcal{S}})$ n'a pas été élagué pendant l'étape d'*évaluation*, l'exploration de l'arbre se poursuit. Un indice $i \in \bar{\mathcal{S}}$ est sélectionné selon une certaine règle et deux descendants sont créés sous le nœud ν en imposant soit "$x_i = 0$", soit "$x_i \neq 0$". Lorsque tous les nœuds ont été soit explorés, soit élagués, l'algorithme de BnB s'arrête et tout candidat donnant la meilleure borne supérieure p_u est un minimiseur de (P).

Le lecteur peut trouver une implémentation efficace de BnB résolvant (P) dans [4]. C'est cette implémentation que nous considérons dans la Section 4 pour générer nos résultats.

3 Tests de *node-screening*

Dans cette section, nous présentons nos nouveaux tests de *node-screening*. Ils visent à identifier, à faible coût, les nœuds de l'arbre de BnB qui ne peuvent pas mener à un minimiseur global de (P).

3.1 Problème dual

Notre contribution repose sur l'identification d'une connexion simple entre les fonctions objectif des problèmes duaux associés à (P_l^ν) à deux nœuds consécutifs. Avant d'exposer ce résultat, définissons pour tout $\mathbf{u} \in \mathbb{R}^m$ et $i \in \{1, \dots, n\}$ trois familles de valeurs *pivots* de la manière suivante

$$\begin{aligned} \boldsymbol{\gamma}_i(\mathbf{u}) &\triangleq M(|\mathbf{a}_i^{\mathrm{T}}\mathbf{u}| - \tfrac{\lambda}{M}) \\ \boldsymbol{\gamma}_i^0(\mathbf{u}) &\triangleq M[|\mathbf{a}_i^{\mathrm{T}}\mathbf{u}| - \tfrac{\lambda}{M}]_+ \\ \boldsymbol{\gamma}_i^1(\mathbf{u}) &\triangleq M[\tfrac{\lambda}{M} - |\mathbf{a}_i^{\mathrm{T}}\mathbf{u}|]_+ \end{aligned} \tag{2}$$

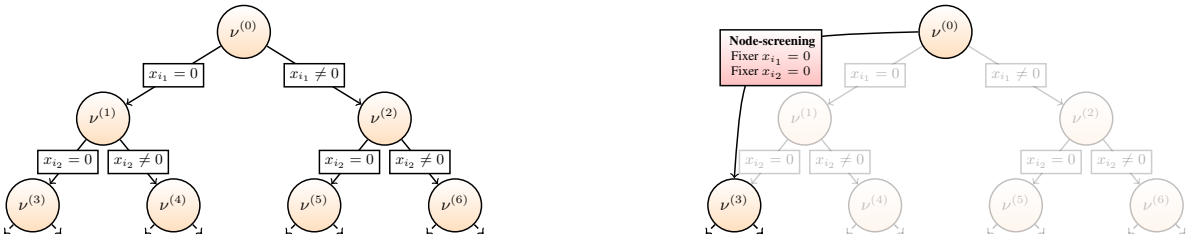
où $[z]_+ \triangleq \max(0, z)$ pour tout scalaire z. Nous exprimons maintenant le problème dual de (P_l^ν).

Proposition 1. *Le problème dual de* (P_l^ν) *est donné par*

$$\begin{aligned} d_l^\nu \;=\; \max_{\mathbf{u}\in\mathbb{R}^m} \mathrm{D}_l^\nu(\mathbf{u}) \triangleq \tfrac{1}{2}\|\mathbf{y}\|_2^2 - \tfrac{1}{2}\|\mathbf{y}-\mathbf{u}\|_2^2 \\ - \textstyle\sum_{i\in\bar{\mathcal{S}}} \boldsymbol{\gamma}_i^0(\mathbf{u}) - \sum_{i\in\mathcal{S}_1} \boldsymbol{\gamma}_i(\mathbf{u}) \end{aligned} \tag{D_l^ν}$$

et la propriété de dualité forte est vérifiée pour (P_l^ν)*-*(D_l^ν)*, i.e.,* $p_l^\nu = d_l^\nu$*. De plus, si* $(\mathbf{x}_l^\star, \mathbf{u}_l^\star)$ *est un couple de solutions primale-duale, on a*

$$\mathbf{u}_l^\star = \mathbf{y} - \mathbf{A}\mathbf{x}_l^\star. \tag{3}$$



(a) (b)

FIGURE 1 – Premiers nœuds explorés dans l'arbre de BnB. Le nœud $\nu^{(0)}$ correspond au problème (P) et chacun de ses descendants correspond au problème (P^ν) avec différentes variables fixées. (a) Implémentation standard d'un BnB où tous les nœuds sont traités. (b) Impact de l'application des tests de *node-screening* sur l'arbre de recherche : au nœud $\nu^{(0)}$, le test (6b) est passé pour les entrées i_1 et i_2 ; on peut donc aller directement au descendant incluant les contraintes "$x_{i_1} = 0$" et "$x_{i_2} = 0$", à savoir $\nu^{(3)}$. Les nœuds transparents ne sont donc pas explorés.

Par conséquent, à un nœud donné ν, l'objectif du dual de (P_l^ν) est la somme d'un terme commun à tous les nœuds et de valeurs pivots. Il est intéressant de noter que la fonction objectif ne diffère que d'une seule valeur pivot entre deux nœuds consécutifs, comme le montre le résultat suivant :

Corollaire 1. *Soit $\nu = (\mathcal{S}_0, \mathcal{S}_1, \bar{\mathcal{S}})$ et $i \in \bar{\mathcal{S}}$, alors $\forall \mathbf{u} \in \mathbb{R}^m$,*

$$\mathrm{D}_l^{\nu \cup \{x_i = 0\}}(\mathbf{u}) = \mathrm{D}_l^\nu(\mathbf{u}) + \boldsymbol{\gamma}_i^0(\mathbf{u}) \tag{4a}$$

$$\mathrm{D}_l^{\nu \cup \{x_i \neq 0\}}(\mathbf{u}) = \mathrm{D}_l^\nu(\mathbf{u}) + \boldsymbol{\gamma}_i^1(\mathbf{u}) \tag{4b}$$

où $\nu \cup \{x_i = 0\}$ et $\nu \cup \{x_i \neq 0\}$ sont les deux descendants de ν pour lesquels l'indice i a été échangé de $\bar{\mathcal{S}}$ à $\mathcal{S}_0$ ou à $\mathcal{S}_1$.

3.2 Tests de *node-screening*

Nous exposons maintenant notre stratégie de *node-screening* permettant d'identifier des nœuds de l'arbre qui ne peuvent pas mener à un minimiseur global de (P). Soit ν un nœud et soit p_u une borne supérieure sur $p^\star$. Par dualité forte entre (P_l^ν)-(D_l^ν),

$$\forall \mathbf{u} \in \mathbb{R}^m, \quad \mathrm{D}_l^\nu(\mathbf{u}) \leq d_l^\nu = p_l^\nu \leq p^\nu. \tag{5}$$

En combinant (5) avec le Corollaire 1, on a le résultat suivant :

Proposition 2. *Soit $i \in \bar{\mathcal{S}}$ un indice. Alors, $\forall \mathbf{u} \in \mathbb{R}^m$,*

$$\mathrm{D}_l^\nu(\mathbf{u}) + \boldsymbol{\gamma}_i^0(\mathbf{u}) > p_u \quad \Longrightarrow \quad p^{\nu \cup \{x_i = 0\}} > p^\star \tag{6a}$$

$$\mathrm{D}_l^\nu(\mathbf{u}) + \boldsymbol{\gamma}_i^1(\mathbf{u}) > p_u \quad \Longrightarrow \quad p^{\nu \cup \{x_i \neq 0\}} > p^\star. \tag{6b}$$

Autrement dit, la Proposition 2 décrit une procédure simple pour identifier certains descendants de ν qui ne peuvent pas donner une solution optimale de (P). Ils peuvent donc être élagués. Si les tests (6a) et (6b) sont tout deux passés pour un indice donné, aucun vecteur admissible par rapport aux contraintes définies au nœud ν ne peut être un minimiseur de (P). Le nœud ν peut donc être lui-même élagué.

La procédure que nous proposons diffère de la méthodologie d'élagage décrite dans la Section 2. L'évaluation des tests ne nécessite que le calcul d'un seul produit scalaire. Ils peuvent donc être implémentés avec un coût marginal par rapport au coût global de l'étape d'évaluation. Ils bénéficient également de la propriété d'imbrication suivante :

Corollaire 2. *Soit $\nu = (\mathcal{S}_0, \mathcal{S}_1, \bar{\mathcal{S}})$ un nœud. Si le test* (6a) *ou* (6b) *est passé pour l'indice $i \in \bar{\mathcal{S}}$, alors il l'est également pour tout descendant $\nu' = (\mathcal{S}_0', \mathcal{S}_1', \bar{\mathcal{S}}')$ de ν tel que $i \in \bar{\mathcal{S}}'$.*

En particulier, le Corollaire 2 implique que si deux indices distincts i et i' passent le test (6a) au nœud ν, alors l'indice i' passera également le test (6a) au nœud $\nu \cup \{x_i \neq 0\}$ (pour la même valeur de $\mathbf{u}$). La même conséquence s'applique pour le test (6b) et pour le nœud $\nu \cup \{x_i = 0\}$. En d'autres termes, si plusieurs tests de *node-screening* passent à un nœud donné, on peut élaguer *simultanément* plusieurs descendants de ν, comme l'illustre la Figure 1b. Cela contraste avec la procédure d'évaluation décrite dans la Section 2 qui doit traiter chaque descendant individuellement avant un potentiel élagage.

3.3 Implémentation

L'implémentation des tests de *node-screening* décrits dans la Proposition 2 nécessite la connaissance d'une borne supérieure p_u sur $p^\star$ et d'un vecteur $\mathbf{u} \in \mathbb{R}^m$ approprié. La valeur de p_u peut être obtenue sans coût via l'étape d'évaluation. Nous obtenons un candidat pertinent pour $\mathbf{u}$ de la manière suivante : en supposant que la méthode utilisée pour résoudre (P_l^ν) génère une séquence d'itérés $\{\mathbf{x}^{(t)}\}_{t \in \mathbb{N}}$, nous utilisons le vecteur dual défini par

$$\forall\, t \in \mathbb{N}, \quad \mathbf{u}^{(t)} = \mathbf{y} - \mathbf{A}\mathbf{x}^{(t)}. \tag{7}$$

Ce choix implique que les performances des tests de *node-screening* dépendent de la séquence $\{\mathbf{u}^{(t)}\}_{t \in \mathbb{N}}$ générée. Néanmoins, il permet de bénéficier de deux propriétés intéressantes. Premièrement, la séquence $\{\mathbf{u}^{(t)}\}_{t \in \mathbb{N}}$ converge vers un maximiseur de (D_l^ν) de par la condition d'optimalité (3). Deuxièmement, $\mathbf{u}^{(t)}$ et $\mathbf{A}^{\mathrm{T}}\mathbf{u}^{(t)}$ sont déjà calculés par la plupart des méthodes de résolution de (P_l^ν) car ils correspondent respectivement à l'erreur résiduelle et à l'opposé du gradient de la partie quadratique de la fonction objectif. Ainsi, l'évaluation des tests de *node-screening* pour n'importe quelle entrée de $\bar{\mathcal{S}}$ n'ajoute aucune complexité supplémentaire à l'algorithme de BnB. Le choix de $\mathbf{u}^{(t)}$ suggère d'effectuer des tests de *node-screening* au sein de l'étape d'évaluation. Ainsi, lorsqu'une entrée de $\mathbf{x}$ passe un test au nœud ν, l'algorithme de BnB passe immédiatement à un descendant, sans avoir besoin de terminer la résolution de (P_l^ν).

4 Résultats numériques

Cette section décrit des résultats de simulation illustrant la pertinence de la méthodologie de *node-screening*.

4.1 Schémas expérimentaux

Nous traitons un problème d'estimation de directions d'arrivée. Dans ce problème, on considère un réseau linéaire uniforme de m antennes mesurant un signal provenant de k sources situées à des angles différents par rapport à l'orientation du réseau. Chaque antenne fournit une mesure exprimée comme la somme des k signaux. Pour une source donnée, le signal capté par deux antennes différentes est légèrement déphasé à cause de la distance qui les sépare. L'objectif est de retrouver les angles d'incidence des signaux grâce à ce déphasage.

Pour modéliser ce problème, on peut construire une discrétisation en n éléments des potentiels angles d'arrivée des signaux. Le vecteur $\mathbf{x}^0 \in \mathbb{R}^n$ représente les "vrais" angles des k signaux : l'indice i de $\mathbf{x}^0$ est non-nul si et seulement si le i-ème élément de la discrétisation correspond effectivement à un des angles des signaux arrivant sur le réseau d'antennes. Ensuite, on peut construire une matrice $\mathbf{A} \in \mathbb{R}^{m\times n}$ où chaque ligne correspond à une des m antennes et où chaque colonne correspond à un des n éléments de la discrétisation. Chaque entrée a_{ji} de la matrice $\mathbf{A}$ correspond à la valeur du déphasage correspondant au i-ème angle de la discrétisation arrivant sur la j-ème antenne du réseau linéaire. Le signal $\mathbf{y} \in \mathbb{R}^m$ résultant des k signaux captés est alors

$$\mathbf{y} = \mathbf{A}\mathbf{x}^0 + \boldsymbol{\epsilon} \tag{8}$$

où $\boldsymbol{\epsilon} \in \mathbb{R}^m$ représente un bruit induit par la mesure. On va donc résoudre (P) pour retrouver $\mathbf{x}^0$ à partir de $\mathbf{y}$ et $\mathbf{A}$. Nous mentionnons qu'il existe également de nombreuses méthodes heuristiques permettant d'approcher la solution de (P) pour ce type d'application [2, Chap. 3-4].

Dans notre scénario de simulation, nous fixons $(m, n) = (300, 1000)$, k est choisi parmi $\{5, 10, 15\}$ et les entrées non-nulles de $\mathbf{x}^0$ sont fixées aléatoirement dans $\{-1, +1\}$. Nous générons également un bruit blanc gaussien $\boldsymbol{\epsilon}$ avec une variance menant un rapport signal-bruit de 10dB ou de 20dB. Enfin, nous réglons λ et M en suivant le protocole décrit dans [4]. Nous comparons trois méthodes de résolution pour (P) : *i)* `Direct`, qui utilise le solveur commercial CPLEX ; *ii)* `BnB`, l'algorithme de BnB présenté dans [4] ; *iii)* `BnB+scr` qui correspond à `BnB` amélioré avec notre méthode de *node-screening*. La version 20.1 de CPLEX est utilisée. `BnB` et `BnB+scr` sont implémentés en Julia v1.6. Les résultats sont moyennés sur 100 instances du problème (P).

4.2 Comparaison des méthodes

La Table 1 compare le nombre moyen de nœuds traités (*i.e.* le nombre de relaxations (P_l^ν) résolues) et le temps de résolution pour les trois méthodes. On observe que `BnB+scr` obtient toujours les meilleures performances. On constate également que, par rapport à `BnB`, la réduction du temps de résolution est plus importante que la réduction du nombre de nœuds traités. Un examen approfondi de nos résultats indique que l'étape d'*évaluation* est effectuée d'autant plus rapidement que de nombreuses variables sont fixées à zéro dans (P_l^ν). Les tests de *node-screening* permettent d'atteindre rapidement des nœuds de l'arbre où la plupart des variables sont déjà fixées à zéro, ce qui accélère d'autant plus la résolution du problème.

		Direct			BnB			BnB+scr		
	k	N	T	F	N	T	F	N	T	F
10dB	5	64	528	0	48	3	0	**33**	**0**	**0**
	10	985	2753	1	421	86	0	**342**	**38**	**0**
	15	4601	10034	71	4385	1529	32	**3871**	**918**	**24**
20dB	5	57	226	0	42	5	0	**29**	**1**	**0**
	10	542	3830	1	180	35	1	**145**	**14**	**0**
	15	3073	8846	66	2984	1936	7	**2362**	**982**	**4**

TABLE 1 – Nombre de nœuds explorés (N), temps de résolution en secondes (T) et nombre d'instances non résolues sous 3 heures (F).

5 Conclusion

Dans cet article, nous avons présenté une nouvelle méthodologie visant à accélérer la résolution du problème des moindres carrés avec pénalité ℓ_0. Notre contribution tire parti d'une propriété d'imbrication entre des tests de *node-screening* à deux nœuds consécutifs. Cela conduit à des améliorations significatives en termes de nombre de nœuds explorés et de temps de résolution de l'algorithme de BnB.

Références

[1] X. CHEN et al. « Complexity of unconstrained $\ell_2 - \ell_p$ minimization ». *Mathematical Programming* 143.1-2 (2014), p. 371-383.

[2] S. FOUCART et H. RAUHUT. *A Mathematical Introduction to Compressive Sensing*. Springer, 2013.

[3] D. BERTSIMAS, A. KING et R. MAZUMDER. « Best subset selection via a modern optimization lens ». *The Annals of Statistics* 44.2 (2016), p. 813-852.

[4] R. MHENNI et al. « Sparse branch and bound for exact optimization of ℓ_0-norm penalized least squares ». *ICASSP*. IEEE. 2020, p. 5735-5739.

[5] L. EL GHAOUI, V. VIALLON et T. RABBANI. « Safe feature elimination for the lasso and sparse supervised learning problems ». *Pacific Journal of Optimization* 8.4 (2010), p. 667-698.

[6] A. ATAMTURK et A. GÓMEZ. « Safe screening rules for L0-regression from perspective relaxations ». *ICML*. PMLR. 2020, p. 421-430.

[7] T. GUYARD, C. HERZET et C. ELVIRA. « Node-screening tests for l0-penalized least-squares problem with supplementary material ». *arXiv preprint arXiv :2110.07308* (2021).