Safe Peeling for ℓ_0-Regularized Least-Squares

Théo Guyard*†, Gilles Monnoyer‡, Clément Elvira§ and Cédric Herzet*

*Inria, Centre de l'Université de Rennes, France
†IRMAR UMR CNRS 6625, INSA Rennes, France
‡ELEN, ICTEAM, UCLouvain, Belgium
§IETR UMR CNRS 6164, CentraleSupelec Rennes Campus, France

***Abstract*—We introduce a new methodology dubbed *"safe peeling"* to accelerate the resolution of ℓ_0-regularized least-squares problems via a Branch-and-Bound (BnB) algorithm. Our procedure enables to tighten the convex relaxation considered at each node of the BnB decision tree and therefore potentially allows for more aggressive pruning. Numerical simulations show that our proposed methodology leads to significant gains in terms of number of nodes explored and overall solving time.**

***Index Terms*—Sparse model, ℓ_0 regularization, Branch-and-Bound algorithm.**

I. Introduction

This paper focuses on the resolution of the so-called "ℓ_0-regularized least-squares" problem given by

$$p^\star = \min_{\mathbf{x}\in\mathbb{R}^n} \ \mathrm{P}(\mathbf{x}) \triangleq \tfrac{1}{2}\|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_0 \qquad (1\text{-}\mathcal{P})$$

where $\mathbf{y}\in\mathbb{R}^m$ and $\mathbf{A}\in\mathbb{R}^{m\times n}$ are input data, $\lambda>0$ is a regularization parameter and $\|\cdot\|_0$ denotes the ℓ_0-pseudonorm which counts the number of non-zero elements in its argument.

Solving (1-$\mathcal{P}$) is of paramount interest in many scientific fields such as statistics, machine learning or inverse problems [1–3]. Unfortunately, this problem also turns out to be NP-hard [4, Th. 1]. Hence, the last decades have seen a flurry of contributions proposing tractable procedures able to recover approximate solutions of (1-$\mathcal{P}$). Canonical examples include greedy algorithms or methodologies based on convex relaxations, see [5, Ch. 3]. Although these procedures successfully recover the actual solutions of (1-$\mathcal{P}$) in "easy" setups, they usually fall short for more challenging instances of the problem. This observation, combined with some recent advances in integer optimization and hardware performance, has revived the interest in methods solving (1-$\mathcal{P}$) exactly. A standard approach is to use a Branch-and-Bound (BnB) algorithm that solves (1-$\mathcal{P}$), see [6–11].

In this paper, we propose a new strategy, dubbed *"safe peeling"*, to accelerate the exact resolution of (1-$\mathcal{P}$). In a nutshell, our contribution is a computationally simple test applied at each node of the BnB decision tree to identify some intervals of $\mathbb{R}^n$ which cannot contain a solution of (1-$\mathcal{P}$). This information allows to construct tighter convex relaxations and more aggressive pruning of the nodes of the decision tree. Our numerical experiments show that the proposed method leads to a significant reduction of the solving time as compared to state-of-the-art concurrent methods. The name *"safe peeling"* comes from the fact that the proposed method enables to reduce (or in more figurative terms, "to peel") the feasible set of the problem at each node of the decision tree while safely preserving the correctness of the BnB procedure.

The rest of the paper is organized as follows. Sec. III describes the main ingredients of BnB methods. Our peeling strategy is presented in Sec. IV and its performance is illustrated in Sec. V. All the proofs are postponed to the technical report accompanying this paper [12].

Gilles Monnoyer is funded by the Belgian FNRS. The research presented in this paper is reproducible. The code associated to our numerical experiments is available at https://github.com/TheoGuyard/BnbPeeling.jl.

II. Notations

We use the following notations. $\mathbf{0}$ and $\mathbf{1}$ denote the all-zero and all-one vectors. The i-th column of a matrix $\mathbf{A}$ is denoted $\mathbf{a}_i$ and the i-th entry of a vector $\mathbf{x}$ is denoted x_i. The superscript $^{\mathrm{T}}$ refers to transposition. Any vectorial relation has to be understood component-wise, *e.g.,* $\mathbf{x}\in[\mathbf{l},\mathbf{u}]$ means $x_i\in[l_i,u_i],\forall i$. Moreover, $\eta(\cdot)$ denotes the indicator function which equals to 0 if the condition in argument is fulfilled and to $+\infty$ otherwise, $[x]_+=\max(x,0)$ refers to the positive-part function and $|\cdot|$ denotes the cardinality of a set. Finally, $[\![1,n]\!]$ with $n\in\mathbb{N}^*$ is a short-hand notation for the set $\{1,\dots,n\}$.

III. Principles of BnB Methods

In this section, we recall the main principles of BnB procedures. Due to space limitation, we only review the elements of interest to introduce the proposed peeling method. We refer the reader to [13, Ch. 7] for an in-depth treatment of the subject.

A. Pruning

The crux of BnB methods consists in identifying and discarding some subsets of $\mathbb{R}^n$ which do not contain a minimizer of (1-$\mathcal{P}$). To do so, one constructs a decision tree in which each node corresponds to a particular subset of $\mathbb{R}^n$. In our context, a tree node is identified by two disjoint subsets of $[\![1,n]\!]$, say ν_0 and ν_1. The goal at node $\nu\triangleq(\nu_0,\nu_1)$ is to detect whether a solution of (1-$\mathcal{P}$) can be attained within

$$\mathcal{X}^\nu \triangleq \{\mathbf{x}\in\mathbb{R}^n \colon \mathbf{x}_{\nu_0}=\mathbf{0},\ \mathbf{x}_{\nu_1}\neq\mathbf{0}\}, \qquad (2)$$

where $\mathbf{x}_{\nu_k}$ denotes the restriction of $\mathbf{x}$ to its elements in ν_k. In particular, let $\mathcal{X}^\star$ be the non-empty set of minimizers of

(1-$\mathcal{P}$). Then, if some upper bound $\bar{p}$ on the optimal value $p^\star$ is known and if we let

$$p^\nu \triangleq \inf_{\mathbf{x}\in\mathbb{R}^n} \mathrm{P}^\nu(\mathbf{x}) \tag{3}$$

with $\mathrm{P}^\nu(\mathbf{x}) \triangleq \mathrm{P}(\mathbf{x}) + \eta(\mathbf{x} \in \mathcal{X}^\nu)$, we obtain the implication

$$p^\nu > \bar{p} \implies \mathcal{X}^\nu \cap \mathcal{X}^\star = \emptyset. \tag{4}$$

In words, if the left-hand side of (4) is satisfied, $\mathcal{X}^\nu$ does not contain any solution of (1-$\mathcal{P}$) and can therefore be discarded from the search space of the optimization problem. This operation is usually referred to as "*pruning*".

B. Bounding and relaxing

Making a pruning decision at node ν requires the knowledge of $\bar{p}$ and p^ν. On the one hand, finding $\bar{p}$ is an easy task since the value of the objective function in (1-$\mathcal{P}$) at any feasible point constitutes an upper bound on $p^\star$. On the other hand, evaluating p^ν is NP-hard. This issue can nevertheless be circumvented by finding a tractable lower bound r^ν on p^ν and relaxing (4) as

$$r^\nu > \bar{p} \implies \mathcal{X}^\nu \cap \mathcal{X}^\star = \emptyset. \tag{5}$$

One ubiquitous approach in the literature [7, 9, 14] to find such a lower bound consists in:

i) Adding an extra term "$\eta(\mathbf{x} \in [\mathbf{l}, \mathbf{u}])$" to the cost function of (1-$\mathcal{P}$), for some *well-chosen* bounds $\mathbf{l} \in \mathbb{R}^n_-$ and $\mathbf{u} \in \mathbb{R}^n_+$.[1] In particular, the new constraint "$\mathbf{x} \in [\mathbf{l}, \mathbf{u}]$" must lead to a problem fully equivalent to (1-$\mathcal{P}$), that is

$$\forall \mathbf{x}^\star \in \mathcal{X}^\star : \ \mathbf{x}^\star \in [\mathbf{l}, \mathbf{u}]. \tag{6}$$

ii) Exploiting the convex relaxation of the function $\|\cdot\|_0$ on the bounded set $\mathcal{X}^\nu \cap [\mathbf{l}, \mathbf{u}]$, given by

$$\|\mathbf{x}\|_0 \geq |\nu_1| + \sum_{i\in\nu_\bullet} \frac{[x_i]_+}{u_i} - \frac{[-x_i]_+}{l_i}, \tag{7}$$

with $\nu_\bullet \triangleq [\![1, n]\!] \backslash (\nu_0 \cup \nu_1)$ and the convention "$0/0 = 0$".

On the one hand, item *i)* implies that the pruning test (4) involves the following quantity (rather than p^ν):

$$p^\nu(\mathbf{l}, \mathbf{u}) = \inf_{\mathbf{x}\in\mathbb{R}^n} \mathrm{P}^\nu(\mathbf{x}; \mathbf{l}, \mathbf{u}) \tag{8-$\mathcal{P}^\nu$}$$

where $\mathrm{P}^\nu(\mathbf{x}; \mathbf{l}, \mathbf{u}) \triangleq \mathrm{P}^\nu(\mathbf{x}) + \eta(\mathbf{x} \in [\mathbf{l}, \mathbf{u}])$. On the other hand, a lower bound $r^\nu(\mathbf{l}, \mathbf{u})$ on $p^\nu(\mathbf{l}, \mathbf{u})$ can be obtained by using (7) and solving

$$r^\nu(\mathbf{l}, \mathbf{u}) = \min_{\mathbf{x}\in\mathbb{R}^n} \mathrm{R}^\nu(\mathbf{x}; \mathbf{l}, \mathbf{u}) \tag{9-$\mathcal{R}^\nu$}$$

where

$$\mathrm{R}^\nu(\mathbf{x}; \mathbf{l}, \mathbf{u}) \triangleq \tfrac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda \sum_{i\in\nu_\bullet} \tfrac{[x_i]_+}{u_i} - \tfrac{[-x_i]_+}{l_i} + \lambda|\nu_1| + \eta(\mathbf{x}_{\nu_0} = \mathbf{0}) + \eta(\mathbf{x} \in [\mathbf{l}, \mathbf{u}]).$$

[1]This additional constraint usually takes the form "$-M \leq x_i \leq M, \ \forall i$" with $M > 0$ and is known as *"Big-M"* constraint, see [7, Sec. 3]

We note that (9-$\mathcal{R}^\nu$) is a convex problem and can be solved efficiently to good accuracy via numerous polynomial-time numerical procedures, see *e.g.,* [15, Ch. 10].

In practice, the choice of $\mathbf{l}$ and $\mathbf{u}$ must respect two conflicting imperatives. First, the new constraint "$\mathbf{x} \in [\mathbf{l}, \mathbf{u}]$" should not modify the solution of our target problem (1-$\mathcal{P}$) and condition (6) must therefore be verified. Since $\mathcal{X}^\star$ is obviously not accessible beforehand, this suggests that the entries of $\mathbf{l}$ and $\mathbf{u}$ should be chosen "large-enough" in absolute values.[2] Second, the tightness of $r^\nu(\mathbf{l}, \mathbf{u})$ with respect to $p^\nu(\mathbf{l}, \mathbf{u})$ degrades with the spread of the set $[\mathbf{l}, \mathbf{u}]$.[3] In particular, the right-hand side of (7) tends to $|\nu_1|$ when $\mathbf{l} \ll \mathbf{x}$ and $\mathbf{x} \ll \mathbf{u}$. Therefore, setting the entries of $\mathbf{l}$ and $\mathbf{u}$ with too large absolute values is likely to degrade the effectiveness of the relaxed pruning decision (5).

In the next section, we propose a solution to address this problem by deriving a methodology which locally tightens the constraint $\mathbf{x} \in [\mathbf{l}, \mathbf{u}]$ at each node of the decision tree while preserving the correctness of the BnB procedure.

IV. Peeling

In this section, we introduce our proposed peeling procedure. As an initial assumption, we suppose that some interval $[\mathbf{l}, \mathbf{u}]$ verifying condition (6) is known. This assumption will be relaxed later on in Sec. IV-C.

Our goal is to find a new interval $[\mathbf{l}', \mathbf{u}']$ such that

$$\forall \mathbf{x} \in [\mathbf{l}, \mathbf{u}] \setminus [\mathbf{l}', \mathbf{u}'] : \ \mathrm{P}^\nu(\mathbf{x}; \mathbf{l}, \mathbf{u}) > \bar{p} \tag{10a}$$

$$[\mathbf{l}', \mathbf{u}'] \subseteq [\mathbf{l}, \mathbf{u}]. \tag{10b}$$

These requirements imply that the pruning decision (4) made at node ν remains unchanged when replacing constraint "$\mathbf{x} \in [\mathbf{l}, \mathbf{u}]$" by "$\mathbf{x} \in [\mathbf{l}', \mathbf{u}']$" in (8-$\mathcal{P}^\nu$). More specifically, the following result holds:

Lemma 1. *Assume* $[\mathbf{l}, \mathbf{u}]$ *and* $[\mathbf{l}', \mathbf{u}']$ *verify* (10a)-(10b)*, then*

$$p^\nu(\mathbf{l}', \mathbf{u}') > \bar{p} \iff p^\nu(\mathbf{l}, \mathbf{u}) > \bar{p}. \tag{11}$$

A proof of this result is available in [12, App. A]. A consequence of preserving the pruning decision is that taking the new constraint "$\mathbf{x} \in [\mathbf{l}', \mathbf{u}']$" into account at node ν does not alter the output of the BnB procedure. In particular, it still correctly identifies the solutions of (1-$\mathcal{P}$). The second requirement (10b) implies that $r^\nu(\mathbf{l}', \mathbf{u}')$ can possibly be larger than $r^\nu(\mathbf{l}, \mathbf{u})$ since the lower bound in (7) is tightened by considering lower absolute values for $\mathbf{l}$ and $\mathbf{u}$. Overall, any choice of $[\mathbf{l}', \mathbf{u}']$ verifying (10a)-(10b) thus keeps unchanged the output of the BnB procedure while allowing for potentially more aggressive pruning decisions.

In the rest of this section, we describe a strategy to find some interval $[\mathbf{l}', \mathbf{u}']$ satisfying (10a)-(10b). Because of the symmetry of the problem at stake, we only focus on the construction of the upper bound $\mathbf{u}'$. The identification of a lower bound $\mathbf{l}'$ can be done along the same lines.

[2]Some heuristics are commonly used in the literature to select proper values of the bounds, see [7, Sec. V.B], [11, Sec. 5.1] or [16, Sec. 4].

[3]This impairment pertains to a large class of mixed-integer problems and is well known in the literature, see *e.g.*, [17].

A. Target peeling strategy

Given some index $j \in \nu_\bullet$ and $\alpha > 0$, we consider the following perturbed versions of (8-$\mathcal{P}^\nu$):

$$p^\nu_\alpha(\mathbf{l},\mathbf{u}) \triangleq \inf_{\mathbf{x}\in\mathbb{R}^n} \mathrm{P}^\nu(\mathbf{x};\mathbf{l},\mathbf{u}) + \eta(x_j > \alpha). \tag{12}$$

Problem (12) corresponds to (8-$\mathcal{P}^\nu$) where x_j is additionally constrained to be strictly greater than α. The following lemma then trivially follows from the definition of $p^\nu_\alpha(\mathbf{l},\mathbf{u})$:

Lemma 2. *If* $\alpha \in [0, u_j[$ *and*

$$p^\nu_\alpha(\mathbf{l},\mathbf{u}) > \bar{p}, \tag{13}$$

then (10a)-(10b) *hold with*

$$u'_i = \begin{cases} \alpha & \text{if } i = j \\ u_i & \text{otherwise.} \end{cases} \tag{14}$$

This result thus states that any $\alpha \in [0, u_j[$ verifying (13) enables to construct some $\mathbf{u}'$ automatically fulfilling (10a)-(10b). Unfortunately, evaluating (13) involves the same computational burden as solving (8-$\mathcal{P}^\nu$). This problem can nevertheless be circumvented by finding some proper lower bound on $p^\nu_\alpha(\mathbf{l},\mathbf{u})$ as described in the next section.

B. Tractable implementation

Leveraging Fenchel-Rockafellar duality for problem (9-$\mathcal{R}^\nu$), it can be shown [12, App. A] that for any $\mathbf{w} \in \mathbb{R}^m$, the following lower bound on $p^\nu_\alpha(\mathbf{l},\mathbf{u})$ holds:

$$p^\nu_\alpha(\mathbf{l},\mathbf{u}) \geq \mathrm{D}^\nu(\mathbf{w};\mathbf{l},\mathbf{u}) + \psi_j(\mathbf{w};\mathbf{l},\mathbf{u}) + \alpha[-\mathbf{a}_j^{\mathrm{T}}\mathbf{w}]_+, \tag{15}$$

where

$$\begin{aligned} \mathrm{D}^\nu(\mathbf{w};\mathbf{l},\mathbf{u}) &\triangleq \tfrac{1}{2}\|\mathbf{y}\|_2^2 - \tfrac{1}{2}\|\mathbf{y}-\mathbf{w}\|_2^2 + \lambda|\nu_1| \\ &\quad - \sum_{i\in\nu_1} \mu_{0,i}(\mathbf{a}_i^{\mathrm{T}}\mathbf{w}) - \sum_{i\in\nu_\bullet} \mu_{\lambda,i}(\mathbf{a}_i^{\mathrm{T}}\mathbf{w}) \\ \psi_j(\mathbf{w};\mathbf{l},\mathbf{u}) &\triangleq \mu_{\lambda,j}(\mathbf{a}_j^{\mathrm{T}}\mathbf{w}) - u_j[\mathbf{a}_j^{\mathrm{T}}\mathbf{w}]_+ + \lambda \end{aligned}$$

and $\mu_{\rho,i}(v) \triangleq [u_i v - \rho]_+ + [l_i v - \rho]_+$.

Using this result, condition (13) can be relaxed as

$$\mathrm{D}^\nu(\mathbf{w};\mathbf{l},\mathbf{u}) + \psi_j(\mathbf{w};\mathbf{l},\mathbf{u}) + \alpha[-\mathbf{a}_j^{\mathrm{T}}\mathbf{w}]_+ > \bar{p}. \tag{16}$$

Hence, choosing any $\alpha \in [0, u_j[$ verifying (16) for some $\mathbf{w} \in \mathbb{R}^m$ defines a new valid constraint via (14), in the sense of (10a)-(10b). Interestingly, the left-hand side of (16) depends linearly on α, thus allowing to precisely characterize the range of possible values satisfying the strict inequality (16). This leads us to the main result of this section.

Proposition 1. *Let* $\mathbf{w} \in \mathbb{R}^m$. *If* $\mathbf{a}_j^{\mathrm{T}}\mathbf{w} \geq 0$ *and*

$$\mathrm{D}^\nu(\mathbf{w};\mathbf{l},\mathbf{u}) + \psi_j(\mathbf{w};\mathbf{l},\mathbf{u}) > \bar{p}, \tag{17}$$

then $\mathbf{u}'$ *defined as in* (14) *with* $\alpha = 0$ *fulfills* (10a)-(10b). *Moreover, if* $\mathbf{a}_j^{\mathrm{T}}\mathbf{w} < 0$, *then* $\mathbf{u}'$ *defined as in* (14) *with any* $\alpha \in [0, u_j[$ *verifying*

$$\alpha > \bar{\alpha} \triangleq \frac{\bar{p} - \mathrm{D}^\nu(\mathbf{w};\mathbf{l},\mathbf{u}) - \psi_j(\mathbf{w};\mathbf{l},\mathbf{u})}{[-\mathbf{a}_j^{\mathrm{T}}\mathbf{w}]_+} \tag{18}$$

fulfills (10a)-(10b).

Our next result shows that Prop. 1 can be applied to all indices $j \in [\![1, n]\!]$ either sequentially or in parallel, while preserving the correctness of the BnB procedure:

Lemma 3. *Let* $[\mathbf{l}', \mathbf{u}']$ *and* $[\mathbf{l}'', \mathbf{u}'']$ *be two intervals satisfying* (10a)-(10b). *Then, the interval* $[\mathbf{l}', \mathbf{u}'] \cap [\mathbf{l}'', \mathbf{u}'']$ *also fulfills* (10a)-(10b).

A proof is available in [12, App. A]. We note that in terms of complexity the parallel application of Prop. 1 to all indices $j \in [\![1, n]\!]$ requires the computation of the inner products $\{\mathbf{a}_i^{\mathrm{T}}\mathbf{w}\}_{i=1}^n$ and *one single* evaluation of $\mathrm{D}^\nu(\mathbf{w};\mathbf{l},\mathbf{u})$. Interestingly, these inner products are already computed in most numerical procedures solving (9-$\mathcal{R}^\nu$) and are thus usually available at no additional cost, see *e.g.,* [11, Sec. 4.3]. The overhead complexity of applying in parallel our proposed peeling strategy thus scales as $\mathcal{O}(n+m)$.

C. Propagating peeling down the tree

In this section, we emphasize that any interval $[\mathbf{l}', \mathbf{u}']$ verifying (10a)-(10b) at node ν can be used as a starting point to apply our peeling procedure at the child nodes of ν. More specifically, the following result holds:

Lemma 4. *Let* $[\mathbf{l}', \mathbf{u}']$ *be some interval verifying* (10a)-(10b) *at node* ν *and let* ν' *be some child node of* ν. *Assume that the peeling procedure defined in Prop. 1 is applied at node* ν' *with* $[\mathbf{l}', \mathbf{u}']$ *as input, rather than* $[\mathbf{l}, \mathbf{u}]$, *to generate a new interval* $[\mathbf{l}'', \mathbf{u}'']$. *Then we have*

$$\forall \mathbf{x} \in [\mathbf{l},\mathbf{u}] \setminus [\mathbf{l}'',\mathbf{u}''] : \mathrm{P}^{\nu'}(\mathbf{x};\mathbf{l},\mathbf{u}) > \bar{p} \tag{19a}$$

$$[\mathbf{l}'',\mathbf{u}''] \subseteq [\mathbf{l},\mathbf{u}]. \tag{19b}$$

A proof of this result is available in [12, App. A]. In other words, Lem. 4 states that any peeled interval $[\mathbf{l}', \mathbf{u}']$ computed at node ν can be used as a starting point to apply a new peeling step at any child node ν'. This allows to propagate the peeled interval $[\mathbf{l}', \mathbf{u}']$ down the decision tree to hopefully improve sequentially the tightness of the convex relaxation (9-$\mathcal{R}^\nu$).

V. Numerical results

This section reports an empirical study demonstrating the effectiveness of the proposed peeling procedure to accelerate the resolution of (1-$\mathcal{P}$) on a synthetic dataset. We refer the reader to [8, 18] for an in-depth study of the statistical properties of the optimizer obtained from this problem.

A. Experimental setup

We consider instances of problem (1-$\mathcal{P}$) with dimensions $(m, n) = (100, 150)$. For each trial, new realizations of $\mathbf{A}, \mathbf{y}$ and λ are generated as follows. Each row of the dictionary $\mathbf{A}$ is drawn from a multivariate normal distribution with zero mean and covariance matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$. The (i,j)th entry of $\mathbf{K}$ is defined as $K_{ij} = 10^{-|i-j|}$, $\forall i, j \in [\![1, n]\!]$. Each realization of $\mathbf{y}$ is generated in two steps. We first create a 5-sparse vector $\mathbf{x}^\dagger \in \mathbb{R}^n$ with evenly-distributed non-zero components. The non-zero entries are defined as $x_i^\dagger = \mathrm{sign}(r_i) + r_i$ where r_i

is an independent realization of a zero-mean Gaussian with variance σ^2. We then set $\mathbf{y} = \mathbf{A}\mathbf{x}^\dagger + \mathbf{n}$ for some zero-mean white Gaussian noise $\mathbf{n}$. The variance of the noise is adjusted so that the SNR $\triangleq 10\log_{10}(\|\mathbf{A}\mathbf{x}^\dagger\|_2^2/\|\mathbf{n}\|_2^2)$ is equal to 15dB. The parameter λ is calibrated for each instance of (1-$\mathcal{P}$) using the cross-validation tools of the L0LEARN package [19] with the default parameters, see [12, Sec. V] for more details.

B. Competing procedures

We consider the following numerical solvers addressing (1-$\mathcal{P}$): *i)* CPLEX [6], a generic mixed-integer problem solver; *ii)* L0BNB [10], a standard BnB procedure using a "breadth-first search" exploration strategy, see [10, Sec. 3.3]; *iii)* SBNB, a standard BnB procedure using a "depth-first search" exploration strategy, see [16, Sec. 2.2]; *iv)* SBNB-N, corresponding to SBNB enhanced with additional "node-screening" techniques, see [11]; *v)* SBNB-P, corresponding to SBNB enhanced with the peeling strategy presented in this paper. L0BNB, SBNB, SBNB-N and SBNB-P all use the same solving procedure for relaxed problem (9-$\mathcal{R}^\nu$), namely a coordinate descent method [20]. We use the C++ implementation of CPLEX[4] and the Python implementation of L0BNB.[5] SBNB, SBNB-N and SBNB-P are implemented in Julia.[6]

For SBNB-P, peeling is applied at each iteration of the numerical procedure solving the relaxed problem (9-$\mathcal{R}^\nu$). We use the current iterate, say $\mathbf{x}^{(k)}$, to define $\mathbf{w} \triangleq \mathbf{y} - \mathbf{A}\mathbf{x}^{(k)}$ and apply the peeling rules defined in Prop. 1 in parallel, *i.e.,* simultaneously for all the components of $\mathbf{x}$. The value of α satisfying (18), if any, is chosen as $\alpha = \bar{\alpha} + 10^{-16}$. The peeled intervals are propagated through the decision tree as described in Sec. IV-C.

All the solving procedures are provided with the initial bounds $\mathbf{l} = -M\mathbf{1}$ and $\mathbf{u} = M\mathbf{1}$ for some proper value of M. This corresponds to the standard *"Big-M"* constraint commonly considered in the literature [7, 9, 14, 16]. As far as our random simulation setup is concerned, it can be shown that (1-$\mathcal{P}$) admits a unique minimizer $\mathbf{x}^\star$ with probability one and we thus choose $M = \gamma\|\mathbf{x}^\star\|_\infty$ for some $\gamma \geq 1$ in our simulations. This requires to solve (1-$\mathcal{P}$) once beforehand to identify $\mathbf{x}^\star$. This operation is here only done for the sake of comparing the sensibility of the solving methods to the choice of γ. More details are given in our companion paper [12, Sec. V].

C. Computational gains

Fig. 1 presents the performance of the considered solving procedures. All results are averaged over 50 problem instances. Experiments were run on one Intel Xeon E5-2660 v3 CPU clocked at 2.60 GHz with 16 GB of RAM. The left column in Fig. 1 represents the average solving time of each procedure as a function of γ (top) and σ (bottom); the right column illustrates the gain allowed by the proposed method in terms of solving time (solid) and number of nodes explored (dashed) as compared to its best competitor, that is SBNB-N.

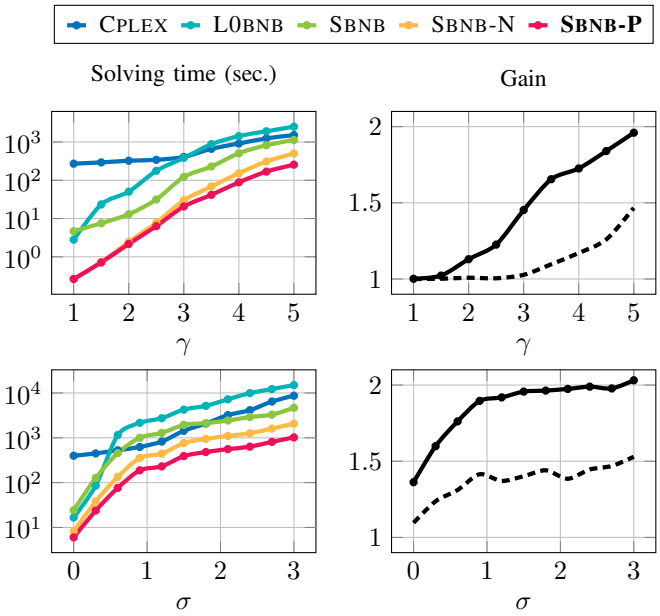

Fig. 1. Left: Solving time as a function of γ (top, $\sigma = 1$) and σ (bottom, $\gamma = 5$). Right: gain in terms of solving time (solid) and number of nodes explored (dashed) with respect to SBNB-N.

We note that SBNB-P leads to the smallest running time in all the considered setups. Since the latter corresponds to SBNB where peeling has been added, the spacing between the red and green curves materializes the gain provided by peeling. As far as our simulation setup is concerned, we see that the proposed method enables an acceleration of almost one order of magnitude with respect to SBNB. It is noticeable that this acceleration occurs even if $\gamma = 1$, that is the *Big-M* constraint is perfectly tuned to the problem at hand. This is due to the fact that peeling can refine *individually* each component of the initial bounds $\mathbf{l}$ and $\mathbf{u}$ at *each node* of the BnB decision tree to fit the local geometry of the problem.

We also notice that SBNB-P improves over SBNB-N, which can be seen as another acceleration of SBNB. In particular, SBNB-P performs always as well as SBNB-N as emphasized by the gains in the right-hand side of Fig. 1. We note in particular the gain provided by peeling in terms of number of nodes processed by the BnB procedure: as expected, peeling allows for more aggressive pruning and thus reduces the number of nodes to be explored.

VI. CONCLUSION

In this paper, we presented a tractable strategy, named "*peeling*", to tighten the box constraints used in a BnB procedure tailored to ℓ_0-regularized least-squares problems. Unlike the standard approach which imposes *one global* constraint to the problem, our strategy aims to locally refine the box constraints *at each node* of the decision tree. This refinement enables to strengthen the convex relaxations used in the pruning decisions made by the BnB procedure and can lead to significant improvements in terms of solving time, as emphasized by our simulation results.

[4]https://github.com/jump-dev/CPLEX.jl
[5]https://github.com/hazimehh/L0Learn
[6]https://github.com/TheoGuyard/BnbPeeling.jl

References

[1] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE transactions on information theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[2] D. L. Donoho, M. Elad, and V. N. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise," *IEEE Transactions on information theory*, vol. 52, no. 1, pp. 6–18, 2005.

[3] J. A. Tropp and S. J. Wright, "Computational methods for sparse solution of linear inverse problems," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 948–958, 2010.

[4] X. Chen, D. Ge, Z. Wang, and Y. Ye, "Complexity of unconstrained l2-lp minimization," *Mathematical Programming*, vol. 143, no. 1-2, pp. 371–383, 2012.

[5] S. Foucart, H. Rauhut, S. Foucart, and H. Rauhut, *An Invitation To Compressive Sensing*. Springer, 2013.

[6] I. I. Cplex, "V12. 1: User's manual for cplex," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[7] S. Bourguignon, J. Ninin, H. Carfantan, and M. Mongeau, "Exact sparse approximation problems via mixed-integer programming: Formulations and computational performance," *IEEE Transactions on Signal Processing*, vol. 64, no. 6, pp. 1405–1419, 2015.

[8] D. Bertsimas, A. King, and R. Mazumder, "Best subset selection via a modern optimization lens," *The Annals of Statistics*, vol. 44, no. 2, pp. 813–852, 2016.

[9] D. Bertsimas, R. Cory-Wright, and J. Pauphilet, "A unified approach to mixed-integer optimization problems with logical constraints," *SIAM Journal on Optimization*, vol. 31, no. 3, pp. 2340–2367, 2021.

[10] H. Hazimeh, R. Mazumder, and A. Saab, "Sparse regression at scale: Branch-and-bound rooted in first-order optimization," *Mathematical Programming*, vol. 196, no. 1-2, pp. 347–388, 2022.

[11] T. Guyard, C. Herzet, and C. Elvira, "Node-screening tests for the l0-penalized least-squares problem," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 5448–5452.

[12] T. Guyard, G. Monnoyer, C. Elvira, and C. Herzet, "Safe peeling for l0-regularized least-squares with supplementary material," *arXiv preprint arXiv:2302.14471*, 2023.

[13] L. A. Wolsey, *Integer Programming*. John Wiley & Sons, 2020.

[14] R. Ben Mhenni, S. Bourguignon, and J. Ninin, "Global optimization for sparse solution of least squares problems," *Optimization Methods and Software*, vol. 37, no. 5, pp. 1740–1769, 2022.

[15] A. Beck, *First-Order Methods In Optimization*. SIAM, 2017.

[16] R. B. Mhenni, S. Bourguignon, M. Mongeau, J. Ninin, and H. Carfantan, "Sparse branch and bound for exact optimization of l0-norm penalized least squares," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 5735–5739.

[17] J. D. Camm, A. S. Raturi, and S. Tsubakitani, "Cutting big M down to size," *Interfaces*, vol. 20, no. 5, pp. 61–66, 1990.

[18] T. Hastie, R. Tibshirani, and R. J. Tibshirani, "Extended comparisons of best subset selection, forward stepwise selection, and the lasso," *arXiv preprint arXiv:1707.08692*, 2017.

[19] H. Hazimeh and R. Mazumder, "Fast best subset selection: Coordinate descent and local combinatorial optimization algorithms," *Operations Research*, vol. 68, no. 5, pp. 1517–1537, 2020.

[20] S. J. Wright, "Coordinate descent algorithms," *Mathematical programming*, vol. 151, no. 1, pp. 3–34, 2015.