Screening Methods for Sparse Linear Problems

Theo Guyard
INSA Rennes, Mathematics Department
Campus de Beaulieu, 35000 Rennes, France
theo.guyard@insa-rennes.fr

Abstract

In this report, we introduce new methods to accelerate the resolution of two particular cases of sparse linear problems : the *Elastic-Net* and the ℓ_0*-regularized least-squares problem*. The Elastic-Net appears in various scientific fields and is often tackled with iterative algorithms. Recently, *screening tests* were developed to reduce the computational burden of such methods by identifying *zero entries* in the solution. We introduce a complementary method, dubbed *relaxing tests*, that identifies *non-zero entries* of the solution, and we leverage on this knowledge to reduce the dimensionality of the problem. Ultimately combining the two methodologies in a "Screen & Relax" strategy on top of an arbitrary iterative solving method, we manage to significantly reduce the overall complexity of the resolution process on various simulation setups. The ℓ_0-regularized least-squares problem is also a sparsity including problem, which directly penalizes non-sparsity using the ℓ_0-norm. This penalization leads to solutions with better statistical properties, but makes the problem *NP-hard*. Nonetheless, it remains tractable with recent custom Branch-and-Bound algorithms. We introduce the counterpart of the Screen & Relax methodology, but for this Mix-Integer Problem. Using new *Zero-One screening tests*, we manage to *fix some binary variables* during the Branch-and-Bound process, ultimately improving the overall number of nodes processed and thereby the solution time.

Acknowledgments

First of all, I would like to deeply thank Cedric and Clement. Since I started working with them, I have learned a lot about many different subjects. From the technical side to the "editorial method" as they say, but also on the strategy to build a research career, they were always there to give me good advice. I already know that my PhD that they will supervise next year can only go well.

I would thank Ayse-Nur Arslan for her support during this internship, and also for her willingness to continue working with me during my PhD.

More generally, I would like to thank the whole Mathematics Department from INSA Rennes which gives students, I think, very high quality skills. I feel lucky to have been able to follow a course where the supervision of the students is so meticulous and which allows to really dig into subjects if you really want to. Without the possibility of following the research course, I would probably not be where I am today.

I would also thank my father who passed on to me his passion for Sciences and Mathematics. Thank you for the interesting discussions we had. I would also like to thank my mother, who taught me all the little things that are useful in my daily life !

I thank Jeremy and Paul for the discussions and exchanges I had with them over a cup of coffee during this internship.

Finally, I thank Louise for trying to understand what I was doing and for bearing my working hours in the evenings and during weekends.

Contents

1 Introduction

The end-of-studies internship is the final step in the curriculum of INSA students. It allows not only to put into practice the knowledge acquired during their courses, but also to prepare students for their future jobs. The work presented in this report has nevertheless some specificities. Although INSA mainly trains engineers, the internship was carried out in a research laboratory. This report will therefore more take the form of a technical report than a classic end-of-studies report. Its purpose is to transcribe, in the most comprehensible way possible, the research work done during this five-months internship.

1.1 Working environment

The work presented in this document was carried out in the IETR laboratory[1], within the SCEE team at CentraleSupelec[2]. The IETR is a public research laboratory specialized in the field of Electronics and Digital Technologies. The SCEE team was created in 2004 and focuses its research on two main areas : Signal Processing and Embedded Systems. It splits between the Rennes 1 campus and various engineering schools in Rennes. Clement Elvira[3] is the official supervisor of this internship. He is part of the CentraleSupelec pedagogical team since September 2020 and works mainly in the field of Applied Mathematics, on topics related to Signal Processing and Machine Learning. Cedric Herzet[4], although unofficial supervisor, was also actively involved in the guidance of the internship. He is part of the SIMSMART team[5] at INRIA Rennes, the laboratory of Applied Mathematics and Computer Science, gathering researchers from various research institutes. These two researchers have been working together for a few years and share the same research areas. They had already co-supervised the previous fourth-year traineeship, which is mandatory for INSA students. Although this end-of-studies internship was carried out within the SCEE team at CentraleSupelec, it mainly consisted in working in trio with the two supervisors, on different research problems. Finally, Ayse-Nur Arslan[6] also participated in the supervision of this internship as reference teacher at INSA Rennes. For most of its duration, the internship was carried out at distance, or in a hybrid working method.

1.2 Internship guidelines

In a way, this internship is the continuation between the fourth-year INSA internship, realized the past year with Cedric Herzet and Clement Elvira, and the PhD thesis that will start in October 2021, also supervised by these two researchers. Topics treated are all related to sparse linear problems, which we will be explained further. Last year, we have started to work on solving methods suited a particular case of a *convex* sparse linear problem, but we did not have time to finish everything. Hence, the first mission of the internship was to finalize this work, which will result, hopefully, in a paper published at the 2022 ICASSP conference[7]. We also plan to extend our results and potentially publish a journal paper later on. In addition, we wanted to study another avenue consisting in carrying the concepts

[1] https://www.ietr.fr
[2] https://www.centralesupelec.fr
[3] https://scholar.google.fr/citations?user=2LSEQYcAAAAJ&hl=fr&oi=ao
[4] https://scholar.google.fr/citations?user=wO7HamwAAAAJ&hl=fr&oi=ao
[5] https://www.inria.fr/fr/simsmart
[6] https://www.researchgate.net/profile/Ayse-Arslan-7
[7] https://2022.ieeeicassp.org

developed for this first convex optimization problem to another similar, yet *combinatorial*, optimization problem. This second project will also lead to the submission of a paper at ICASSP 2022 and may also be extended in a future journal paper. Finally, the last objective was to start getting familiar with the concepts that will be addressed next year is thesis, namely sparse linear problems in *infinite* dimension.

1.3 Report outline

In Section 2, we first introduce basic concepts about sparse linear problems that are at the center of topics addressed in the report. Then in Section 3, we focus on our new *Screen & Relax* method suited for the Elastic-Net, which is a particular case of convex sparse linear problem. This corresponds to the continuation of the previous year's internship project. In Section 4, we focus on our new *Zero-One screening* methodology, extending Screen & Relax concepts from the Elastic-Net to the ℓ_0-Regularized Least-Squares problem. This is also a sparse linear problem, which is combinatorial, and therefore NP-hard. Finally in Section 5, we give a conclusion to the two previous parts and a global conclusion to this report, with personal remarks and prospects.

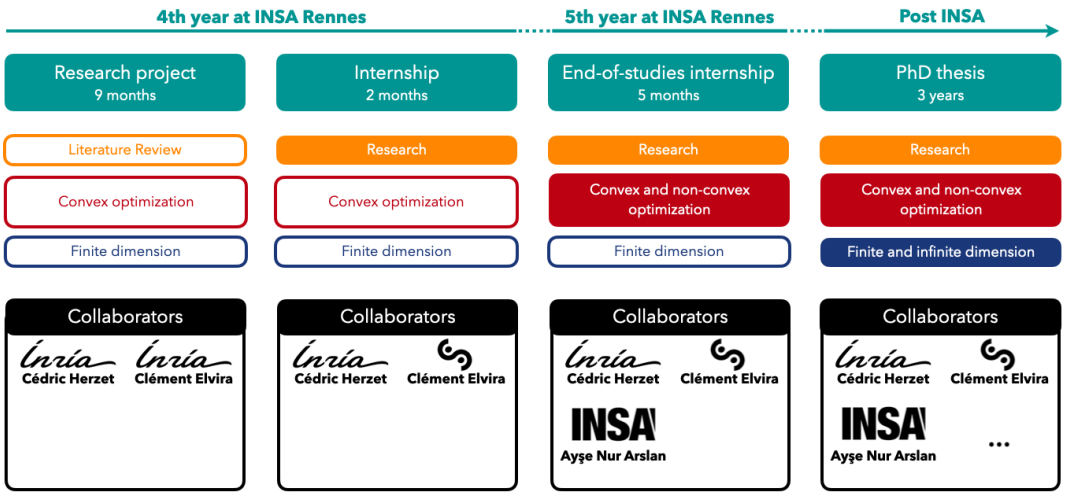

Figure 1: Continuation and evolution of the topics covered between the research project, the end-of studies internship and the future PhD thesis.

2 Sparse linear problems

Throughout this report, we focus on sparse linear problems and on methods aiming to improve their resolution. In this section, we go through key concepts needed to better understand the rest of the report.

2.1 First concepts

Linear inverse problems emerge from most fields linked to Engineering and to Mathematical Sciences (Ribes et al., 2008), from Signal processing to Statistics and through Economics. They aim to decompose a given signal using a linear combination of basic elements. In most applications, such problems are ill-conditioned or not well-posed (Bertero, 1989). Imposing new constraints is often useful to recover interesting solutions. For instance, one can impose *sparsity*, meaning that we want to use only few of the basic elements in the decomposition. This sparsity constraint also appears naturally in many practical contexts. In the literature, linear inverse problems with an additional sparsity driving are dubbed as *sparse linear problems*. They are made of three principal ingredients :

- $\mathbf{y}$: the signal to be decomposed, also called the *observation*
- $\mathbf{A}$: the *dictionary* gathering the basic components –or *atoms*– that can be used to decompose the signal
- $\mathbf{x}$: the *sparse decomposition*, *ie.* the manner to decompose the observation using the atoms

The observation is always a given data of the problem. In some cases, atoms are not directly provided, and a first problem is to select the right dictionary to perform an appropriate decomposition of the observation. In this report, we assume that the dictionary is fixed and that one only seeks to find how to decompose the observation using the given atoms. Therefore, the unknown of the problem is $\mathbf{x}$ whereas $\mathbf{y}$ and $\mathbf{A}$ are given data.

The observation, the dictionary and the sparse solution can be various mathematical objects, depending on the practical context. We consider the standard *discrete* case where $\mathbf{y} \in \mathbf{R}^m$ and $\mathbf{x} \in \mathbf{R}^n$ are vectors, and where $\mathbf{A} \in \mathbf{R}^{m \times n}$ is a matrix. Columns $\{\mathbf{a}_i\}_{i=1}^n$ of $\mathbf{A}$ correspond to the atoms. This encompasses most cases that are tackled in practice (Tropp et al., 2010). In the considered setup, a sparse linear problem can be roughly expressed as

$$\textit{Find } \mathbf{x} \textit{ sparse such that } \mathbf{y} \simeq \mathbf{A}\mathbf{x} \tag{2.1}$$

In other words, one seeks a sparse linear combination of the *columns* of $\mathbf{A}$ that well approximates $\mathbf{y}$. Coefficients of $\mathbf{x}$ weight each atom of the decomposition. In sparse linear problems, one usually needs $m \ll n$ to be able to provide a good approximation of the observation (Foucart et al., 2013). Note that in the latter problem, the exact reconstruction of $\mathbf{y}$ by $\mathbf{A}\mathbf{x}$ is not necessarily targeted, and a certain error can be allowed, depending on the practical use. As it, problem (2.1) is not well-posed. One has to define how to measure the *sparsity* of $\mathbf{x}$, how to measure the *data-fidelity* of $\mathbf{A}\mathbf{x}$ to $\mathbf{y}$, and what are the actual objectives and constraints of the problem.

2.2 Basic examples

Before going more into details, we focus on three basic examples of sparse linear models that illustrate their relevance and practical use. The first example is linked with the Statistical inference of some data, while the two others are linked to Signal Processing problems.

2.2.1 Crime prediction

Consider a country where police services wish to predict the crime ratio in cities in order to improve safety of citizens. One can assume that the crime rate can be modelled as a linear combination of various factors. Thus, recording data in some reference cities can allow identifying key factors impacting the crime ratio in these cities and to extrapolate results to other cities. An example of such dataset can be found in the UCI website[8]. It is presented as :

City	Pop. ($\times 10^6$)	Dens. (pop/km^2)	Pop. 12-21	Employ.	Incomes ($/mon.)	...
Lakewood	0.33	10,374	21%	76%	2978	...
Tukwila	0.16	5,932	24%	89%	3145	...
Aberdeen	0.42	7,342	18%	75%	3103	...
...	...	...	...	...	...	...

Table 1: Crime dataset from UCI database

Let $\mathbf{A}$ be the matrix representing the dataset, where each line corresponds to a reference city and where each column represents a factor observed. Furthermore, let denote $\mathbf{y}$ the rate of crime observed in each reference city. A simple manner to identify key factors that lead to a high crime rate is to find $\mathbf{x}$ such that $\mathbf{A}\mathbf{x} \simeq \mathbf{y}$. The value of each entry of $\mathbf{x}$ indicates the importance of each factor in the model. In practice however, this naïve model may involve many different factors in the prediction, making results difficult to interpret. To improve the understandability of the model, one can impose sparsity of $\mathbf{x}$ in order to recover a model including only few factors, but only the ones that really impact the crime rate. This directly lead to the sparse linear problem (2.1).

2.2.2 Image de-blurring

Image deblurring is a classical problem in natural image processing, where the goal is to remove the blurring of an image due to camera motion. In this scenario, the blurred image can be modelled as the product between a *blur kernel* that models the sensor motion and the sharp natural image. Natural images are usually not sparse but their spectral decomposition, in the Fourier basis for instance, is typically sparse and compressible (Levin et al., 2007). Hence, the spectral decomposition of the blurred image can be modelled as the product of between the blur kernel and a sparse signal, that is the spectral decomposition of the deblurred image.

In this problem, we can encode in $\mathbf{y}$ the spectral decomposition of the blurred image and in $\mathbf{A}$ the blur kernel. Directly finding $\mathbf{x}$ such that $\mathbf{y} \simeq \mathbf{A}\mathbf{x}$ does not necessarily ensure that the right spectral decomposition of the sharp image will be recovered. However, we can force $\mathbf{x}$ to be sparse in order to recover a sparse spectral decomposition that is more likely

[8] See https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

to correspond to a natural image. In this way, we improve chances to recover the original image. One more time, this directly lead to a problem of type (2.1).

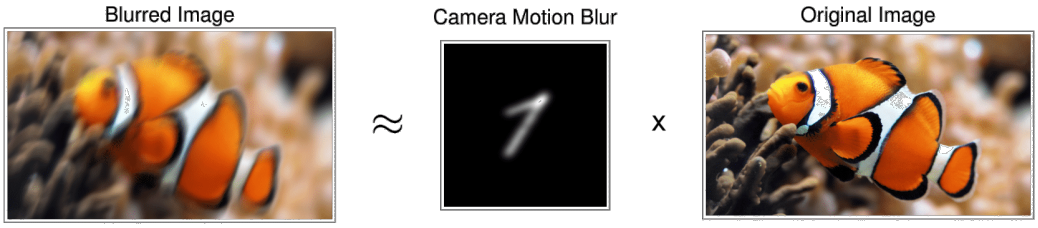

Figure 2: Illustration of a de-blurring problem. Source : DeconvLab.

2.2.3 Signal compression

When dealing with large scale and high quality images, it is often necessary to resort to a compression method to be able to send it, through a network for instance. As in the previous example, we can take advantage of the sparse property of the spectral decomposition of natural images. To construct the spectral decomposition of an image encoded in a vector $\mathbf{y}$, one can use a Discrete Cosine Transform matrix $\mathbf{A}$ (Strang, 1999), where each column corresponds to shifted cosine functions. By finding $\mathbf{x}$ such that, $\mathbf{y} \simeq \mathbf{A}\mathbf{x}$, we can capture periodicity in the image : the larger the magnitude of a coefficient in $\mathbf{x}$, the more important the periodicity of the corresponding column in $\mathbf{A}$. To compress the image, we can only truncate the decomposition and select a small proportion of the coefficients in $\mathbf{x}$ that have the higher magnitude. However, this does not necessarily ensure that the compressed image will be of good quality. We can rather solve the problem by directly imposing the sparsity of $\mathbf{x}$, which generally yields a compressed image of better quality.

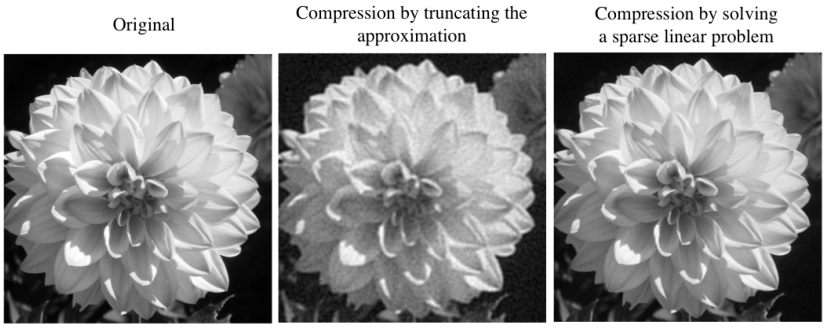

Figure 3: Sparse linear models and denoising course, New York University, Spring 2016.

2.3 Proper formulation of a sparse linear problem

We now focus on defining a proper formulation for sparse linear problems. The goal is to formulate an optimization problem for which the solution responds in a satisfactory manner to (2.1).

2.3.1 A bi-objective optimization problem

There exists different ways to measure how good an approximation of the signal is. In the literature, a function measuring the *data-fidelity* of $\mathbf{Ax}$ to $\mathbf{y}$ is commonly called a *loss function*. To measure sparsity of a vector, we can use *sparsity-including* functions, whose value increases when the input in less sparse. Losses are generally *convex* functions, while sparsity-including functions are commonly *mixed-norm* functions.

In problem (2.1), the solution $\mathbf{x}$ must ensure both data-fidelity and sparsity. A simple way to express an optimization problem giving a solution to (2.1) is

$$\min_{\mathbf{x}\in\mathbf{R}^n} \left\{ \mathrm{F}(\mathbf{y}, \mathbf{Ax}) \ , \ \Omega(\mathbf{x}) \right\} \tag{2.2}$$

where $\mathrm{F}(\mathbf{y}, \mathbf{Ax})$ is a *loss function* ensuring a good approximation of $\mathbf{y}$ by $\mathbf{Ax}$ and where $\Omega(\mathbf{x})$ is a *sparsity-including* function. By convention, functions respectively penalize low data-fidelity and non-sparsity, so a minimization is performed. Different losses are used in the literature, and we can mention for instance the likelihood loss, the log loss or ℓ_p-norm losses, among many others (Janocha et al., 2017). The most straightforward sparsity-including function is the ℓ_0-norm that counts the number of non-zero coefficients in its input. However, this function is not always very convenient to deal with and one can also resort to *convex* sparsity-including function such as the ℓ_1-norm or the ℓ_2-norm.

As it, problem (2.2) is still not easy to handle because it is a *bi-objective* problem. The optimality of such problems can be defined using *Pareto optimality* (Luc, 2008) but it is not convenient to compare different solutions and to tell which one should be used on practical cases. One most often solves optimization problems with a single objective and potentially some constraints, yielding an unique solution.

2.3.2 Choosing the right paradigm

Depending on the practical context, one of the two objectives of (2.2) is often more important than the other and the problem can be formulated under a more standard form. For instance, one can choose a sparsity threshold k and find the best approximation $\mathbf{Ax}$ of $\mathbf{y}$ regarding to a given loss, with the constraint $\Omega(\mathbf{x}) \leq k$. Alternatively, one can select a data-fidelity threshold ϵ and find the minimum of $\Omega(\mathbf{x})$ such that $\mathrm{F}(\mathbf{y}, \mathbf{Ax}) \leq \epsilon$. In this report, we focus on a third alternative, where a *trade-off* between the two objectives in minimized in a *unique* objective function. Given a tuning parameter $\lambda > 0$, the problem is formulated as

$$\min_{\mathbf{x}\in\mathbf{R}^n} \left\{ \mathrm{F}(\mathbf{y}, \mathbf{Ax}) + \lambda\Omega(\mathbf{x}) \right\} \tag{2.3}$$

This form is probably the most used in practice because it allows to easily balance the two objectives of the problem : the larger λ, the sparser the solution but the worst the data fidelity. In the rest of the report, we focus on efficient resolution methods suited for problems of the form (2.3).

2.4 Resolution approaches

In general, losses used are always convex functions, so the choice of the loss does not impact a lot the resolution method. In the opposite, sparsity-including function are of various types : convex, non-convex, smooth, non-smooth... Thus, even if problem (2.3) has a standard form, different methodologies are suited to solved it, regarding to the choice of the sparsity-including function.

2.4.1 Mixed-Integer programs

As said earlier, the most intuitive sparsity-including function is the ℓ_0-norm that counts the number of non-zero components of $\mathbf{x}$. Even if it is called a norm, it is not one, and it is even not a pseudo-norm. This function is highly non-linear, non-convex, and it is not always easy to deal with. Yet with some additional assumptions, problem (2.3) can be reformulated as a *Mixed-Integer Program* (MIP), with both continuous and integer variables (Wolsey, 1989). Even tough MIPs are NP-hard, one can hope to benefit from the huge improvement made during past decades in the resolution of such problems. MIP solvers such as `CPLEX`, `GUROBI` or `SCIP` are improved each year and allow to efficiently tackle problems with integer variables (Davies et al., 2013). Furthermore, the exponential increase in computing power make MIPs to become more and more tractable in practice. Many custom algorithms are also developed to exploit particular properties of sparse linear problems and manage to reduce significantly their resolution time compared to a resolution with generic MIP solvers. Finally, there exists a wide range of heuristic methods for NP-hard sparse linear problems (T. Zhang, 2008).

2.4.2 Convex optimization

To avoid NP-hardness of MIPs, one can rather use a *convex* penalty instead of the ℓ_0-norm. Even if such penalties do not directly penalize non-sparsity, interesting solutions can still be recovered under mild assumptions (Foucart et al., 2013). For instance the ℓ_1-norm, which is the convex relaxation of the ℓ_0-norm, is widely used and leads to the well-known LASSO problem when resorting to a Least-Squares loss (Tibshirani, 2011). The objective becomes fully convex and one can benefit from a large choice of methods that are suited for convex optimization problems. When a moderate accuracy on the solution is targeted, *iterative methods* are usually used. They construct a sequence of iterates that converges toward the optimizer. For instance, interior point methods, gradient-based methods or the ADMM have already proven their worth in the optimization community (Helmberg et al., 1996; Parikh et al., 2014; Boyd et al., 2011). When the exact solution is targeted, one can use *pivoting methods*, such as the Active-Set, that try to find the right form of optimality conditions of the problem and solve them to recover the optimizer exactly, up to some numerical errors (Wright et al., 1999). However, they are usually more expensive in computations.

2.4.3 Cross-cutting issues

Apart from complications inherent to problem properties, *cross-cutting* issues appear whatever the loss and the sparsity-including penalty used. On *large-scale* data, every solving method suffers from computational bottlenecks because each matrix-vector product involving $\mathbf{A}$ is performed with a complexity at least in $O(mn)$ in general. When both n and m are large, this can lead to prohibitive calculations. Thus, it can be convenient to use dictionaries with a particular structure, providing an economical way to compute $\mathbf{Ax}$ or $\mathbf{A}^{\mathrm{T}}\mathbf{u}$. For example, the cost of these products goes down to $O(n \log(n))$ when the dictionary is constructed from Fourier or Wavelet bases (Brigham et al., 1967). Another issue arises when there is an *high correlation* between the atoms, because this leads to ill-conditioned problems. In a simplified fashion, the problem is tricker because it is harder to detect which atom to use in the decomposition within a group of correlated atoms. Unfortunately, such correlated dictionaries naturally appear in practical cases and make things more difficult.

A final cross-cutting issue appears when targeting an *high accuracy* on the solution. Exact methods are usually more expensive in computations than inexact methods.

2.5 First intuitions about complexity reduction

In both Section 3 and Section 4, we present methods aiming to make solving algorithms for (2.3) computationally cheaper by identifying particular components of the solution *during* the solving process. Even if they are developed for two different sparse linear problems, the underlying idea is the same. Assume that one has the knowledge of some entries of the optimizer of (2.3) that will be *zero*. Then removing them as well as corresponding atom in $\mathbf{A}$ does *not* change (in general) the solution of the problem. In addition, this reduces the problem dimension and makes the solving process more economic in computations. In an analogous way, assume that one has the knowledge of some entries of the solution that will be *non-zero*, one can hope to benefit from this information to improve the speed of the resolution process. As only few coefficients are non-zero in a sparse linear problem context, knowing a coefficient that is used in the optimal decomposition provides a lot of information about the problem structure and allows to tackle it more efficiently.

In the case of convex penalties, *zero* entries of the optimizer can be identified thanks to *screening tests* developed during the last decade (Ghaoui et al., 2010). In Section 3 we develop our contribution, dubbed *relaxing tests*, that are complementary to screening tests and allow identifying *non-zero* entries of the solution. With both screening and relaxing tests, we manage to significantly improve the convergence speed and address computational issues when a high accuracy is targeted. In Section 4, we develop our second contribution that adapts the screening and relaxing tests idea to the case of a sparse linear problem with an ℓ_0-penalty. In particular, we manage to identify the value of some integer coefficients and fix them during the solving process. As it is the integrity of coefficients that makes the problem NP-hard, our approach allows to significantly improve the resolution process.

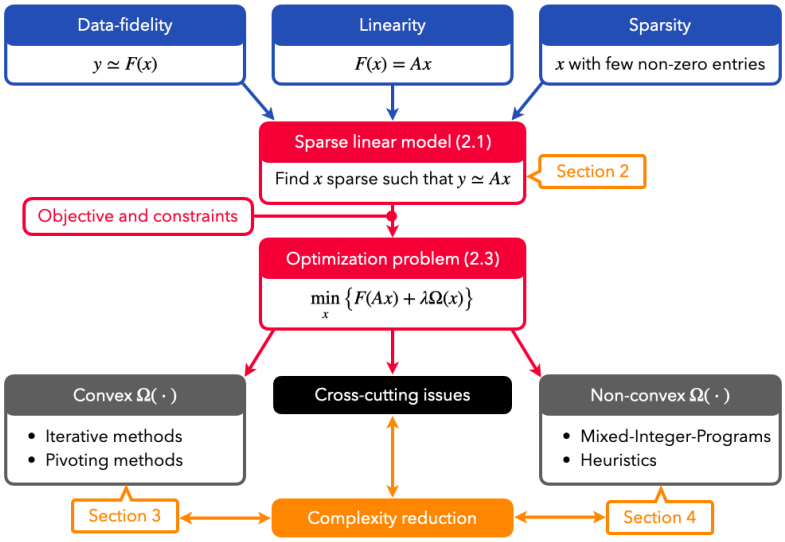

Figure 4: Key ingredients of sparse linear problems and outline of this report.

3 Screen & Relax : A fast solving procedure for the Elastic-Net

In this section, we present a new method aiming to accelerate an arbitrary gradient-based method solving the Elastic-Net problem. We first discuss our motivations and goals. Then, we give some mathematical tools and some key ingredients required to develop our method. We finally present a generic *Screen & Relax* algorithm that can be implemented on top of an arbitrary solving method for the Elastic-Net. We assess it with the projected-gradient algorithm as an example.

3.1 The Elastic-Net problem

The Elastic-Net problem is a widely used sparse linear problem. It was introduced in (Zou et al., 2005) and ensures data-fidelity using the *least-squares* loss $\mathrm{F}(\mathbf{y}, \mathbf{A}\mathbf{x}) \triangleq \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$. The convex sparsity-including penalty $\Omega(\mathbf{x}) \triangleq \alpha\|\mathbf{x}\|_1 + \frac{\beta}{2}\|\mathbf{x}\|_2^2$ is used, where $\alpha > 0$ and $\beta > 0$ are two tuning parameters. The extreme case where $\beta = 0$ corresponds to the celebrated LASSO problem (Tibshirani, 2011) while the extreme case where $\alpha = 0$ can be re-arranged into a Least-Squares problem (Lawson et al., 1995). The ℓ_1-norm is the convex relaxation of the ℓ_0-norm and enforces sparsity of the solution. The additional ℓ_2-norm allows to archive a better statistical accuracy on the solution of real-case problems. It also encourages a grouping effect and entries of the solution corresponding to correlated atoms tend to be either zero or non-zero together, which can be interesting when dealing with real-world data (You et al., 2016). Finally, the Elastic-Net arises in maximization-minimization procedures used to solve inverse problems (Rakotomamonjy et al., 2019).

Methods such as gradient-based algorithms or the ADMM are often implemented to solve the Elastic-Net as it is a convex, but non-smooth, optimization problem (Parikh et al., 2014; Boyd et al., 2011). However, such methods suffer from computational bottlenecks on large-scale data and usually struggle to converge when a high-accuracy is targeted. Recently, *screening tests* were introduced to reduce the computational cost of these methods (Ghaoui et al., 2010). They exploit the sparsity of the problem by identifying *zero entries* of the solution and removing it from the problem. In this way, the dimension is reduced, leading to potential complexity savings. Since a decade, various screening methods were developed and have proven their worth in the optimization community (Bonnefoy et al., 2014; Xiang, Y. Wang, et al., 2016).

Contributions We derive a complementary method to screening that identifies *non-zero entries* of the Elastic-Net solution using new *relaxing tests*. We dub our method this way since it allows to drop some constraints of the target problem upon the identification of non-zero coefficients and, similarly to screening, end up with a problem of reduced dimension. Ultimately combining the two methodologies in a *"Screen & Relax"* strategy, we manage to significantly reduce the overall complexity of the resolution on various simulation setups.

3.2 Dimensionality reduction

In the following, we denote $\mathbf{A}_{\mathcal{I}}$ (*resp.* $\mathbf{x}_{\mathcal{I}}$) the matrix (*resp.* vector) whose columns (*resp.* indices) entries are restricted to the set $\mathcal{I}$.

3.2.1 Elastic-Net derivations

Without loss of generality, we restrict our scope to the *non-negative* Elastic-Net problem. The standard Elastic-Net can always be reformulated in this way and the non-negative version leads to simpler derivations, on top of being more general. Thus, we rather tackle

$$\mathbf{x}^\star = \underset{\mathbf{x}\geq\mathbf{0}_n}{\operatorname{argmin}} \left\{ \tfrac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \alpha \mathbf{1}^{\mathrm{T}}\mathbf{x} + \tfrac{\beta}{2}\|\mathbf{x}\|_2^2 \right\} \tag{$\mathcal{P}$}$$

Problem ($\mathcal{P}$) is β-strongly convex and thereby admits a *unique* minimizer (Nesterov et al., 2018). We assume that $\alpha < \alpha_{\max} \triangleq \max(\mathbf{A}^{\mathrm{T}}\mathbf{y})$ since otherwise $\mathbf{x}^\star$ is identically null. Screening and relaxing tests leverage on properties of the Fenchel dual (Bot, 2009) of ($\mathcal{P}$), given by

$$\mathbf{u}^\star = \underset{\mathbf{u}\in\mathbf{R}^m}{\operatorname{argmax}} \left\{ \tfrac{1}{2}\big(\|\mathbf{y}\|_2^2 - \|\mathbf{y} - \mathbf{u}\|_2^2 - \tfrac{1}{\beta}\|[\mathbf{A}^{\mathrm{T}}\mathbf{u} - \alpha\mathbf{1}]_+\|_2^2\big) \right\} \tag{$\mathcal{D}$}$$

where $[w]_+ \triangleq \max(w, 0)$, but also on the first-order optimality condition

$$\mathbf{x}^\star = \beta^{-1}[\mathbf{A}^{\mathrm{T}}\mathbf{u}^\star - \alpha\mathbf{1}]_+ \tag{$\mathcal{C}$}$$

linking the primal)dual optimizer couple. If one is able to solve ($\mathcal{D}$), then the solution of ($\mathcal{P}$) is available in closed-form. Unfortunately, identifying $\mathbf{u}^\star$ turns out to be as challenging as obtaining $\mathbf{x}^\star$. One can nevertheless leverage on ($\mathcal{C}$) to design procedures that can be exploited to accelerate the resolution of the Elastic-Net.

3.2.2 Screening tests

Screening tests developed during the last decade aim to identify elements of the set $\mathcal{I}_0 \triangleq \{i \mid x_i^\star = 0\}$. They use the following *necessary and sufficient condition* derived from relation ($\mathcal{C}$).

Proposition 3.1. *For all* $i \in [\![1, n]\!]$,

$$\mathbf{a}_i^{\mathrm{T}}\mathbf{u}^\star \leq \alpha \iff i \in \mathcal{I}_0 \tag{3.1}$$

In other words, $\mathcal{I}_0$ can be entirely identified if $\mathbf{u}^\star$ is available. Assume that a subset $\mathcal{I} \subseteq \mathcal{I}_0$ is known, then $\mathbf{A}$ (*resp.* $\mathbf{x}$) can be replaced by $\mathbf{A}_{\mathcal{I}}$ (*resp.* $\mathbf{x}_{\mathcal{I}}$) in ($\mathcal{P}$), which does *not* change the solution of the problem. In this way, the dimension is reduced by $|\mathcal{I}|$ and the complexity of the solving process may decrease. As already said, it turns out that $\mathbf{u}^\star$ is as challenging to compute as $\mathbf{x}^\star$. To circumvent this issue, one can construct a *safe region* to devise a weaker version of (3.1) that can be implemented with a low computational burden (Xiang and Ramadge, 2012).

Let $\mathcal{R}$ be a *safe region*, *ie.* a region containing $\mathbf{u}^\star$, then a direct corollary of Proposition 3.1 is

Proposition 3.2. *For all* $i \in [\![1, n]\!]$,

$$\max_{\mathbf{u}\in\mathcal{R}} \{\mathbf{a}_i^{\mathrm{T}}\mathbf{u}\} \leq \alpha \implies i \in \mathcal{I}_0 \tag{3.2}$$

Proposition 3.2 only gives a *sufficient condition* but does not involve $\mathbf{u}^\star$, so it is a practical way to identify elements $\mathcal{I}_0$. The performance of test (3.2) obviously depends on the safe region $\mathcal{R}$ used, and one cannot necessarily ensure that $\mathcal{I}_0$ can be entirely identified.

Typically, safe regions that are cheap to construct, providing a closed-form for (3.2) and allowing to identify as many elements of $\mathcal{I}_0$ as possible are preferred.

Fortunately, various safe regions were proposed in the literature, each one with its own specific characteristics and strengths. The region is usually tightened during the solving process with new iterates provided by the iterative method to identify more and more elements of $\mathcal{I}_0$. They are built-up with values *already computed* at each iteration so as not to over-penalize the solving process. For discussions on the devise of relevant safe spheres, the reader may refer to (Herzet et al., 2018; J. Wang et al., 2013; Ndiaye et al., 2016).

3.2.3 Relaxing tests

Our new relaxing tests aim to identify elements of $\mathcal{I}_+ \triangleq \{i \mid x_i^\star > 0\}$, that is the complementary of $\mathcal{I}_0$. For each element of $\mathcal{I}_+$ identified, we *relax* the corresponding constraint in ($\mathcal{P}$) and by performing a linear variable shift, we exclude the corresponding coefficient from the problem in an analogous way to screening. Relaxing tests leverage on the negation of Proposition 3.1, which is

Proposition 3.3. *For all* $i \in [\![1, n]\!]$,

$$\mathbf{a}_i^{\mathrm{T}}\mathbf{u}^\star > \alpha \iff i \in \mathcal{I}_+ \tag{3.3}$$

As for screening, the test (3.3) is not usable in practice because it requires $\mathbf{u}^\star$ to be known. Therefore, we also resort to safe regions. Assume that a safe region $\mathcal{R}$ containing $\mathbf{u}^\star$ is known, then a direct corollary to Proposition 3.3 is

Proposition 3.4. *For all* $i \in [\![1, n]\!]$,

$$\min_{\mathbf{u}\in\mathcal{R}} \{\mathbf{a}_i^{\mathrm{T}}\mathbf{u}\} > \alpha \implies i \in \mathcal{I}_+ \tag{3.4}$$

In other words, test Proposition 3.4 is a practical way to identify $\mathcal{I}_+$, given a safe region $\mathcal{R}$. Here again, the performance of test (3.4) depends on the safe region used. Any region originally designed for screening tests can also be used for relaxing tests. In general, a region that is efficient for screening tests is also efficient for relaxing tests and shares the same properties for both.

3.2.4 Shrinking dimension

Assume that a subset $\mathcal{I} \subseteq \mathcal{I}_+$ is known and let denote $\bar{\mathcal{I}}$ its complementary. Removing the constraints on $\mathbf{x}_\mathcal{I}$ in ($\mathcal{P}$) does *not* change the solution of the problem because it is *inactive* at optimality. In addition, a byproduct of ($\mathcal{C}$) allows to express $\mathbf{x}_\mathcal{I}$ *in function of* $\mathbf{x}_{\bar{\mathcal{I}}}$ by

$$\mathbf{x}_\mathcal{I} = \mathbf{B}\mathbf{x}_{\bar{\mathcal{I}}} + \mathbf{b} \tag{3.5}$$

where

$$\begin{cases} \mathbf{G} &= \mathbf{A}_\mathcal{I}^{\mathrm{T}}\mathbf{A}_\mathcal{I} + \beta\mathbf{I} \\ \mathbf{B} &= -\mathbf{G}^{-1}\mathbf{A}_\mathcal{I}^{\mathrm{T}}\mathbf{A}_{\bar{\mathcal{I}}} \\ \mathbf{b} &= \mathbf{G}^{-1}\big(\mathbf{A}_\mathcal{I}^{\mathrm{T}}\mathbf{y} - \alpha\mathbf{1}\big) \end{cases} \tag{3.6}$$

In the latter, $\mathbf{I}$ is the identity matrix of the appropriate size. By injecting (3.5) into ($\mathcal{P}$), we obtain the following equivalent reformulation of the Elastic-Net problem

$$\mathbf{x}_{\bar{\mathcal{I}}}^\star = \underset{\mathbf{x}\geq\mathbf{0}_{n-|\mathcal{I}|}}{\operatorname{argmin}} \left\{ \tfrac{1}{2}\|\tilde{\mathbf{y}} - \tilde{\mathbf{A}}\mathbf{x}\|_2^2 + \boldsymbol{\theta}^{\mathrm{T}}\mathbf{x} + \tfrac{\beta}{2}\|\mathbf{x}\|_\mathbf{M}^2 \right\} \tag{$\mathcal{P}_{\bar{\mathcal{I}}}$}$$

up to some constant terms, with

$$\begin{cases} \tilde{\mathbf{y}} & = \mathbf{y} - \mathbf{A}_{\mathcal{I}}\mathbf{b} \\ \tilde{\mathbf{A}} & = \mathbf{A}_{\bar{\mathcal{I}}} + \mathbf{A}_{\mathcal{I}}\mathbf{B} \\ \boldsymbol{\theta} & = \alpha\mathbf{1} + \mathbf{B}^{\mathrm{T}}(\alpha\mathbf{1} + \beta\mathbf{b}) \\ \mathbf{M} & = \mathbf{I} + \mathbf{B}^{\mathrm{T}}\mathbf{B} \end{cases} \tag{3.7}$$

and where $\|\mathbf{x}\|_{\mathbf{M}}^2 \triangleq \mathbf{x}^{\mathrm{T}}\mathbf{M}\mathbf{x}$. The expression of ($\mathcal{P}_{\bar{\mathcal{I}}}$) is very similar to ($\mathcal{P}$) and it can be tackled by any solving method originally suited for the Elastic-Net with only slight modifications. Notice that with the reformulation, the dimension of the problem is reduced by $|\mathcal{I}|$ which makes it less costly to solve with an iterative method.

3.2.5 Reduced problem construction

Both screening and relaxing tests allow to reduce the problem dimension by transforming it. The reduced problem construction costs nothing with screening tests because one only has to remove coefficients in $\mathbf{x}$ and columns in $\mathbf{A}$. However, this is not the case with relaxing tests, because values (3.6)-(3.7) have to be computed to construct ($\mathcal{P}_{\bar{\mathcal{I}}}$). Safe regions are usually strengthened at each iteration with the new iterate provided by the solving method and new elements of $\mathcal{I}_+$ are progressively identified. Therefore, we can take advantage of *recursive update rules* to recompute (3.6)-(3.7) every time that a new element of $\mathcal{I}_+$ is identified, instead of recomputing all from scratch (Hager, 1989). Every time that a new coefficient of $\mathcal{I}_+$ is identified, the complexity to update ($\mathcal{P}_{\bar{\mathcal{I}}}$) is $O(|\bar{\mathcal{I}}|^2 + |\bar{\mathcal{I}}| \times m)$. As only *few* coefficients are non-zero in the context of sparse-linear models, values (3.6)-(3.7) are not updated too many times. Thus, the computational cost to update ($\mathcal{P}_{\bar{\mathcal{I}}}$) every time that new elements of $\mathcal{I}_+$ are identified is negligible compared to the overall resolution cost.

3.3 Screen & Relax algorithm

In the previous subsection, we derived how to identify $\mathcal{I}_0$ using screening tests and how to identify $\mathcal{I}_+$ using relaxing tests, given a safe region. We now present how to implement these tests in order to improve an *arbitrary* solving method for the Elastic-Net.

3.3.1 Combining screening and relaxing tests

Both screening and relaxing tests can be performed on top of an arbitrary solving method. In Algorithm 1, the procedure `MethodStep` represents one iteration of this underlying solving process. We denote by $\boldsymbol{\eta}^k$ all the computations done during the procedure `MethodStep` at iteration k that can be re-used during the rest of the iteration. For instance, $\boldsymbol{\eta}^k$ can store the gradient value, the current objective value, *etc*... Notice that $\boldsymbol{\eta}^k$ is passed as an argument to `ScreeningTest` and `RelaxingTest` procedures that identify elements of $\mathcal{I}_0$ and $\mathcal{I}_+$. This emphasizes the key ingredient of screening that is to re-use values already computed in order to implement tests (3.2) and (3.4) in a cheap way.

The set $\mathcal{I}_\bullet$ represents all the indices that have *not* been identified yet as belonging to $\mathcal{I}_0$ or $\mathcal{I}_+$. Through iterations, screening and relaxing tests are strengthened with new iterates computed and more and more elements are progressively removed from $\mathcal{I}_\bullet$. In this way, the problem dimension as well as the complexity of `MethodStep` is decreased. Every time that a screening or a relaxing test passes, problem data (3.6)-(3.7) are updated with a low computational burden using recursive update rules.

3.3.2 Direct convergence up to a machine-epsilon accuracy

Using both screening and relaxing tests provides another level of improvement. The optimizer of ($\mathcal{P}$) can be recovered up to a machine-epsilon accuracy using the following proposition.

Proposition 3.5. *If $\mathcal{I} \equiv \mathcal{I}_+$, one has*

$$\mathbf{x}^\star_{\mathcal{I}} = \mathbf{b} \qquad \textit{and} \qquad \mathbf{x}^\star_{\bar{\mathcal{I}}} = \mathbf{0} \tag{3.8}$$

where $\mathbf{b}$ *is defined in* (3.6).

In other words, if $\mathcal{I}_+$ has been entirely identified with relaxing tests, Proposition 3.5 allows recovering $\mathbf{x}^\star$ *exactly* because it admits a closed-form expression. As $\mathbf{b}$ is already computed to construct ($\mathcal{P}_{\bar{\mathcal{I}}}$), the solution of the Elastic-Net problem is recovered with a *machine-epsilon accuracy* at *no cost*. If all elements of $\mathcal{I}_+$ are identified soon enough, this allows to stop the algorithm rapidly with a convergence certificate. It is particularly interesting with gradient-based methods that usually struggle to converge when a high-accuracy is targeted.

One cannot ensure that $\mathcal{I}_+$ is identified entirely with every safe region and using relaxing tests *only*. Thankfully, some safe regions are guaranteed to identify *all* elements of $\mathcal{I}_0$ and $\mathcal{I}_+$ when the current iterate $\mathbf{x}^k$ is closed enough to $\mathbf{x}^\star$. Also, when using *both* screening and relaxing tests, we are sure that $\mathcal{I}_+$ is entirely identified as soon as all the elements of $\mathcal{I}_\bullet$ have been classified into $\mathcal{I}_0$ or $\mathcal{I}_+$.

Algorithm 1: Screen & Relax

Input: $\mathbf{x}^0$, $\mathbf{A}$, $\mathbf{y}$, α, β

$\mathcal{I}_\bullet \leftarrow [\![1, n]\!]$
while *stopping criterion is not met* **do**
  $\mathbf{x}^k, \boldsymbol{\eta}^k \leftarrow \texttt{MethodStep}(\mathbf{x}^{k-1}, \mathbf{A}, \mathbf{y}, \alpha, \beta)$
  $\mathcal{I}_\bullet \leftarrow \mathcal{I}_\bullet \setminus \texttt{ScreeningTest}(\boldsymbol{\eta}^k)$ // with (3.2)
  $\mathcal{I}_\bullet \leftarrow \mathcal{I}_\bullet \setminus \texttt{RelaxingTest}(\boldsymbol{\eta}^k)$ // with (3.4)
  Update problem with recursive update rules if needed
  if $\mathcal{I}_\bullet \equiv \emptyset$ **then**
    return $\mathbf{x}^\star = [\mathbf{b}, \mathbf{0}]$ // Proposition 3.5
  end
end

3.4 Numerical results

We assess the Screen & Relax methodology using a projected-gradient (PG) algorithm, with an optimized step-size selection as solving process for ($\mathcal{P}$). This algorithm alternates between a gradient step toward a descent direction and a projection onto the positive orthan to ensure feasibility of the new iterate (Calamai et al., 1987). We first explain our experimental setup. Then, we focus on three of the improvements drawn by the Screen & Relax methodology, naming a reduction in the number of iterations, a conditioning improvement and a dimensionality reduction. In a final experiment, we show that our Screen & Relax method effectively reduces the cost of solving the Elastic-Net problem.

3.4.1 Experimental setup

We use dictionaries with size 100×300 of four different types. *Gaussian* and *Uniform* dictionaries are generated with entries sampled using the corresponding density law. *DCT* dictionaries are made-up with atoms corresponding to shifted cosine functions and are often used in signal processing (Strang, 1999), see *eg.* Section 2.2.3. Finally, *Toeplitz* dictionaries are diagonal-constant matrices arising in sparse-deconvolution problems (Böttcher et al., 2005), see *eg.* Section 2.2.2. As stated in Section 2.4.3, dictionaries with correlated atoms, such as Uniform and Toeplitz ones, make the problem harder to solve, contrary to Gaussian and DCT dictionaries that have low-correlated atoms. The observation $\mathbf{y}$ is generated randomly according to a centered Gaussian density with unit variance for Gaussian and DCT dictionaries and according to a Uniform density on $[0, 1]$ for Uniform and Toeplitz dictionaries. Each atom as well as the observation are ℓ_2-normalized in pre-processing. We use the two different couples of tuning parameters $(\alpha, \beta) = (0.2 \times \alpha_{\max}, 0.5 \times \alpha_{\max})$ and $(\alpha, \beta) = (0.5 \times \alpha_{\max}, 0.2 \times \alpha_{\max})$ to test different working regimes. To illustrate improvements drawn by screening and relaxing tests independently and jointly, we compare a classical PG method (`PG`), a PG with screening tests *only* (`PG+s`), a PG with relaxing tests *only* (`PG+r`) and a PG with *both* screening and relaxing tests (`PG+sr`). We note that the first three algorithms correspond to incomplete versions of `PG+sr` where some steps have been disabled in order to emphasize the computational gain induced by screening and relaxing methodologies.

To perform both screening and relaxing tests, we use the so-called *GAP safe region*, which is a sphere centred on the point $\mathbf{u}^k \triangleq \mathbf{y} - \mathbf{A}\mathbf{x}^k$ at iteration k, and whose radius decreases with the *duality-gap* (Ndiaye et al., 2016). The duality gap is the difference between the primal and the dual objectives for a primal-dual feasible couple, and goes to zero as the solving method converges toward the optimum. This ensures that both $\mathcal{I}_0$ and $\mathcal{I}_+$ are identified if $\mathbf{x}^k$ is close enough to $\mathbf{x}^\star$ and that Proposition 3.5 can be used to directly met a machine-epsilon accuracy. The GAP region allows to implement tests (3.2) and (3.4) in only $O(1)$ by re-using the gradient value at each iteration. Thus, adding screening and relaxing tests to the PG is *cost-free*, except for the few updates of (3.6)-(3.7) needed each time that an element of $\mathcal{I}_+$ is identified.

3.4.2 Improving the convergence speed

The first experiment carried out aims to highlight the acceleration effect of knowing some non-zero elements at the *initialization* of the algorithm. For different initial knowledge of $\mathcal{I}_+$, we first begin by transforming ($\mathcal{P}$) into ($\mathcal{P}_{\bar{\mathcal{I}}}$). Then we run the classical PG method without screening and relaxing test to solve ($\mathcal{P}_{\bar{\mathcal{I}}}$). Thus, we directly observe how the resolution of the problem behaves, given some addition knowledge on non-zero entries of the optimizer at the initialization.

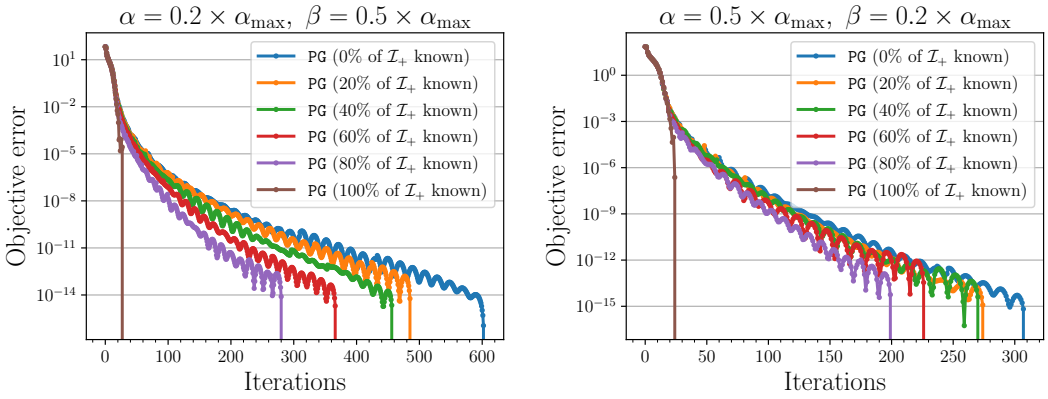

Figure 5: Objective error at a given iteration for different initial knowledge of $\mathcal{I}_+$ and with a Gaussian dictionary.

We can see in Figure 5 that the better the knowledge of $\mathcal{I}_+$, the lower the number of iterations required to met a given accuracy on the approximation of the objective value. The reformulated problem ($\mathcal{P}_{\bar{\mathcal{I}}}$) seems to be better handled by the PG method. We can also notice that when α is small, the improvement is even more significant. Indeed, a low α weighting linear term in ($\mathcal{P}$) makes the PG method slower because fewer atoms have to be selected, so the choice is trickier. In the reformulation ($\mathcal{P}_{\bar{\mathcal{I}}}$), the linear term is weighted by a more complex term that does not only depend on α, but also on β and other parameters. Hence, the better behavior of PG.

3.4.3 Conditionning improvment

In addition of reducing the number of iterations, relaxing tests also continually improve the conditioning of the problem during the resolution process. To emphasize this improvement, we measure the evolution of the *mutual coherence* μ when a given duality gap τ is met (Lu et al., 2018). The mutual coherence measures the correlation between the atoms of $\mathbf{A}$ and the larger the coherence, the worst the conditioning. A problem with a smaller coherence is generally easier to solve.

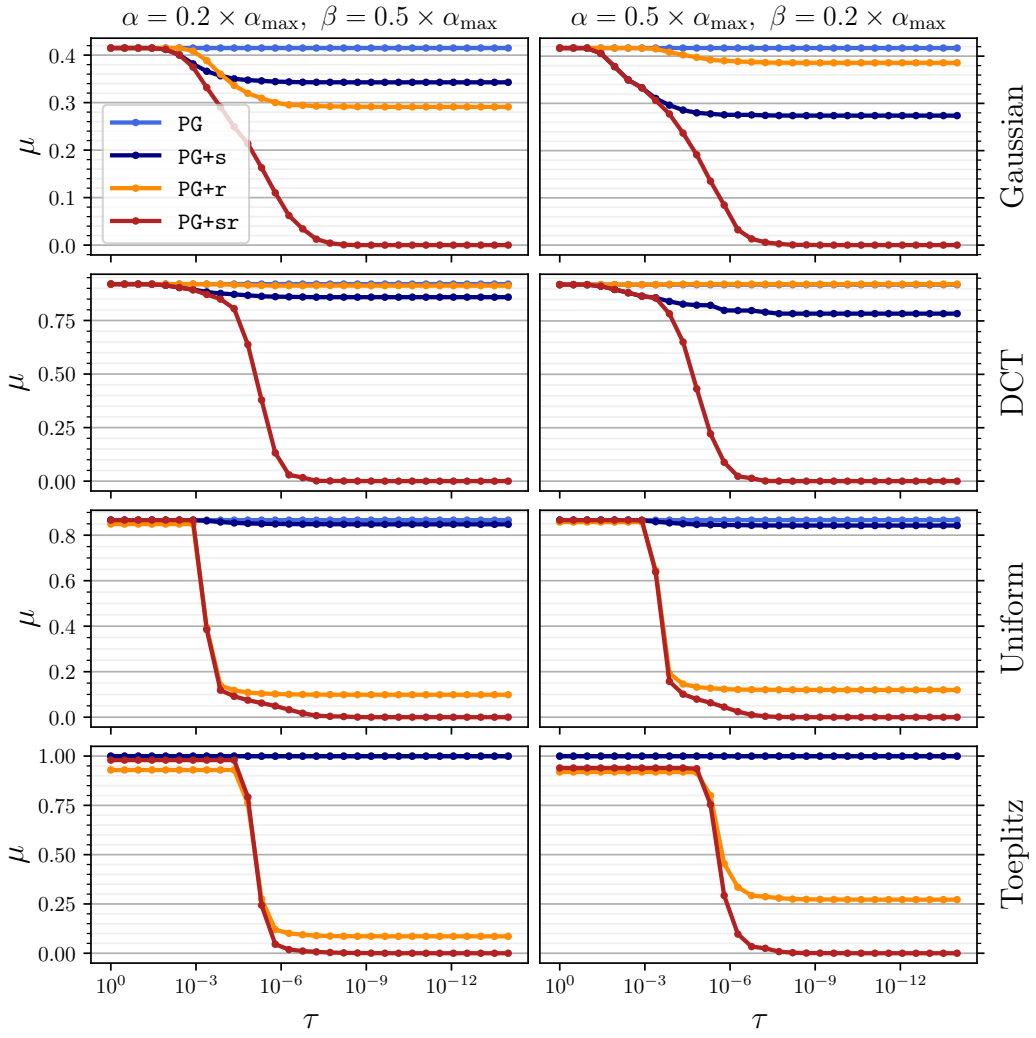

Figure 6: Mutual coherence diminution using screening and relaxing tests.

In Figure 6, we can clearly see that using a Screen & Relax methodology, the mutual coherence is progressively decreased because of the removing of atoms during the problem resolution. Hence, as the solving method iterates, its convergence speed is also enhanced by the problem modification and the conditioning improvements. Thus, a better accuracy can potentially be met with fewer iterations. Notice the huge decreasing of mutual coherence in the case of correlated dictionaries. Indeed, relaxing tests usually remove the most correlated atoms first because they are more likely to pass test (3.4). Hence, the mutual coherence is even more decreases.

3.4.4 Dimensionnality reduction

In addition to the conditioning improvement, combining screening and relaxing tests allows to decrease the dimension of the problem, which lead to cheaper iterations. In the following figure, we plot the number of remaining atoms $|\mathcal{I}_\bullet|$ when a given duality gap τ is met.

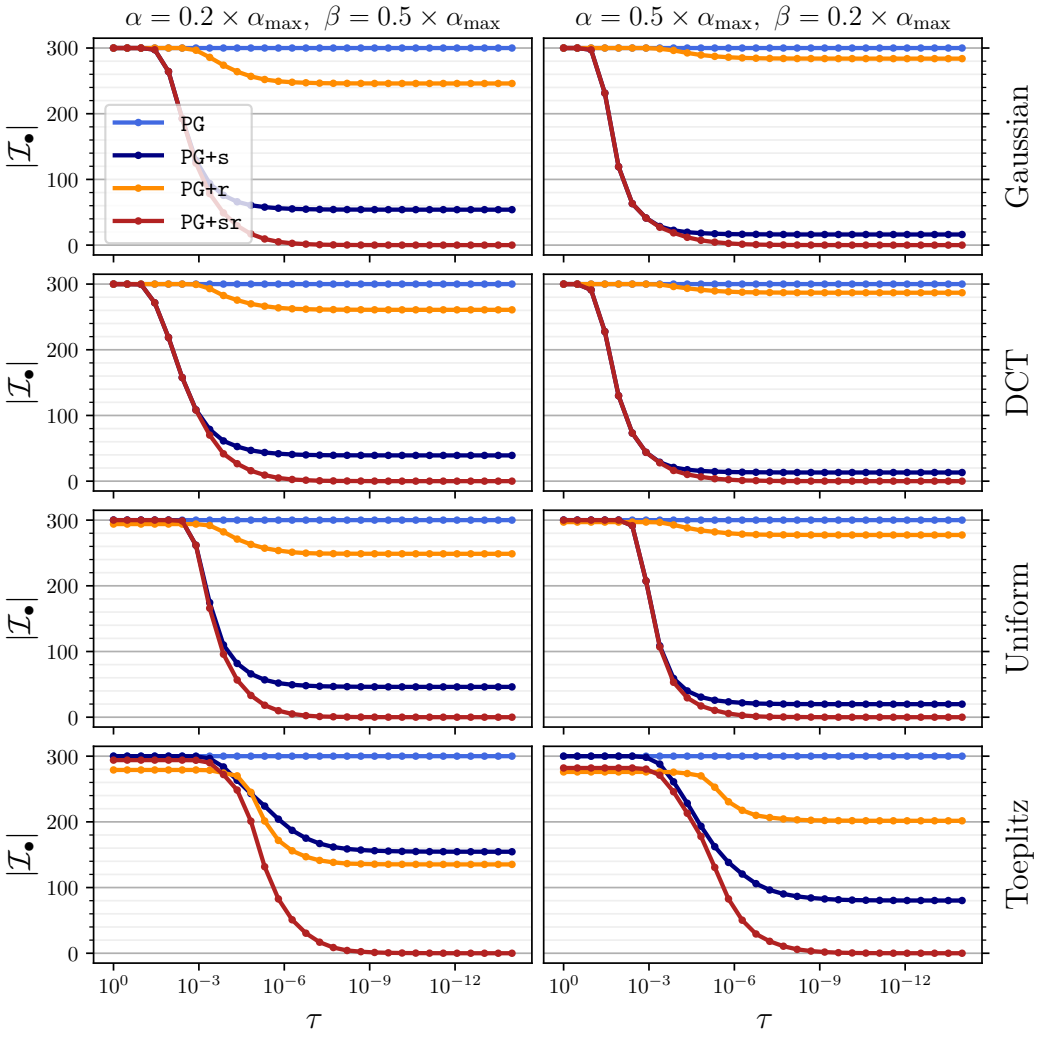

Figure 7: Dimension reduction of the problem using screening and relaxing tests.

A large part of the dimension reduction is done by screening tests. Nonetheless, relaxing tests allow to remove atoms that cannot be treated by screening tests. By combining these two methods, we can see that the Screen & Relax strategy brings the dimension down to zero rapidly. Notice that when $|\mathcal{I}_\bullet| = 0$, then Proposition 3.5 can directly be used to recover $\mathbf{x}^\star$ exactly.

3.4.5 Performance profiles

In this final experiment, we gather all previous observations in a single graphic. We plot Dolan-Moré performance profiles (Dolan et al., 2002) to measure the empirical percentage $\rho_{\%}$ of problems solved over 100 trials for a fixed complexity budget expressed in FLOPs, the number of floating-point operations performed (Hunger, 2005), and for a target accuracy τ on the duality gap. In particular, we set a budget of 2×10^6 FLOPs for Gaussian and DCT dictionaries and a budget of 2×10^7 FLOPs for Uniform and Toeplitz dictionaries. Thus, this final experiment aims to show the improvement in the number of problems solved with a limited complexity budget.

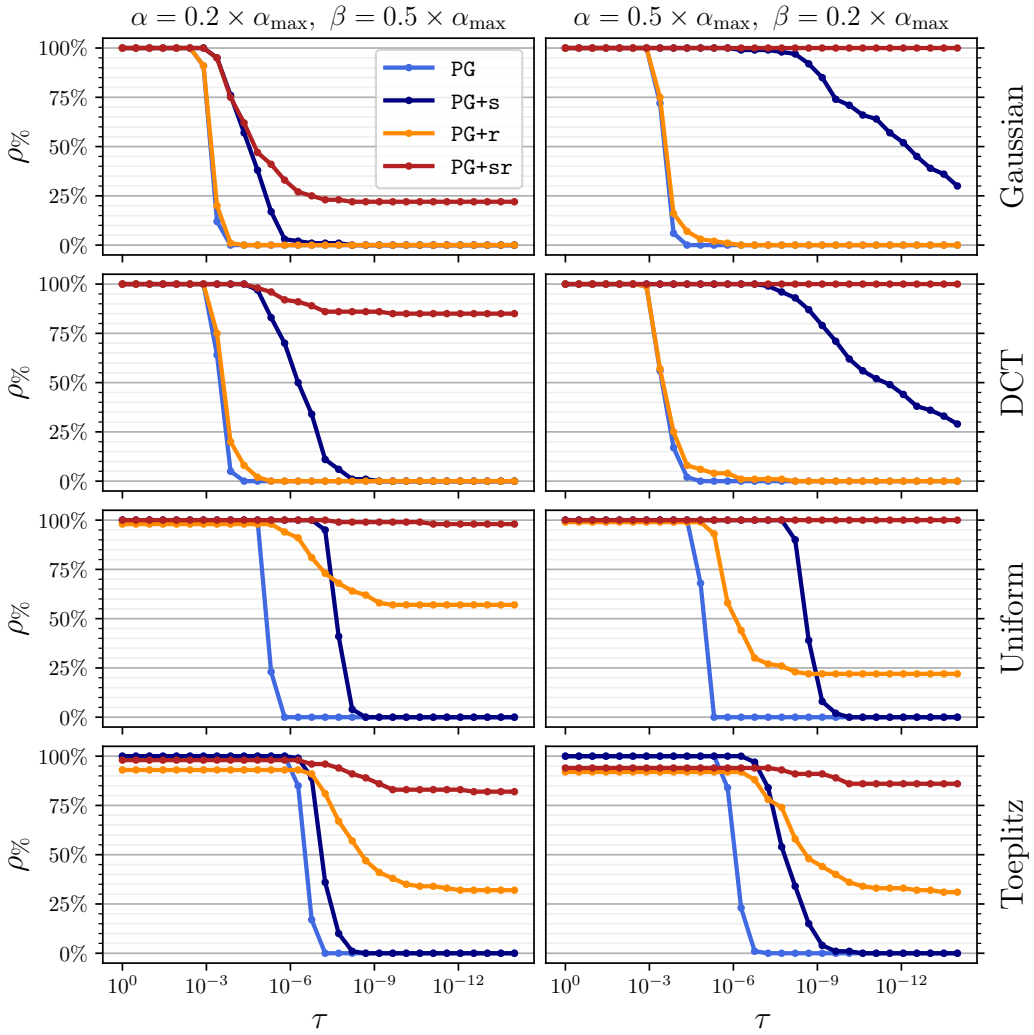

Figure 8: Dolan-Moré performance profiles.

Figure 8 confirms that the *Screen & Relax* method reduces the complexity of the solving process as more problems are solved at a given accuracy for a fixed budget. It also emphasizes that using Proposition 3.5, the number of problems solved to an high accuracy is significantly increased because of the problem dimension reduction, but also because of the conditioning improvement draw by the Screen & Relax method. These improvements are due to the three factors lightened in the previous experiments, namely a reduction in the number of iterations, a conditioning improvement and a dimensionality reduction.

4 ℓ_0-screening : Fixing integer variables in MIP sparse linear problems

In the previous section, we presented an acceleration method for the Elastic-Net problem, tackled with convex optimization methods. In this sequel, we present how to transpose screening and relaxing concepts from *convex problems* to *MIPs* in the context of sparse linear problems. The aim is to make sparse linear problems with an ℓ_0-norm as sparsity-including function more tractable by the identification and the fixing of particular solution indices during the resolution process.

4.1 MIP formulation of a sparse linear problem

As stated in Section 2, sparsity-including functions that directly penalize the non-sparsity of a vector give more interesting solutions to the sparse linear problem (2.1). For instance, one can use the ℓ_0-norm that counts the number of non-zero coefficients in its input, but this leads to NP-hard problems (Woeginger, 2003). Such problems were originally considered intractable in practice. However, the discrete optimization community began to address them since few years and already managed to design efficient solving methods (Bertsimas et al., 2016). In this section, we are interested in the ℓ_0-redularized least-squares problem, given by[9]

$$(\mathbf{x}^\star, p^\star) \ : \min_{\mathbf{x}\in\mathbf{R}^n} \ \left\{ \tfrac{1}{2}\|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_0 \right\} \tag{$\mathcal{P}$}$$

where $\mathbf{x}^\star$ is an optimizer of the problem and $p^\star$ is the associated optimal objective value. A Least-Squares loss ensures data-fidelity, and an ℓ_0-norm enforces sparsity of the solution. As it, this problem is not a MIP, strictly speaking. However, assume that one can choose M large enough so that $\|\mathbf{x}^\star\|_\infty \leq M$ for any optimizer $\mathbf{x}^\star$ of ($\mathcal{P}$), then an equivalent formulation of the problem is

$$(\mathbf{x}^\star, \mathbf{z}^\star, p^\star) \ : \ \left\{ \begin{array}{ll} \min & \tfrac{1}{2}\|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2 + \lambda \mathbf{1}^{\mathrm{T}}\mathbf{z} \\ \text{st.} & -M\mathbf{z} \leq \mathbf{x} \leq M\mathbf{z} \\ & \mathbf{x}\in\mathbf{R}^n, \ \mathbf{z}\in\{0,1\}^n \end{array} \right. \tag{$\mathcal{P}$}$$

The binary variable $\mathbf{z}$ encodes the nullity or the non-nullity of $\mathbf{x}$[10]. This reformulation of the problem is a MIP and can be handled by many solvers such as `CPLEX`, `GUROBI` or `SCIP`. Recently, custom Branch-and-Bounds (BnBs) were also designed for ($\mathcal{P}$) and usually outperform generic solvers by exploiting the sparse property of the problem (Bourguignon et al., 2015).

Contributions We introduce *zero-one screening tests* implemented inside an *arbitrary* BnB suited for ($\mathcal{P}$). These tests allow to fix some binary variables within the BnB exploration. In this way, we avoid processing some nodes that lead to provably suboptimal solutions *during* the resolution process, and thereby to reduce the solution time. As far as we know, only a similar idea was developed for ($\mathcal{P}$) with an additional ℓ_2-norm penalty. Tests were done differently and were only performed *before* the solving process (Atamturk, 2020).

[9] Some notations may be used in both Section 3 and Section 4 but they are totally independent !

[10] To lighten notations, we mostly use the formation with the ℓ_0-norm so the use of a binary variable $\mathbf{z}$ in the model is implied.

4.2 Custom Branch-and-Bound

Zero-one screening tests are performed on top of an arbitrary BnB solving ($\mathcal{P}$). First, we present the most recent and probably most efficient BnB for this problem introduced in (Ben Mhenni et al., 2020). We will use it as an example to derive further explanations.

4.2.1 Key ingredients of a Branch-and-Bound

The BnB is an algorithm design paradigm for discrete and combinatorial optimization problems (Lawler et al., 1966). It consists of a systematic testing of candidate solutions by enumerating integer constraints. A tree is firstly created with a root where none of the integer variables are fixed. The algorithm then explores branches of this tree, representing subsets of the solution set where some integer variables are fixed. Before enumerating the candidate solutions of a branch, it is checked against upper and lower estimated bounds on the optimal solution, and is pruned if it cannot produce a better solution than the best one found so far by the algorithm. The efficiency of the BnB mostly depends on the estimation of the lower and upper bounds of each node of the tree. In the worst case, the algorithm performs an exhaustive search in the tree. For the particular case of problem ($\mathcal{P}$), the root corresponds to a problem where no constraints are set about the nullity of variable $\mathbf{x}$. Each new node is created either by including the constraint $\{x_i = 0\}$ or $\{x_i \neq 0\}$ for an index i that is defined by branching rules.

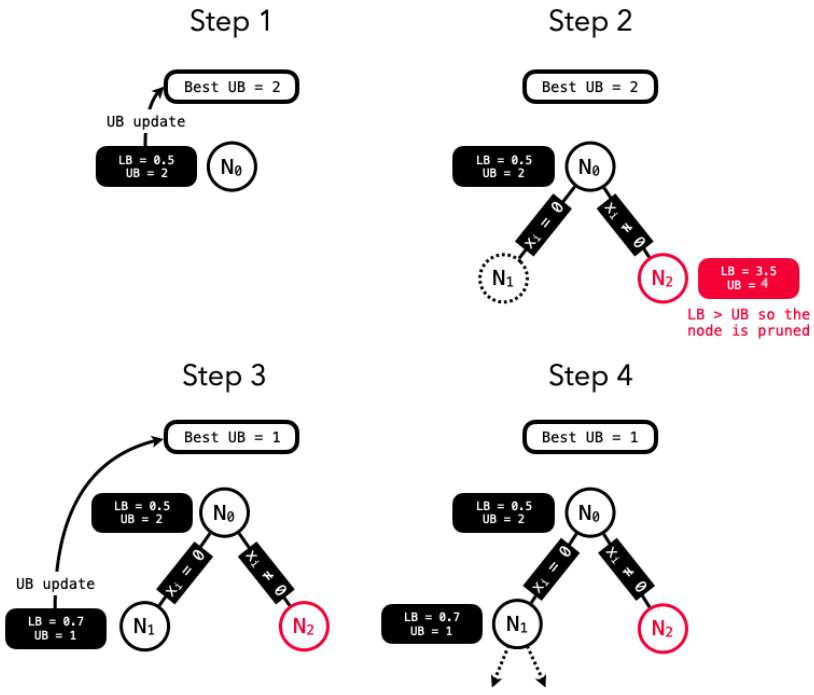

Figure 9: Illustration of a BnB algorithm.

4.2.2 Bounding step

In the tree, each node is *uniquely* defined by its current set of fixed constraints, denoted $\mathcal{S} \triangleq (\mathcal{S}_0, \mathcal{S}_1)$. In the latter

- $\mathcal{S}_1$ is the set of entries in $\mathbf{x}$ fixed to non-zero
- $\mathcal{S}_0$ is the set of entries in $\mathbf{x}$ fixed to zero

Variables fixed to zero can be removed from the problem without changing it. At a node, decisions are to be made concerning the remaining variables, indexed by $\bar{\mathcal{S}}$. We write $\mathcal{S} \subseteq \mathcal{S}'$ if $\mathcal{S}'$ is a subnode of $\mathcal{S}$ in reference to the inclusion of their constraints.

Rather than directly including constraints in ($\mathcal{P}$), it is more clever to re-formulate the sub-problem of each node. At node $\mathcal{S}$ the corresponding subproblem writes

$$(\bar{\mathbf{x}}^{\mathcal{S}}, \bar{p}^{\mathcal{S}}) \ : \ \left\{ \begin{array}{ll} \min & \frac{1}{2}\|\mathbf{y} - \mathbf{A}_{\mathcal{S}_1}\mathbf{x}_{\mathcal{S}_1} - \mathbf{A}_{\bar{\mathcal{S}}}\mathbf{x}_{\bar{\mathcal{S}}}\|_2^2 + \lambda\|\mathbf{x}_{\bar{\mathcal{S}}}\|_0 + \lambda \mathrm{card}(\mathcal{S}_1) \\ \text{st.} & \|\mathbf{x}_{\mathcal{S}_1}\|_\infty \leq M, \ \|\mathbf{x}_{\bar{\mathcal{S}}}\|_\infty \leq M \\ & \mathbf{x}_{\mathcal{S}_1} \in \mathbf{R}^{|\mathcal{S}_1|}, \ \mathbf{x}_{\bar{\mathcal{S}}} \in \mathbf{R}^{|\bar{\mathcal{S}}|} \end{array} \right. \tag{$\bar{\mathcal{P}}^{\mathcal{S}}$}$$

Recall that we denote $\mathbf{A}_{\mathcal{S}}$ (*resp.* $\mathbf{x}_{\mathcal{S}}$) the matrix (*resp.* vector) whose columns (*resp.* indices) are in $\mathcal{S}$. Notice that variables $\mathbf{x}_{\bar{\mathcal{S}}}$ and $\mathbf{x}_{\mathcal{S}_1}$ are split and that variables fixed to zero are removed from the problem, which does not change it. Furthermore, observe that the penalty of variables fixed to non-zero is directly expressed in function of $\mathcal{S}_1$, but this does not necessarily ensure that $\mathbf{x}_{\mathcal{S}_1}$ will have only non-zero components.

To test if the current branch may yield an optimal solution, a lower bound on $\bar{p}^{\mathcal{S}}$ is computed by solving the *continuous relaxation* of ($\bar{\mathcal{P}}^{\mathcal{S}}$). This relaxation writes

$$(\mathbf{x}^{\mathcal{S}}, p^{\mathcal{S}}) \ : \ \left\{ \begin{array}{ll} \min & \frac{1}{2}\|\mathbf{y} - \mathbf{A}_{\mathcal{S}_1}\mathbf{x}_{\mathcal{S}_1} - \mathbf{A}_{\bar{\mathcal{S}}}\mathbf{x}_{\bar{\mathcal{S}}}\|_2^2 + \frac{\lambda}{M}\|\mathbf{x}_{\bar{\mathcal{S}}}\|_1 + \lambda \mathrm{card}(\mathcal{S}_1) \\ \text{st.} & \|\mathbf{x}_{\mathcal{S}_1}\|_\infty \leq M, \ \|\mathbf{x}_{\bar{\mathcal{S}}}\|_\infty \leq M \\ & \mathbf{x}_{\mathcal{S}_1} \in \mathbf{R}^{|\mathcal{S}_1|}, \ \mathbf{x}_{\bar{\mathcal{S}}} \in \mathbf{R}^{|\bar{\mathcal{S}}|} \end{array} \right. \tag{$\mathcal{P}^{\mathcal{S}}$}$$

Only the ℓ_1-norm and its weight differ between the relaxed and the non-relaxed problem but ($\mathcal{P}^{\mathcal{S}}$) is convex whereas ($\bar{\mathcal{P}}^{\mathcal{S}}$) is still NP-hard. When $p^{\mathcal{S}}$ has been computed, it is compared to the best upper bound computed so far on $p^\star$, that is denoted $\overline{p}$. If $p^{\mathcal{S}} > \overline{p}$, then the branch will necessarily yield a suboptimal solution, so it is pruned. Otherwise, two child nodes are created by branching on a particular index defined by a branching rule that is derived further. We also derive later how to efficiently solve ($\mathcal{P}^{\mathcal{S}}$).

4.2.3 Upper bound update

To be efficient, a BnB must be able to compute both tight lower and upper bounds on $p^\star$. Although lower bounds are computed for every node, only one upper bound is used through the tree, and it corresponds to the best upper bound on $p^\star$ known so far. This upper bound is updated when a *perfect relaxation* is found or using heuristics. The relaxation is perfect when $|x_i^{\mathcal{S}}| \in \{0, M\}$ for all $i \in \bar{\mathcal{S}}$ because $\frac{\lambda}{M}\|\mathbf{x}_{\bar{\mathcal{S}}}^{\mathcal{S}}\|_1 = \lambda\|\mathbf{x}_{\bar{\mathcal{S}}}^{\mathcal{S}}\|_0$. This means that $\mathbf{x}^{\mathcal{S}}$ is also an optimizer of ($\bar{\mathcal{P}}^{\mathcal{S}}$) and thereby $p^{\mathcal{S}} = \bar{p}^{\mathcal{S}} \geq p^\star$. Thus, we can try to update the best upper bound as $\overline{p} \leftarrow \min(\overline{p}, p^{\mathcal{S}})$. If the relaxation is not perfect, one can heuristically fix $\mathbf{x}_{\bar{\mathcal{S}}} = \mathbf{0}$ in ($\bar{\mathcal{P}}^{\mathcal{S}}$), which yields the problem

$$(\mathbf{x}_f^{\mathcal{S}}, p_f^{\mathcal{S}}) \ : \ \left\{ \begin{array}{ll} \min & \frac{1}{2}\|\mathbf{y} - \mathbf{A}_{\mathcal{S}_1}\mathbf{x}_{\mathcal{S}_1}\|_2^2 + \lambda \mathrm{card}(\mathcal{S}_1) \\ \text{st.} & \|\mathbf{x}_{\mathcal{S}_1}\|_\infty \leq M, \ \mathbf{x}_{\mathcal{S}_1} \in \mathbf{R}^{\mathcal{S}_1} \end{array} \right. \tag{$\mathcal{P}_f^{\mathcal{S}}$}$$

This is a Least-Squares problem with additional box-constraints. It can be solved with the BVLS algorithm (Stark et al., 1995) in polynomial time. As $\mathbf{x}_f^{\mathcal{S}}$ is a feasible value for ($\mathcal{P}$), $p_f^{\mathcal{S}}$ is also a candidate to update $\overline{p}$.

4.2.4 Branching step

Once a relaxation has been solved and if the node has not been pruned, two child nodes are created by either including the constraint $\{x_i = 0\}$ or $\{x_i \neq 0\}$. The branching index i is defined by a branching rule. In general, it is chosen as the most fractional variables among integer ones that are still unfixed. The branching rule

$$i \in \underset{j \in \bar{\mathcal{S}}}{\operatorname{argmax}} |x_j^{\mathcal{S}}| \tag{4.1}$$

seems more efficient in practice. It is inspired by forward search method of greedy algorithms (T. Zhang, 2008).

4.2.5 Active-Set algorithm for the relaxation resolution

The relaxation (4.1) is a convex-constrained problem that can be solved with various optimization methods. The efficiency of the BnB presented in (Ben Mhenni et al., 2020) mostly relies on the fast resolution of relaxation that leverages on the fact that only one constraint differ between two nodes. They use an Active-Set algorithm, suited for quadratic-constrained problems, that explores the faces of the polytop defining the feasible region, decreasing the objective function at each iteration until convergence. It can be viewed as a generalization of the simplex method for problems with a quadratic objective (Dantzig, 1990). As the polytop is only slightly modified between two nodes, performing a warm-start with the parent node solution allows converging with very few iterations.

First, let define some *working sets* that encode the *support configuration* of the current iterate :

$$\left\{\begin{array}{lllll}
\bar{\mathcal{S}}_0 & \triangleq \{i \in \bar{\mathcal{S}}, & \text{st. } 0 = & |x_i| & \} \\
\bar{\mathcal{S}}_{\text{in}} & \triangleq \{i \in \bar{\mathcal{S}}, & \text{st. } 0 < & |x_i| < M & \} \\
\bar{\mathcal{S}}_{\square} & \triangleq \{i \in \bar{\mathcal{S}}, & \text{st. } & |x_i| = M & \} \\
\mathcal{S}^1_{\text{in}} & \triangleq \{i \in \mathcal{S}_1, & \text{st. } & |x_i| < M & \} \\
\mathcal{S}^1_{\square} & \triangleq \{i \in \mathcal{S}_1, & \text{st. } & |x_i| = M & \}
\end{array}\right. \tag{4.2}$$

These sets indicate which constraints are saturated in ($\mathcal{P}^{\mathcal{S}}$). Using this notation, we can write optimality conditions for the problem as

$$\left\{\begin{array}{l}
\mathbf{c}_{\bar{\mathcal{S}}_{\text{in}}} = (\lambda/M) \odot \operatorname{sign}(\mathbf{x}_{\bar{\mathcal{S}}_{\text{in}}}) \\
\mathbf{c}_{\mathcal{S}^1_{\text{in}}} = \mathbf{0}
\end{array}\right. \quad \text{and} \quad \left\{\begin{array}{lll}
|\mathbf{c}_{\bar{\mathcal{S}}_0}| & < & \lambda/M \\
\mathbf{c}_{\bar{\mathcal{S}}_{\square}} \odot \operatorname{sign}(\mathbf{x}_{\bar{\mathcal{S}}_{\square}}) & \geq & \lambda/M \\
\mathbf{c}_{\mathcal{S}^1_{\square}} \odot \operatorname{sign}(\mathbf{x}_{\mathcal{S}^1_{\square}}) & \geq & \mathbf{0}
\end{array}\right. \tag{4.3}$$

where $\odot$ is the Hadamard product and where $\mathbf{c} \triangleq \mathbf{A}^{\mathrm{T}}(\mathbf{y} - \mathbf{A}_{\bar{\mathcal{S}}}\mathbf{x}_{\bar{\mathcal{S}}} - \mathbf{A}_{\mathcal{S}_1}\mathbf{x}_{\mathcal{S}_1})$. Let $\boldsymbol{\theta}$ be the sign of $\mathbf{x}$, then the solution of the first system of (4.3) reads

$$\left\{\begin{array}{ll}
\mathbf{x}_{\bar{\mathcal{S}}_{\text{in}}} & = (\mathbf{A}^{\mathrm{T}}_{\bar{\mathcal{S}}_{\text{in}}} \mathbf{B} \mathbf{A}_{\bar{\mathcal{S}}_{\text{in}}})^{-1} (\mathbf{A}^{\mathrm{T}}_{\bar{\mathcal{S}}_{\text{in}}} \mathbf{B}\mathbf{r} - \frac{\lambda}{M} \boldsymbol{\theta}_{\bar{\mathcal{S}}_{\text{in}}}) \\
\mathbf{x}_{\mathcal{S}^1_{\text{in}}} & = (\mathbf{A}^{\mathrm{T}}_{\mathcal{S}^1_{\text{in}}} \mathbf{A}_{\mathcal{S}^1_{\text{in}}})^{-1} (\mathbf{A}^{\mathrm{T}}_{\mathcal{S}^1_{\text{in}}} \mathbf{r} - \mathbf{A}^{\mathrm{T}}_{\mathcal{S}^1_{\text{in}}} \mathbf{A}_{\mathcal{S}^1_{\text{in}}} \mathbf{x}_{\bar{\mathcal{S}}_{\text{in}}})
\end{array}\right. \tag{4.4}$$

where $\mathbf{B} \triangleq \mathbf{I} - \mathbf{A}_{\mathcal{S}^1_{\text{in}}} \left(\mathbf{A}_{\mathcal{S}^1_{\text{in}}}^{\mathrm{T}} \mathbf{A}_{\mathcal{S}^1_{\text{in}}}\right)^{-1} \mathbf{A}_{\mathcal{S}^1_{\text{in}}}^{\mathrm{T}}$, $\mathbf{I}$ is the identity matrix of appropriate size and where $\mathbf{r} \triangleq \mathbf{y} - \mathbf{A}_{\bar{\mathcal{S}}_\square} \mathbf{x}_{\bar{\mathcal{S}}_\square} - \mathbf{A}_{\mathcal{S}^1_\square} \mathbf{x}_{\mathcal{S}^1_\square}$.

The algorithm starts from any support configuration and defines an initial point by computing its active variables with (4.4). Then, the inactive variable in $\mathbf{x}$ that most violates the inequality optimality conditions in the right system of (4.3) is put into $\bar{\mathcal{S}}_{\text{in}}$ or $\mathcal{S}^1_{\text{in}}$. A new point $\mathbf{x}^{\text{new}}$ is obtained by computing the new active variables with (4.4) and fixed variables are left unchanged. Then, a discrete line search is performed from $\mathbf{x}$ to $\mathbf{x}^{\text{new}}$. Let $f(t)$ denote the value of the objective function in ($\mathcal{P}^{\mathcal{S}}$) at $\mathbf{x}_t \triangleq \mathbf{x} + t(\mathbf{x}^{\text{new}} - \mathbf{x})$ for $t \in [0, t_{\max}]$, where $t_{\max}$ defines the maximum value of $t \leq 1$ such that $\|\mathbf{x}_t\|_\infty \leq M$. By construction, $f(\cdot)$ is continuous and piecewise quadratic on $[0, t_{\max}]$, and its expression changes at each sign change of one component of $\mathbf{x}_t$. Consequently, the minimization of f is performed exactly by evaluating f over a finite (and in practice, often small) number of points. Each iteration decreases the value of the objective function and explores a new support configuration, until all optimality conditions in (4.3) are satisfied. The algorithm converges toward the solution in a finite number of iterations (Lee et al., 2007).

From the computational point of view, each iteration mainly consists in solving the two linear systems in (4.4), with respective size $|\bar{\mathcal{S}}_{\text{in}}|$ and $|\mathcal{S}^1_{\text{in}}|$. In practice, the support configuration only slightly changes between two consecutive iterations, therefore recursive updates can be used advantageously. The blockwise inversion technique seems to be the most efficient method in practice (F. Zhang, 2006). The algorithm is initialized by the previously computed solution at the parent node, since the two problems only differ by one variable.

4.3 Fixing variables with zero-one screening tests

During a BnB, binary variables are progressively fixed with branching rules and by solving successive relaxations. In the following, we show how to perform *zero-one screening tests* ensuring that fixing $\{x_i = 0\}$ or $\{x_i \neq 0\}$ for some $i \in \bar{\mathcal{S}}$ at node $\mathcal{S}$ will necessarily yield a *suboptimal* solution, without having to solve further relaxations. If such test passes at node $\mathcal{S}$, one can directly fix the opposite constraint safely in the node, which divides by two the size of the subtree below node $\mathcal{S}$. Thus, we can hope to reduce the overall number of nodes treated and thereby the solution time. Principal advantages of these tests are that they can be performed with a *low computational burden* and can test *any* of variables that are still unfixed at the node where they are performed.

Zero-one screening tests leverage on the strong relation linking the objective of Fenchel dual of relaxations through different nodes of the BnB tree. The dual of the relaxation at node $\mathcal{S}$ writes

$$(\mathbf{u}^{\mathcal{S}}, d^{\mathcal{S}}) \;:\; \max_{\mathbf{u} \in \mathbf{R}^m} \Big\{ \mathrm{D}^{\mathcal{S}}(\mathbf{u}) = \tfrac{1}{2}\|\mathbf{y}\|_2^2 - \tfrac{1}{2}\|\mathbf{y} - \mathbf{u}\|_2^2 - M \sum_{i \in \bar{\mathcal{S}}} [|\mathbf{a}_i^{\mathrm{T}}\mathbf{u}| - \tfrac{\lambda}{M}]_+ - M \sum_{i \in \mathcal{S}_1} (|\mathbf{a}_i^{\mathrm{T}}\mathbf{u}| - \tfrac{\lambda}{M}) \Big\} \qquad (\mathcal{D}^{\mathcal{S}})$$

where $[x]_+ \triangleq \max(x, 0)$. Note that the objective function is particular since it is composed of a basic term, a term corresponding to $\bar{\mathcal{S}}$ and a term corresponding to $\mathcal{S}_1$. Hence between

two consecutive nodes, only one component differs ant it is easy to link their objective. Let define *pivot values* as

$$\boldsymbol{\gamma}_i^0(\mathbf{u}) \triangleq M[|\mathbf{a}_i^{\mathrm{T}}\mathbf{u}| - \tfrac{\lambda}{M}]_+ \qquad \text{and} \qquad \boldsymbol{\gamma}_i^1(\mathbf{u}) \triangleq M[\tfrac{\lambda}{M} - |\mathbf{a}_i^{\mathrm{T}}\mathbf{u}|]_+$$

One only has to add $\boldsymbol{\gamma}_i^0(\cdot)$ or $\boldsymbol{\gamma}_i^1(\cdot)$ to $\mathrm{D}^{\mathcal{S}}(\cdot)$ in order to express the dual objective of the two child nodes constructed by branching on i. Using this remark, we can derive the main result used to build zero-one screening tests.

Proposition 4.1. *At node $\mathcal{S}$, let $\overline{p}$ be an upper bound on $p^\star$ and let $i \in \bar{\mathcal{S}}$, then $\forall\, \mathbf{u} \in \mathbf{R}^m$*

$$\mathrm{D}^{\mathcal{S}}(\mathbf{u}) + \boldsymbol{\gamma}_i^0(\mathbf{u}) > \overline{p} \implies \textit{Node } \mathcal{S} \cup \{x_i = 0\} \textit{ cannot yield an optimal solution} \tag{4.5}$$

$$\mathrm{D}^{\mathcal{S}}(\mathbf{u}) + \boldsymbol{\gamma}_i^1(\mathbf{u}) > \overline{p} \implies \textit{Node } \mathcal{S} \cup \{x_i \neq 0\} \textit{ cannot yield an optimal solution} \tag{4.6}$$

The first implication is called a zero-screening test and the second implication is called a one-screening test, in reference to the constraint that is identified –or screened– as not leading to the optimal solution.

The left-hand-side of test (4.5) is equal to the dual objective of node $\mathcal{S} \cup \{x_i = 0\}$ evaluated in $\mathbf{u}$. In other words, we compute a lower bound on $d^{\mathcal{S}\cup\{x_i=0\}}$ with an arbitrary $\mathbf{u}$, that is also a lower bound on $\bar{p}^{\mathcal{S}\cup\{x_i=0\}}$. If the test passes, we know that this child node will yield a suboptimal solution. Hence, we can directly impose the constraint $\{x_i \neq 0\}$ safely at node $\mathcal{S}$. This divides by two the number of child nodes below $\mathcal{S}$. The same reasoning holds for the second test. We also provide another simple result :

Proposition 4.2. *Assume that (4.5) and (4.6) pass simultaneously for the same $i \in \bar{\mathcal{S}}$, but not necessarily for the same $\mathbf{u}$. Then node $\mathcal{S}$ can directly be pruned from the tree as it necessarily yields a suboptimal solution with either $\{x_i = 0\}$ or $\{x_i \neq 0\}$.*

The advantages of these tests are tree-fold. First, tests can be performed with any $\mathbf{u}$ and do *not* require to solve the relaxation associated to a child node. Second, they allow to tests *any* $i \in \bar{\mathcal{S}}$ and not only the index defined by the branching rule. Thus, we can hope to add several constraints at a given node. Finally, they can be performed with a *low computational burden* by re-using the value of $\mathbf{A}^{\mathrm{T}}\mathbf{u}$ which is usually already computed during the relaxation solving process since it is involved in the gradient computation.

4.4 Practical implementation

In the previous section, we explained how to identify some branching constraints that necessarily lead to a suboptimal solution, given some $\mathbf{u} \in \mathbf{R}^m$ and some upper bound $\overline{p}$. We now derive how to plug zero-one screening tests within an arbitrary BnB in order to pass as many screening tests as possible. To improve the chances to pass tests of Proposition 4.1, we seek to use both a tight upper bound $\overline{p}$ and an $\mathbf{u}$ that increases the most the left-hand-side of tests. We also want to avoid to over-penalize the BnB with too many additional computations.

An upper bound $\overline{p}$ is already provided by the BnB process, so we use this bound which is recovered at no cost. It remains to find a dual point $\mathbf{u} \in \mathbf{R}^m$ in order to perform zero-one screening tests. We propose to use the dual point $\mathbf{u}^k \triangleq \mathbf{y} - \mathbf{A}\mathbf{x}^k$, where $\mathbf{x}^k$ is the current iterate at iteration k during the resolution of ($\mathcal{P}^{\mathcal{S}}$). Evaluating the objective gradient

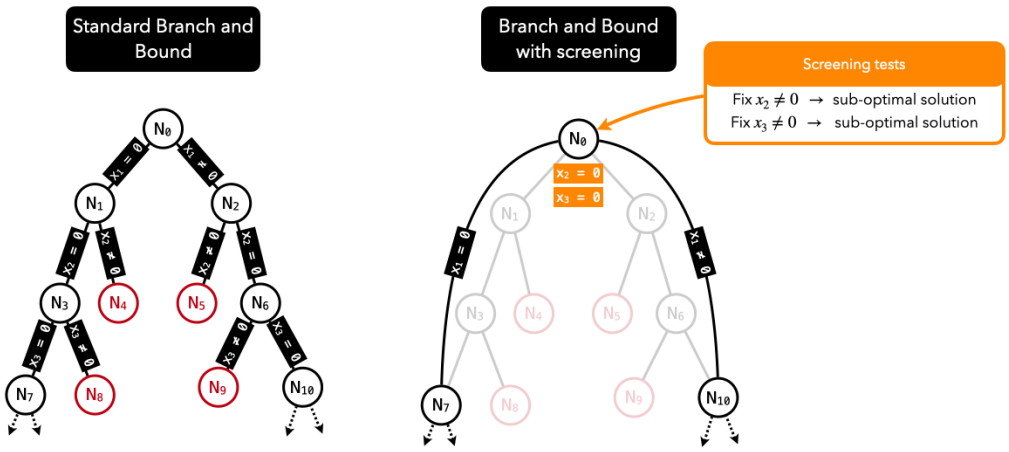

Figure 10: Left : Standard BnB. Node pruned by bounds are in red. Right : BnB with screening tests. At node N_0, screening tests yield that fixing $\{x_2 \neq 0\}$ and $\{x_3 \neq 0\}$ necessarily leads to a suboptimal solution. Thus, opposite constraints are directly added at node N_0 and then the classical branching scheme is performed. This allows to bypass 8 nodes.

at $\mathbf{x}^k$ also allows recovering values of $\mathbf{u}^k$ and $\mathbf{A}^{\mathrm{T}}\mathbf{u}^k$ at no cost, which allows to perform tests (4.5)-(4.6) in only $O(m)$. This computation is done in most solving method suited for ($\mathcal{P}^{\mathcal{S}}$) and in particular in the Active-Set used in the BnB presented in Section 4.2. To avoid performing to many additional computations, we only trigger zero-one screening tests when $\mathbf{x}^k$ improves the objective function. This is more likely to strengthen tests. When an zero-one screening test is passed, we can directly fix a new variable inside the subproblem. Hence, if zero-one screening tests are passed during the resolution of ($\mathcal{P}^{\mathcal{S}}$) and that variables are fixed, the optimizer recovered at the end of the relaxation solving process if not the one of node $\mathcal{S}$, but directly the one of node $\mathcal{S}$ plus the additional fixing constraints added.

Algorithm 2: Resolution of ($\mathcal{P}^{\mathcal{S}}$) with intra-node screening

Input: $\mathbf{x}^0, \mathbf{A}, \mathbf{y}, \lambda, M, \mathcal{S}_0, \mathcal{S}_1, \bar{\mathcal{S}}$

while *convergence criterion is not met* **do**
  $\mathbf{x}^k \leftarrow$ `SolvingMethodStep`$(\mathbf{x}^{k-1}, \mathbf{A}, \mathbf{y}, \lambda, M, \mathcal{S}_0, \mathcal{S}_1, \bar{\mathcal{S}})$
  if *objective decrases* **then**
    $\mathbf{u}^k \leftarrow \mathbf{y} - \mathbf{A}\mathbf{x}^k$ `// Or recovered from the method step`
    $\mathcal{S}_1 \leftarrow \mathcal{S}_1 \bigcup$ `ZeroScreeningTest`$(\mathbf{u}^k, \overline{p})$
    $\mathcal{S}_0 \leftarrow \mathcal{S}_0 \bigcup$ `OneScreeningTest`$(\mathbf{u}^k, \overline{p})$
  end
end

4.5 Numerical results

We now assess our new Zero-one screening method by comparing the resolution of ($\mathcal{P}$) with CPLEX, with the BnB presented in Section 4.2 and with the same BnB but overloaded with screening tests.

To generate problem data, we first sample a k-sparse vector $\mathbf{x}_{real} \in \mathbf{R}^n$ and we test different values of k. Spike locations are drawn uniformly within the indices and to avoid

too small spike values, each amplitude is sampled as $s + \text{sign}(s)$ where $s \sim \mathcal{N}(0, 1)$. Then, we use two different manners to construct the dictionary :

- **Sparse deconvolution :** The dictionary $\mathbf{A} \in \mathbb{R}^{120\times100}$ is the *discrete convolution matrix* corresponding to a sinus-cardinal sampled within $[-10; 10]$.
- **Subset selection :** The dictionary $\mathbf{A} \in \mathbb{R}^{500\times1000}$ is constructed with columns sampled as $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ for all $i \in \{1, \dots, 1000\}$ that are normalized.

Finally, we construct the data as $\mathbf{y} = \mathbf{A}\mathbf{x}_{real} + \epsilon$ where ϵ has a 10dB *signal-to-noise ratio*[11]. The regularization parameter is tuned statistically as $\lambda = 2\sigma^2 \log(\frac{n}{k} - 1)$ and we set $M = 1.5 \times \|\mathbf{A}^{\mathrm{T}}\mathbf{y}\|_\infty$. The sparse deconvolution setup is similar to practical cases encountered in image-processing contexts. It is hard to solve due to the strong correlation between atoms, even tough it is in relatively small dimension. The second setup is a more synthetic one, with a low correlation between atoms but with a higher dimension. We refer to (Bourguignon et al., 2015; Ben Mhenni et al., 2020) for more details about motivations to use these setups.

Solving methods compared are denoted

- `CPLEX` : ($\mathcal{P}$) is directly tackled using the CPLEX
- `BnB` : solve ($\mathcal{P}$) using the BnB algorithm presented in Section 4.2
- `BnB+s` : `BnB` with additional zero-one screening tests

Experiments are run on a macOS equipped with an i7 CPU, with 4 cores clocked at 2.2 GHz and with 8Go of RAM. Results are averaged over 30 instances. We plot both the number of nodes explored and the solution time for the three methods compared, for the two setups presented and for three different values of k used to generate $\mathbf{x}_{real}$.

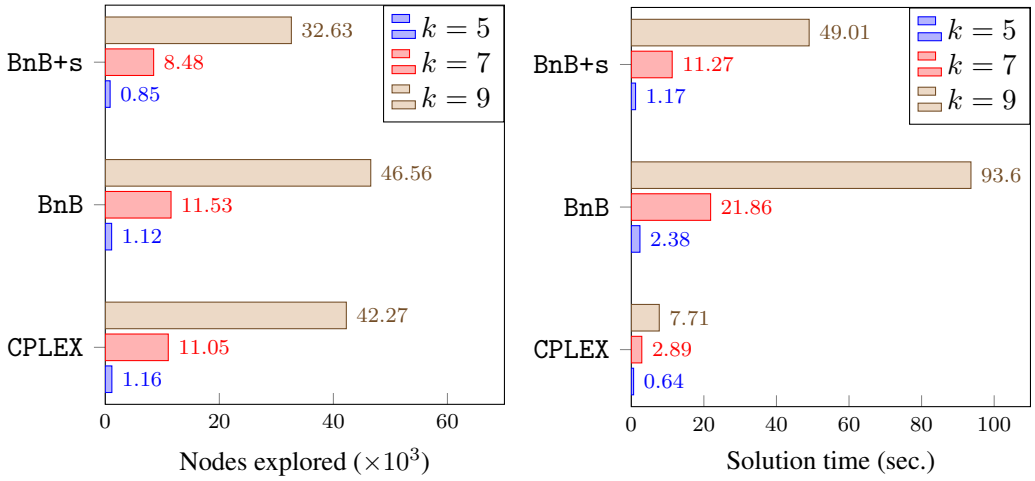

Figure 11: Performance profiles. Sparse deconvolution setup.

First on sparse deconvolution setups, we can see the reduction in the number of nodes explored when adding zero-one screening tests, whatever the value of k used. As a result, we also observe a reduction in the solution time of `BnB+s`, which outperforms `BnB` on all

[11] This corresponds to $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ with $\sigma^2 = \frac{\|\mathbf{A}\mathbf{x}_{real}\|_2^2}{10m}$.

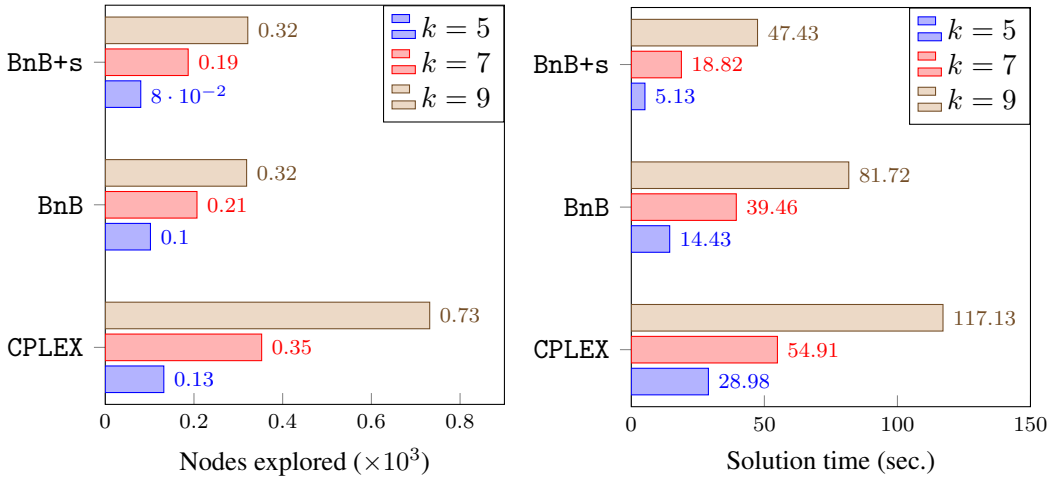

Figure 12: Performance profiles. Subset selection setup.

trials. We must mention that both `BnB` and `BnB+s` are implemented in `Julia` while `CPLEX` is implemented in `C++`, which usually runs faster. Hence, we still have a large room for code speed-up for methods `BnB` and `BnB+s`. Furthermore, `CPLEX` exploits parallelization capabilities, contrary to our `Julia` code. In (Ben Mhenni et al., 2020), author show an implementation of `BnB` that outperforms `CPLEX`, so our `BnB+s` method should also be more efficient.

On subset selection setups, the overall number of nodes processed with or without Zero-one screening tests is not improved. However, we can still notice a solution time improvement. Indeed, zero-one screening tests allow fixing binary variables to 0, removing them from the problem and making the resolution process cheaper in computation. In practice, we observe that many zero-one screening tests are passed at leaf nodes. Even if they will be pruned anyhow with or without screening tests (hence the same number of processed nodes), their relaxation is solved faster, accelerating the BnB process. In this setup, we can notice that even with an implementation in `Julia`, our code runs faster than `CPLEX`, which is written in `C++`.

5 Conclusion

In this last section, conclusions are given about Section 3 and Section 4. An overall conclusion for this report is also given, with personal remarks and prospects.

5.1 Screen & Relax

A first mission of this internship was to finish the work began last year about screening methods. We responded to this assignment by developing a new method to jointly identify zero and non-zero coefficients during the resolution of the Elastic-Net by an arbitrary gradient method. We have shown how to take advantage of this identification in a Screen & Relax strategy to reduce the dimensionality of the problem, thus reducing the complexity of the solution. Numerical simulations have shown the interest of this method. This work will lead to the submission of a paper to the ICASSP 2022 conference which will present our results. Further work will extend our method with new results in a journal article. Finally, we also believe that it is possible to extend the Screen & Relax methodology to more general sparse linear problems, such as the LASSO.

5.2 ℓ_0-screening

The second task of this internship was to extend the notion of screening tests, originally developed for convex problems, to sparse linear problems expressed as MIPs. In Section 4, we showed how it is possible to fix some binary variables during a Branch-and-Bound algorithm, which allows to reduce the overall number of nodes processed and thereby reduce the solution time. Once our code will be more optimized and that more satisfactory results will have been obtained, we will also submit paper at ICASSP 2022. Our new zero-one screening approach will be extended to greedy methods in a future work. Finally, we may combine our zero-one screening strategy with standard screening tests in the relaxation resolution to devise an even better BnB. This last idea should lead to a journal paper submission.

5.3 Prospects for the future PhD

Through this report, we have focused on finite dimensional sparse linear problems and have treated both a convex and a non-convex case. The PhD which will start next October will focus on the extension of our results in the case of sparse linear models with continuous dictionaries, *ie.* in infinite dimension. To do so, a large bibliographic work will have to be carried out because infinite dimensional problems resort to complex mathematical objects which we are not necessarily specialists in. We will first have to define the problem that is addressed and then propose efficient methods to solve it. We also believe that it will be possible to use our methods on the practical cases, such as the automatic detection of liver diseases. This thesis will therefore be a logical extension of the topics we have already started to deal with, but going a step further in terms of technic and application.

5.4 Personal reflections

By finishing this last year at INSA, I really feel that I have acquired the necessary engineering knowledge in Mathematics required to work in a company. Even tough all topics were obviously not covered, I have learned how to deal with a subject that I do not

necessarily master on my own. On top of that, the specialized research course allowed me to put a first foot in this world, by doing bibliographic works and research projects. Internships I was able to do allowed me to really look into research topics and make my first contributions. I think I made the right choice in choosing this studies and always gave my best to get there. Even if INSA is not supposed to train researchers, I think I was able to acquire all the necessary background to continue in this way.

In addition to the courses, INSA has allowed me to learn many non-scientific skills through the associative world. Since my first year in this school, I have been deeply involved in many associations. I was able to learn how to manage projects, resources and people. The highlight was the participation in the AEIR, which deals with the global management of associations at INSA. Even though it was a complicated and exhausting year, I would do it again if I were asked to. I was able to learn many skills that are useful to me every day, and I met many interesting people. Even if such a deep involvement is not always well seen by a part of the pedagogical team, I am fully convinced that it is this particularity that makes the strength of INSA engineers.

During this last internship, I really felt like I was collaborating as a (junior) researcher and not just as a trainee, which was really pleasant and rewarding. I hope that this was possible thanks to the skills I acquired during my studies and my different experiences. I could really experience different phases of the researcher's life. For example, I learned that it can sometimes be a long way to get satisfactory results. On the other hand, the feeling can only be better when you can finally value your work. In addition to the technical side, I learned a lot about the publication strategy, about the good working methods to adopt or about the vulgarization and sharing of my work with people outside the project.

As a final conclusion, I would like to say that this choice to study science and to continue in research contributes to making me see the world through a particular prism, which allows me to understand it better. In these particular times of increasing climate change and growing social inequalities, I think it will be a very useful tool to evolve in tomorrow's society.

Notations

- $\mathbf{y}$: Observation of a sparse linear problem
- $\mathbf{A}$: Dictionary of a sparse linear problem
- $\mathbf{a}_i$: i^{th} column of $\mathbf{A}$
- $\mathbf{x}$: Primal variable
- $\mathbf{u}$: Dual variable
- $\mathbf{z}$: Binary variable encoding nullity of entires in $\mathbf{x}$
- n : Dimension of $\mathbf{x}$, number of atoms in $\mathbf{A}$
- m : Dimension of $\mathbf{u}$, dimension of atoms
- λ, α, β : Tuning parameters
- M : Big-M value in box-constraints
- $\boldsymbol{\gamma}_i^0(\cdot)$, $\boldsymbol{\gamma}_i^0(\cdot)$: Pivot value
- $\mathrm{F}(\cdot)$: Data-fidelity function
- $\Omega(\cdot)$: Sparsity including function
- $\mathcal{R}$: Safe region
- $\mathcal{I}_+$, $\mathcal{I}_0$, $\mathcal{S}_0$, $\mathcal{S}_1$, $\bar{\mathcal{S}}$: Sets of entries of $\mathbf{x}$
- $\bar{\mathcal{I}}$: Complementary of $\mathcal{I}$
- $\mathbf{A}_{\mathcal{I}}$: $\mathbf{A}$ whose columns are restricted to $\mathcal{I}$
- $\mathbf{x}_{\mathcal{I}}$: $\mathbf{x}$ whose entries are restricted to $\mathcal{I}$
- $\mathcal{S}$: Node in the Branch-and-Bound
- MIP : Mixed-Integer Program
- PG : Projected Gradient
- BnB : Branch-and-Bound

References

Atamturk, Alper (2020). "Safe screening rules for L0-regression from perspective relaxations". In: *International conference on machine learning*.

Ben Mhenni, Ramzi et al. (2020). "Sparse Branch and Bound for Exact Optimization of L0-Norm Penalized Least Squares". In: *ICASSP 2020, Speech and Signal Processing*.

Bertero, Mario (1989). "Linear inverse and III-posed problems". In: *Advances in electronics and electron physics*.

Bertsimas, Dimitris, Angela King, and Rahul Mazumder (2016). "Best subset selection via a modern optimization lens". In: *The annals of statistics*.

Bonnefoy, Antoine et al. (2014). "A dynamic screening principle for the lasso". In: *2014 22nd European Signal Processing Conference (EUSIPCO)*. IEEE.

Bot, Radu Ioan (2009). *Conjugate duality in convex optimization*. Springer Science & Business Media.

Böttcher, Albrecht and Sergei M Grudsky (2005). *Spectral properties of banded Toeplitz matrices*. SIAM.

Bourguignon, Sébastien et al. (2015). "Exact sparse approximation problems via mixed-integer programming: Formulations and computational performance". In: *IEEE Transactions on Signal Processing*.

Boyd, Stephen, Neal Parikh, and Eric Chu (2011). *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc.

Brigham, E Oran and RE Morrow (1967). "The fast Fourier transform". In: *IEEE spectrum*.

Calamai, Paul H and Jorge J Moré (1987). "Projected gradient methods for linearly constrained problems". In: *Mathematical programming*.

Dantzig, George B (1990). "Origins of the simplex method". In: *A history of scientific computing*.

Davies, Jessica and Fahiem Bacchus (2013). "Exploiting the power of MIP solvers in MaxSat". In: *International conference on theory and applications of satisfiability testing*. Springer.

Dolan, Elizabeth D and Jorge J Moré (2002). "Benchmarking optimization software with performance profiles". In: *Mathematical programming*.

Foucart, Simon and Holger Rauhut (2013). "An invitation to compressive sensing". In: *A mathematical introduction to compressive sensing*. Springer.

Ghaoui, Laurent El, Vivian Viallon, and Tarek Rabbani (2010). "Safe feature elimination for the lasso and sparse supervised learning problems". In:

Hager, William W (1989). "Updating the inverse of a matrix". In: *SIAM review*.

Helmberg, Christoph et al. (1996). "An interior-point method for semidefinite programming". In: *SIAM Journal on optimization*.

Herzet, Cédric and Angélique Drémeau (2018). "Joint screening tests for Lasso". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Hunger, Raphael (2005). *Floating point operations in matrix-vector calculus*. Munich University of Technology, Inst. for Circuit Theory and Signal.

Janocha, Katarzyna and Wojciech Marian Czarnecki (2017). "On loss functions for deep neural networks in classification". In: *arXiv preprint arXiv:1702.05659*.

Lawler, Eugene L and David E Wood (1966). "Branch-and-bound methods: A survey". In: *Operations research* 14.4, pp. 699–719.

Lawson, Charles L and Richard J Hanson (1995). *Solving least squares problems*. SIAM.

Lee, Honglak et al. (2007). “Efficient sparse coding algorithms”. In: *Advances in neural information processing systems*.

Levin, Anat et al. (2007). “Image and depth from a conventional camera with a coded aperture”. In: *ACM transactions on graphics (TOG)*.

Lu, Canyi, Huan Li, and Zhouchen Lin (2018). “Optimized projections for compressed sensing via direct mutual coherence minimization”. In: *Signal Processing* 151, pp. 45–55.

Luc, Dinh The (2008). “Pareto optimality”. In: *Pareto optimality, game theory and equilibria*.

Ndiaye, Eugene et al. (2016). “Gap safe screening rules for sparse-group lasso”. In: *Advances in neural information processing systems*.

Nesterov, Yurii et al. (2018). *Lectures on convex optimization*. Vol. 137. Springer.

Parikh, Neal and Stephen Boyd (2014). “Proximal algorithms”. In: *Foundations and Trends in optimization*.

Rakotomamonjy, Alain, Gilles Gasso, and Joseph Salmon (2019). “Screening rules for lasso with non-convex sparse regularizers”. In: *International Conference on Machine Learning*. PMLR.

Ribes, Alejandro and Francis Schmitt (2008). “Linear inverse problems in imaging”. In: *IEEE Signal Processing Magazine*.

Stark, Philip B and Robert L Parker (1995). “Bounded-variable least-squares: an algorithm and applications”. In: *Computational Statistics*.

Strang, Gilbert (1999). “The discrete cosine transform”. In: *SIAM review*.

Tibshirani, Robert (2011). “Regression shrinkage and selection via the lasso: a retrospective”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*.

Tropp, Joel A and Stephen J Wright (2010). “Computational methods for sparse solution of linear inverse problems”. In: *Proceedings of the IEEE* 98.6, pp. 948–958.

Wang, Jie et al. (2013). “Lasso Screening Rules via Dual Polytope Projection.” In: *NIPS*. Citeseer.

Woeginger, Gerhard J (2003). “Exact algorithms for NP-hard problems: A survey”. In: *Combinatorial optimization—eureka, you shrink!* Springer.

Wolsey, Laurence (1989). “Strong formulations for mixed integer programming: a survey”. In: *Mathematical Programming*.

Wright, Stephen, Jorge Nocedal, et al. (1999). “Numerical optimization”. In: *Springer Science*.

Xiang, Zhen James and Peter J Ramadge (2012). “Fast lasso screening tests based on correlations”. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Xiang, Zhen James, Yun Wang, and Peter J Ramadge (2016). “Screening tests for lasso problems”. In: *IEEE transactions on pattern analysis and machine intelligence*.

You, Chong et al. (2016). “Oracle based active set algorithm for scalable elastic net subspace clustering”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Zhang, Fuzhen (2006). *The Schur complement and its applications*. Springer Science & Business Media.

Zhang, Tong (2008). “Adaptive forward-backward greedy algorithm for sparse learning with linear models”. In: *Advances in neural information processing systems*.

Zou, Hui and Trevor Hastie (2005). “Regularization and variable selection via the elastic net”. In: *Journal of the royal statistical society: series B (statistical methodology)*.