Summary of: Branch-and-Bound Algorithms for L0-Regularized Problems

Théo Guyard
Inria, Centre de l'Université de Rennes
Insa Rennes, IRMAR – CNRS UMR 6625
theo.guyard@insa-rennes.fr

The manuscript focuses on ℓ_0-regularized optimization problems given by

$$p^\star = \min_{\mathbf{x}\in\mathbf{R}^n} f(\mathbf{x}) + \underbrace{\lambda\|\mathbf{x}\|_0 + h(\mathbf{x})}_{g(\mathbf{x})} \tag{P}$$

where $f\colon \mathbf{R}^n \to \mathbf{R}\cup\{+\infty\}$ is a loss function and $g\colon \mathbf{R}^n \to \mathbf{R}\cup\{+\infty\}$ is a regularization involving an ℓ_0-norm with weight $\lambda > 0$ to promote sparse solutions by counting the number of non-zeros in its argument, as well as another penalty term $h\colon \mathbf{R}^n \to \mathbf{R}\cup\{+\infty\}$ used to promote other properties than sparsity. Although various heuristics have been proposed to tackle this problem [9], exact methods are of paramount interest for many applications in signal processing, machine learning or statistics, among other fields [2, 8]. In this vein, state-of-the-art solvers are based on Branch-and-Bound (BnB) algorithms [1, 5]. So far, they have only been designed for some particular instances of the problem and can still suffer from poor numerical performance on real-world data. In this view, the manuscript aims to:

1. Provide a unified BnB algorithm to tackle problem (P) in a generic fashion,
2. Propose novel acceleration strategies to improve its numerical efficiency,
3. Design a convenient toolbox for practitioners.

From Sections 1 to 3, we describe the main ingredients required to implement a BnB algorithm tailored to problem (P), outline the contributions presented in the manuscript, and give an overview of the implications stemming from this work for signal processing applications.

1 Branch-and-Bound Ingredients

BnB is a generic algorithm to address optimization problems. It explores *regions* in the feasible space and *prunes* those that cannot contain solutions. When specialized to problem (P), the construction of regions is driven by the sparsity of the optimization variable, that is, by fixing some entries to zero or non-zero. Stated otherwise, regions are defined as

$$\nu = \left\{ \mathbf{x}\in\mathbf{R}^n \;\middle|\; \begin{array}{ll} x_i = 0 & \forall i\in\nu_0 \\ x_i \neq 0 & \forall i\in\nu_1 \\ x_i \in \mathbf{R} & \forall i\in\nu_\bullet \end{array} \right\} \tag{1}$$

from a partition $(\nu_0, \nu_1, \nu_\bullet)$ of the index set $\{1,\dots,n\}$. Starting with a region resulting from the partition $(\emptyset, \emptyset, \{1,\dots,n\})$ which corresponds to the whole feasible space $\mathbf{R}^n$, the BnB algorithm iteratively selects a new region to process, performs a *pruning test*, and if no pruning decision can be taken, splits it into two new subregions by fixing a new index to zero or non-zero. Once the entire feasible space has been explored, solutions to problem (P) can be extracted from the regions that have not been pruned during the search. The BnB complexity depends on the total number of regions processed and the cost to evaluate a pruning test for a given region.

Pruning test. The central mechanism of the BnB algorithm is a pruning test that aims to detect regions that cannot contain any solution to problem (P). Denoting

$$p^\nu = \min_{\mathbf{x}\in\nu} f(\mathbf{x}) + g(\mathbf{x}) \tag{P^ν}$$

the smallest objective value achievable in problem (P) over a region ν, then no solutions are contained in this part of the feasible space if $p^\nu > p^\star$ and the region can thus be pruned. However, this inequality cannot be checked directly since $p^\star$ is unknown and recovering p^ν is still an NP-hard task. In practice, one rather implements a pruning test expressed as

$$p^\nu_{\text{lb}} > p^\star_{\text{ub}} \tag{2}$$

using some lower bound $p^\nu_{\text{lb}} \leq p^\nu$ and upper bound $p^\star_{\text{ub}} \geq p^\star$ as surrogates. Implementing the BnB algorithm then amounts to devising strategies to compute these bounds, typically sought *tight* and *tractable* to ensure the numerical efficiency of the method.

Bounds computation. Various heuristics-based strategies have been proposed to obtain an upper bound $p^\star_{\text{ub}} \geq p^\star$ of good quality at a reasonable cost. The crux of the BnB algorithm mainly lies in the computation of a suitable lower bound $p^\nu_{\text{lb}} \leq p^\nu$ for each region explored. This can be achieved by solving a *relaxation* of problem (P^ν), expressed as

$$r^\nu = \min_{\mathbf{x}\in\mathbf{R}^n} f(\mathbf{x}) + g^\nu(\mathbf{x}) \tag{R^ν}$$

for some $g^\nu \colon \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$. One requires $g^\nu(\mathbf{x}) \leq g(\mathbf{x})$ for all $\mathbf{x} \in \nu$ so that setting $p^\nu_{\text{lb}} = r^\nu$ in the pruning test (2) leads to a valid BnB algorithm. Moreover, one usually desires g^ν convex so that the relaxation (R^ν) can be addressed efficiently via cutting-edge convex optimization methods.

Limitations. Existing BnB algorithms suffer from two main limitations. First, existing strategies to construct the function g^ν in the relaxation (R^ν) are tied to the expression of the function h, thereby restricting the BnB algorithm scope of application. Second, these BnB algorithms can still suffer from poor performance on some real-world instances. Indeed, one relaxation needs to be solved per region explored to evaluate the associated pruning test, which can represent a significant computational burden.

2 Contributions

The manuscript presents contributions disseminated from Chapters 4 to 7 to address the BnB algorithm limitations aforementioned. In the following, we outline the main results of each chapter.

Unified Branch-and-Bound Framework. Chapter 4 proposes a unified BnB framework to address (P) through a generic strategy for constructing the relaxation (R^ν). Specifically, we consider any function $h(\mathbf{x}) = \sum_{i=1}^n h_i(x_i)$ separable into splitting terms $\{h_i\}_{i=1}^n$ that are proper, closed, convex, even, supercoercive, and verify $h_i \geq h_i(0) = 0$ with $0 \in \operatorname{int}\operatorname{dom}(h_i)$.[1] In this context, we show that the *convex envelope* of the function g over a region ν defined by (1) can be expressed as

$$g^\nu(\mathbf{x}) = \sum_{i=1}^n g_i^\nu(x_i) \quad \text{with} \quad g_i^\nu(x) = \begin{cases} \eta(x=0) & \text{if } i \in \nu_0 \\ h_i(x) + \lambda & \text{if } i \in \nu_1 \\ \tau_i|x| & \text{if } i \in \nu_\bullet \text{ with } |x| \leq \mu_i \\ h_i(x) + \lambda & \text{if } i \in \nu_\bullet \text{ with } |x| > \mu_i \end{cases} \tag{3}$$

where $\tau_i = \sup\{z \in \mathbf{R} \mid h_i^*(z) \leq \lambda\}$ and $\mu_i = \sup\{z \in \mathbf{R} \mid z \in \partial h_i^*(\tau)\}$. Here, the notation $\eta(\cdot)$ stands for the convex indicator function and h_i^* for the convex conjugate of h_i. Our construction strategy is *optimal* in the sense that it gives the best convex approximation of the function g over the region ν. Notably, it encompasses prior contributions as particular cases, and, in contrast to them, does not depend on a specific expression of the function h. We additionally characterize various operators associated with the function g^ν enabling the use of modern convex optimization methods to address the resulting relaxation (R^ν). From this contribution stems a unified BnB framework that can address problem (P) generically.

[1] Our results can be extended to cases where h_i is only assumed coercive with $h_i(0) \in \mathbf{R}$.

Simultaneous Pruning. Chapter 5 introduces a novel acceleration strategy for the BnB algorithm. We first point out that a valid choice for the lower-bound involved in the pruning test (2) is to set $p_{\text{lb}}^{\nu} = D^{\nu}(\mathbf{u})$ for any $\mathbf{u} \in \mathbf{R}^n$, where D^{ν} denotes the objective of the dual problem associated with the relaxation (R^{ν}). More importantly, we show that for nested regions $\nu' \subseteq \nu$ defined as in (1), such *dual* lower-bounds verify

$$D^{\nu'}(\mathbf{u}) = D^{\nu}(\mathbf{u}) + \Delta_{\text{sp}}^{\nu'}(\mathbf{u}) \tag{4}$$

where $\Delta_{\text{sp}}^{\nu'} \colon \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ can be evaluated in $\mathcal{O}(1)$. From a *single* evaluation of $D^{\nu}(\mathbf{u})$ when processing the region ν, we can then *simultaneously* check several nested regions $\nu' \subseteq \nu$ against a pruning test through their dual bound $D^{\nu'}(\mathbf{u})$ obtained *at virtually no cost* from (4). In contrast, the standard pruning strategy requires solving one relaxation per region processed. This shift of paradigm in the pruning process allows accelerating the BnB algorithm up to several orders of magnitude in terms of solving time.

Bound Tightening. Chapter 6 focuses on the specific case where $h(\mathbf{x}) = 0$ when $\boldsymbol{\alpha} \leq \mathbf{x} \leq \boldsymbol{\beta}$ for some $(\boldsymbol{\alpha}, \boldsymbol{\beta}) \in \mathbf{R}_{-}^n \times \mathbf{R}_{+}^n$, and $h(\mathbf{x}) = +\infty$ otherwise. These instances are of interest for various applications but lead to two conflicting imperatives: the bounds $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ are usually required to have large entries in absolute value, but increasing their magnitude degrades the performance of the BnB algorithm. To face this issue, we propose a *bound tightening* strategy to refine the initial bounds $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ during the BnB algorithm. Similarly to simultaneous pruning, we show that for a region of the form $\nu' = \nu \setminus \{\mathbf{x} \in \mathbf{R}^n \mid \boldsymbol{\alpha}' \leq \mathbf{x} \leq \boldsymbol{\beta}'\}$ with $\boldsymbol{\alpha} \leq \boldsymbol{\alpha}' \leq \boldsymbol{\beta}' \leq \boldsymbol{\beta}$, we can perform the pruning test using a dual bound $p_{\text{lb}}^{\nu} = D^{\nu}(\mathbf{u})$ verifying

$$D^{\nu'}(\mathbf{u}) = D^{\nu}(\mathbf{u}) + \Delta_{\text{bt}}^{\nu'}(\mathbf{u}) \tag{5}$$

where $\Delta_{\text{bt}}^{\nu'} \colon \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ can be evaluated in $\mathcal{O}(1)$. When processing region ν, we can then check any nested region ν' against a pruning test using its dual bound obtained at virtually no cost via (5). If the latter is verified, the region ν' can be pruned, which amounts to tightening the bounds associated with region ν as $(\boldsymbol{\alpha}, \boldsymbol{\beta}) \leftarrow (\boldsymbol{\alpha}', \boldsymbol{\beta}')$. This process is implemented dynamically in the BnB algorithm to refine the initial bounds $(\boldsymbol{\alpha}, \boldsymbol{\beta})$, which progressively enhance its performance during the resolution process.

Relaxation Processing. Chapter 7 proposes an acceleration strategy for numerical procedures tailored to the relaxation (R^{ν}), but more generally tailored to any *convex* optimization problem of the form

$$r^{\star} = \min_{\mathbf{x} \in \mathbf{R}^n} f(\mathbf{x}) + \tilde{g}(\mathbf{x}) \tag{R}$$

where $\tilde{g}$ promotes sparse solutions, which are ubiquitous in the literature. While prior contributions have proposed *screening tests* [4] to detect the position of zero entries in the problem solutions, we propose *smoothing tests* to detect the position of non-zero entries. We combine these two complementary strategies to dynamically reduce the dimension and smooth the objective function of problem (R) during its resolution process. When interleaved within standard optimization procedures such as proximal gradient, coordinate descent, or ADMM, this results in performance improvements up to several orders of magnitude in terms of solving time.

Practical Toolbox. The contributions presented in the manuscript are implemented in `El0ps`,[2] an open-source Python package providing convenient interfaces for problem (P). It includes several built-in instances, but also offers the flexibility to create new ones. Users are only required to specify some operators associated with the function f and h. Once done, the BnB solver provided by `El0ps` can automatically address the resulting instance of problem (P). In contrast, other existing BnB solvers are restricted to one particular instance. Numerical experiments outline that `El0ps` achieves state-of-the-art performance, with acceleration factors up to several orders of magnitude in terms of solving times compared to its best competitors, even on instances that were previously thoroughly addressed in the literature.

[2] https://github.com/TheoGuyard/El0ps

3 Numerical Illustration

Figure 1 gives a glimpse of the implications of the manuscript's contributions for signal processing applications. Here, we consider Bernoulli-Mixture inverse problems [7] aiming to reconstruct a sparse signal $\mathbf{x}^\star \in \mathbf{R}^{1000}$ known to verify $x_i^\star \sim \mathcal{D}$ with probability $\rho = 0.01$ for some distribution $\mathcal{D}$, and $x_i^\star = 0$ otherwise. To this end, we have access to a linear measurement $\mathbf{y} = \mathbf{A}\mathbf{x}^\star + \boldsymbol{\epsilon}$ where $\mathbf{A} \in \mathbf{R}^{500\times 1000}$ is a given matrix and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ is a noise with 10dB signal-to-noise ratio. In this setup, the Maximum A Posteriori estimation of the sparse signal corresponds to any solution of problem (P) with a loss $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$, a parameter $\lambda = \sigma^2 \log((1-\rho)/\rho)$, and a penalty h whose expression depends on the underlying distribution $\mathcal{D}$ of the non-zero entries. To conduct our experiment, we select different choices of distributions (Normal, Gauss-Laplace, Half-Normal, Exponential), independently generate 100 instances of the corresponding problem (P), and represent the percentage solved within a given time budget using different numerical procedures. Compared to generic optimization solvers such as `Cplex` and `Mosek` [6], but also to solution methods specifically designed for problem (P) such as `L0bnb` [5] and `OA` [3], our solver `El0ps` obtains the best performances and can also handle a broader variety of instances. For instance, it is the only method able to process cases corresponding to the Exponential distribution.

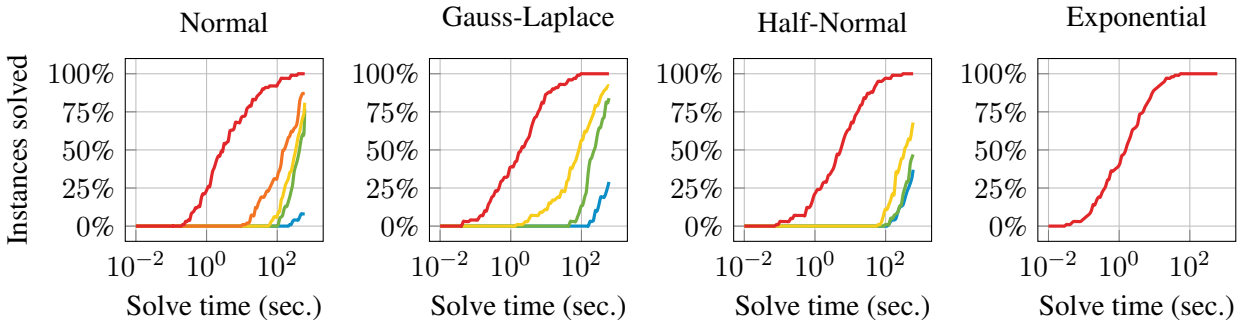

Figure 1: Instances solved for a given time budget using: — `Cplex`, — `Mosek`, — `OA`, — `L0bnb`, — `El0ps`. The absence of a curve indicates that the solver was unable to process the corresponding problem instance.

References

[1] R. Ben Mhenni, S. Bourguignon, and J. Ninin. Global optimization for sparse solution of least squares problems. *Optimization Methods and Software*, 37(5):1740–1769, 2022.

[2] D. Bertsimas, A. King, and R. Mazumber. Best subset selection via a modern optimization lens. *The Annals of Statistics*, 44(2):813–852, 2016.

[3] D. Bertsimas, R. Cory-Wright, and J. Pauphilet. A unified approach to mixed-integer optimization problems with logical constraints. *SIAM Journal on Optimization*, 31(3):2340–2367, 2021.

[4] L. El Ghaoui. Safe feature elimination for the Lasso and sparse supervised learning problems. *Pacific Journal of Optimization*, 8(4):667, 2012.

[5] H. Hazimeh, R. Mazumder, and A. Saab. Sparse regression at scale: Branch-and-bound rooted in first-order optimization. *Mathematical Programming*, 196(1):347–388, 2022.

[6] J. Kronqvist, D. E. Bernal, A. Lundell, and I. E. Grossmann. A review and comparison of solvers for convex minlp. *Optimization and Engineering*, 20:397–455, 2019.

[7] C. Soussen, J. Idier, D. Brie, and J. Duan. From Bernoulli–Gaussian deconvolution to sparse signal restoration. *IEEE Transactions on Signal Processing*, 59(10):4572–4584, 2011.

[8] A. M. Tillmann, D. Bienstock, A. Lodi, and A. Schwartz. Cardinality minimization, constraints, and regularization: a survey. *SIAM Review*, 66(3):403–477, 2024.

[9] J. A. Tropp and S. J. Wright. Computational methods for sparse solution of linear inverse problems. *Proceedings of the IEEE*, 98(6):948–958, 2010.