Branch-and-bound algorithm for exact ℓ_0-norm sparse spectral unmixing

Mehdi Latif[⋆], Gwen Samain[⋄], Nils Foix-Colonier[⋆], Sébastien Bourguignon[⋆]
Nantes Université, École Centrale Nantes, CNRS
Laboratoire des Sciences du Numérique de Nantes (LS2N), UMR 6004
F-44000, Nantes, France
[⋆]Surname.Name@ls2n.fr, [⋄]samain.gwen@laposte.net

***Abstract*—We propose an algorithm that exactly solves the cardinality-constrained sparse spectral unmixing problem. Based on recent works on ℓ_0-norm exact optimization, a branch-and-bound architecture is specifically developed for sparse unmixing, under nonnegativity and sum-to-one constraints. The procedure boils down to solving a finite number of sum-to-one constrained nonnegative least-squares problems for upper- and lower-bounding the global optimal value, which are solved efficiently. Numerical simulations show that our method outperforms competing ones in terms of support identification and estimation, and that it remains computationally tractable as long as the problem size is limited or the signal-to-noise ratio is high enough. A free C++ implementation is made available.**

***Index Terms*—spectral unmixing, sparsity, ℓ_0 norm, global optimization, branch-and-bound.**

I. Introduction

Hyperspectral imaging extracts information about the electromagnetic spectra reflected by an illuminated scene, collected over a set of spatial locations. It has found many application fields, including remote sensing for Earth and planetary science, agriculture, food industry, chemistry. Among the numerous hyperspectral data processing techniques, spectral unmixing (SU) is certainly one of the most studied topics [1]. Originally formulated as a blind source separation problem, it aims to factorize an hyperspectral data cube as a collection of pure spectral signatures (*endmembers*) and associated weights at each pixel location. In particular, the linear mixing model [2] assumes that a measured reflectance spectrum decomposes as the linear combination of endmembers weighted by their *fractional abundances*. Since abundances are proportions, abundance non-negativity (ANC) and sum-to-one (ASC) constraints are usually added to the model. Abundance estimation is then known as Fully Constrained Least-Squares (FCLS) [3], which minimizes the ℓ_2-norm misfit under ANC and ASC.

In many cases, the abundance set is expected to be *sparse*, since only a few endmembers are expected to be active in each decomposition. Although FCLS solutions are easily computed and do exhibit some sparsity due to nonnegativity, estimated abundances often spread over a substantial number of components with low values, which lacks interpretability and, most of all, can fail in detecting the true endmembers [4]. Enforcing more sparsity may merely ensure a better mixture estimation. Sparse spectral unmixing has been addressed in the literature, either by incorporating sparsity-enhancing penalties or constraints in the least-squares fit [5], [6], [7], or by iterative forward [5] or backward [8] methods. In this paper, we address the ℓ_0-norm formulation for sparse unmixing (hereafter, ℓ_0-SU), where sparsity is explicitly imposed by constraining the number of non-zero abundances. We consider pixel-wise abundance estimation, where the reflectance spectrum $\boldsymbol{y} \in \mathbb{R}^{N_\lambda}$ is decomposed into K endmembers taken in dictionary $\mathbf{S} = [\boldsymbol{s}_1, \ldots, \boldsymbol{s}_P] \in \mathbb{R}^{N_\lambda \times P}$, with related abundances $\boldsymbol{a} \in \mathbb{R}^P$, by solving the cardinality-constrained optimization problem:

$$\mathcal{P} : \min_{\boldsymbol{a} \in [0,1]^P} \tfrac{1}{2} \left\| \boldsymbol{y} - \mathbf{S}\boldsymbol{a} \right\|_2^2 \text{ s.t. } \mathbf{1}^\intercal \boldsymbol{a} = 1, \|\boldsymbol{a}\|_0 \leq K, \qquad (1)$$

where $\mathbf{1}$ denotes the vector composed of ones of appropriate dimension, and the ℓ_0 "norm" $\|\boldsymbol{a}\|_0$ counts the number of non-zero elements in $\boldsymbol{a}$.

Although *exact* ℓ_0-norm optimization (the solution is *proved* to be optimal) is $\mathcal{NP}$-hard [9], it has been successfully addressed in the general setting (*i.e.,* without ANC and ASC) for relatively small problems, first with mixed-integer programming (MIP) reformulations and generic MIP solvers [10], then with dedicated, faster, branch-and-bound algorithms [11], [12], [13]. Exact ℓ_0-SU as in (1) was also formulated as a MIP in [4], where solutions were shown to achieve better estimates than competing methods. In particular, in many unmixing problems, the number of active elements searched in the decomposition rarely exceeds a few units, therefore the complexity, although combinatorial, remains limited.

This paper presents a branch-and-bound (*BB*) algorithm for ℓ_0-SU. In Section II, a specific architecture is built considering ANC and ASC. Section III explores its practical implementation. Performance in terms of solution quality and computational efficiency is assessed in Section IV. The discussion in Section V concludes the paper.

II. Branch-and-bound design

The *BB* method [14] is an algorithm paradigm widely used in operations research to tackle hard optimization problems for which exhaustive search is unconceivable. It implicitly enumerates the entire search space by solving a sequence of simpler problems, which provide bounds on the optimal value;

This work has been partially funded by the French national research agency (ANR), project ANR-16-CE33-0005-01 MIMOSA (Mixed integer programming methods for sparse approximation), https://mimosa.ls2n.fr/.

these bounds are then used to avoid exploration of regions that are proved to yield suboptimal solutions. In parallel, a tree structure is constructed in which a node represents a part of the search space. Children of this node are produced by partitioning it into subspaces and related *subproblems*; this partition is obtained by setting the value of a decision variable: *the branching procedure*. Resulting children are stored in a list, waiting to be solved. The aim is to prune as many nodes as possible by *the bounding procedure*. The optimal solution is found once all nodes have been implicitly explored, which occurs in a finite number of steps, all the smaller as such procedures are performed efficiently. This Section details the main constitutive elements of our algorithm to solve ℓ_0-SU.

A. Branching procedure

Since the ℓ_0-norm constraint (the hard part) mostly indicates binary decisions (*which variables should be non-zero?*), a decision tree is built, where the branching procedure generates two subproblems of the form: "include $\boldsymbol{s}_p$" (left branch) vs. "discard $\boldsymbol{s}_p$" (that is, $a_p = 0$, right branch). Note that the first choice does not impose $a_p \neq 0$ (which is hard to consider) and the two branches are not mutually exclusive, but this does not question the validity of the procedure. Moreover, it will be shown in Section II-C that the branching variable choice ensures $a_p \neq 0$ in the related subproblem.

At a given node, the solution space has undergone a certain number of left and right branchings. Let $\mathcal{Z}$ index variables that were set to 0, $\overline{\mathcal{Z}}$ denote the complementary index set among $[1, \ldots, P]$, and $\mathcal{C} \subset \overline{\mathcal{Z}}$ index the variables explicitly included in the solution. Let also $\overline{\mathcal{C}} = \overline{\mathcal{Z}} \setminus \mathcal{C}$ index the remaining *undecided* variables. The resulting subproblem then just reads:

$$\min_{\boldsymbol{a}_{\overline{\mathcal{Z}}} \in [0,1]^{\#\overline{\mathcal{Z}}}} \tfrac{1}{2} \left\| \boldsymbol{y} - \mathbf{S}_{\overline{\mathcal{Z}}} \boldsymbol{a}_{\overline{\mathcal{Z}}} \right\|_2^2 \text{ s.t. } \mathbf{1}^\intercal \boldsymbol{a}_{\overline{\mathcal{Z}}} = 1, \|\boldsymbol{a}_{\overline{\mathcal{C}}}\|_0 \leq K - \#\mathcal{C} \tag{2}$$

where $\#$ denotes the cardinality, $\mathbf{S}_{\overline{\mathcal{Z}}}$ is the submatrix formed by columns of $\mathbf{S}$ indexed by $\overline{\mathcal{Z}}$, and $\boldsymbol{a}_{\overline{\mathcal{Z}}}$ is the corresponding subvector of $\boldsymbol{a}$. The value of the ℓ_0-norm term is the number of remaining non-zero variables that can still be included in the solution since, by construction, $\|\boldsymbol{a}_{\mathcal{Z}}\|_0 = 0$ and $\|\boldsymbol{a}_{\mathcal{C}}\|_0 = \#\mathcal{C}$.

After K left branchings, one has $\#\mathcal{C} = K$, that is, $\boldsymbol{a}_{\overline{\mathcal{C}}} = \mathbf{0}$. The node is a *leaf* that does not have to be divided any more: the subproblem solution is K-sparse and is obtained by restricting $\overline{\mathcal{Z}}$ to $\mathcal{C}$ in (2) (see (3) in Section II-B1).

B. Bounding operations

At each node, an upper and a lower bound for the corresponding problem are defined, that should be easily computed.

1) Upper bound and reduced-size FCLS: In *BB* design, an upper bound is obtained by adding constraints to the subproblem, *simplifying* its hard part. Here, we consider the particular solution with all undecided variables $\boldsymbol{a}_{\overline{\mathcal{C}}}$ set to zero:

$$\mathcal{P}_{\text{UB}} : \text{UB} = \min_{\boldsymbol{a}_{\mathcal{C}} \in [0,1]^{\#\mathcal{C}}} \tfrac{1}{2} \left\| \boldsymbol{y} - \mathbf{S}_{\mathcal{C}} \boldsymbol{a}_{\mathcal{C}} \right\|_2^2 \text{ s.t. } \mathbf{1}^\intercal \boldsymbol{a}_{\mathcal{C}} = 1. \tag{3}$$

When exploration reaches the node where $\mathcal{C}$ identifies with the support of the global optimizer, the latter is the solution to (1) (without being proven yet) and UB gives the global minimum.

2) Lower bound and inefficiency of ℓ_1-norm relaxation: A lower bound is generally obtained by *relaxing* the hard part of the subproblem. Finding a relaxation for the problem (2) amounts to relax the ℓ_0-norm constraint. Note that the lower bound definition only makes sense if $K > \#\mathcal{C}$ in (2). When $K = \#\mathcal{C}$, the considered node is a leaf and its subproblem is solved exactly (see Section II-A).

The ℓ_1 norm is often considered for convex relaxation of the ℓ_0 norm—which requires additional assumptions (such as box constraints) in order to cope with the non-homogeneity of the ℓ_0 function. It was used for sparse branch-and-bound design in a generic context, *e.g.* [11], [13], [12]. In our case, however, such a choice is useless due ANC and ASC. More precisely, let $\mathcal{S}_p^L := \{\boldsymbol{a} \in \mathbb{R}^P; \|\boldsymbol{a}\|_p \leq L\} \cap [0,1]^P$ for $p \in \{0;1\}$ and $L \in \mathbb{N}^*$. We have the following proposition.

Proposition 1. The convex hull of the ℓ_0-norm ball on $[0,1]^P$ is the ℓ_1-norm ball on $[0,1]^P$. That is: **conv** $\left(\mathcal{S}_0^L\right) = \mathcal{S}_1^L$.

The proof can be found as supplementary material `there`; a graphical illustration is given in Figure 1.

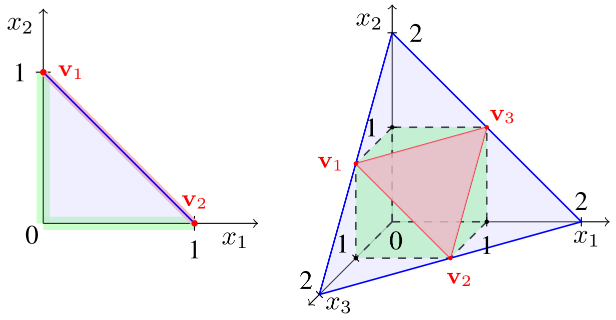


Fig. 1. Illustration of Proposition 1 for $P = 2$ and $L = 1$ (left), and for $P = 3$ and $L = 2$ (right): ℓ_0-norm ball on $[0,1]^P$ (S_0^L, green) and ℓ_1-norm ball on $[0,1]^P$ (S_1^L, blue). In red, the boundary $\{\|\boldsymbol{a}\|_1 = L\} \cap [0,1]^P$ and its vertices $\mathbf{v}_i$.

Proposition 1 shows that convexifying the problem in (2) amounts to substituting the ℓ_0 norm by the ℓ_1 norm, that is, replacing constraint $\|\boldsymbol{a}_{\overline{\mathcal{C}}}\|_0 \leq K - \#\mathcal{C}$ by $\|\boldsymbol{a}_{\overline{\mathcal{C}}}\|_1 \leq K - \#\mathcal{C}$. However, the ASC imposes $\|\boldsymbol{a}_{\overline{\mathcal{Z}}}\|_1 = \|\boldsymbol{a}_{\mathcal{C}}\|_1 + \|\boldsymbol{a}_{\overline{\mathcal{C}}}\|_1 = 1$, hence $\|\boldsymbol{a}_{\overline{\mathcal{C}}}\|_1 \leq 1$, which dominates the former constraint. In other words, there is no better convex relaxation of the problem in (2) than just removing the ℓ_0-norm constraint. Then, the lower bound computed at each node just reads:

$$\mathcal{P}_{\text{LB}} : \text{LB} = \min_{\boldsymbol{a}_{\overline{\mathcal{Z}}} \in [0,1]^{\#\overline{\mathcal{Z}}}} \tfrac{1}{2} \left\| \boldsymbol{y} - \mathbf{S}_{\overline{\mathcal{Z}}} \boldsymbol{a}_{\overline{\mathcal{Z}}} \right\|_2^2 \text{ s.t. } \mathbf{1}^\intercal \boldsymbol{a}_{\overline{\mathcal{Z}}} = 1. \tag{4}$$

Note that some works investigated the use of non-convex approaches to sparse unmixing, such as ℓ_p-norms with $p \in]0,1[$ (*e.g.* [7], [15]). In our case, however, global optimization of the relaxed subproblem is required so that it gives a valid lower bound, for which convexity is a guarantee.

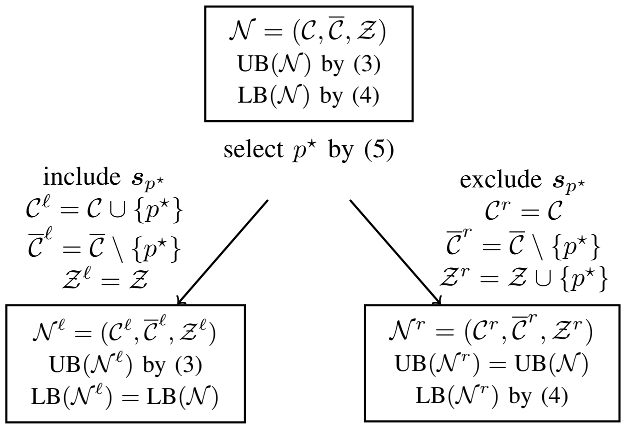
C. Branching variable

The choice for the branching variable usually exploits the solution of the considered relaxed subproblem. We use the strategy proposed in [11], which considers the component with maximum amplitude among undecided variables in the solution to (4):

$$p^\star = \arg\max_{p \in \overline{\mathcal{C}}} a_p^*, \text{ with } \boldsymbol{a}_{\overline{\mathcal{Z}}}^* = \arg\min \mathcal{P}_{\text{LB}}. \tag{5}$$

It assumes that component $\boldsymbol{s}_{p^\star}$ is likely to belong to the solution of the subproblem, so that developing the search tree should quickly identify the optimal support.

Figure 2 finally summarizes the core steps of our algorithm, where a node is described by the index partition $(\mathcal{C}, \overline{\mathcal{C}}, \mathcal{Z})$.



$\mathcal{N} = (\mathcal{C}, \overline{\mathcal{C}}, \mathcal{Z})$
UB($\mathcal{N}$) by (3)
LB($\mathcal{N}$) by (4)
select $p^\star$ by (5)
include $\boldsymbol{s}_{p^\star}$
$\mathcal{C}^\ell = \mathcal{C} \cup \{p^\star\}$
$\overline{\mathcal{C}}^\ell = \overline{\mathcal{C}} \setminus \{p^\star\}$
$\mathcal{Z}^\ell = \mathcal{Z}$
exclude $\boldsymbol{s}_{p^\star}$
$\mathcal{C}^r = \mathcal{C}$
$\overline{\mathcal{C}}^r = \overline{\mathcal{C}} \setminus \{p^\star\}$
$\mathcal{Z}^r = \mathcal{Z} \cup \{p^\star\}$
$\mathcal{N}^\ell = (\mathcal{C}^\ell, \overline{\mathcal{C}}^\ell, \mathcal{Z}^\ell)$
UB($\mathcal{N}^\ell$) by (3)
LB($\mathcal{N}^\ell$) = LB($\mathcal{N}$)
$\mathcal{N}^r = (\mathcal{C}^r, \overline{\mathcal{C}}^r, \mathcal{Z}^r)$
UB($\mathcal{N}^r$) = UB($\mathcal{N}$)
LB($\mathcal{N}^r$) by (4)

Fig. 2. Main steps of our ℓ_0-SU algorithm: node splitting, variable selection, subset updates and bound updates.

III. Practical implementation

A. Tree exploration and optimality proof

The algorithm starts at the *root node*, where all variables are undecided. With notations introduced in Section II-A, $\mathcal{Z} = \emptyset$, $\overline{\mathcal{Z}} = [1, \ldots, P]$, $\mathcal{C} = \emptyset$ and $\overline{\mathcal{C}} = \overline{\mathcal{Z}}$. Problem (2) is solved on all variables, providing a first lower bound. From the variable selection rule in Section II-C, the variable with maximal amplitude in the corresponding minimizer, say $a_{p_1}^*$, is selected to operate a first division (see Section II-A and Figure 2). If $a_{p_1}^* = 0$, then $\boldsymbol{a}^* = 0$ (which is unlikely to occur at first iteration). Otherwise, the node is divided. On the left child node, an upper bound is computed by solving the problem in (3) for $\mathcal{C} = \{p_1\}$. Let us observe that, as $a_{p_1}^* \neq 0$, the set of non-zero variables $\overline{\mathcal{Z}}$ is unchanged, therefore the lower bound inherits that of its parent node. On the contrary, for the right child node, the set $\mathcal{C}$, indexing variables included in the solution, is inherited from its parent node, and so is the upper bound, but a new lower bound needs to be computed, since p_1 was removed from $\overline{\mathcal{Z}}$. Such propagations of bounds are summarized in Fig. 2.

The list of existing nodes, say $\mathcal{L}$, is stored in memory with their related lower bounds, and the best (*i.e.,* lowest) upper bound among all computed ones is kept, denoted by $\underline{\text{UB}}$, as well as the corresponding minimizer, denoted by $\boldsymbol{a}_{\underline{\text{UB}}}^*$. The process is repeated by selecting a new node in $\mathcal{L}$. Let LB denote the lower bound obtained at that node (see (4)), and let $\boldsymbol{a}_{\text{LB}}^*$ denote the corresponding minimizer. Then:

- if the node is a leaf (that is, $\#\mathcal{C} = K$), then $\boldsymbol{a}_{\text{LB}}^*$ is also feasible, both lower and upper bounds coincide: if necessary, $\underline{\text{UB}}$ and $\boldsymbol{a}_{\underline{\text{UB}}}^*$ are updated;
- if LB $\geq \underline{\text{UB}}$, then it is proved that all solutions obtained from the current node will be suboptimal (feasible solutions will give even higher values than LB): the node is fathomed by *dominance* and removed from $\mathcal{L}$;
- if LB $< \underline{\text{UB}}$ and $\|\boldsymbol{a}_{\text{LB}}^*\|_0 \leq K$, i.e. $\boldsymbol{a}_{\text{LB}}^*$ is feasible — which may happen since ANC favors zero values— then the current node is pruned by *optimality*, and updates ($\underline{\text{UB}}$ = LB, $\boldsymbol{a}_{\underline{\text{UB}}}^* = \boldsymbol{a}_{\text{LB}}^*$) are performed;
- otherwise, the node cannot be pruned and is divided, and its two children are added to the list of existing nodes (see Section II-C and Fig. 2).

We use *depth-first search* [14] in order to schedule the node exploration, which is a standard choice in the sparse branch-and-bound literature [11], [13]. It explores in priority the nodes furthest from the root, aiming at quickly finding feasible K-sparse solutions. Once the search space has been completely explored, *i.e.*, $\mathcal{L} = \emptyset$, the optimality proof is achieved, $\boldsymbol{a}_{\underline{\text{UB}}}^*$ being the global minimizer of Problem (1).

The pseudo-code for our algorithm is given in Algorithm 1.

B. Computation of bounds

Most computations consist in evaluating upper and lower bounds at each node (Eqs. (3) and (4), respectively). Lower bound problems can be viewed as particular instances of underdetermined ($\#\mathcal{C} > N_\lambda$), ℓ_1-norm-constrained, sparse problems, for which the ℓ_1 norm equals 1. We solve them by an homotopy continuation method [16], certainly one of the most efficient strategies for the targeted problem sizes. Starting from the zero solution, a sequence of ℓ_1-norm penalized problems are solved, where the penalty parameter is iteratively decreased at the sequence of values where the support of the solution changes, until the ℓ_1 norm sums to one. This strategy can easily include box constraints—in our case, *i.e.*, $a_p \in [0, 1]$ [11]. On the contrary, upper bound computations in (3), which only involve a few variables, are quadratic programs that can be solved very efficiently by off-the-shelf solvers. In this work, such computations are run with the `qpOASES` software [17], under licenses LGPL-2.1.

Attached to this work, we distribute a free C++ implementation under license LGPL-3.0. All material including source code, installation procedure and usage instructions can be found at the Mimosa (*mixed integer programming methods for sparse approximation*) `repository`.

IV. Numerical results

In this section, algorithmic performance is evaluated on numerical simulations using the United States Geological Survey library [18], composed of 498 spectra covering $N_\lambda = 224$ spectral bands. A sub-dictionary is created by randomly extracting P columns, among which K-sparse mixtures are generated by randomly selecting the active endmembers. No library pruning was performed, therefore endmembers are

Algorithm 1: *BB* sparse unmixing algorithm

```
Data   : $\boldsymbol{y}$, $\mathbf{S}$, $K$
Result: $\hat{\boldsymbol{a}} := \textbf{Argmin}(\mathcal{P})$
// Initialization
$\mathcal{L} \leftarrow \emptyset$, $\mathcal{C} \leftarrow \emptyset$, $\overline{\mathcal{C}} \leftarrow \{1,\dots,P\}$, $\mathcal{Z} \leftarrow \emptyset$, $i = 0$
$\hat{\boldsymbol{a}} \leftarrow \mathbf{0}$, $\underline{\mathrm{UB}} \leftarrow \frac{1}{2}\|\boldsymbol{y}\|_2^2$
// add root node to the list
$\mathcal{N} \leftarrow (\mathcal{C}, \overline{\mathcal{C}}, \mathcal{Z})$,  $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{N}$
// Main loop
while $\mathcal{L} \neq \emptyset$ do
    $i \leftarrow i + 1$
    // take last node in $\mathcal{L}$ (depth-first search)
    pop $\mathcal{N} = (\mathcal{C}, \overline{\mathcal{C}}, \mathcal{Z})$, $\mathcal{L} \leftarrow \mathcal{L} \setminus \mathcal{N}$
    $prune \leftarrow$ false
    // Node evaluation procedure
    if $i = 0$ or $\mathcal{N}$ is left child then
        UB $:= \min \mathcal{P}_{\mathrm{UB}}$, $\boldsymbol{a}^{\mathrm{UB}} := \textbf{argmin}\, \mathcal{P}_{\mathrm{UB}}$ //Eq. (3)
    if $i = 0$ or $\mathcal{N}$ is right child then
        LB $:= \min \mathcal{P}_{\mathrm{LB}}$, $\boldsymbol{a}^{\mathrm{LB}} := \textbf{argmin}\, \mathcal{P}_{\mathrm{LB}}$ //Eq. (4)
    // Bound procedure
    if $UB < \underline{UB}$ then
        $\underline{\mathrm{UB}} \leftarrow$ UB, $\hat{\boldsymbol{a}} \leftarrow \boldsymbol{a}^{\mathrm{UB}}$
    if $LB \geq \underline{UB}$ then
        // Fathom by dominance
        $prune \leftarrow$ true
    else if $\|\boldsymbol{a}^{LB}\|_0 \leq K$ then
        // Fathom by feasibility
        $prune \leftarrow$ true, $\hat{\boldsymbol{a}} \leftarrow \boldsymbol{a}^{\mathrm{LB}}$, $\underline{\mathrm{UB}} \leftarrow$ LB
    end
    if $\#\mathcal{C} = K$ then
        $prune \leftarrow$ true        // Fathom by optimality
    // Branch procedure
    if $\overline{\mathcal{C}} \neq \emptyset$ and $prune = false$ then
        $q \leftarrow \textbf{argmax}_{i \in \overline{\mathcal{C}}}\, a_i^{\mathrm{LB}}$
        // Insert Right and Left children in $L$
        $\mathcal{N}^r \leftarrow (\mathcal{C}, \overline{\mathcal{C}} \setminus \{q\}, \mathcal{Z} \cup \{q\})$, $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{N}^r$
        $\mathcal{N}^\ell \leftarrow (\mathcal{C} \cup \{q\}, \overline{\mathcal{C}} \setminus \{q\}, \mathcal{Z})$, $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{N}^\ell$
    end
end
```

quite highly correlated. Amplitudes are generated uniformly in $\{\boldsymbol{a} \in [\tau, 1]^K |\ \sum_p a_p = 1\}$, with $\tau = 0.05$. Gaussian white noise is added with variance σ^2, by controlling the signal-to-noise ratio $\mathrm{SNR_{dB}} = 10\log_{10}(\|\mathbf{S}\boldsymbol{a}\|^2/(N_\lambda\,\sigma^2))$. In the following simulations, P varies from 20 to 400, K varies from 2 to 8, SNR varies from 60 to 30 dB, and results are averaged over 10 instances for each configuration. Typical mixtures at different SNRs are shown in Figure 3 (top).

A. *Solution quality*

The ℓ_0-SU solution is compared to estimates obtained by:

- restricting the FCLS solution to its K biggest values;
- considering ℓ_1-norm sparsity. To that aim, we remove the ASC, and the ℓ_1-norm constraint is varied until the solution has exactly K non-zero values;
- ℓ_p-norm sparsity, with $p = 0.5$, still without ASC. The (local) optimizer is that provided in [7];
- the backward elimination method in [8], which iteratively removes the small components in the FCLS solution.

All solutions are tuned to be K-sparse. Due to lack of space, only partial results are shown. More results and all data sets are available as supplementary material[1]. Let $\mathring{\boldsymbol{a}}$ and $\widehat{\boldsymbol{a}}$ denote the true and estimated abundance vectors, respectively. Centre and bottom rows in Fig. 3 respectively show the *support error* (SE) and the *signal to reconstruction error* (SRE), defined by:

$$\mathrm{SE} := \frac{1}{2}\frac{\|\mathring{\boldsymbol{a}} - \widehat{\boldsymbol{a}}\|_0}{K} \times 100, \quad \mathrm{SRE} := 10\log_{10}\frac{\|\mathring{\boldsymbol{a}}\|^2}{\|\mathring{\boldsymbol{a}} - \widehat{\boldsymbol{a}}\|^2},$$

averaged over instances and problem cardinality. On such

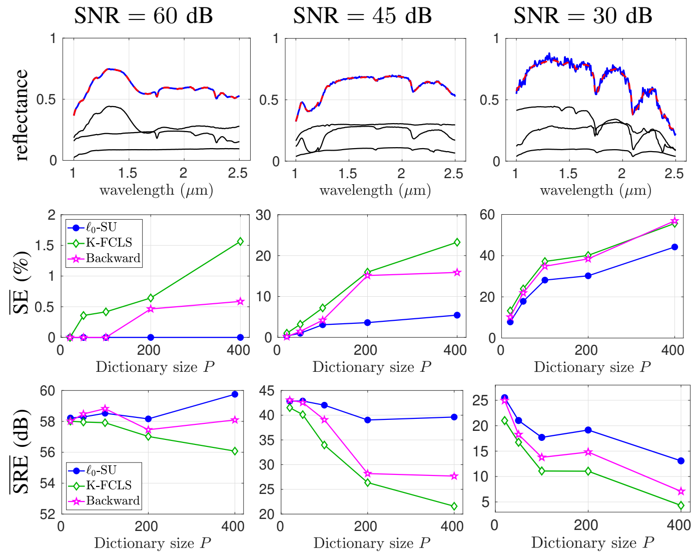


Fig. 3. Typical spectral mixtures at different noise levels with $K = 3$ (top). Noise-free mixture (red), data (blue) and elementary components (black). Support error (center) and signal to reconstruction error (bottom) for ℓ_0-SU and competing methods, averaged over instances and problem cardinality.

problems, ℓ_1- and $\ell_{0.5}$-norm solutions were far from reaching comparable performance to other methods, in particular due to the disadvantaging constraint imposing them to be exactly K-sparse. ℓ_0-SU always gives lower errors than its competitors. At the highest SNR, ℓ_0-SU always retrieves the truth, for any K and P. Of course, the performance worsens as SNR decreases: the global optimum found by solving (1) may more likely be obtained on a support other than the true one.

B. *Computational efficiency*

Computing times are evaluated as a function of the problem difficulty. Optimization is run on a single thread, using a laptop computer on Ubuntu 22.04.5 equipped with 32 Go RAM and Intel Core Ultra 7 165U processors. Some representative results are given in Table I; more complete ones can be found as supplementary material[1]. Complexity obviously depends on the combinatorial parameters K and P; it is also impacted by

[1]Material can be downloaded `there`.

the noise level: as noise gets stronger, discriminating solutions becomes harder, so the number of explored nodes increases. As long as the problem size (K, P) is relatively small or the SNR is high, problems can be solved in a few milliseconds, which is very efficient regarding the combinatorial complexity. For example, the average 35-ms time for SNR = 45 dB, $K = 6$ and $P = 100$ corresponds to $\binom{100}{6} \sim 1.2\,10^9$ possible combinations, whereas the search tree developed by our algorithm evaluates only 125 nodes in average. When both the noise level and the problem size increase, ℓ_0-SU faces exponential complexity and computing times can reach several hundreds of seconds, or even exceed the 1 000 s time limit.

SNR 60	$P = 50$	$P = 100$	$P = 200$	$P = 400$
$K = 2$	**1m** (6m)	**1m** (13m)	**4m** (36m)	**11m** (0.1)
$K = 4$	**3m** (11m)	**5m** (19m)	**11m** (43m)	**43m** (0.5)
$K = 6$	**4m** (9m)	**9m** (45m)	**21m** (1.0)	**0.3** (2)
$K = 8$	**5m** (16m)	**13m** (0.3)	**0.1** (12)	70 (**12**)$^{(1)}$
SNR 45	$P = 50$	$P = 100$	$P = 200$	$P = 400$
$K = 2$	**1m** (6m)	**2m** (12m)	**4m** (35m)	**13m** (0.1)
$K = 4$	**2m** (9m)	**32m** (56m)	**28m** (64m)	**1** (8)
$K = 6$	**7m** (21m)	**35m** (95m)	83 (**18**)	**133**$^{(1)}$ (170)$^{(1)}$
$K = 8$	**25m** (69m)	**1** (1)	**7**$^{(1)}$ (15)$^{(1)}$	**15**$^{(7)}$ (27)$^{(7)}$
SNR 30	$P = 50$	$P = 100$	$P = 200$	$P = 400$
$K = 2$	**1m** (6m)	**9m** (32m)	**28m** (68m)	**53m** (0.4)
$K = 4$	**13m** (33m)	**0.1** (0.2)	**1** (1)	**14**$^{(2)}$ (39)$^{(1)}$
$K = 6$	**0.1** (0.2)	1 (**1**)	157$^{(1)}$ (**30**)$^{(1)}$	116$^{(4)}$ (**103**)$^{(3)}$
$K = 8$	**0.2** (0.2)	20 (**3**)	109$^{(1)}$ (**58**)	NaN$^{(10)}$ (392)$^{(9)}$

TABLE I
COMPUTING TIMES FOR OUR *BB* ALGORITHM (FOR THE GUROBI MIP SOLVER) AVERAGED OVER 10 INSTANCES. TIMES IN SECONDS, EXCEPT 'M' (MILLISECONDS). THE EXPONENT MARKS THE NUMBER OF INSTANCES FOR WHICH THE 1 000- S TIME LIMIT WAS REACHED.

Table I also gives the computation times for the Gurobi commercial optimizer [19], that solves the equivalent MIP formulation [4]:

$$\min_{\boldsymbol{b}\in\{0,1\}^P, \boldsymbol{a}\in[0,1]^P} \tfrac{1}{2}\left\| \boldsymbol{y} - \mathbf{S}\boldsymbol{a} \right\|_2^2 \text{ s.t. } \begin{cases} \mathbf{1}_P^{\intercal}\,\boldsymbol{a} = 1, \sum_p b_p \leq K \\ 0 \leq a_p \leq b_p. \end{cases}$$

In most cases, our approach is more efficient, and even more when computing times are small. On the most difficult tested instances, however, Gurobi becomes faster. Note that such state-of-the art software benefits from many developments (mathematical refinements, computer engineering), that are not implemented in our algorithm, which opens up promising improvement perspectives.

V. CONCLUSION

The proposed branch-and algorithm, tailored for spectral unmixing problems with nonnegativity and sum-to-one constraints, is able to efficiently solve exact ℓ_0-norm sparse unmixing problems, as long as the number of endmembers or the cardinality constraint is small. This is particularly interesting for strongly supervised problems with a small library (or after library pruning [20]), or in unsupervised contexts, where the number of endmembers extracted from the observed scene rarely exceeds a few tenths. Of course, combinatorial complexity makes it harder to solve in higher dimensional problems, and so is it when the noise level increases. If the computational burden of ℓ_0-SU surely exceeds that of simpler methods, better estimation capacity may still make it worth in particular contexts. We also remark that, in general, the optimal solution is reached far before optimality was proved, therefore unguaranteed solutions obtained by limiting the computing time may also be considered with interest. Finally, if sparsity helps regularizing unmixing problems, tuning the sparsity level K may be a challenging practical issue. To that purpose, our current research is investigating multi-objective formulations *via* dedicated branch-and-bound algorithms.

Interested readers are invited to use and challenge our solver, available `there`.

REFERENCES

[1] N Keshava and J F Mustard. Spectral unmixing. *IEEE signal processing magazine*, 19(1):44–57, 2002.

[2] R B Singer and T B McCord. Mars-large scale mixing of bright and dark surface materials and implications for analysis of spectral reflectance. In *Lunar and Planetary Science Conference Proceedings*, volume 10, pages 1835–1848, 1979.

[3] D C Heinz and Chein-I-Chang. Fully constrained least squares linear spectral mixture analysis method for material quantification in hyperspectral imagery. *IEEE Trans. Geosci. Remote Sens.*, 39(3):529–545, 2001.

[4] R Ben Mhenni, S Bourguignon, J Ninin, and F Schmidt. Spectral unmixing with sparsity and structuring constraints. In *Proc. IEEE WHISPERS*, Amsterdam, The Netherlands, September 2018.

[5] M. D. Iordache, J. M. Bioucas-Dias, and A Plaza. Sparse unmixing of hyperspectral data. *IEEE Trans. Geosci. Remote Sens.*, 49(6), June 2011.

[6] J M Bioucas-Dias, A Plaza, N Dobigeon, M Parente, Q Du, P Gader, and J Chanussot. Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches. *IEEE J. Sel. Topics Appl. Earth Observ.*, 5(2):354–379, April 2012.

[7] D Tuia, R Flamary, and M Barlaud. Nonconvex regularization in remote sensing. *IEEE Trans. Geosci. Remote Sens.*, 54(11):6470–6480, Nov 2016.

[8] J Greer. Sparse demixing of hyperspectral images. *IEEE Trans. Image Process.*, 21:219–28, 06 2011.

[9] B K Natarajan. Sparse approximate solutions to linear systems. *SIAM J. Comput.*, 24:227–234, 1995.

[10] D Bertsimas, A King, and R Mazumder. Best subset selection via a modern optimization lens. *Ann. Stat.*, 44(2):813 – 852, 2016.

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

[12] T Guyard, C Herzet, C Elvira, and A-N Arslan. A new branch-and-bound pruning framework for ℓ_0-regularized problems. In *Proc. ICML*, 2024.

[13] H Hazimeh, R Mazumder, and A Saab. Sparse regression at scale: Branch-and-bound rooted in first-order optimization. *Math. Prog.*, 2021.

[14] L A Wolsey and G L Nemhauser. *Integer and Combinatorial Optimization*. John Wiley & Sons, July 1999.

[15] L Drumetz, T Meyer, J Chanussot, A Bertozzi, and C Jutten. Hyperspectral Image Unmixing with Endmember Bundles and Group Sparsity Inducing Mixed Norms. *IEEE Trans. Image Process.*, 28(7):3435–3450, July 2019.

[16] MR Osborne, B Presnell, and BA Turlach. A new approach to variable selection in least squares problems. *IMA J. Numer. Anal.*, 20(3):389–403, 07 2000.

[17] H J Ferreau, C Kirches, A Potschka, H G Bock, and M Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Math. Prog. Comput.*, 6(4):327–363, 2014.

[18] R N Clark et al. USGS digital spectral library splib05a. *US Geological Survey, Digital Data Series*, 231, 2003.

[19] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024.

[20] X Xu, B Pan, Z Chen, Z Shi, and T Li. Simultaneously multiobjective sparse unmixing and library pruning for hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 59(4):3383–3395, 2021.