Group-Sparse Model Selection: Hardness and Relaxations

Luca Baldassarre, Nirav Bhan, Volkan Cevher, and Anastasios Kyrillidis

Abstract

Group-based sparsity models are proven instrumental in linear regression problems for recovering signals from much fewer measurements than standard compressive sensing. The main promise of these models is the recovery of "interpretable" signals along with the identification of their constituent groups. To this end, we establish a combinatorial framework for group-model selection problems and highlight the underlying tractability issues revolving around such notions of interpretability when the regression matrix is simply the identity operator. We show that, in general, claims of correctly identifying the groups with convex relaxations would lead to polynomial time solution algorithms for a well-known NP-hard problem, called the weighted maximum cover problem. Instead, leveraging a graph-based understanding of group models, we describe group structures which enable correct model identification in polynomial time via dynamic programming. We also show that group structures that lead to totally unimodular constraints have tractable discrete as well as convex relaxations. Finally, we study the Pareto frontier of budgeted group-sparse approximations for the tree-based sparsity model and illustrate identification and computation trade-offs between our framework and the existing convex relaxations.

Index Terms

Signal Approximation, Structured Sparsity, Interpretability, Tractability, Dynamic Programming, Compressive Sensing.

I. Introduction

INFORMATION in many natural and man-made signals can be exactly represented or well approximated by a sparse set of nonzero coefficients in an appropriate basis [1]. Compressive sensing (CS) exploits this fact to recover signals from their compressive samples, which are dimensionality reducing, non-adaptive random measurements. According to the CS theory, the number of measurements for stable recovery is proportional to the signal sparsity, rather than to its Fourier bandwidth as dictated by the Shannon/Nyquist theorem [2]–[4]. Unsurprisingly, the utility of sparse representations also goes well-beyond CS and permeates a lot of fundamental problems in signal processing, machine learning, and theoretical computer science.

Recent results in CS extend the simple sparsity idea to consider more sophisticated *structured* sparsity models, which describe the interdependency between the nonzero coefficients [5]–[8]. There are several compelling reasons for such extensions: The structured sparsity models allow to significantly reduce the number of required measurements for perfect recovery in the noiseless case and be more stable in the presence of noise. Furthermore, they facilitate the interpretation of the signals in terms of the chosen structures, revealing information that could be used to better understand their properties.

An important class of structured sparsity models is based on groups of variables that should either be selected or discarded together [8]–[12]. These structures naturally arise in applications such as neuroimaging [13], [14], gene expression data [11], [15], bioinformatics [16], [17] and computer vision [7], [18]. For example, in cancer research, the groups might represent genetic pathways that constitute cellular processes. Identifying which processes lead to the development of a tumor can allow biologists to directly target certain groups of genes instead of others [15]. Incorrect identification of the active/inactive groups can thus have a rather dramatic effect on the speed at which cancer therapies are developed.

This work was supported in part by the European Commission under Grant MIRG-268398, ERC Future Proof and SNF 200021-132548.

The authors are with LIONS, EPFL, Lausanne, Switzerland (email: {luca.baldassarre, volkan.cevher, anastasios.kyrillidis}@epfl.ch, bhannirav@gmail.com).

Authors are listed in alphabetical order

As a result, in this paper, we consider *group-based* sparsity models, denoted as $\mathfrak{G}$. These structured sparsity models feature collections of groups of variables that could overlap arbitrarily, that is $\mathfrak{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_M\}$ where each $\mathcal{G}_j$ is a subset of the index set $\{1, \ldots, N\}$, with N being the dimensionality of the signal that we model. Arbitrary overlaps mean that we do not restrict the intersection between any two sets $\mathcal{G}_j$ and $\mathcal{G}_\ell$ from $\mathfrak{G}$.

In this paper, we address the *signal approximation* problem based on a known group structures $\mathfrak{G}$. That is, given a signal $\mathbf{x} \in \mathbb{R}^N$, we seek an $\hat{\mathbf{x}}$ closest to it in the Euclidean sense, whose *support* (i.e., the index set of its non-zero coefficients) consists of the union of at most G groups from $\mathfrak{G}$, where $G > 0$ is a user-defined group budget:

$$\hat{\mathbf{x}} \in \operatorname*{argmin}_{\mathbf{z} \in \mathbb{R}^N} \left\{ \|\mathbf{x} - \mathbf{z}\|_2^2 : \operatorname{supp}(\mathbf{z}) \subseteq \bigcup_{\mathcal{G} \in \mathcal{S}} \mathcal{G}, \mathcal{S} \subseteq \mathfrak{G}, |\mathcal{S}| \leq G \right\},$$

where $\operatorname{supp}(\mathbf{z})$ is the support of the vector $\mathbf{z}$. We call such an approximation as *G-group-sparse* or in short *group-sparse*.

More importantly, we seek to identify the *G-group-support* of the approximation $\hat{\mathbf{x}}$, that is the G groups that constitute its support. We call this the group-sparse *model selection* problem. The G-group-support of $\hat{\mathbf{x}}$ allows us to "interpret" the original signal and discover its properties so that we can, for example, target specific groups of genes instead of others [15] or focus more precise imaging techniques on certain brain regions only [19]. As a result, we study under which circumstances we can correctly and tractably identify the group-support of a given signal.

Previous work. Recent works in compressive sensing and machine learning with group sparsity have mainly focused on leveraging the group structures for lowering the number of samples required for recovering signals [5]–[8], [11], [20]–[22]. While these results have established the importance of group structures, many of these works have not fully addressed the relevant issue of model selection.

For the special case of non-overlapping groups, dubbed as the block-sparsity model, the problem of model selection does not present computational difficulties and features a well-understood theory [20]. The first convex relaxations for group-sparse approximation [23] considered only non-overlapping groups. Its extension to overlapping groups [24] has the drawback of selecting supports defined as the complement of a union of groups (see also [10]).

For overlapping groups, on the other hand, Eldar et al. [5] consider the union of subspaces framework and cast the model selection problem as a block-sparse model selection one by duplicating the variables that belong to overlaps between the groups. Their uniqueness condition [5][Prop. 1], however, is infeasible for any group structure with overlaps, because it requires that the subspaces intersect only at the origin, while two subspaces defined by two overlapping groups of variables intersect on a subspace of dimension equal to the number of elements in the overlap.

The recently proposed convex relaxations [11], [22] for group-sparse approximations select group-supports that consist of union of groups. However, the group-support recovery conditions in [11], [22] should be taken with care, because they are defined with respect to a particular subset of group-supports and are not general. As we numerically demonstrate in this paper, the group-supports recovered with these methods might be incorrect. Furthermore, the required consistency conditions in [11], [22] are unverifiable *a priori*. For instance, they require tuning parameters to be known beforehand to obtain the correct group-support.

Contributions. This paper is an extended version of a prior submission to the IEEE International Symposium on Information Theory (ISIT), 2013. This version contains all the proofs that were previously omitted due to lack of space, refined explanations of the concepts, and provides the full description of the proposed dynamic programming algorithms.

In stark contrast to the existing literature, we take an explicitly discrete approach to identifying group-supports of signals given a budget constraint on the number of groups. This fresh perspective enables us to show that the group-sparse model selection problem is NP-hard: if we can solve the group model selection problem in general, then we can solve any weighted maximum coverage (WMC) problem instance in polynomial time. However, WMC is known to be NP-Hard. Given this connection, we can only hope to characterize a subset of instances which are tractable or find guaranteed and tractable approximations.

We then present characterizations of group structures that lead to computationally tractable problems via dynamic programming. We do so by leveraging a graph-based representation of the groups and exploiting properties of the induced graph. We present and describe a novel dynamic program that solves the WMC problem for a specific class of group structures and could be of interest by itself. We also identify tractable discrete relaxations of the group-sparse model selection problem that lead to efficient algorithms. Specifically, we relax the constraint on the number of groups into a penalty term and show that if the remaining group constraints satisfy a property related to the concept of total unimodularity [25], then the relaxed problem can be efficiently solved using linear program solvers. We also extend the discrete model to incorporate an overall sparsity constraint and allowing to select individual elements from each group, leading to within-group sparsity. Furthermore, we discuss how this extension can be used to model hierarchical relationships between variables. We present a novel dynamic program that solves the hierarchical model selection problem exactly and discuss a tractable discrete relaxation.

We also interpret the implications of our results in the context of other group-based recovery frameworks. For instance, the convex approaches proposed in [5], [11], [22] also relax the discrete constraint on the cardinality of the group support. However, they first need to decompose the approximation into vector components whose support consists only of one group and then penalize the norms of these components. It has been observed [11] that these relaxations produce approximations that are group-sparse, but their group-support might include irrelevant groups. We concretely illustrate these cases via a Pareto frontier example.

Paper structure. The paper is organized as follows. In Section 2, we present definitions of group-sparsity and related concepts, while in Section III, we formally define the approximation and model-selection problems and connect them to the WMC problem. We present and analyze discrete relaxations of the WMC in Section IV and consider convex relaxations in Section V. In Section VI, we illustrate via a simple example the differences between the original problem and the relaxations. The generalized model is introduced and analyzed in Section VII, while numerical simulations are presented in Section VIII. We conclude the paper with some remarks in Section IX. The appendices contain the detailed descriptions of the dynamic programs.

II. Basic Definitions

Let $\mathbf{x} \in \mathbb{R}^N$ be a vector and $\mathcal{N} = \{1, \dots, N\}$ be the ground set of its indices. We use $|\mathcal{S}|$ to denote the cardinality of an index set $\mathcal{S}$. We use $\mathbb{B}^N$ to represent the space of N-dimensional binary vectors and define $\iota : \mathbb{R}^N \to \mathbb{B}^N$ to be the indicator function of the nonzero components of a vector in $\mathbb{R}^N$, i.e., $\iota(\mathbf{x})_i = 1$ if $x_i \neq 0$ and $\iota(\mathbf{x})_i = 0$, otherwise. We let $\mathbf{1}_N$ to be the N-dimensional vector of all ones and $\mathbf{I}_N$ the $N \times N$ identity matrix. The support of $\mathbf{x}$ is defined by the set-valued function $\operatorname{supp}(\mathbf{x}) = \{i \in \mathcal{N} : x_i \neq 0\}$. Note that we use bold lowercase letters to indicate vectors and bold uppercase letters to indicate matrices.

Definition II.1. *A* **group structure** $\mathfrak{G} = \{\mathcal{G}_1, \dots, \mathcal{G}_M\}$ *is a collection of index sets, named* groups*, with* $\mathcal{G}_j \subseteq \mathcal{N}$ *and* $|\mathcal{G}_j| = g_j$ *for* $1 \leq j \leq M$ *and* $\bigcup_{\mathcal{G} \in \mathfrak{G}} \mathcal{G} = \mathcal{N}$.

We can represent a group structure $\mathfrak{G}$ as a bipartite graph, where on one side we have the N variables nodes and on the other the M group nodes. An edge connects a variable node i to a group node j if $i \in \mathcal{G}_j$. Fig. 1 shows an example. The bi-adjacency matrix $\mathbf{A}^{\mathfrak{G}} \in \mathbb{B}^{N \times M}$ of the bipartite graph encodes the group structure,

$$A^{\mathfrak{G}}_{ij} = \begin{cases} 1, & \text{if } i \in \mathcal{G}_j; \\ 0, & \text{otherwise.} \end{cases}$$

Another useful representation of a group structure is via a *group graph* $(\mathcal{V}, \mathcal{E})$ where the nodes $\mathcal{V}$ are the groups $\mathcal{G} \in \mathfrak{G}$ and the edge set $\mathcal{E}$ contains e_{ij} if $\mathcal{G}_i \cap \mathcal{G}_j \neq \emptyset$, that is an edge connects two groups that *overlap*. A sequence of connected nodes $v_1, v_2, \dots, v_n$, is a *loop* if $v_1 = v_n$.

In order to illustrate these concepts, consider the group structure $\mathfrak{G}^1$ defined by the following groups, $\mathcal{G}_1 = \{1\}$, $\mathcal{G}_2 = \{2\}$, $\mathcal{G}_3 = \{1, 2, 3, 4, 5\}$, $\mathcal{G}_4 = \{4, 6\}$, $\mathcal{G}_5 = \{3, 5, 7\}$ and $\mathcal{G}_6 = \{6, 7, 8\}$. $\mathfrak{G}^1$ can be represented by the variables-groups bipartite graph of Fig. 1 or by the group graph of Fig. 2, which is bipartite and contains loops.

An important group structure is given by *loopless pairwise overlapping groups*. This group structure consists of groups such that each element of the ground set occurs in at most two groups and the induced graph does not contain loops. Therefore the group graph for these structures is actually a tree or a forest and hence bipartite. For example,

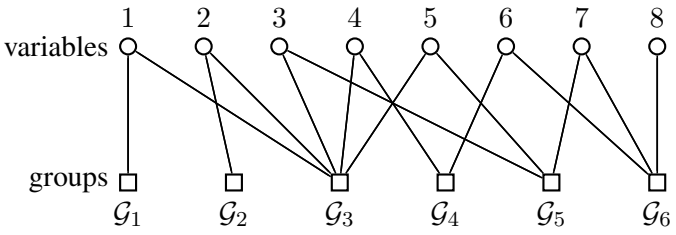

Fig. 1. Example of bipartite graph between variables and groups induced by the group structure $\mathfrak{G}^1$, see text for details.

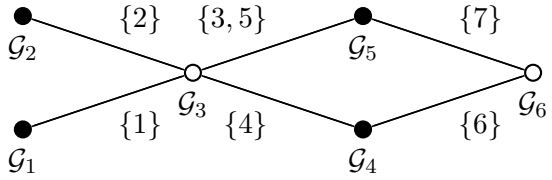

Fig. 2. Bipartite group graph with loops induced by the group structure $\mathfrak{G}^1$, where on each edge we report the elements of the intersection.

consider $\mathcal{G}_1 = \{1, 2, 3\}$, $\mathcal{G}_2 = \{3, 4, 5\}$, $\mathcal{G}_3 = \{5, 6, 7\}$, which can be represented by the graph in Fig. 3(Left). If $\mathcal{G}_3$ were to include an element from $\mathcal{G}_1$, for example $\{2\}$, we would have the loopy graph of Fig. 3(Right). Note that $\mathfrak{G}^1$ is pairwise overlapping, but not loopless, since $\mathcal{G}_3, \mathcal{G}_4, \mathcal{G}_5$ and $\mathcal{G}_6$ form a loop.

We anchor our analysis of the tractability of interpretability via selection of groups on covering arguments.

Definition II.2. *A* **group cover** $\mathcal{S}(\mathbf{x})$ *for a signal* $\mathbf{x} \in \mathbb{R}^N$ *is a collection of groups such that* $\operatorname{supp}(\mathbf{x}) \subseteq \bigcup_{\mathcal{G} \in \mathcal{S}(\mathbf{x})} \mathcal{G}$. *An alternative equivalent definition is given by*

$$\mathcal{S}(\mathbf{x}) = \{\mathcal{G}_j \in \mathfrak{G} : \boldsymbol{\omega} \in \mathbb{B}^M,\ \omega_j = 1,\ \mathbf{A}^{\mathfrak{G}}\boldsymbol{\omega} \geq \iota(\mathbf{x})\}\ .$$

The binary vector $\boldsymbol{\omega}$ indicates which groups are active and the constraint $\mathbf{A}^{\mathfrak{G}}\boldsymbol{\omega} \geq \iota(\mathbf{x})$ makes sure that, for every non-zero component of $\mathbf{x}$, there is at least one active group that covers it. We also say that $\mathcal{S}(\mathbf{x})$ *covers* $\mathbf{x}$. Note that the group cover is often not unique and $\mathcal{S}(\mathbf{x}) = \mathfrak{G}$ is a group cover for any signal $\mathbf{x}$. This observation leads us to consider more restrictive definitions of group cover.

Definition II.3. *A* G**-group cover** $\mathcal{S}^G(\mathbf{x}) \subseteq \mathfrak{G}$ *is a group cover for* $\mathbf{x}$ *with at most* G *elements,*

$$\mathcal{S}^G(\mathbf{x}) = \{\mathcal{G}_j \in \mathfrak{G} : \boldsymbol{\omega} \in \mathbb{B}^M,\ \omega_j = 1,\ \mathbf{A}^{\mathfrak{G}}\boldsymbol{\omega} \geq \iota(\mathbf{x}),\ \sum_{j=1}^{M} \omega_j \leq G\}\ .$$

It is not guaranteed that a G-group cover always exists for any value of G. Finding the smallest G-group cover lead to the following definitions.

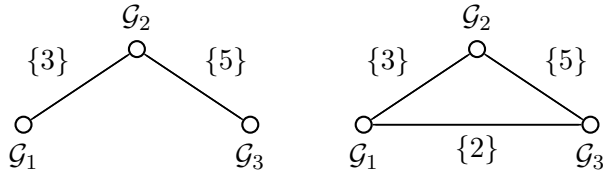

Fig. 3. (Left) Loopless pairwise overlapping groups. (Right) By adding one element from $\mathcal{G}_1$ into $\mathcal{G}_3$, we introduce a loop in the graph.

Definition II.4. *The* **group ℓ_0-"norm"** *is defined as*

$$\|\mathbf{x}\|_{\mathfrak{G},0} := \min_{\boldsymbol{\omega}\in\mathbb{B}^M} \left\{ \sum_{j=1}^{M} \omega_j : \mathbf{A}^{\mathfrak{G}}\boldsymbol{\omega} \geq \iota(\mathbf{x}) \right\} . \tag{1}$$

Definition II.5. *A* **minimal group cover** *for a signal $\mathbf{x} \in \mathbb{R}^N$ is defined as $\mathcal{M}(\mathbf{x}) = \{\mathcal{G}_j \in \mathfrak{G} : \hat{\omega}(\mathbf{x})_j = 1\}$, where $\hat{\boldsymbol{\omega}}$ is a minimizer for* (1)*,*

$$\hat{\boldsymbol{\omega}}(\mathbf{x}) \in \operatorname*{argmin}_{\boldsymbol{\omega}\in\mathbb{B}^M} \left\{ \sum_{j=1}^{M} \omega_j : \mathbf{A}^{\mathfrak{G}}\boldsymbol{\omega} \geq \iota(\mathbf{x}) \right\} .$$

A *minimal group cover* $\mathcal{M}(\mathbf{x})$ is a group cover for $\mathbf{x}$ with minimal cardinality. Note that there exist pathological cases where for the same group ℓ_0-"norm", we have different minimal group cover models.

Definition II.6. *A signal $\mathbf{x}$ is G-***group sparse** *with respect to a group structure $\mathfrak{G}$ if $\|\mathbf{x}\|_{\mathfrak{G},0} \leq G$.*

In other words, a signal is *G-group sparse* if its support is contained in the union of at most G groups from $\mathfrak{G}$.

III. Tractability of interpretations

A group-based interpretation of a signal consists in identifying the groups that constitute the support of its approximation. In this section, we establish the hardness of finding group-based interpretations of signals in general and characterize a class of group structures that lead to tractable interpretations. In particular, we present a polynomial time algorithm that finds the correct G-group-support of the G-group-sparse approximation of $\mathbf{x}$, given a positive integer G and the group structure $\mathfrak{G}$.

We first define the G-group sparse approximation $\hat{\mathbf{x}}$ and then show that it can be easily obtained from its G-group cover $\mathcal{S}^G(\hat{\mathbf{x}})$, which is the solution of the model selection problem. We then reformulate the model selection problem as the weighted maximum coverage problem. Finally, we present our main result, the polynomial time dynamic program for loopless pairwise overlapping group structures.

Problem 1 (Signal approximation)**.** *Given a signal $\mathbf{x} \in \mathbb{R}^N$, a best G-group sparse approximation $\hat{\mathbf{x}}$ is given by*

$$\hat{\mathbf{x}} \in \operatorname*{argmin}_{\mathbf{z}\in\mathbb{R}^N} \left\{ \|\mathbf{x}-\mathbf{z}\|_2^2 : \|\mathbf{z}\|_{\mathfrak{G},0} \leq G \right\} . \tag{2}$$

If we already know the G-group cover of the approximation $\mathcal{S}^G(\hat{\mathbf{x}})$, we can obtain $\hat{\mathbf{x}}$ as $\hat{\mathbf{x}}_{\mathcal{I}} = \mathbf{x}_{\mathcal{I}}$ and $\hat{\mathbf{x}}_{\mathcal{I}^c} = 0$, where $\mathcal{I} = \bigcup_{\mathcal{G}\in\mathcal{S}^G(\hat{\mathbf{x}})} \mathcal{G}$ and $\mathcal{I}^c = \mathcal{N} \setminus \mathcal{I}$. Therefore, we can solve Problem 1 by solving the following discrete problem.

Problem 2 (Model selection)**.** *Given a signal $\mathbf{x} \in \mathbb{R}^N$, a G-group cover model for its G-group sparse approximation is expressed as follows*

$$\mathcal{S}^G(\hat{\mathbf{x}}) \in \operatorname*{argmax}_{\substack{\mathcal{S}\subseteq\mathfrak{G} \\ |\mathcal{S}|\leq G}} \left\{ \sum_{i\in\mathcal{I}} x_i^2 : \mathcal{I} = \bigcup_{\mathcal{G}\in\mathcal{S}} \mathcal{G} \right\} . \tag{3}$$

To show the connection between the two problems, we first reformulate Problem 1 as

$$\min_{\mathbf{z}\in\mathbb{R}^N} \left\{ \|\mathbf{x}-\mathbf{z}\|_2^2 : \operatorname{supp}(\mathbf{z}) = \mathcal{I}, \mathcal{I} = \bigcup_{\mathcal{G}\in\mathcal{S}} \mathcal{G}, \mathcal{S} \subseteq \mathfrak{G}, |\mathcal{S}| \leq G \right\},$$

which can be rewritten as

$$\min_{\substack{\mathcal{S}\subseteq\mathfrak{G} \\ |\mathcal{S}|\leq G \\ \mathcal{I}=\bigcup_{\mathcal{G}\in\mathcal{S}}\mathcal{G}}} \quad \min_{\substack{\mathbf{z}\in\mathbb{R}^N \\ \operatorname{supp}(\mathbf{z})=\mathcal{I}}} \|\mathbf{x}-\mathbf{z}\|_2^2 .$$

The optimal solution is not changed if we introduce a constant, change sign of the objective and consider maxi-

mization instead of minimization

$$\max_{\substack{\mathcal{S}\subseteq\mathfrak{G}\\ |\mathcal{S}|\leq G\\ \mathcal{I}=\bigcup_{\mathcal{G}\in\mathcal{S}}\mathcal{G}}} \quad \max_{\substack{\mathbf{z}\in\mathbb{R}^N\\ \operatorname{supp}(\mathbf{z})=\mathcal{I}}} \left\{\|\mathbf{x}\|_2^2-\|\mathbf{x}-\mathbf{z}\|_2^2\right\}.$$

The internal maximization is achieved for $\hat{\mathbf{x}}$ as $\hat{\mathbf{x}}_{\mathcal{I}}=\mathbf{x}_{\mathcal{I}}$ and $\hat{\mathbf{x}}_{\mathcal{I}^c}=0$, so that we have, as desired,

$$\mathcal{S}^G(\hat{\mathbf{x}})\in \operatorname*{argmax}_{\substack{\mathcal{S}\subseteq\mathfrak{G}\\ |\mathcal{S}|\leq G\\ \mathcal{I}=\bigcup_{\mathcal{G}\in\mathcal{S}}\mathcal{G}}} \|\mathbf{x}_{\mathcal{I}}\|_2^2 .$$

The following reformulation of Problem 2 as a binary problem allows us to characterize its tractability.

Lemma 1. *Given $\mathbf{x}\in\mathbb{R}^N$ and a group structure $\mathfrak{G}$, we have that $\mathcal{S}^G(\hat{\mathbf{x}})=\{\mathcal{G}_j\in\mathfrak{G}:\omega_j^G=1\}$, where $(\boldsymbol{\omega}^G,\mathbf{y}^G)$ is an optimal solution of*

$$\max_{\boldsymbol{\omega}\in\mathbb{B}^M,\ \mathbf{y}\in\mathbb{B}^N}\left\{\sum_{i=1}^N y_i x_i^2 : \mathbf{A}^{\mathfrak{G}}\boldsymbol{\omega}\geq\mathbf{y}, \sum_{j=1}^M\omega_j\leq G\right\}. \tag{4}$$

Proof. The proof follows along the same lines as the proof in [26]. Note that in (4), $\boldsymbol{\omega}$ and $\mathbf{y}$ are binary variables that specify which groups and which variables are selected, respectively. The constraint $\mathbf{A}^{\mathfrak{G}}\boldsymbol{\omega}\geq\mathbf{y}$ makes sure that for every selected variable at least one group is selected to cover it, while the constraint $\sum_{j=1}^M\omega_j\leq G$ restricts choosing at most G groups. □

Problem (4) can produce all the instances of the weighted maximum coverage problem (WMC), where the weights for each element are given by x_i^2 $(1\leq i\leq N)$ and the index sets are given by the groups $\mathcal{G}_j\in\mathfrak{G}$ $(1\leq j\leq M)$. Since WMC is in general NP-hard and given Lemma 1, the tractability of (3) directly depends on the hardness of (4), which leads to the following result.

Proposition III.1. *The model selection problem* (3) *is in general NP-hard.*

It is possible to approximate the solution of (4) using the greedy WMC algorithm [27]. At each iteration, the algorithm selects the group that covers new variables with maximum combined weight until G groups have been selected. However, we show next that for certain group structures we can find an exact solution.

Our main result is an algorithm for solving (4) for loopless pairwise overlapping groups structures. The proof is given in Appendix A.

Proposition III.2. *Given a loopless pairwise overlapping group structure $\mathfrak{G}$, there exists a polynomial time dynamic programming algorithm that solves* (4)*.*

IV. Discrete relaxations

Relaxations are useful techniques that allow to obtain approximate, or even sometimes exact, solutions while being computationally less demanding. In our case, by relaxing the constraint on the number of groups in (4) into a regularization term with parameter $\lambda>0$, we obtain the following binary linear program

$$(\boldsymbol{\omega}^\lambda,\mathbf{y}^\lambda)\in\operatorname*{argmax}_{\boldsymbol{\omega}\in\mathbb{B}^M,\mathbf{y}\in\mathbb{B}^N}\left\{\sum_{i=1}^N y_i x_i^2-\lambda\sum_{j=1}^M\omega_j : \mathbf{A}^{\mathfrak{G}}\boldsymbol{\omega}\geq\mathbf{y}\right\} \tag{5}$$

We can rewrite the previous program in standard form. Let $\mathbf{u}^\top=[\mathbf{y}^\top\ \boldsymbol{\omega}^\top]\in\mathbb{B}^{N+M}$, $\mathbf{w}^\top=[x_1^2,\ldots,x_N^2,-\lambda\mathbf{1}_M^\top]$ and $\mathbf{C}=[\mathbf{I}_N,\ -\mathbf{A}^{\mathfrak{G}}]$. We then have that (5) is equivalent to

$$\mathbf{u}^\lambda\in\operatorname*{argmax}_{\mathbf{u}\in\mathbb{B}^{N+M}}\left\{\mathbf{w}^\top\mathbf{u} : \mathbf{C}\mathbf{u}\leq 0\right\} \tag{6}$$

In general, (6) is NP-hard, however, it is well known [25] that if the constraint matrix $\mathbf{C}$ is Totally Unimodular (TU), then it can be solved in polynomial-time. While the concatenation of two TU matrices is not TU in general,

the concatenation of the identity matrix with a TU matrix results in a TU matrix. Thus, due to its structure, $\mathbf{C}$ is TU if $\mathbf{A}^{\mathfrak{G}}$ is TU [25].

Group structures that can be represented by a bipartite graph, such as the one in Fig. 2, lead to constraint matrices $\mathbf{A}^{\mathfrak{G}}$ that are TU [25].

Lemma 2. *Loopless pairwise overlapping groups lead to totally unimodular constraints.*

Proof. We first use a result that establishes that if a matrix is TU, then its transpose is also TU [25][Prop.2.1]. We then apply [25][Corollary 2.8] to $\mathbf{A}^{\mathfrak{G}}$, swapping the roles of rows and columns. Given a binary matrix whose columns can be partitioned into two disjoint sets and with no more than two nonzero elements in each row, this result provides two sufficient conditions for it being totally unimodular. In our case, the columns of $\mathbf{A}^{\mathfrak{G}}$ can be partitioned in two sets, $\mathcal{S}_1$ and $\mathcal{S}_2$ because the group graph for loopless pairwise overlapping groups is bipartite. The two sets represents groups which have no common overlap. Furthermore, each row of $\mathbf{A}^{\mathfrak{G}}$ contains at most two nonzero entries due to the pairwise overlap. We can now easily check that the two conditions on $\mathbf{A}^{\mathfrak{G}}$ are satisfied:

- If two nonzero entries in a row have the same sign, then the column of one is in $\mathcal{S}_1$ and the other is in $\mathcal{S}_2$: indeed if an element belongs to two groups, these groups must lie in two different sets;
- If two nonzero entries in a row have opposite signs, then their columns are both in $\mathcal{S}_1$ or both in $\mathcal{S}_2$: there are no such rows in our case.

□

Even though for this group structure we can use the dynamic program of Prop. III.2, for very large problems it may be computationally faster to solve the binary linear program. The next proposition establishes when the regularized solution coincides with the solution of (4).

Lemma 3. *If the value of the regularization parameter λ is such that the solution $(\boldsymbol{\omega}^\lambda, \mathbf{y}^\lambda)$ of* (5) *satisfies $\sum_j \omega_j^\lambda = G$, then $(\boldsymbol{\omega}^\lambda, \mathbf{y}^\lambda)$ is also a solution for* (4)*.*

Proof. This lemma is a direct consequence of Prop. IV.1 below. □

However, as we numerically show in Section VIII, given a value of G it is not always possible to find a value of λ such that the solution of (5) is also a solution for (4). Let the set of points $\mathcal{P} = \{G, (f(G))\}_{G=1}^{M}$, where $f(G) = \sum_{i=1}^{N} y_i^G x_i^2$, be the Pareto frontier of (4). We then have the following characterization of the solutions of the discrete relaxation.

Proposition IV.1. *The discrete relaxation* (5) *yields only the solutions that lie on the intersection between the Pareto frontier of* (4)*, $\mathcal{P}$, and the boundary of the convex hull of $\mathcal{P}$.*

Proof. On the one hand, the solutions of (4) for all possible values of G are the Pareto optimal solutions of the following vector-valued minimization problem with respect to the positive orthant of $\mathbb{R}^2$, which we denote by $\mathbb{R}^2_+$,

$$\begin{array}{ll} \min\limits_{\boldsymbol{\omega}\in\mathbb{B}^M,\ \mathbf{y}\in\mathbb{B}^N} & f(\boldsymbol{\omega}, \mathbf{y}) \\ \text{subject to} & \mathbf{A}^{\mathfrak{G}}\boldsymbol{\omega} \geq \mathbf{y} \end{array} \tag{7}$$

where $f(\boldsymbol{\omega}, \mathbf{y}) = \left(\|x\|^2 - \sum_{i=1}^{N} y_i x_i^2, \sum_{j=1}^{M} \omega_j \right) \in \mathbb{R}^2_+$.

On the other hand, the scalarization of (7) yields the following discrete problem, with $\lambda > 0$

$$\begin{array}{ll} \min\limits_{\boldsymbol{\omega}\in\mathbb{B}^M,\ \mathbf{y}\in\mathbb{B}^N} & \|\mathbf{x}\|^2 - \sum_{i=1}^{N} y_i x_i^2 + \lambda \sum_{j=1}^{M} \omega_j \\ \text{subject to} & \mathbf{A}^{\mathfrak{G}}\boldsymbol{\omega} \geq \mathbf{y} \end{array} \tag{8}$$

whose solutions are the same as for (5). Therefore, the relationship between the solutions of (4) and (5) can be inferred by the relationship between the solutions of (7) and (8). It is known that the solutions of (8) are also Pareto optimal solutions of (7), but only the Pareto optimal solutions of (7) that admit a supporting hyperplane for the feasible objective values of (7) are also solutions of (8) [28]. In other words, the solutions obtainable via scalarization belong to the intersection of the Pareto optimal solution set and the boundary of its convex hull. □

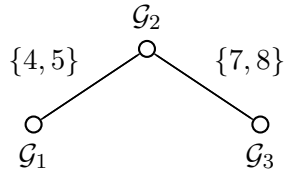

Fig. 4. The group-graph for the example in Section VI

V. Convex relaxations

For tractability and analysis, convex proxies to the group ℓ_0-norm have been proposed (e.g., [22]) for finding group-sparse approximations of signals. Given a group structure $\mathfrak{G}$, an example generalization is defined as

$$\|\mathbf{x}\|_{\mathfrak{G},\{1,p\}} := \inf_{\substack{\mathbf{v}^1,\dots,\mathbf{v}^M \in \mathbb{R}^N \\ \forall j, \operatorname{supp}(\mathbf{v}^j) = \mathcal{G}_j}} \left\{ \sum_{j=1}^{M} d_j \|\mathbf{v}^j\|_p : \sum_{j=1}^{M} \mathbf{v}^j = \mathbf{x} \right\}, \tag{9}$$

where $\|\mathbf{x}\|_p = \left(\sum_{i=1}^N x_i^p\right)^{1/p}$ is the ℓ_p-norm, and d_j are positive weights that can be designed to favor certain groups over others [11]. This norm can be seen a weighted instance of the atomic norm described in [8], where the authors leverage convex optimization for signal recovery, but not for model selection.

One can in general use (9) to find a group-sparse approximation under the chosen group norm

$$\hat{\mathbf{x}} \in \operatorname*{argmin}_{\mathbf{z}\in\mathbb{R}^N} \left\{ \|\mathbf{x}-\mathbf{z}\|_2^2 : \|\mathbf{z}\|_{\mathfrak{G},\{1,p\}} \leq \lambda \right\}, \tag{10}$$

where $\lambda > 0$ controls the trade-off between approximation accuracy and group-sparsity. However, solving (10) does not yield a group-support for $\hat{\mathbf{x}}$: even though we can recover one through the decomposition $\{\mathbf{v}^j\}$ used to compute $\|\hat{\mathbf{x}}\|_{\mathfrak{G},\{1,p\}}$, it may not be unique as observed in [11] for $p = 2$. In order to characterize the group-support for $\mathbf{x}$ induced by (9), in [11] the authors define two group-supports for $p = 2$. The *strong group-support* $\breve{\mathcal{S}}(\mathbf{x})$ contains the groups that constitute the supports of each decomposition used for computing (9). The *weak group-support* $\mathcal{S}(\mathbf{x})$ is defined using a dual-characterisation of the group norm (9). If $\breve{\mathcal{S}}(\mathbf{x}) = \mathcal{S}(\mathbf{x})$, the group-support is uniquely defined. However, [11] observed that for some group structures and signals $\mathbf{x}$, even when $\breve{\mathcal{S}}(\mathbf{x}) = \mathcal{S}(\mathbf{x})$, the group-support does not capture the minimal group-cover of $\mathbf{x}$. Hence, the equivalence of ℓ_0 and ℓ_1 minimization [2], [3] in the standard compressive sensing setting does not hold in the group-based sparsity setting.

VI. Case study: discrete vs. convex interpretability

The following stylized example illustrates situations that can potentially be encountered in practice. In these cases, the group-support obtained by the convex relaxation will not coincide with the discrete definition of group-cover, while the dynamical programming algorithm of Prop. III.2 is able to recover the correct group-cover.

Let $\mathcal{N} = \{1,\dots,11\}$ and let $\mathfrak{G} = \{\mathcal{G}_1 = \{1,\dots,5\},\ \mathcal{G}_2 = \{4,\dots,8\},\ \mathcal{G}_3 = \{7,\dots,11\}\}$ be the loopless pairwise overlapping groups structure with 3 groups of equal cardinality. Its group graph is represented in Fig. 4. Consider the 2-group sparse signal $\mathbf{x} = [0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0]^\top$, with minimal group-cover $\mathcal{M}(\mathbf{x}) = \{\mathcal{G}_1, \mathcal{G}_3\}$.

The dynamic program of Prop. III.2, with group budget $G = 2$, correctly identifies the groups $\mathcal{G}_1$ and $\mathcal{G}_3$. The TU linear program (5), with $0 < \lambda \leq 2$, also yields the correct group-cover. Conversely, the decomposition obtained via (9) with unitary weights is unique, but is not group sparse. In fact, we have $\mathcal{S}(\mathbf{x}) = \breve{\mathcal{S}}(\mathbf{x}) = \mathfrak{G}$. We can only obtain the correct group-cover if we use the weights $[1\ d\ 1]$ with $d > \frac{2}{\sqrt{3}}$, that is knowing beforehand that $\mathcal{G}_2$ is irrelevant.

Remark 1. *Indeed, if the convex relaxation always recovered the correct minimal group-cover, it would be possible to solve the discrete NP-hard problem in polynomial time.*

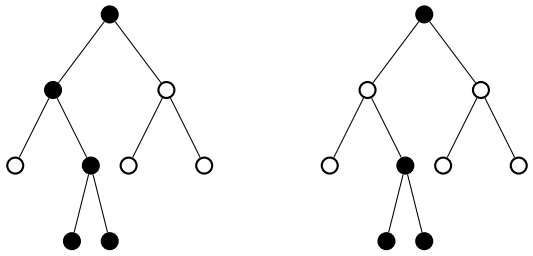

Fig. 5. Hierarchical constraints. Each node represent a variable. (Left) A valid selection of nodes. (Right) An *invalid* selection of nodes.

VII. Generalizations

In this section, we first present a generalization of the discrete approximation problem (4) by introducing an additional overall sparsity constraint. Secondly, we show how this generalization encompasses approximation with hierarchical constraints that can be solved exactly via dynamic programming. Finally, we show that the generalized problem can be relaxed into a linear binary problem and that hierarchical constraints lead to totally unimodular matrices for which there exists efficient polynomial time solvers.

A. Sparsity within groups

In many applications, for example genome-wide association studies [17], it is desirable to find approximations that are not only group-sparse, but also sparse in the usual sense (see [29] for an extension of the group lasso). To this end, we generalize our original problem (4) by introducing a sparsity constraint K and allowing to individually select variables within a group. The generalized integer problem then becomes

$$\max_{\boldsymbol{\omega}\in\mathbb{B}^M,\ \mathbf{y}\in\mathbb{B}^N} \left\{ \sum_{i=1}^{N} y_i x_i^2 : \mathbf{A}^{\mathfrak{G}}\boldsymbol{\omega} \geq \mathbf{y}, \sum_{i=1}^{N} y_i \leq K, \sum_{j=1}^{M} \omega_j \leq G \right\} . \tag{11}$$

This problem is in general NP-hard too, but it turns out that it can be solved in polynomial time for the same group structures that allow to solve (4).

Proposition VII.1. *Given a loopless pairwise overlapping groups structure $\mathfrak{G}$, there exists dynamic programming algorithm that solves* (11) *with complexity linear in N and K.*

Proof. The dynamic program is described in Appendix A alongside the proof that it has a polynomial running time. □

B. Hierarchical constraints

The generalized model allows to deal with hierarchical structures, such as regular trees, frequently encountered in image processing (e.g. denoising using wavelet trees). In such cases, we often require to find K-sparse approximations such that the selected variables form a rooted connected subtree of the original tree, see Fig. 5. Given a tree $\mathcal{T}$, the rooted-connected approximation can be cast as the solution of the following discrete problem

$$\max_{\mathbf{y}\in\mathbb{B}^N} \left\{ \sum_{i=1}^{N} y_i x_i^2 : \mathbf{y} \in \mathcal{T}_K \right\} , \tag{12}$$

where $\mathcal{T}_K$ denotes all rooted and connected subtrees of the given tree $\mathcal{T}$ with at most K nodes.

This type of constraint can be represented by a group structure, where for each node in the tree we define a group consisting of that node and all its ancestors. When a group is selected, we require that all its elements are selected as well. We impose an overall sparsity constraint K, while discarding the group constraint G.

For this particular problem, for which convex approximations have been proposed [30], we present an exact dynamic program that runs in polynomial time. While preparing the final version of this manuscript, [**?**] independently

proposed a similar dynamic program for tree projections. Our initial complexity analysis resulted in $\mathcal{O}(NK^2D)$, where D is the maximum number of children that a node in the tree can have. Following [**?**], we improved the analysis of our algorithm to $\mathcal{O}(NKD)$.

Proposition VII.2. *Given a hierarchical group structure $\mathfrak{G}$, there exists a dynamic programming algorithm that solves* (12) *in $\mathcal{O}(NKD)$ operations.*

Proof. The description of the algorithm and the proof of its time complexity can be found in Appendix B. □

C. Additional discrete relaxations

By relaxing both the group budget and the sparsity budget in (11) into regularization terms, we obtain the following binary linear program

$$(\boldsymbol{\omega}^\lambda, \mathbf{y}^\lambda) \in \underset{\boldsymbol{\omega}\in\mathbb{B}^M, \mathbf{y}\in\mathbb{B}^N}{\operatorname{argmax}} \left\{ \mathbf{w}^\top \mathbf{u} : \mathbf{u}^\top = [\mathbf{y}^\top \ \boldsymbol{\omega}^\top \ \mathbf{y}^\top],\ \mathbf{C}\mathbf{u} \leq 0 \right\} \tag{13}$$

where $\mathbf{w}^\top = [x_1^2, \ldots, x_N^2, -\lambda_G \mathbf{1}_M^\top, -\lambda_K \mathbf{1}_N^\top]$ and $\mathbf{C} = [\mathbf{I}_N,\ -\mathbf{A}^{\mathfrak{G}},\ \mathbf{0}_N]$ and $\lambda_G, \lambda_K > 0$ are two regularization parameters that indirectly control the number of active groups and the number of selected elements. (13) can be solved in polynomial time if the constraint matrix $\mathbf{C}$ is totally unimodular. Due to its structure, $\mathbf{C}$ is totally unimodular if $\mathbf{A}^{\mathfrak{G}}$ is totally unimodular [25]. The next results proves that the constraint matrix of hierarchical group structures is totally unimodular.

Proposition VII.3. *Hierarchical group structures lead to totally unimodular constraints.*

Proof. We use the fact that a binary matrix is totally unimodular if there exists a permutation of its columns such that in each row the 1s appear consecutively [25]. For hierarchical group structures, such permutation is given by a depth-first ordering of the groups. In fact, a variable is included in the group that has it as the leaf and in all the groups that contain its descendants. Given a depth-first ordering of the groups, the groups that contain the descendants of a given node will be consecutive. □

VIII. Pareto Frontier Example

The purpose of this numerical simulation is to illustrate the limitations of relaxations for correctly estimating the G-group cover of an approximation. We consider the problem of finding a K-sparse approximation of a signal imposing hierarchical constraints. We generate a piecewise constant signal of length $N = 64$, to which we apply the Haar wavelet transformation, yielding a 25-sparse vector of coefficients $\mathbf{x}$ that satisfies hierarchical constraints on a binary tree of depth 5, see Fig. 6(Left).

We compare the proposed dynamic program (DP) to the regularized totally unimodular linear program approach and two convex relaxations that use group-based norms. The first [8] uses the Latent Group Lasso penalty (10) with groups defined as all father-child relations in the tree. This formulation will not enforce all hierarchical constraints to be satisfied, but will only 'favor' them. Therefore, we also report the number of hierarchical constraint violations. The second [30] considers a hierarchy of groups where $\mathcal{G}_j$ contains node j and all its descendants. Hierarchical constraints are enforced by the group lasso penalty $\Omega_{GL}(\mathbf{x}) = \sum_{\mathcal{G}\in\mathfrak{G}} \|\mathbf{x}_\mathcal{G}\|_2$, where $\mathbf{x}_\mathcal{G}$ is the restriction of $\mathbf{x}$ to $\mathcal{G}$. We call this method Hierarchical Group Lasso. Once we determine the support of the solution, we assign to the components in the support the values of the corresponding components of the original signal.

In Fig. 6(Right), we show the approximation error $\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2$ as a function of the solution sparsity K for the methods. The values of the DP solutions form the discrete Pareto frontier of the optimization problem controlled by the parameter K. Note that there are points in the Pareto frontier that do not lie on its convex hull, hence these solutions are not achievable by the TU linear relaxation. We observe that the Hierarchical Group Lasso[1] also misses the solutions for $K = 21$ and $K = 23$, while the Latent Group Lasso[2] approach achieves more levels of sparsity

[1]We used the code provided in http://spams-devel.gforge.inria.fr/.

[2]We used the duplication of variables approach and solved the resulting Group Lasso problem using SpaRSA: http://www.lx.it.pt/~mtf/SpaRSA/

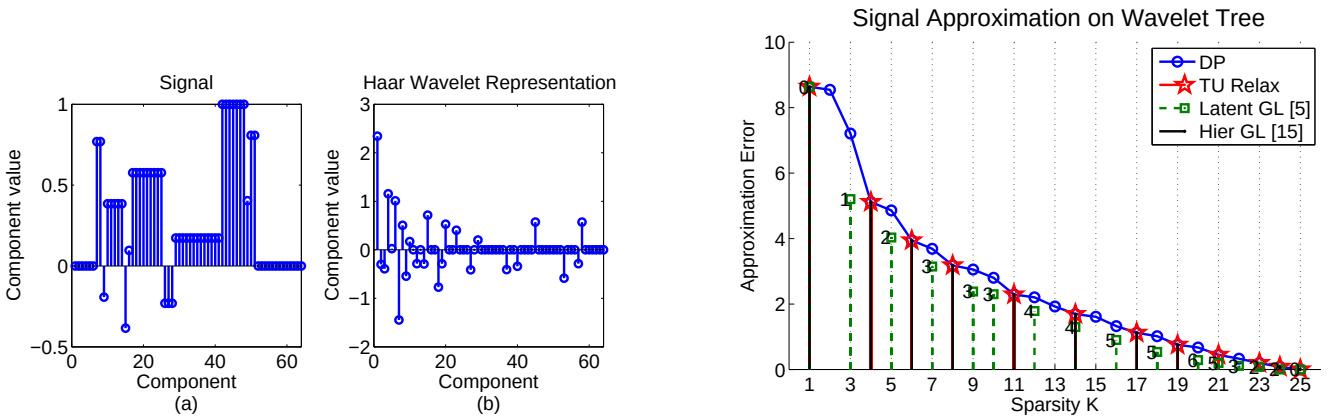

Fig. 6. (Left) (a) Original piecewise constant signal. (b) Haar wavelet representation. (Right) Signal approximation on a binary tree. The original signal is 25-sparse and satisfies hierarchical constraints. The numbers next to the Latent Group Lasso solutions indicate the number of constraint violations.

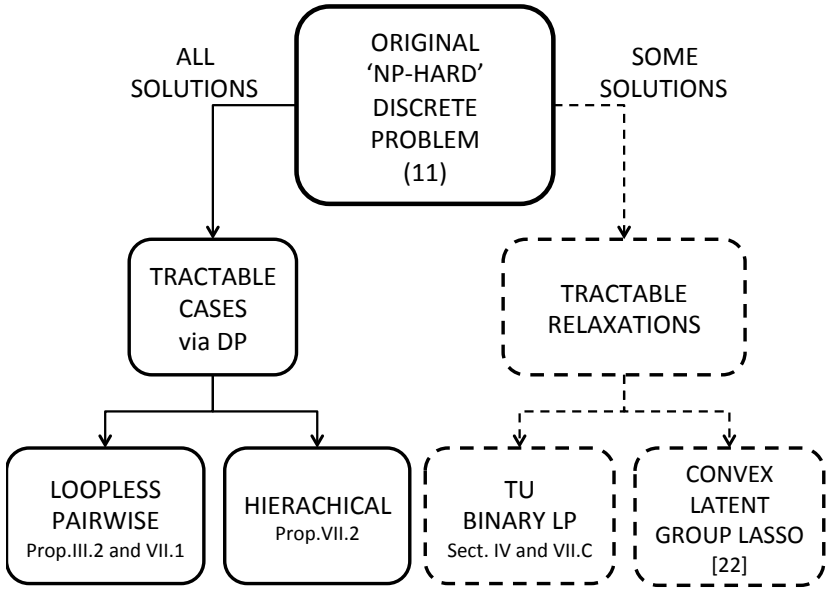

Fig. 7. Characterization of tractability for group-based interpretations.

(but still missing the solutions for $K = 2, 13$ and 15), although at the price of violating some of the hierarchical constraints. These observations lead us to conclude that, in some cases, relaxations of the original discrete problem might not be able to find the correct group-based interpretation of a signal.

IX. Conclusions

Many applications benefit from group sparse representations. Unfortunately, our result in this paper shows that finding a group-based interpretation of a signal is an integer optimization problem, which is in general NP-hard. To this end, we characterize group structures for which a dynamical programming algorithm can find a solution in polynomial time and also delineate discrete relaxations for special structures (i.e., totally unimodular constraints) that can obtain correct solutions in special circumstances.

Our examples and numerical simulations show the deficiencies of convex relaxations. We observe that such methods only recover group-covers that lie in the convex hull of the Pareto frontier determined by the solutions of the original integer problem for different values of the group budget G (and sparsity budget K for the generalized model). This, in turn, implies that convex and non-convex relaxations might miss some important groups or include spurious ones in the group-sparse model selection. We summarize our findings in Fig. 7.

APPENDIX A
DYNAMICAL PROGRAMMING FOR SOLVING (11) FOR LOOPLESS PAIRWISE OVERLAPPING GROUPS

Here, we give the proof of Prop. VII.1. The proof of Prop. III.2 follows along similar lines.

Proof. The proof consists in describing the algorithm and showing it is polynomial time. (11) can be equivalently described by the following problem: given a signal $\mathbf{x} \in \mathbb{R}^N$ and a group structure $\mathfrak{G}$ consisting of M groups defined over the index set $\mathcal{N} = \{1, \ldots, N\}$, with each index having an associated (non-negative) weight (i.e., $x_i^2,\ \forall\, i \in \mathcal{N}$), find the optimal selection of at most K indices, to maximize the sum of their weights, such that the indices are contained in a union of at most G groups.

We highlight the optimal substructure inherent in this problem, which allows us to solve it using a dynamic programming approach. The optimal substructure is somewhat involved: we represent it below by two properties.

1) Suppose we know the G groups that constitute the optimal solution. Then the optimal choice of elements corresponds to picking the K largest weight elements belonging to the union of the G groups.
2) Suppose we know the groups and elements selected in the optimal solution under a G-groups and K-elements constraint. Partition the set of chosen groups into two sets, $\mathcal{S}_1$ and $\mathcal{S}_2$, consisting of g_1 and g_2 groups respectively ($g_1 + g_2 = G$). Suppose $\mathcal{S}_1$ contains k_1 of the elements in the optimal solution, and suppose $\mathcal{S}_2$ contains additional k_2 elements *excluding elements covered by* $\mathcal{S}_1$ ($k_1 + k_2 = K$). Then, given the selection of groups and elements in $\mathcal{S}_1$, $\mathcal{S}_2$ represents the optimal selection of at most k_2 elements from at most g_2 groups in $\mathcal{S}_1^c$ (i.e. $\mathfrak{G} \setminus \mathcal{S}_1$), after the elements in $\mathcal{S}_1$ have been removed from the groups in $\mathcal{S}_1^c$.

These properties lead us to a dynamic programming based method for obtaining the solution. The underlying graph has as nodes groups in $\mathfrak{G}$. The algorithm gradually explores every node in the group-graph, storing the optimal solution among the visited nodes and it is defined by two rules: the **Node Picking Rule** controls how the graph must be explored in order to minimize the number of values to store; the **Value Update Rule** describes how the stored values are updated when a new node is considered. Due to the looplessness constraint, the graph can be represented as a tree or a forest. Choose an arbitrary node and call it the root node.

Let $\mathfrak{G}$ be the set of all nodes and let $\mathcal{S} \subseteq \mathfrak{G}$ be the set of currently explored nodes with $|\mathcal{S}| = m$. Define 3-valued indicator variables, $I_1, I_2, \ldots, I_M$ for each of the M nodes. $I_j = 1$ indicates that the jth node is selected, $I_j = 0$ shows that it is forbidden, while $I_j = 2$ represents a "don't care" state: there is no restriction on the jth node being either chosen or not. For unexplored nodes, the indicator variables are always in the "don't care" state. At every step of the algorithm, we store the optimal value for choosing at most k elements, from at most g nodes, $1 \leq k \leq K$ and $1 \leq g \leq G$ from the currently explored set of nodes, $\mathcal{S}$. These optimal values are stored in the function, $F(\mathcal{S}, g, k, I_1, I_2, \ldots, I_M)$.

We define an additional function $H(\mathcal{S}, k)$ to represent the optimal value for choosing k elements from a set $\mathcal{S}$. The set $\mathcal{S}$ could be a single group, a union of groups, or any well-defined collection of elements. As noted earlier, the optimal selection corresponds to simply picking the k largest weight elements in $\mathcal{S}$.

Our aim is to obtain the value: $F(\mathfrak{G}, G, K, I_1 = 2, I_2 = 2, \ldots, I_M = 2)$. All indicator variables are set to 2, as we do not care about any particular group being selected or rejected in the final selection. Notice that by definition,

$$\begin{aligned} F(\mathcal{S}, g, k, I_1 = i_1, \ldots, I_j = 2, \ldots, I_M = i_M) = \max\{&F(\mathcal{S}, g, k, I_1 = i_1, \ldots, I_j = 0, \ldots, I_M = i_M), \\ &F(\mathcal{S}, g, k, I_1 = i_1, \ldots, I_j = 1, \ldots, I_M = i_M)\}, \end{aligned}$$

i.e., the optimal value when we do not care about a particular group being selected, is simply the maximum over the two cases of the group being selected and being rejected, *ceteris paribus*.

Note that the function F has an input space which is exponential in M (since the indicator variables combined can take exponentially many values). Therefore, if we tried to determine the values of F at all possible points, we would need an exponential amount of space and time. However, we shall see that our algorithm needs to work with only a small set of values of F, and hence runs in polynomial time. This happens because the values of the indicator variables will be important only for certain specific nodes, called boundary nodes. We define a **boundary node** as an explored node adjacent to an unexplored node. Hence the groups defined by the boundary node and the adjacent unexplored node overlap.

Base Case. We start by taking $\mathcal{S} = \emptyset$. For this case, all values of F will be set to 0: $F(\emptyset, g, k, I_1 = i_1, I_2 = i_2, \ldots, I_M = i_M) = 0 \ \forall g, k, i_1, \ldots, i_M$.

Assume that we have ordered the nodes from 1 to M according to some criteria. We shall now explore the nodes in this order and use the following value-update rules.

Value Update Rule. Suppose at a particular step, we have explored the first m nodes. We assume that we have stored the values of F for each g and k. Further, we assume that we have stored the values of F separately for each value of the indicator variable for each boundary nodes. Denote by $\mathcal{S}_i$ the set of the first i nodes. Denote by $\mathcal{B}_m$ the set of boundary nodes when m nodes have been explored. Denote by $\mathcal{X}_m$ the mth group. We assume that the following values are available at this step: $F(\mathcal{S}_m, g, k, I_1 = i_1, \ldots, I_M = i_M)$ for all $1 \leq g \leq G$ and $1 \leq k \leq K$ and all $i_1, \ldots, i_M$ such that $i_j \in \{0, 1\}$ for $j \in \mathcal{B}_m$ and $i_j = 2$ for $j \in \{1, \ldots, M\} \setminus \mathcal{B}_m$. Note that the indicator variables for all non-boundary nodes, as well as the unexplored nodes are set equal to 2. Thus the total number of values that have to be stored equals $GK2^{|\mathcal{B}_M|}$

The value update rule is divided into 3 cases and a final condensation step.

1) *The new node is rejected.* All optimal values for all k and all g remain the same as for m nodes. The added node is treated as a new boundary node and the stored values are associated to the new node being rejected.

$$\begin{aligned} &F(\mathcal{S}_{m+1}, g, k, I_1 = i_1, \ldots, I_m = i_m, \mathbf{I_{m+1}} = \mathbf{0}, I_{m+2} = 2, \ldots, I_M = 2) \\ &= F(\mathcal{S}_m, g, k, I_1 = i_1, \ldots, I_m = i_m, \mathbf{I_{m+1}} = \mathbf{2}, \ldots, I_M = 2) \end{aligned}$$

for all $1 \leq g \leq G$ and $1 \leq k \leq K$ and all $i_1, \ldots, i_M$ such that $i_j \in \{0, 1\}$ for $j \in \mathcal{B}_m$ and $i_j = 2$ for $j \in \{1, \ldots, m\} \setminus \mathcal{B}_m$.

2) *The new node is accepted (no overlap with any explored node).* Since the new node is selected, we can choose at most $g - 1$ explored nodes. We first compute the sum of the optimal value for choosing the best ℓ elements from the new node and the optimal value for choosing $k - \ell$ elements from the $g - 1$ explored nodes, for any ℓ such that $1 \leq \ell \leq k$. Then, the new optimal value for each g and k is given by taking the maximum of these sums over ℓ.

$$\begin{aligned} &F(\mathcal{S}_{m+1}, g, k, I_1 = i_1, \ldots, I_m = i_m, \mathbf{I_{m+1}} = \mathbf{1}, I_{m+2} = 2, \ldots, I_M = 2) \\ &= \max_{1 \leq \ell \leq k} \{F(\mathcal{S}_m, \mathbf{g - 1}, \boldsymbol{k - \ell}, I_1 = i_1, \ldots, I_m = i_m, \mathbf{I_{m+1}} = \mathbf{0}, \ldots, I_M = 2) + H(\mathcal{X}_{m+1}, \ell)\} \end{aligned}$$

for all $1 \leq g \leq G$ and $1 \leq k \leq K$ and all $i_1, \ldots, i_M$ such that $i_j \in \{0, 1\}$ for $j \in \mathcal{B}_m$ and $i_j = 2$ for $j \in \{1, \ldots, m\} \setminus \mathcal{B}_m$.

3) *The new node is accepted (overlaps with some explored nodes).* The update rule is the same as for case 2, but the elements in the region of overlap between the new node and the selected explored nodes must not be considered as being part of the new node. In other words, the new node must be 'cleaned' of the overlapping region before updating. For this step, we need to know exactly which nodes have been chosen while computing an optimal value. This is the reason why we need to store separate values for each boundary node. We further assume that the cleaning operation can be done in $O(1)$ time, leading to a total complexity of $O(GK^2 2^{|\mathcal{B}_m|})$.

$$\begin{aligned} &F(\mathcal{S}_{m+1}, g, k, I_1 = i_1, \ldots, I_m = i_m, \mathbf{I_{m+1}} = \mathbf{1}, I_{m+2} = 2, \ldots, I_M = 2) \\ &= \max_{1 \leq \ell \leq k} \left\{F(\mathcal{S}_m, \mathbf{g - 1}, \boldsymbol{k - \ell}, I_1 = i_1, \ldots, I_m = i_m, \mathbf{I_{m+1}} = \mathbf{0}, \ldots, I_M = 2) + H(\mathcal{X}'_{m+1}, \ell)\right\} \end{aligned}$$

for all $1 \leq g \leq G$ and $1 \leq k \leq K$ and all $i_1, \ldots, i_M$ such that $i_j \in \{0, 1\}$ for $j \in \mathcal{B}_m$ and $i_j = 2$ for $j \in \{1, \ldots, m\} \setminus \mathcal{B}_m$. We also define $\mathcal{X}'_{m+1} = \mathcal{X}_{m+1} \setminus \bigcup_{j \in \mathcal{B}} \mathcal{X}_j$, with $\mathcal{B} = \{j \in \mathcal{B}_m : i_j = 1\}$. That is we "clean" $\mathcal{X}_m$ of the overlap with the currently selected boundary nodes.

4) *Condensation.* After these steps, the number of stored values will be (at most) doubled. We can reduce them: for each boundary node which has fallen into the interior of the explored nodes, we combine the optimal values for it being picked or unpicked, into a single value by taking the larger of the two values. Each such

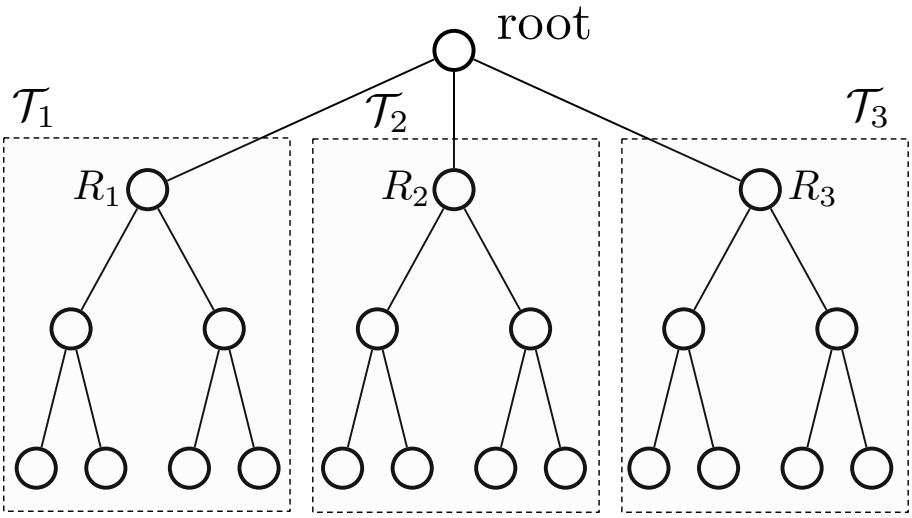

Fig. 8. Node Picking Rule: explore nodes in the order $\mathcal{T}_1, \text{root}, \mathcal{T}_2, \mathcal{T}_3$ where $D_1 \geq D_2 \geq D_3$. For the subtree $\mathcal{T}_1$, the node connected to `root` should be considered the root of $\mathcal{T}_1$, which we denote by R_1; similarly for the other subtrees.

operation reduces the number of stored values by half and we perform it after each value update

$$F(\mathcal{S}_{m+1}, g, k, I_1 = i_1, \ldots, \mathbf{I_j} = \mathbf{2}, \ldots, I_M = i_M) = \max\{F(\mathcal{S}_m, g, k, I_1 = i_1, \ldots, \mathbf{I_j} = \mathbf{0}, \ldots, I_M = i_M), \\ F(\mathcal{S}_m, g, k, I_1 = i_1, \ldots, \mathbf{I_j} = \mathbf{1}, \ldots, I_M = i_M)\}$$

for all $j \in (\mathcal{B}_m \cup X_{m+1}) \setminus \mathcal{B}_{m+1}$ and for all $1 \leq g \leq G$ and $1 \leq k \leq K$ and for all $i_1, \ldots, i_M$.

Time Complexity. Let B be the maximum number of boundary nodes encountered by the algorithm, then the number of steps is bounded by $O(2^B K^2 G M)$. We now give an algorithm to explore the graph so that B is logarithmic in M, establishing polynomial complexity.

Node Picking Rule. In order to minimize the number of boundary nodes encountered by the algorithm, we must explore the graph in a particular fashion. The order with which the nodes are picked is determined by a value associated to each subtree of the graph, which we call the D-value. In the following we describe how it is computed, how it depends logarithmically on the number of nodes in the graph and how the number of boundary nodes is bounded by the D-value.

The Node Rule Picking rule is defined as follows. We first order all rooted subtrees with respect to the the D-value, so that $D_1 \geq \ldots \geq D_R$ for subtrees $T_1, T_2, \ldots, T_R$. We then pick the subtrees in the order $\{T_1, \text{root}, T_2, \ldots, T_R\}$ and recurse until the explored subtree has only one node, see Fig. 8.

The procedure for computing the D-values is also recursive. If the tree has only one node, $D = 1$. Now, assume the subtrees at a node Q have values $D_1 \geq \ldots \geq D_R$. Then, $D(Q) = \max(D_1, D_2 + 1)$. In case there is no 2^{nd} subtree, $D_2 = 0$. We then have the following bound on the D-values.

Lemma 4. *The D-value of a rooted tree graph is logarithmic in the number of nodes, i.e. $D(G) \leq \log_2(M) + 1$.*

Proof. Let D be a positive integer and $N(D)$ be the minimum number of nodes that a rooted tree must have in order to have D-value of D. We prove by induction that

$$N(D) \geq 2^{D-1}. \tag{14}$$

Base case: $D = 1$. A tree with only one node will have a D-value of 1. Hence (14) is satisfied.

Inductive case: $D > 0$. Let $\mathcal{T}$ be the smallest rooted tree graph whose D-value is equal to D. Spread out $\mathcal{T}$ in the form of root and subtrees. Let the subtrees be $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$; with corresponding D-values $D_1, D_2, \ldots, D_k$. Without loss of generality, assume that $D_1 \geq D_2 \geq \ldots \geq D_k$. By definition, $D(\mathcal{T}) = \max(D_1, D_2 + 1)$.

By our assumption, $\mathcal{T}$ is the smallest graph with D-value equal to D, hence we cannot have $D_1 = D(G) = D$, since that would give us a smaller rooted tree graph ($\mathcal{T}_1$) with a D-value of D. This means that $D_1 < D$, and hence $D_2 + 1 = D$, i.e. $D_2 = D - 1$. Since $D_1 \geq D_2 = D - 1$ and $D_1 < D$, then $D_1 = D_2 = D - 1$. Thus, the graph $\mathcal{T}$ has 2 subtrees ($\mathcal{T}_1$ and $\mathcal{T}_2$), with D-values of $D - 1$ each. By definition, any rooted subtree with a D-value of $D - 1$ must have at least $N(D - 1)$ nodes. By our induction hypothesis, $N(D - 1) \geq 2^{D-2}$. Therefore, $\mathcal{T}$ has at least $2 \times 2^{D-2} = 2^{D-1}$ nodes. But since $\mathcal{T}$ was the smallest rooted tree graph with D-value of D, this means that $N(D) \geq 2^{D-1}$, as required. □

We now link the number of boundary nodes visited by the algorithm to the D-value of the group graph.

Lemma 5. *The total number of boundary nodes encountered by the node-picking algorithm cannot exceed the D-value of the graph.*

Proof. Let $\mathcal{T}$ be the given rooted tree graph, with M nodes. We shall consider the number of boundary nodes when there is a *ghost node* connected to the root node. This implies that the root, when explored, will always be counted as a boundary node. The ghost node captures the fact when we are running the algorithm recursively on a subtree, there will be an additional (potentially unexplored) node connected to the root of the subtree, which may lead to the root being counted as a boundary node. Let $B^*(\mathcal{T})$ denote the maximum number of boundary nodes encountered on $\mathcal{T}$ when we pick nodes according to our algorithm, and let $B^*_G(\mathcal{T})$ represent the same when we also have the ghost node. Clearly, $B^*_G(\mathcal{T}) \geq B^*(\mathcal{T})$, hence it is enough to prove the following:

$$B^*_G(\mathcal{T}) \leq D(\mathcal{T}). \tag{15}$$

We prove this by induction on M.

Base Case. Suppose the rooted tree graph $\mathcal{T}$ has only 1 node. Then the maximum number of boundary nodes encountered is obviously 1, which is equal to the D-value of the graph (by definition). Hence $B^*_G(\mathcal{T}) \leq D(\mathcal{T})$.

Inductive Case. When the graph $\mathcal{T}$ consists of M nodes, $M > 1$, consider the graph to be spread out in the form of root and subtrees. Compute the D-values for each subtree, where w.l.o.g., $D_1 \geq D_2 \geq \ldots D_k$. Let $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ be the corresponding subtrees. Our algorithm explores nodes in the sequence: $\mathcal{T}_1, \text{root}, \mathcal{T}_2, \mathcal{T}_3, \ldots \mathcal{T}_k$.

Since each subtree has strictly fewer than M nodes, each subtree satisfies (15) by the induction hypothesis. Also, notice that when exploring the subtree $\mathcal{T}_1$ of $\mathcal{T}$, the number of boundary nodes encountered is less than or equal to the number of boundary nodes encountered when exploring $\mathcal{T}_1$ as a standalone rooted-tree-graph, with a ghost node connected to its root. By construction, this is exactly equal to $B^*_G(\mathcal{T}_1)$, which by our induction hypothesis is bounded by D_1. Therefore, the number of boundary nodes encountered while exploring $\mathcal{T}_1$ in $\mathcal{T}$ cannot exceed D_1. Once we are finished with $\mathcal{T}_1$, we pick the root, so the total number of boundary nodes is 1. We now proceed to pick $\mathcal{T}_2$. By a similar argument, the maximum number of boundary nodes in $\mathcal{T}_2$ at any point cannot exceed the number of boundary nodes encountered while exploring $\mathcal{T}_2$ as a standalone graph with attached ghost node. In addition, the root of $\mathcal{T}$ can contribute at most 1 additional boundary node (In fact, the ghost node for $\mathcal{T}$ ensures that the root, once picked, will always contribute an additional boundary node). Therefore, the total number of boundary nodes in $\mathcal{T}$ while exploring $\mathcal{T}_2$ is at most $D_2 + 1$. Similar arguments hold for all other subtrees — the maximum number of boundary nodes while exploring the k-th subtree will be at most $D_k + 1$, which is at most $D_2 + 1$.

Therefore, the maximum number of boundary nodes encountered at any step while exploring $\mathcal{T}$ is $B^*_G(\mathcal{T}) \leq \max(D_1, D_2 + 1)$. By definition, $D(\mathcal{T}) = \max(D_1, D_2 + 1)$. Therefore $B^*_G(\mathcal{T}) \leq D(\mathcal{T})$. □

Combining Lemmas 4 and 5, we have the following result.

Proposition A.1. *The maximum number of boundary nodes at any step of the algorithm is logarithmic in the number of nodes, i.e., $B \leq \log_2(M) + 1$.*

The previous proposition establishes the polynomial time complexity of the dynamic program for solving the generalized integer problem (11).

Theorem 1. *The proposed dynamic program solves, in polynomial time, the problem of Weighted Maximum Cover with an additional constraint on element sparsity for loopless pairwise overlapping groups. In particular, its time*

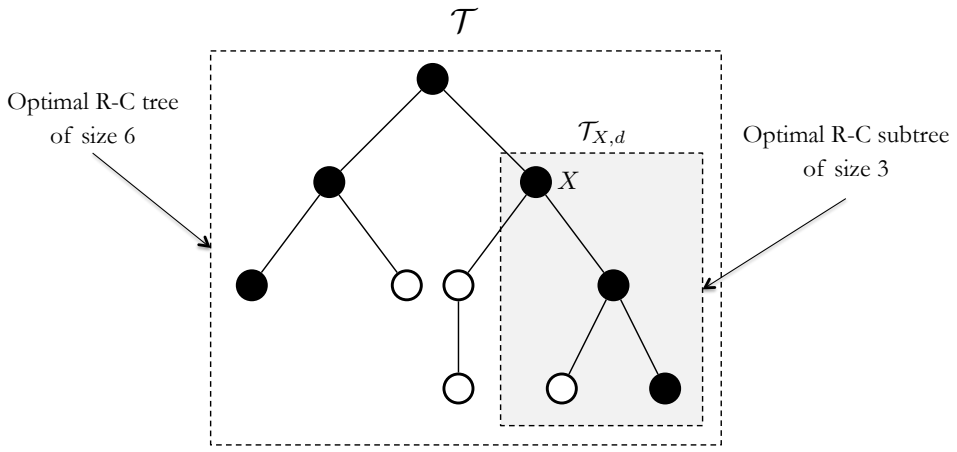

Fig. 9. Example of a nested subproblem in hierarchical groups model

complexity is $O(M^2GK^2)$, where M is the number of groups, G is the group sparsity constraint and K is the element sparsity budget.

□

APPENDIX B
DYNAMICAL PROGRAMMING FOR SOLVING THE HIERARCHICAL SIGNAL APPROXIMATION PROBLEM (12)

Here, we give the proof of Prop. VII.2. We start describing the dynamic program and then prove that its running time is linear in N, number of variables, and K, sparsity.

Proof. Problem (12) can be equivalently rephrased as the following optimization problem. Given a rooted tree $\mathcal{T}$ with each node having at most D children, a non-negative real number (weight) assigned to every node and a positive integer K, choose a subset of its nodes forming a rooted-connected subtree that maximizes the sum of weights of the chosen elements, such that the number of selected nodes does not exceed K. In our case, (12), the weight of a node is the square of the value of the component of the signal associated to that node. The proposed algorithm leverages the optimal substructure of the problem.

Nested Sub-problems. Suppose that a particular node X belongs to the optimal K-node rooted-connected subtree. Consider the subtree $\mathcal{T}_{X,d}$ obtained by choosing X, d of its children ($1 \leq d \leq D$) and all descendants of these children. Consider the set of nodes $\mathcal{S}$ consisting of all the nodes of $\mathcal{T}_{X,d}$ which are also present in the optimal K-node rooted-connected subtree. Suppose there are L nodes in $\mathcal{S}$. Then the nodes in $\mathcal{S}$ form the optimal L-node rooted-connected subtree at X, for the subgraph $\mathcal{T}_{X,d}$. See Fig. 9 for an example.

Dynamic Programming method. For every node X, we store the weight of the optimal k-node rooted-connected subtree at X, using only the nodes in the d rightmost children of X and their descendants, for each k and d such that $1 \leq k \leq K$ and $1 \leq d \leq D$. We define a function $F(X, k, d)$, to store these optimal values. We start from the leaf nodes and move upwards, for each node assessing all its subtrees from right to left, eventually covering the entire tree. At the end, the optimal value will be given by $F(\text{root}, K, D)$, that is the value of the best K-node rooted connected subtree of the root considering all its descendants.

Base Case. For every leaf node X and for all $1 \leq k \leq K$ and $1 \leq d \leq D$, we set $F(X, k, d) = \text{Weight}(X)$.

Inductive Case. By induction, for every non-leaf node X, all the F-values are known for the descendants of X. Let $X_1, X_2, \ldots X_d$ be the d children of X in the right-to-left order, where $1 \leq d \leq D$. Then, we compute the F-values of X using the following value update rules.

Value Update Rules.

1) For all $1 \leq k \leq K$

$$F(X, k, 1) = \text{Weight}(X) + F(X_1, k-1, D)\ .$$

The optimal value for choosing a k-node subtree rooted at X, when only the rightmost child X_1 is allowed, equals the weight of X itself (since X must be chosen), plus the optimal value for choosing a rooted connected subtree with $k-1$ nodes from the rightmost child X_1.

2) For all $1 \leq k \leq K$ and $1 \leq i \leq d$

$$F(X,k,i) = \max_{1 \leq \ell \leq k} \{F(X,\ell,i-1) + F(X_d, k-\ell, D)\} \ .$$

For choosing the best k-node rooted connected subtree from the rightmost d children, choose a positive integer $\ell \leq k$, pick the best ℓ-node subtree at X by including the rightmost $d-1$ children and pick the remaining $k-\ell$ nodes from the subtree of the dth child. We then take the maximum over all ℓ, $1 \leq \ell \leq k$ (since at least 1 node must be chosen from the rightmost $d-1$ nodes, this node will be the root).

3) For all $1 \leq k \leq K$ and $d \leq i \leq D$

$$F(X,k,i) = F(X,k,d) \ .$$

For convenience, when a node has only d children, where d is strictly less than D, we set F-values for cases involving more than d children equal to the value for d children.

Theorem 2. *The time complexity of the dynamic program for D-regular trees is $\mathcal{O}(KDN)$.*

Proof. The proof follows the arguments in [**?**]. Suppose there are J levels in our tree, hence the maximum number of nodes that can be selected for a sub-tree with root in level j is $S_j = 1+D+D^2+\cdots+D^{J-j}$ where $j \in \{1,2,\ldots,J\}$. At each step, the dynamic program considers selecting at most K elements to form a sub-tree. Hence for a sub-tree with root at level j, we can select a maximum number of $\mathcal{O}(l(j)) = \mathcal{O}(\min(K, \frac{D^{J+1-j}-1}{D-1})) = \mathcal{O}(\min(K, D^{J-j}))$ for $D \geq 3$ and $\mathcal{O}(l(j)) = \mathcal{O}(\min(K, D^{J+1-j}))$ for $D = 2$. Note that we do not require any computation for level J. The update step of $F(X,k,i) = \max_{1 \leq \ell \leq \min(k, l(j+1))} F(X, k-\ell, i-1) + F(X_d, \ell, D)$ then requires $\mathcal{O}(k)$ operations and for X in level j this needs to be calculated $\forall 1 \leq k \leq l(j)$ and $1 \leq i \leq D$ which requires at most $\mathcal{O}(D\sum_{k=1}^{l(j)} k) = \mathcal{O}(Dl(j)^2)$ or $\mathcal{O}(D\sum_{k=1}^{l(j)} l(j+1)) = \mathcal{O}(Dl(j)l(j+1))$ operations. By considering that at level j there at most D^{j-1} nodes, the total number of operations can be written as

$$\sum_{j=1}^{J} \mathcal{O}(Dl(j)l(j+1)D^{j-1}) \ . \tag{16}$$

Let j' be such that $K \leq D^{J-j}$ for all $j \leq j'$. We then have $j' = J - \lfloor \log_D K \rfloor$. Now assume that $D^{p-1} < K < D^p \iff p-1 < \log_D K < p$. So $j' = J - p + 1$. Hence we can break (16) into

$$
\begin{aligned}
&\sum_{j=1}^{J} \mathcal{O}(D^j \min(K^2, D^{2J-2j-1})) \\
&= \sum_{j=1}^{j'-1} \mathcal{O}(D^j K^2) + \sum_{j=j'}^{J} \mathcal{O}(D^j D^{2J-2j-1}) \\
&= \mathcal{O}\left(K^2 \sum_{j=1}^{j'-1} D^j\right) + \mathcal{O}\left(\sum_{j=j'}^{J} D^{2J-j-1}\right) \\
&= \mathcal{O}\left(K^2 D^{j'-1} \sum_{m=0}^{j'-2} D^{-m} + D^{2J-1} \sum_{j=j'}^{J} D^{-j}\right) \\
&\leq \mathcal{O}\left(K^2 \frac{D^{j'-1}}{1-D^{-1}} + D^{2J-1} \frac{D^{-j'}}{1-D^{-1}}\right) \\
&\leq \mathcal{O}\left(K \frac{D^J}{1-D^{-1}} + K \frac{D^{J-1}}{1-D^{-1}}\right) \\
&= \mathcal{O}\left(K \frac{D^J}{1-D^{-1}}\right) .
\end{aligned}
$$

For D-regular trees (with $D \geq 3$), we have $N = \frac{D^J - 1}{D-1} \approx \frac{D^J}{D-1} = \frac{D^{J-1}}{1-D^{-1}}$, so that the time complexity will be $\mathcal{O}(KDN)$. When $D = 2$, we can follow the same steps to show that the complexity is $\mathcal{O}(ND^2K)$. But for small values of D, $\mathcal{O}(ND^2K) = \mathcal{O}(NDK)$. Hence we can say that the overall complexity is $\mathcal{O}(NDK)$. □

Theorem 3. *Given the total number of nodes N of the form $\frac{D^J-1}{D-1}$ for any positive integer J and that a parent can have a maximum of D children, a regular tree is the one with maximum complexity.*

Proof. We prove this by induction. First we make two assumptions.

Assumption 1: Without loss of generality, we start from the highest level (i.e $root$) and construct a tree maximizing its complexity. We prove that this process leads to a regular tree.

Assumption 2: We assume that all children in one level (say j) connect to some parent in the upper level $(j-1)$ and not to some node in level $j-2, \ldots, 1$. If one node in level j connects to a parent in level $j-2$, then we place that node in level $j-1$ and not in j.

Base case: We start from the highest level with one $root$. When we go one level down, let $root$ connect to x children. Note that the complexity of running DP on this tree would be $x * l(j)^2 = x * (x+1)^2$ for $1 < x \leq D$. We see that this complexity is maximized when $x = D$. So we form a regular tree.

Inductive case: We have formed a regular tree with J levels starting from the root and increase the tree to level $J+1$. We cannot place any nodes anywhere above the root considering the fact that it would contradict assumption 1. We cannot place nodes anywhere in level 1 to J as all the parents have D children and cannot accommodate more. So we grow the tree downwards. Let the elements in level J have $d_1, d_2, \ldots, d_M$ children respectively, where M is the number of nodes in level J. Let N_{J-1} be the number of nodes in the tree up to level $J-1$ included and X_i be one such node: S_i is the cardinality of the subtree $\mathcal{T}_i$, up to level J, which has X_i as root and $\mathcal{D}(X_i)$ is the set of all the descendants of node X_i, including level $J+1$ (see Fig.10 for an illustration). The complexity of the

Fig. 10. Illustration of complexity computation (17)

total tree is then

$$\begin{aligned}
&\sum_{i=1}^{N_{J-1}} \mathcal{O}(D\min(K, S_i + \sum_{\substack{\ell \in J+1 \\ \ell \in D(X_i)}} d_\ell)^2) + \text{(complexity of nodes in level J)} \\
&= \sum_{i=1}^{N_{J-1}} \mathcal{O}(D\min(K, S_i + \sum_{\substack{\ell \in J+1 \\ \ell \in D(X_i)}} d_\ell)^2) + \sum_{i=1}^{M} d_i l(J)^2 \\
&= \sum_{i=1}^{N_{J-1}} \mathcal{O}(D\min(K, S_i + \sum_{\substack{\ell \in J+1 \\ \ell \in D(X_i)}} d_\ell)^2) + \sum_{i=1}^{M} d_i (d_i+1)^2 \\
&\leq \sum_{i=1}^{N_{J-1}} \mathcal{O}(D(S_i + \sum_{\substack{\ell \in J+1 \\ \ell \in D(X_i)}} d_\ell)^2) + \sum_{i=1}^{M} d_i (d_i+1)^2 \ . \qquad (17)
\end{aligned}$$

Equation (17) is increasing in d_i for all $i = 1, \ldots, M$. Since $d_i \leq D \ \forall i$ by hypothesis, we then have that the complexity is maximized when $d_i = D \ \forall i$, which corresponds to a D-regular tree.

□

Theorem 4. *The time complexity of the dynamic program for hierarchical structures is $\mathcal{O}(ND^2K)$ for non-regular trees*

Proof. For any N in the form of $\frac{D^J-1}{D-1}$, by Theorems 2 and 3 we have that the complexity of the dynamic program is $\mathcal{O}(NDK)$. For other values of N, we can write $\frac{D^J-1}{D-1} < N < \frac{D^{J+1}-1}{D-1}$. The complexity of the dynamic program for any tree with N nodes is less than complexity of any tree with $N' = \frac{D^{J+1}-1}{D-1}$, which is $\mathcal{O}(N'DK)$.

We see that in the approximation of N to N', we will make a maximum error when $N = \frac{D^J-1}{D-1} + 1$ and $N' = \frac{D^{J+1}-1}{D-1}$. Therefore, we can bound the approximation error as

$$\begin{aligned}
\frac{N'}{N} &< \frac{\frac{D^{J+1}-1}{D-1}}{\frac{D^J-1}{D-1}+1} \\
&< \frac{D^{J+1}-1}{D^J-2+D} \\
&< \frac{D^{J+1}}{D^J} \ (\text{as } D \geq 2) = D
\end{aligned}$$

Hence the maximum complexity for non regular trees is $\mathcal{O}(N'DK) \leq \mathcal{O}(ND^2K)$ □

□

ACKNOWLEDGEMENT

We thank Siddhartha Satpathi for many useful discussions and Nikhil Rao for providing the code for block signal recovery with the Latent Group Lasso approach.

REFERENCES

[1] S. Mallat, *A wavelet tour of signal processing*. Academic press, 1999.

[2] D. Donoho, "Compressed sensing," *Information Theory, IEEE Transactions on*, vol. 52, no. 4, pp. 1289–1306, 2006.

[3] E. J. Candès, "Compressive sampling," in *Proceedings oh the International Congress of Mathematicians: Madrid, August 22-30, 2006: invited lectures*, 2006, pp. 1433–1452.

[4] R. Baraniuk, "Compressive sensing," *Signal Processing Magazine, IEEE*, vol. 24, no. 4, pp. 118–121, 2007.

[5] Y. Eldar and M. Mishali, "Robust recovery of signals from a structured union of subspaces," *Information Theory, IEEE Transactions on*, vol. 55, no. 11, pp. 5302–5316, 2009.

[6] T. Blumensath and M. Davies, "Sampling theorems for signals from the union of finite-dimensional linear subspaces," *Information Theory, IEEE Transactions on*, vol. 55, no. 4, pp. 1872–1882, 2009.

[7] R. Baraniuk, V. Cevher, M. Duarte, and C. Hegde, "Model-based compressive sensing," *Information Theory, IEEE Transactions on*, vol. 56, no. 4, pp. 1982–2001, 2010.

[8] N. Rao, B. Recht, and R. Nowak, "Signal recovery in unions of subspaces with applications to compressive imaging," *arXiv preprint arXiv:1209.3079*, 2012.

[9] R. Baraniuk, V. Cevher, and M. Wakin, "Low-dimensional models for dimensionality reduction and signal recovery: A geometric perspective," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 959–971, 2010.

[10] R. Jenatton, J.-Y. Audibert, and F. Bach, "Structured variable selection with sparsity-inducing norms," *Journal of Machine Learning Research*, vol. 12, pp. 2777–2824, 2011.

[11] G. Obozinski, L. Jacob, and J. Vert, "Group lasso with overlaps: The latent group lasso approach," *arXiv preprint arXiv:1110.0413*, 2011.

[12] N. Rao, R. Nowak, S. Wright, and N. Kingsbury, "Convex approaches to model wavelet sparsity patterns," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*, 2011, pp. 1917–1920.

[13] A. Gramfort and M. Kowalski, "Improving m/eeg source localizationwith an inter-condition sparse prior," in *IEEE International Symposium on Biomedical Imaging*, 2009.

[14] R. Jenatton, A. Gramfort, V. Michel, G. Obozinski, F. Bach, and B. Thirion, "Multi-scale mining of fmri data with hierarchical structured sparsity," in *Pattern Recognition in NeuroImaging (PRNI)*, 2011.

[15] A. Subramanian, P. Tamayo, V. Mootha, S. Mukherjee, B. Ebert, M. Gillette, A. Paulovich, S. Pomeroy, T. Golub, E. Lander *et al.*, "Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 43, pp. 15 545–15 550, 2005.

[16] F. Rapaport, E. Barillot, and J. Vert, "Classification of arraycgh data using fused svm," *Bioinformatics*, vol. 24, no. 13, pp. i375–i382, 2008.

[17] H. Zhou, M. Sehl, J. Sinsheimer, and K. Lange, "Association screening of common and rare genetic variants by penalized regression," *Bioinformatics*, vol. 26, no. 19, p. 2375, 2010.

[18] V. Cevher, C. Hegde, M. Duarte, and R. Baraniuk, "Sparse signal recovery using markov random fields," in *NIPS*, 2009.

[19] V. Michel, A. Gramfort, G. Varoquaux, E. Eger, and B. Thirion, "Total variation regularization for fmri-based prediction of behavior," *Medical Imaging, IEEE Transactions on*, vol. 30, no. 7, pp. 1328 –1340, july 2011.

[20] M. Stojnic, F. Parvaresh, and B. Hassibi, "On the reconstruction of block-sparse signals with an optimal number of measurements," *Signal Processing, IEEE Transactions on*, vol. 57, no. 8, pp. 3075–3085, 2009.

[21] J. Huang, T. Zhang, and D. Metaxas, "Learning with structured sparsity," *The Journal of Machine Learning Research*, vol. 12, pp. 3371–3412, 2011.

[22] L. Jacob, G. Obozinski, and J. Vert, "Group lasso with overlap and graph lasso," in *International Conference on Machine Learning*, 2009.

[23] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.

[24] P. Zhao, G. Rocha, and B. Yu, "The composite absolute penalties family for grouped and hierarchical variable selection," *The Annals of Statistics*, vol. 37, no. 6A, pp. 3468–3497, 2009.

[25] L. Wolsey and G. Nemhauser, *Integer and Combinatorial Optimization*. Wiley, 1999.

[26] A. Kyrillidis and V. Cevher, "Combinatorial selection and least absolute shrinkage via the clash algorithm," in *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, 2012.

[27] G. Nemhauser, L. Wolsey, and M. Fisher, "An analysis of approximations for maximizing submodular set functions — I," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[28] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[29] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, "A sparse-group lasso," *Journal of Computational and Graphical Statistics*, vol. 10, 2012.

[30] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach, "Proximal methods for hierarchical sparse coding," *Journal of Machine Learning Reasearch*, vol. 12, pp. 2297–2334, 2011.

Luca Baldassarre received his M.Sc. in Physics in 2006 and his Ph.D. in Machine Learning in 2010 at the University of Genoa, Italy. He then joined the Computer Science Department of University College London, UK, to work with Prof. Massimiliano Pontil on structured sparsity models for machine learning and convex optimization. Currently he is with the LIONS of Prof. Volkan Cevher at the Ecole Polytechnique Federale de Lausanne, Switzerland. His research interests include model-based machine learning and compressive sensing and optimization.

Nirav Bhan is currently an undergraduate student at the Indian Institute of Technology-Bombay, India. He is expected to graduate in August 2013, with a B.Tech degree in Electrical Engineering and a minor in Computer Science. He has worked as a research intern at EPFL, Switzerland, during the period of May to July, 2012. His interests are in optimization, machine learning and signal processing.

Volkan Cevher received his BSc degree (valedictorian) in Electrical Engineering from Bilkent University in 1999, and his PhD degree in Electrical and Computer Engineering from Georgia Institute of Technology in 2005. He held Research Scientist positions at University of Maryland, College Park during 2006-2007 and at Rice University during 2008-2009. Currently, he is an Assistant Professor at Ecole Polytechnique Federale de Lausanne and a Faculty Fellow at Rice University. He coauthored (with C. Hegde and M. Duarte) the Best Student Paper at the 2009 International Workshop on Signal Processing with Adaptive Sparse Structured Representations (SPARS). In 2011, he received an ERC Junior award. His research interests include signal processing theory, machine learning, graphical models, and information theory.

Anastasios Kyrillidis received his 5-year diploma and M.Sc. in Electronic and Computer Engineering from Technical University of Crete in 2008 and 2010, respectively. Currently, he is a graduate student at Ecole Polytechnique Federale de Lausanne. His research interests includes machine learning, high-dimensional data analysis and mining.