Geometric group sparsity in image analysis

Alhussein Fawzi

Master’s thesis
Ecole Polytechnique Federale de Lausanne

Supervisor: Prof. Pascal Frossard

2011-2012

Abstract

In this project, we consider the sparse approximation problem in redundant dictionaries under structured sparsity constraints. We propose a new framework for modeling structured sparsity that allows encoding structures representing meaningful objects or parts of objects in an image. The main challenge comes from the fact that the orientation, position and size of the structures is arbitrary since an object has the same ‘semantic meaning’ as its geometric transformed versions. We propose to model such transformation-invariant structures using graphs where edges encode the relative transformations between the components of an object. Unlike other related approaches, this model allows us to define a whole class of structures by geometric transformations from a template structure. We then propose two greedy algorithms that use this framework to solve the structured sparse approximation problem and we show theoretically and experimentally the benefits of these methods over an approach that uses standard sparsity without any structural priors. We give illustrative examples of these algorithms in pattern recognition and denoising.
B Proofs of the GOMP algorithm
B.1 Intrastructure coherence ........................................... 40
B.2 Proof of theorem 3 .................................................. 40
B.3 Proof of Proposition 4 .............................................. 42
B.4 Proof of Proposition 5 .............................................. 43

C Pseudocode of DETECTSTRUCTURE and SPIKESNEIGHBOURS functions 44
List of Figures

1 Geometrical interpretation of sparse recovery problem ........................................... 7
2 Comparison of $\ell_2$, $\ell_1$ and $\ell_1/\ell_2$ norm balls ........................................ 9
3 Effect of regularizing with the $\ell_1/\ell_2$ norm for overlapping groups ................... 10
4 Benefit of group lasso in denoising ................................................................. 11
5 Gaussian dictionary ............................................................................................. 13
6 Approximation of a face with Gaussian atoms ....................................................... 14
7 Semantically similar sets ....................................................................................... 14
8 Complete graph and tree semantic structures ...................................................... 16
9 Decomposable sets ............................................................................................... 17
10 Comparison between OMP and GOMP .............................................................. 21
11 Upper bound on the residual norm ...................................................................... 24
12 Support success recovery rate for GOMP with coefficients: influence of the coefficients on the performance ................................................................. 27
13 Rectangle and arrow semantic structures ............................................................ 27
14 Performance of GOMP ......................................................................................... 28
15 Handwritten digits ............................................................................................... 28
16 Result of denoising handwritten digits with GOMP .............................................. 29
17 Evolution of the structure promoting function ..................................................... 32
18 Scheme block of the PSP algorithm ...................................................................... 33
19 Denoising example using the Progressive Structure Promoting algorithm .......... 34
20 Denoising with several semantic structures in the image ...................................... 35
21 Denoising using OMP, GOMP and PSP .............................................................. 36
22 Diagram representing the approximate decision criteria ...................................... 37

List of Tables

1 Transformations formulas ....................................................................................... 12
2 Performance of GOMP as a classifier .................................................................... 29
1 Introduction

Among the many ways to model signals, a recent and popular approach employs sparse and redundant representations. This model suggests that a high dimensional sparse data source contains relatively little information compared to its ambient dimension. That is, a sparse signal can be represented by a vector with few non-zero coefficients. The process of finding a reliable sparse representation of an arbitrary signal is a well-studied problem under the name of sparse approximation with applications to compression, denoising and classification to name a few. Solving the sparse approximation problem under the sole prior of sparsity has attracted a lot of attention in the past few years and has led to many efficient algorithms. However, in many practical scenarios, we also have information on the structure of the signal, which is merely disregarded by typical pure sparsity priors. Encoding more sophisticated prior knowledge that takes into account structured sparsity can improve both the performance and interpretability of sparse recovery algorithms. Much effort has been spent in recent years in designing structures that would allow encoding successfully the notion of structured sparsity priors. Several models have been considered and have led to successful applications in bioinformatics [1], computer vision [2, 3], image denoising [4], topic modeling and dictionary learning [5]. For example, the hierarchical model has been considered in wavelet based denoising since it allows encoding the dependencies between the coefficients across different scales. Similarly, graphical models have been proposed to encode the dependencies across different components of an image [1, 6, 3].

In this project, we consider structures that typically represent meaningful objects or parts of objects in images. We thus refer to this approach as semantic structured sparsity. Because a structure and its geometric transformed versions often refer to the same ‘semantic object’, the major constraint that semantic structures should satisfy is to be transformation invariant. That is, the definition of one structure referring to an object has to encompass every other object that differs from the original one with a transformation. In this approach, we need not define a structure for each of the transformed versions. To the best of our knowledge, there has been no proposed structured sparsity model that exploits this invariance. Using semantic structures does not only lead to better performance compared to other types of structures, but also provides a more natural and elegant way to define structures since it offers a more accurate description of the perception process. Our first contribution consists in establishing a new framework for modeling semantic structures. Then, we propose two greedy algorithms that use this novel representation to solve the sparse approximation problem under semantic structured sparsity constraints. We perform a theoretical analysis and conduct experiments in illustrative problems in pattern recognition and image denoising to show the benefits of our approach over a method that does not exploit structures in sparse approximation. Even though our approach is not yet ready to be applied to ‘natural signals’, the proposed structure model as well as the algorithms show that pursuing on this work might lead to promising results in image processing applications.

The report is organized as follows. Section 2 gives an overview of the previous work on structured and unstructured sparsity models and signal recovery. We emphasize on the fact that proposed methods do not allow to capture efficiently the notion of transformation invariant structures. In section 3, we give a definition of semantic structures and justify that such a definition takes into account transformation invariance. Armed with these structures, we formulate the structured sparse recovery and approximation problems in section 4. In section 5, we propose an iterative algorithm (GOMP) that extends Orthogonal Matching Pursuit to the selection of semantic structures and give theoretical results on the performance of the algorithm. Experiments are carried to show the benefits of GOMP and we discuss the shortcomings of the algorithm. In section 6, we propose an alternative approach (PSP) that selects iteratively atoms and tries to fit them into structures. We conclude in section 7 and discuss future work.
2 A quick tour of unstructured and structured sparse signal recovery

2.1 Sparse models

2.1.1 Sparse recovery and approximation problems

Low dimensional signal models have recently received a lot of attention from the signal processing community. This interest came as a natural response to the need of acquiring and processing increasingly large volumes of data. The most popular of these models is perhaps the sparse model which states that the intrinsic dimensionality of signals is often quite small with respect to their ambient dimensionality. Mathematically, a discrete signal \( s \in \mathbb{R}^N \) is said to be \( S \)-sparse if it contains at most \( S \) non zero elements. Let \( \Sigma_S \) be the set of \( S \)-sparse signals:

\[
\Sigma_S = \{ x \in \mathbb{R}^N : \| x \|_0 \leq S \},
\]

where \( \| \cdot \|_0 \) is the \( \ell_0 \) pseudo norm defined by: \( \| x \|_0 = | \{ x_i : x_i \neq 0, i \in \{1, \ldots, N\} \} | \). The sparsity model consists in restricting the space of valid signals to \( \Sigma_S \). However, \( \Sigma_S \) fails to model natural signals as relatively few sparse signals exist; rather, signals tend to admit a sparse representation in some dictionary. A dictionary \( \mathcal{D} = \{ \phi_1, \ldots, \phi_K \} \) is a collection of unit norm vectors \( \phi_i \in \mathbb{R}^N \) (called atoms) with no additional constraints on the number of vectors or the orthogonality between them. In practice, the dictionary size \( K \) is often taken to be larger than the dimension of the signal \( N \). Considering dictionaries instead of orthonormal bases dramatically increases the class of signals that can be sparsely represented. Let \( \Sigma_{S,D} \) be the space of signals that admit a \( S \)-sparse representation in \( \mathcal{D} \):

\[
\Sigma_{S,D} = \{ Dx : x \in \Sigma_S \},
\]

where \( D = [\phi_1, \ldots, \phi_K] \) is the matrix notation for \( \mathcal{D} \). The sparsity model reduces the degrees of freedom of admissible signals from \( K \) to \( S \) for a fixed dictionary \( D \), which represents a huge gain if \( S \ll K \). Geometrically speaking, a sparsity model is seen as the union of \( \binom{K}{S} \) linear subspaces, each of dimension \( S \). That is, \( \Sigma_S = \cup_{i=1}^{\binom{K}{S}} \{ x_\mathcal{A}_i = 0 \} \) where \( \mathcal{A}_i = \{ x \in \mathbb{R}^K : x_i = 0 \} \). This interpretation has important algorithmic consequences in solving the sparse recovery problem (paragraph 2.1.3), defined as follows.

Given a signal \( s \) that admits a \( S \)-sparse representation in a given dictionary \( \mathcal{D} \), the sparse recovery problem \( SR(\mathcal{D}, S) \) consists in finding \( x \in \Sigma_S \) verifying:

\[
s = Dx. \tag{1}
\]

Note that Eq.(1) typically has several solutions as \( D \) is a redundant dictionary. The sparsity constraint specifies that we should look for the vector \( x \in \mathbb{R}^K \) which has at most \( S \) non zero components. In some cases, this constraint is sufficient to guarantee the uniqueness of the solution. The following theorem, featured in [7], gives an upper bound on \( S \) in order to guarantee a unique representation of \( s \) in \( D \).

**Theorem 1** ([7]). Let \( \text{spark}(D) \) be the minimum number of columns in \( D \) that form a linearly dependent family. All sparse representations over \( S \) atoms in \( D \) are unique if and only if \( S < \frac{1}{2} \text{spark}(D) \).

**Proof.** I include the proof of this important theorem as it provides an insight on the appearance of the spark. Suppose first that \( S < \frac{1}{2} \text{spark}(D) \) and that there exists \( x, x' \) both in \( \Sigma_S \) such that \( Dx = Dx' \). Then, \( D(x-x') = 0 \) and \( x-x' \in \text{Ker}(D) \), where \( \text{Ker}(D) \) denotes the kernel of \( D \). As \( x \) and \( x' \) are both \( S \)-sparse, \( x-x' \in \Sigma_{2S} \cap \text{Ker}(D) \). Thus, there exists a linear combination of at most \( 2S \) atoms of \( D \) that is in the kernel of \( D \). The condition \( 2S < \text{spark}(D) \) guarantees that this linear combination is necessarily all zero, which results in \( x = x' \) and hence the uniqueness result. Conversely, suppose that: if \( x, x' \) belong to \( \Sigma_S \) and \( Dx = Dx' \) then \( x = x' \) and that \( \text{spark}(D) \leq 2S \). This last inequality guarantees the existence of a vector \( y \in \Sigma_{2S} \) such that \( Dy = 0 \). As \( y \) is \( 2S \)-sparse it is possible to write \( y = x-x' \) with \( x \) and \( x' \) in \( \Sigma_S \). This contradicts the uniqueness assumption. \( \square \)

We suppose in the sequel, for simplicity, that \( s \) and \( D \) satisfy Theorem 1 and thus that Eq.(1) has a unique solution in \( \Sigma_S \).

In practice very few signals have an exact sparse representation in a redundant dictionary [8]. In this case, the sparse recovery problem is replaced by the sparse approximation problem \( SA(\mathcal{D}, S) \) formulated as follows:

\[
SA(\mathcal{D}, S) : \text{Find } x \in \Sigma_S \text{ that minimizes } \| s - Dx \|_2.
\]

In words, this problem corresponds to finding the sparse vector which provides the best approximant of \( s \).

Note that in the problems that we have defined, the dictionary was always fixed in advance. It is possible to consider an alternative problem where the dictionary (as well as the sparse coefficient vector) are the unknowns [9]. The main advantage of such an approach is that an optimized dictionary is found for the particular signal that we are considering, thus yielding to potentially sparser representations than for the fixed case. On the other hand, adding the dictionary as an unknown makes the problem more complex and thus leads to methods that are computationally expensive.
2.1.2 Sparsity promoting norms

The most natural sparsity promoting norm is undoubtely the $\ell_0$ norm, which we have used in the definition of the sparsity constraints in the sparse recovery/approximation problems. However, as the $\ell_0$ norm is non convex, the problems $SR(\mathcal{D}, S)$ and $SA(\mathcal{D}, S)$ become very difficult to solve exactly. In the next paragraph, we describe a greedy algorithm which provides an approximate solution to such problems. Another alternative is to replace the $\| \cdot \|_0$ norm in the formulation of the problems with its convex approximation $\| \cdot \|_1$. By doing so, we transform a computationally intractable problem into a convex one, which is relatively easy to solve. Formally, the exact sparse recovery problem becomes:

$$\min \|x\|_1 \text{ s.t. } s = Dx,$$

and sparse approximation problem becomes:

$$\min \|x\|_1 \text{ s.t. } \|s - Dx\|_2 \leq \epsilon.$$  \hspace{1cm} (3)

It remains to show that the $\ell_1$ norm is a sparsity promoting norm. We provide the following geometric interpretation. For simplicity, let us consider the sparse recovery problem in $\mathbb{R}^2$ and denote by $A$ the set of all possible solutions: $A = \{x: s = Dx\}$. Solving the relaxed sparse recovery problem in Eq. (2) consists in computing the closest point in $A$ to the origin, where the distance is defined using the $\ell_1$ norm. To do so, we can imagine growing an $\ell_1$ ball centered at the origin 0 until it intersects with $A$. The resulting solution thus generally occurs on a singularity of the ball. An example is given in Fig. 1. In this case, the solution $\hat{x}$ is 1-sparse and indeed occurs on a singularity of the $\ell_1$ ball. Fig. 1 also illustrates the geometry of Eq.(2) with the $\ell_1$ norm replaced by $\ell_2$ and $\ell_\infty$ norms. The $\ell_2$ and $\ell_\infty$ norms yield vectors which tend to be evenly spread among the two coefficients. While the $\ell_2$ norm treats all directions similarly and hence results generally in a non sparse vector, the $\ell_\infty$ norm ball has singularities located at non sparse positions in the plane. The singularities of the considered norm ball thus control somehow the sparsity of the solution to the sparse approximation/recovery problem. This geometric interpretation will be of particular importance in structured sparsity inducing norms in section 2.2.

![Figure 1: Solution to the problem: $\min \|x\|_p \text{ s.t. } s = Dx$ for $p = 1, 2, \infty$ in $\mathbb{R}^2$. $\hat{x}$, the solution of this problem, is represented in each case. [10]](image)

2.1.3 Algorithms for sparse signals recovery

As we have seen previously, there are basically two ways to approach the sparse signal recovery/approximation problem. In the first one, the intuitive $\ell_0$ norm is used to model the sparsity constraint, while in the second, the $\ell_1$ norm is considered for computational efficiency reasons. In this paragraph, we review briefly some of the techniques used to solve both approaches.

Greedy algorithms

The major difficulty of $SR(\mathcal{D}, S)$ and $SA(\mathcal{D}, S)$ problems lies in the optimal selection of the support of $x$ (i.e., $\{i : x_i \neq 0\}$), or equivalently, the optimal choice of the linear subspace in which $x$ is lying. Indeed, once the correct support $\Omega$ is chosen, the non zero coefficients $x|_{\Omega}$ could be obtained by finding the least squares solution to the equation $D_{|\Omega}x|_{\Omega} = s$ where $D_{|\Omega}$ is a matrix of dimension $N \times |\Omega|$ whose columns are equal to the columns of $D$ indexed by $\Omega$. This can be done via the calculation of the pseudo inverse of $D_{|\Omega}$ that we shall denote $D_{|\Omega}^{-1}$. As pointed out in section 2.1, there exists as many as $\binom{K}{s}$ linear subspaces in which $x$ potentially lies. Hence, testing all sparsity patterns is unrealistic. The problem of selecting the best subspace is provably NP hard [11], thus, we cannot hope to perform much better than a full search to obtain an optimal solution. We highlight here one of the most widely used greedy iterative algorithm that constructs an approximation for the sparse recovery/approximation problem. Orthogonal Matching Pursuit (OMP) [12] is an algorithm which identifies and selects one by one the atoms in $\mathcal{D}$ which best match the data. More precisely, at each iteration, the algorithm correlates the atoms in $\mathcal{D}$ with the signal residual, which is obtained by subtracting the contribution of a partial estimate of the signal from the original vector. The atom achieving
The algorithm is formally defined in Algorithm 1. Note that other greedy algorithms solving the sparse recovery/approximation problems exist: notably, the family of iterated thresholding algorithms have recently attracted a considerable attention due to its simplicity and good practical results. For an overview of these algorithms, refer to [13].

Algorithm 1 Orthogonal Matching Pursuit (OMP)

**Input:** signal \( s \), sparsity \( S \), dictionary \( \mathcal{D} \).

**Output:** sparse representation \( \hat{x} \).

1. **Initialization:** \( r_0 \leftarrow s \), \( \Omega_0 \leftarrow \emptyset \), \( \hat{x}_0 \leftarrow 0 \).
2. While \( 1 \leq i \leq S \), do:
   1. **Selection step:** Find \( k_i \):
      
      \[
      k_i \leftarrow \arg\max_{k \in \{1, \ldots, K\}} |\langle r_{i-1}, \phi_k \rangle| 
      \]
   2. **Update step:** Update the support, signal estimate and residual:
      
      \[
      \Omega_i \leftarrow \Omega_{i-1} \cup \{k_i\} \\
      \hat{x}_i|_{\Omega_i} \leftarrow D|_{\Omega_S}^\dagger \hat{x}_i|_{\Omega_T} \leftarrow 0 \\
      r_i \leftarrow s - D\hat{x}_i 
      \]
3. Return the sparse representation: \( \hat{x} \leftarrow \hat{x}_S \)

The \( \ell_1 \) minimization approach

The \( \ell_1 \) minimization approach provides a robust framework for recovering sparse signals. We have discussed in section 2.1.2 that the use of the \( \ell_1 \) norm to promote sparsity has interesting theoretical foundations. Besides, the formulations in Eq. (2) and Eq. (3) are convex optimization problems for which efficient numerical solvers exist. The majority of the \( \ell_1 \) minimization algorithms in the literature (see for example [14] for a review) have however considered an unconstrained version of the sparse approximation problem, namely:

\[
\min_{x \in \mathbb{R}^K} \{\|s - Dx\|_2 + \lambda \|x\|_1\}, \tag{4}
\]

where \( \lambda \) is the regularization parameter whose value governs the sparsity of the solution. Large values of \( \lambda \) produce sparser results. For a fixed \( \lambda \), there exists some \( \epsilon \) such that the unconstrained minimization problem Eq.(4) and constrained problem Eq.(3) are equivalent. However, the correspondence relation between the two equivalent problems is generally unknown a priori, which makes \( \lambda \) rather difficult to choose. Note that efficient solvers [15] also exist for the constrained formulation which has a more natural parametrization, as \( \epsilon \) is generally determined by the quantization/noise level.

2.2 Structured sparsity models

In this section, we consider models that encode an additional structural prior information on top of sparsity. That is, we suppose that vectors we are trying to estimate are not only sparse, but also form non zero patterns with a specific structure. Structured sparsity models have received much attention recently and different ways of encoding the structure were considered. For instance, the most natural form of structured sparsity is perhaps group sparsity that models the fact that variables in the same group tend to be zero or non zero simultaneously.

As for standard sparsity, there are basically two main approaches to formulating recovery problems under a structured sparsity priors. The first deals with nonconvex and combinatorial formulations that are generally computationally intractable and are thus addressed with greedy algorithms. The second considers convex relaxations solved with convex optimization methods.

2.2.1 Nonconvex models

Recall from section 2.1 that a \( S \)-sparse signal \( s \) lives in \( \Sigma_S \), which is a union of \( \binom{K}{S} \) subspaces of dimension \( S \). In [16], Baraniuk et. al. introduce a structure model that constraints the sparsity pattern of the solution to be in a predefined set \( \Sigma^\text{structure}_S \) included in \( \Sigma_S \). Given this a priori knowledge, the authors adapt the greedy CoSaMP recovery algorithm [17] to take into account the structure and address the following nonconvex structured sparse approximation problem:

\[
\min_{x \in \mathbb{R}^K} \|s - Dx\|_2 \text{ s.t. } x \in \Sigma^\text{structure}_S. 
\]
In the framework of Compressed Sensing [10], this additional structural prior significantly reduces the number of measurements needed for the signal $s$ in order to have a correct signal recovery.

Related to the above method, [2] focuses on a formulation of the structure inspired from information theory. In this approach, a coding complexity is first defined on the blocks (i.e., subsets of $\{1, \ldots, K\}$) which in turn automatically defines a coding complexity $c_l$ on vectors. The structure is taken into account through the choice of the coding complexity imposed on the blocks. As a result, vectors that can be expressed as a union of low-complexity blocks are promoted while vectors whose active components lie on high-complexity blocks are ignored. The optimization problem is very similar to $SA(G, S)$ where the $\ell_0$ norm is replaced with the structure promoting term $c_l$:

$$
\min \|s - Dx\|_2 \text{ s.t. } c_l(x) \leq S.
$$

The authors address this optimization problem using a greedy method similar to matching pursuit. This approach is interesting in the sense that the user is free to choose the coding complexity function and thus results in a ‘personalized’ structure promoting term.

### 2.2.2 Convex structured sparse inducing models

In the same way where sparse vectors are promoted using the $\ell_1$ norm, the idea is to promote structured sparsity through a well designed convex norm. The general scheme of convex approaches to structured sparsity consists in solving the following optimization problem:

$$
\min \|s - Dx\|_2 + \lambda N(x),
$$

where $N$ is a well-chosen norm that encourages structured sparse vectors. A popular choice for $N(x)$ is the following:

$$
N(x) = \sum_{G \in G} \|x_{|G}\|_2,
$$

where $G$ denotes the set of predefined groups (i.e., $G \subseteq 2^{\{1, \ldots, K\}}$) and $x_{|G}$ is the restriction of $x$ on $G$. $N$ is usually referred to as the $\ell_1/\ell_2$ norm or the group lasso penalty [18] in the statistics literature. Note that when $G = \{\{1\}, \ldots, \{K\}\}$, $N$ is the $\ell_1$ norm. When the different groups in $G$ do not overlap (i.e., no two different groups include the same variable), regularizing with the $N$ norm leads variables in the same group to be selected or set to zero together. Figure 2 gives a geometric interpretation in $\mathbb{R}^3$. Recall from section 2.1.2 that the $\ell_1$ norm treats the three coordinate directions differently and thus promotes sparsity in individual coefficients (notice the pyramidal shape of its norm ball in $\mathbb{R}^3$, diamond shape in $\mathbb{R}^2$). On the contrary, the $\ell_2$ norm is an isotropic, rounded ball that does not favor any particular direction and thus does not induce sparsity. The $\ell_1/\ell_2$ norm is naturally in between: it encourages sparsity but only at the group level (notice for example the absence of singularities at points $(1,0,0)$ and $(0,1,0)$).

![Figure 2: Comparison of $\ell_2$, $\ell_1$ and $\ell_1/\ell_2$ norm balls. The singular points lying on these balls provides information on the (structured) sparsity inducing behaviour of the norm. The geometrical figures are taken from [19].](image)

When there exists a natural grouping that forms a partition of $\{1, \ldots, K\}$ (i.e., the groups do not overlap), explicitly encoding the structure by using the group lasso norm (Eq. (7)) instead of the $\ell_1$ norm leads to provably better prediction performance and interpretability in statistical learning [20, 18]. The case where groups do not overlap is however too idealistic especially in image processing applications where variables represent geometric atoms. Indeed, this constraint is too demanding as potential groups will inevitably contain atoms...
which are shared between different groups. Besides, it turns out that studying the behaviour of the $\ell_1/\ell_2$ norm in the case where groups overlap allow to encode quite interesting forms of structured sparsity.

In the case where $G$ contains groups of variables that overlap, if the groups span the entire set of variables, $N(x)$ is still a norm and the sparsity inducing behaviour of $N(x)$ is the same: $N(x)$ will force entire groups to be set to zero together. While in the non overlapping case this implied immediately that supports (i.e., non zero patterns) are contained in some union of groups, this is no more true in the overlapping case. Indeed, in the overlapping case, $N(x)$ leads to the estimation of vectors whose support are included in the intersection of a certain number of complements of groups. For a geometrical interpretation, refer to [19] or [1]. While this may be interesting for a number of applications like encoding hierarchical structures [5, 4], we are interested here in penalties which induce the opposite effect: that the support of the estimated vector is in a union of groups.

For that purpose, Jacob et. al. propose in [1] the following norm:

$$N'(x) = \min_{v=(v_g)_{g \in G} \in \mathbb{R}^{p \times |G|}} \sum_{g \in G} \|v^g\|_2 \text{ s.t } x = \sum_{g \in G} v^g \text{ and } \forall g, \text{Supp}(v^g) \subseteq g. \quad (8)$$

Intuitively, in this formulation, $x$ is decomposed into a sum of latent vectors whose supports are included in each group. Applying the $\ell_1/\ell_2$ norm on the latent vectors promotes putting to zero some entire vectors $v^g$. This results in a vector $x$ which has its support in the union of some groups. Notice that the main difference between the $N$ and $N'$ is that $N$ applies the $\ell_1/\ell_2$ norm directly on the vector $x$, whereas $N'$ applies it on the decomposition of $x$. Figure 3 illustrates this fact.

A major problem with this approach is however its computational efficiency. In order to implement such a method, one needs to duplicate variables in $D$ as many times as they appear in the different groups. This results in a new matrix $D'$ of size $N \times \sum_{g \in G} |g|$. If groups have important overlaps, this might significantly increase the dimensionality of the problem.

### 2.3 Concluding example

We conclude this section with a denoising example which illustrates some of the points we have previously mentioned. Consider the image shown in Figure 4(a) and suppose that, by some means (transmission errors for example), the image is corrupted with white gaussian noise (Figure 4(b)). Our task is to denoise the image. The first assumption we do in order to tackle the problem is the sparsity assumption: we suppose that the original image is sparse in the orthonormal Dirac basis (assume that this prior knowledge was given to us by some oracle). We then address the denoising by solving the sparse approximation problem (Eq. 4) where $D = I$ (identity), $s$ is the noisy signal and $x$ is the estimate. SPGL1 solver [21] is used and the result is shown in Figure 4(c). Different regularization parameters $\lambda$ were tested and the one leading to best image quality was chosen.

Suppose now that we have additional information on the unknown image: its main components are letters ‘a’, ‘b’ and ‘c’. However, we do not have any prior knowledge on i) the number of letters composing the image, ii) the position of these letters in the image. As the positions are unknown, one has to consider a group at each potential position. Using this straight-forward design of groups, we are in the case where the support of the image is a union of some groups, and where groups do overlap (consider for example groups at neighbouring positions). That is why we use the penalty of Eq. (8). We implemented this penalty norm and the result is shown in Figure 4(d) (as before, the best regularization parameter $\lambda$ is chosen). It is clear that the
structural regularizer outperforms the regularization with the $\ell_1$ sparsity norm. As opposed to the $\ell_1$ penalty function which results in images that lack a regular pattern, the norm in Eq. (8) leads to estimations which are constructed from the predefined structures.

![Figure 4: Denoising example.](image)

Even though the obtained result with the structural prior is satisfactory, the dimension of the optimization problem is quite big and thus makes the approach unscalable to bigger images and larger number of semantic groups (i.e., letters a, b, c). The high dimensionality comes from two reasons: i) Different groups are defined for each position. In our example, the total number of groups that we defined equals 1200 ($3 \times (20 \times 20)$, where 3 is the number of groups, and $20 \times 20$ is the number of pixels). This number can grow very large in practical examples which results in computational intractability. ii) As mentioned before, the implementation of the regularizer in Eq. (8) necessitates the duplication of the variables as many times as they overlap in different groups. In our example, this leads to $\sim 10^5$ variables (i.e., number of columns of $D$) instead of just $20 \times 20 = 400$. We are aware that in [22], the authors propose an implementation method without duplication. Nevertheless, we did not notice a huge performance gain with respect to the traditional implementation when we tested it on images. Furthermore, even if this method avoided the duplication, the problem posed by i) would remain.

In this work, we are interested in defining groups which have a semantic meaning. That is, semantic structures are defined in terms of the information they represent. Because transformed versions of semantic groups often refer to the same information, the intrinsic definition of semantic groups must obey transformation invariance constraints. Typically, in the above example, there are three semantic groups: 'a', 'b' and 'c'. These semantic groups are obviously independent of the position at which they appear in the image. Thus, in this case, semantic groups should be translation-invariant. Note that by defining only 3 groups instead of 1200 groups in the previous example, we substantially simplify the problem and thus reduce its dimension and complexity. Besides, defining the groups in such a way is more natural and elegant in image processing applications as it offers a more accurate description of the visual perception process.

To the best of our knowledge, no such definition of groups was previously made. In fact, most of the work related to group sparsity in the literature is done in the context of statistical learning. Our work is different since we target more image processing applications in which transformation-invariance is crucial. This naturally leads to the definition of semantic structures. In the next section, we propose a representation of semantic structure which obeys transformation invariance constraints.
3 Representation of semantic structures

3.1 Geometric structured dictionaries

Let \( \mathcal{D} = \{ \phi_\gamma : \gamma \in \mathcal{P} \} \) be a geometric, structured and possibly redundant dictionary that spans the input space. Geometric means that the indices \( \gamma \) have some geometric interpretation. For instance, \( \gamma \) can be a translation and/or rotation parameter. A structured dictionary is a dictionary that is constructed by applying geometric transformations on a mother function \( \phi \). The atoms of a geometric structured dictionary can be seen as building blocks that allow us to construct rich patterns or meaningful objects.

3.1.1 Geometric parameters

In a 2D image, the operations we want to apply on the mother function are typically translations in the image plane by \( \mathbf{b} = (b_1, b_2) \in \mathbb{R}^2 \), isotropic scaling by a parameter \( a > 0 \), and rotation \( \theta \in [0, 2\pi) \) around the origin. Composing these transformations together leads to a transformation vector \( \gamma = (\mathbf{b}, a, \theta) \). Table 1 summarizes the action on the plane for the translation, rotation, scaling and combination of these. \( r_\theta \) denotes the rotation matrix: \( r_\theta = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \).

<table>
<thead>
<tr>
<th>( \gamma )</th>
<th>Translation ( \mathbf{b} = (b_1, b_2) )</th>
<th>Rotation ( \theta )</th>
<th>Scale ( a )</th>
<th>Composition of translation, rotation and scale ( (b, a, \theta) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tilde{\gamma}(x, y) = )</td>
<td>( (x + b_1, y + b_2) )</td>
<td>( r_\theta ) ( \begin{bmatrix} x \ y \end{bmatrix} )</td>
<td>( (ax, ay) )</td>
<td>( ar_\theta ) ( \begin{bmatrix} x + b_1 \ y + b_2 \end{bmatrix} )</td>
</tr>
<tr>
<td>( \tilde{\gamma}^{-1}(x, y) = )</td>
<td>( (x - b_1, y - b_1) )</td>
<td>( r_{-\theta} ) ( \begin{bmatrix} x \ y \end{bmatrix} )</td>
<td>( (\frac{x}{a}, \frac{y}{a}) )</td>
<td>( \frac{r_{-\theta}}{a} ) ( \begin{bmatrix} x - b_1 \ y - b_2 \end{bmatrix} )</td>
</tr>
</tbody>
</table>

Table 1: Action on the plane of translation, rotation, scaling and their combination. The transformation is parametrized with a vector \( \gamma \) and \( \tilde{\gamma} \) denotes its associated transformation. The inverse transformation, \( \tilde{\gamma}^{-1} \) is also showed as it is used in Eq.(9).

Up to a normalizing factor, any atom \( \phi_\gamma \) in \( \mathcal{D} \) is simply obtained from the mother function \( \phi \) using the following change of coordinates:

\[
\phi_\gamma(x, y) = \phi(\tilde{\gamma}^{-1}(x, y)),
\]

where \( \tilde{\gamma} \) denotes the transformation of the plane associated to \( \gamma \). The calculated atom \( \phi_\gamma \) is an atom centered at \( \tilde{\mathbf{b}} = (b_1, b_2) \), rotated with \( \theta \) around its center and scaled by \( a \).

Together, translation, rotation and isotropic scaling operations form a four-parameter group, called the similitude group of the plane and denoted by \( \text{SIM}(2) \) [23]. The composition rule of \( \text{SIM}(2) \) is the following:

\[
(\tilde{\mathbf{b}}, a, \theta) \circ (\tilde{\mathbf{b}}, a, \theta) = (\tilde{\mathbf{b}} + a_0 r_\theta \tilde{\mathbf{b}}, a_0 a, \theta_0 + \theta).
\]

for any \((\tilde{\mathbf{b}}, a, \theta) \) and \((\tilde{\mathbf{b}}, a, \theta) \) in \( \mathcal{P} \). Applying a transformation \( \alpha \) on an atom \( \phi_\gamma \) results in the atom \( \phi_{\alpha \circ \gamma} \) where \( \alpha \circ \gamma \) is the composition of \( \alpha \) and \( \gamma \) using Eq. (10). We define the function \( T \) which maps any two elements \( \gamma_1 \) and \( \gamma_2 \) in \( \mathcal{P} \) to an element in \( \mathcal{P} \) as follows:

\[
T(\gamma_1, \gamma_2) = \gamma_1^{-1} \circ \gamma_2.
\]

The function \( T \) will be the key for us to define structures which are invariant under global transformations, as we shall see in section 3.3. It offers a convenient way to encode the relation between two atoms. The following proposition gives some important properties about \( T \):

**Proposition 1 (Properties of \( T \)).** Let \( \gamma_1, \gamma_2, \gamma_3 \) and \( \alpha \) be arbitrary elements in \( \mathcal{P} \). The function \( T \) defined in Eq. (11) satisfies the following properties:

1. \( \gamma_1 \circ T(\gamma_1, \gamma_2) = \gamma_2 \).
2. **Transformation invariance:** \( T(\gamma_1, \gamma_2) = T(\alpha \circ \gamma_1, \alpha \circ \gamma_2) \).
3. \( T(\gamma_1, \gamma_2) = T(\gamma_2, \gamma_1)^{-1} \).
4. **Transitivity:** \( T(\gamma_1, \gamma_2) \circ T(\gamma_2, \gamma_3) = T(\gamma_1, \gamma_3) \).

**Proof.** 1. \( \gamma_1 \circ T(\gamma_1, \gamma_2) = \gamma_1 \circ \gamma_1^{-1} \circ \gamma_2 = \gamma_2 \).
2. \( T(\alpha \circ \gamma_1, \alpha \circ \gamma_2) = (\alpha \circ \gamma_1)^{-1} \circ (\alpha \circ \gamma_2) = \gamma_1^{-1} \circ \alpha^{-1} \circ \alpha \circ \gamma_2 = \gamma_1^{-1} \circ \gamma_2 = T(\gamma_1, \gamma_2). \)
3. \( T(\gamma_2, \gamma_1)^{-1} = (\gamma_2^{-1} \circ \gamma_1)^{-1} = \gamma_1^{-1} \circ \gamma_2 = T(\gamma_1, \gamma_2). \)
4. \( T(\gamma_1, \gamma_2) \circ T(\gamma_2, \gamma_3) = (\gamma_1^{-1} \circ \gamma_2) \circ (\gamma_2^{-1} \circ \gamma_3) = \gamma_1^{-1} \circ \gamma_3 = T(\gamma_1, \gamma_3). \)

We calculate explicitly this function when \( \mathcal{P} = \text{SIM}(2) \):

**Proposition 2.** Let \( \gamma_1 = (b_1, a_1, \theta_1) \) and \( \gamma_2 = (b_2, a_2, \theta_2) \). Then,

\[
T(\gamma_1, \gamma_2) = \left( r_{-\theta_1} \left( \frac{b_2 - b_1}{a_1}, \frac{a_2}{a_1}, \theta_2 - \theta_1 \right) \right).
\]

**Proof.** If \( \gamma = (b, a, \theta) \in \mathcal{P} \), it follows from the composition rule in Eq. (10) that \( \gamma^{-1} \) is given by:

\[
\gamma^{-1} = \left( r_{-\theta} \left( \frac{\vec{b}}{a}, \frac{1}{a}, -\theta \right) \right)
\]

As \( T(\gamma_1, \gamma_2) = \gamma_1^{-1} \circ \gamma_2 \), we combine the composition rule of Eq. (10) and the inverse formula of Eq. (12) to obtain the stated expression.

### 3.1.2 Mother functions

Generating functions should capture the most important features in the image. A possible example is the Gaussian function:

\[
\phi(x, y) = \frac{1}{C} \exp \left[ -\left( \frac{x}{q_x} \right)^2 + \left( \frac{y}{q_y} \right)^2 \right],
\]

where \( C \) is the normalization factor and \( q_x, q_y \) control the anisotropy of the Gaussian. These atoms typically capture smooth parts or textures in the image. Another mother function which is widely used to capture edges or sharp discontinuities is the anisotropic refinement (AR) function:

\[
\psi(x, y) = \frac{1}{C} \exp \left( -4x^2 - 2 \right).
\]

This function is a Gaussian in one direction and a mexican hat in the orthogonal direction.

In Figure 5, we show some examples of atoms taken from a dictionary \( \mathcal{D} \) generated using a Gaussian mother function and the \( \text{SIM}(2) \) parameter space. Because computers do not handle continuous spaces, we discretize the space \( \mathcal{P} \) of geometric parameters in numerical simulations, and the discretization parameters are denoted below Fig. 5. The Gaussian dictionary is in fact powerful since it allows to approximate images using a relatively small number of atoms. For example in Fig. 6 we see that one can obtain a good representation of a human face using a relatively small number of atoms.

![Example of atoms generated with the Gaussian mother function \( \phi \). Parameters: \( q_x = 1/4, q_y = 1 \).
Discretization of \( \mathcal{P} \): \( \{1, \ldots, 30\}, \{1, \ldots, 30\}, \{1, 1.5\}, \{0, \frac{q_x}{2}, \frac{q_y}{2}, \frac{3q_x}{4}\} \).](image)

### 3.2 Semantic similarity between sets

We now define the key notion of **semantic similarity** between sets of atoms. Intuitively, the semantic similarity expresses the fact that two sets of atoms hold the same ‘information’. That is, two sets are semantically similar if there is a global transformation in \( \mathcal{T} \) that transforms one set to the other. For simplicity, we consider in this work that transformations live in the same space as the space that is used to create the dictionary: \( \mathcal{T} = \mathcal{P} \).
Figure 7: Semantically similar sets. Each image is equal to: \( \sum_{\gamma \in \Gamma} \phi_{\gamma_i} \) with \( \gamma_i \) the set of atoms. The same dictionary as in Figure 5 is used.

The goal consists now to offer a simple characterization of semantically similar sets. As such sets are related by a global transformation, it is convenient to use the function \( T \) (section 3.1.1) which was shown to be transformation-invariant. We give in Proposition 3 two alternative characterizations of semantically similar sets that use the function \( T \).

**Proposition 3.** Let \( \Gamma = \{ \gamma_1, \ldots, \gamma_r \} \subset \mathcal{P} \) and \( \Gamma' = \{ \gamma'_1, \ldots, \gamma'_r \} \subset \mathcal{P} \). We say that \( \Gamma \) and \( \Gamma' \) are semantically similar if there exists a permutation \( \chi \) of \( \{1, \ldots, r\} \) and \( \alpha \in \mathcal{P} \) such that the following equality holds for any \( i \in \{1, \ldots, r\} \):

\[
\gamma'_i = \alpha \circ \gamma_{\chi(i)}.
\]

(13)

Definition 1 formally states the basic intuition that the atoms of semantically similar sets are related by one transformation \( \alpha \). The role of the permutation \( \chi \) is to map the corresponding atoms (i.e., the atoms that are equal up to a transformation \( \alpha \)) in the two structures. Note that when \( r = 1 \), we have semantic similarity between any two subsets (i.e., atoms) as these are related by a transformation, thanks to the parametric nature of the dictionary. We show in Fig. 7 some examples of semantically similar sets. In Fig. 7 as well as in any figure of this report, a set of parameters \( \Gamma' \) will be depicted by the image \( \sum_{\gamma \in \Gamma} \phi_{\gamma} \).

![Figure 6: Approximation of a face with OMP algorithm, using the Gaussian dictionary of Figure 5. The image has 900 pixels.](image)

![Figure 7: Semantically similar sets. Each image is equal to: \( \sum_{\gamma \in \Gamma} \phi_{\gamma_i} \) with \( \gamma_i \) the set of atoms. The same dictionary as in Figure 5 is used.](image)
• 3 ⇒ 1: Suppose that there exists a permutation \( \chi \) such that for all \( j \in \{2, \ldots, r\} \), \( T(\gamma_{(i)}, \gamma_{(j)}) = T(\gamma'_{(1)}, \gamma'_{(j)}) \). Thus, for any \( j \in \{1, \ldots, r\} \),

\[
(\gamma'_{(i)})^{-1} \circ \gamma'_{(j)} = \gamma_{(i)}^{-1} \circ \gamma_{(j)}
\]

\[
\gamma'_{(j)} = (\gamma'_{(i)} \circ \gamma_{(i)}) \circ \gamma_{(j)}
\]

\[
\gamma'_{(j)} = \alpha \circ \gamma_{(j)},
\]

by letting \( \alpha = \gamma'_{(i)} \circ \gamma_{(i)}^{-1} \).

These characterizations are quite natural to understand as semantically similar sets differ only by a global transformation and hence, the internal relations between the atoms (that are captured by the function \( T \)) are conserved.

### 3.3 Semantic structures

A **semantic structure** can be seen as a virtual container that contains all the semantically similar sets\(^1\). That is, if \( \Gamma \) denotes any subset in \( \mathcal{P} \), the semantic structure of \( \Gamma \) should ‘include’ any subset \( \Gamma' \) which is semantically similar to \( \Gamma \). Thus, it has to be defined in such a way to be transformation invariant with respect to any transformation in \( \mathcal{P} \). We propose to define such transformation invariant structures using graphs and we give in the following two methods to do so.

#### 3.3.1 Complete graph design of semantic structures

The point 2 from Proposition 3 suggests to represent semantic structures with a complete graph, where nodes represent atoms and where edges are given by the function \( T \).

**Definition 2** (Complete graph definition of semantic structures). Let \( \Gamma = \{\gamma_1, \ldots, \gamma_r\} \) be a subset of \( \mathcal{P} \). The complete graph semantic structure of \( \Gamma \) is the complete graph \( G = (V, E) \) where:

- \( V = \{1, \ldots, r\} \)
- \( E = \{E_{i \rightarrow j}, (i, j) \in \{1, \ldots, r\}^2\} \) with \( E_{i \rightarrow j} = T(\gamma_i, \gamma_j) \) and \( i \) and \( j \) are the labels of the nodes.

It is straightforward to see that such a definition of semantic structure encompasses all semantically similar sets. Indeed, semantic structures built from two semantically similar sets are equal up to a permutation of their nodes (Point 2 of Proposition 3). Figure 8(c) illustrates the complete graph structure of two semantically similar structures \( \Gamma \) and \( \Gamma' \).

#### 3.3.2 Tree design of semantic structures

The complete graph representation of a semantic structure is highly redundant in the sense that a semantic structure can be defined precisely with less information. This can be seen from the equivalence between points 2 and 3 in Proposition 3 which suggests that Eq.(14) needs to be verified for a single \( i \) in order to judge accurately about the similarity of the structures. Alternatively, one could understand this redundancy noting that the complete graph \( G = (V, E) \) that represents the semantic structure is a very particular one as it satisfies:

\[
E_{i \rightarrow j} = E_{j \rightarrow i}^{-1},
\]

\[
E_{i \rightarrow k} = E_{i \rightarrow j} \circ E_{j \rightarrow k}.
\]

for any \( i, j, k \) in \( \mathcal{P} \), thanks to properties 3 and 4 of Proposition 1. Thus, it may seem useless to define a semantic structure as a complete graph with \( r(r - 1) \) edges whereas only \( r - 1 \) edges are needed to completely define the structure (\( r \) is equal to the number of atoms in the structure). We thus introduce the following alternative definition of a semantic structure:

**Definition 3** (Tree definition of semantic structures). Let \( \Gamma = \{\gamma_1, \ldots, \gamma_r\} \) be a subset of \( \mathcal{P} \). The tree semantic structure of \( \Gamma \) is the tree \( G = (V, E) \) where:

- \( V = \{1, \ldots, r\} \)
- \( E = \{E_{1 \rightarrow j}, j \in \{1, \ldots, r\}\} \) with \( E_{1 \rightarrow j} = T(\gamma_1, \gamma_j) \) where 1 and \( j \) denote the labels of the nodes.

Similarly as for the complete graph definition, two sets are semantically similar if and only if they have equal structures, up to a permutation of their nodes. Figure 8(d) illustrates the tree structure of two semantically similar structures \( \Gamma \) and \( \Gamma' \).

\(^1\)Note that a semantic structure can be seen as an equivalence class under the equivalence relation of semantic similarity.
3.3.3 Notions of fitting and decomposability

In the sparse recovery problem, our main task is to fit a predefined structure to the data. We thus need to define the notion of fitting between a set and a semantic structure:

**Definition 4.** Let \( \Gamma \subset \mathcal{P} \) and \( G \) be a semantic structure generated from some set \( \Gamma' \). We say that \( \Gamma \) fits to \( G \) if \( \Gamma \) and \( \Gamma' \) are semantically similar. Equivalently, \( \{ \chi \}_{\chi \in \chi} \) is a coefficient that represents its weight with respect to the other atoms. We adapt the definition of model with the relative importance of the atoms in the structure. That is, we assign to each element in the space \( \mathcal{P} \). In section 6, we explore one possible way of defining such a metric.

When the data can be explained by several structures, we say that the data is decomposable, however we do not pursue this here and we restrict our attention to exact fitting. Nevertheless, the basic intuition would be to replace the equalities in Definition 4 by inequalities on the distance between the two terms in the equality. In order to do so, we need to define some metric on the parameter space \( \mathcal{P} \). In section 6, we explore one possible way of defining such a metric.

3.4 Semantic structures augmented with coefficients

Up to this point, we have defined structures based on the co-occurrence of atoms. We propose to augment this model with the relative importance of the atoms in the structure. That is, we assign to each element in the structure a coefficient that represents its weight with respect to the other atoms. We adapt the definition of semantic similarity to this new model.

**Definition 6.** Let \( \Gamma = \{ \gamma_1, \ldots, \gamma_r \} \subset \mathcal{P} \) and \( \Gamma' = \{ \gamma'_1, \ldots, \gamma'_r \} \subset \mathcal{P} \). Let \( c = \{ c_1, \ldots, c_r \} \) and \( c' = \{ c'_1, \ldots, c'_r \} \) be in \( \mathbb{R}^r \). We say that \( \{(\gamma_1, c_1), \ldots, (\gamma_r, c_r)\} \) and \( \{(\gamma'_1, c'_1), \ldots, (\gamma'_r, c'_r)\} \) are semantically similar if there exists a permutation \( \chi \) of \( \{1, \ldots, r\} \), \( \alpha \in \mathcal{P} \) and \( \nu \in \mathbb{R} \) such that the following equalities holds for any \( i \in \{1, \ldots, r\} \):

\[
\gamma'_i = \alpha \circ \gamma_{\chi(i)}, \quad c'_i = \nu c_{\chi(i)}.
\]
Figure 9: 1 and 2-decomposable sets in \( \{ G_1, G_2 \} \). A semantic structure \( G \) is represented by an image: \( \sum_{\gamma \in \Gamma} \phi_\gamma \) with \( \Gamma \) an arbitrary set which fits into \( G \). Similarly, a set \( \Gamma' \) is represented by the image: \( \sum_{\gamma' \in \Gamma'} \phi_{\gamma'} \). The numbers indicate the different geometric atoms. Notice that in example (e), we have a set which is decomposable into two overlapping structures, as the set can be written as a union of two (overlapping) sets, each fitting into one structure.

This definition of semantic similarity is thus more strict than the previous one. That is, two sets are semantically similar if they are equal up to a global transformation and their coefficients are equal up to a proportionality factor \( \nu \).

We follow the same approach as before: we give a characterization of semantic similarity which is independent of the transformation \( \alpha \) and the proportionality factor \( \nu \), define the two structures (complete graph and tree), and finally define notions of fitting and decomposability. For readability reasons, we put the main definitions of the notions with the new model in Appendix A.

3.5 Discussion

We have introduced two ways of defining transformation invariant structures using graphs. While the tree approach has the advantage of being non redundant, it has an asymmetric topology which can be a main shortcoming with respect to the complete graph representation. Consider for example an algorithm that selects an atom iteratively and tries to fit it on-the-fly into some incomplete structure. In such an approach, using a tree representation of the structure is not adapted since recovered atoms do not follow any specific ordering and the root node of the tree is arbitrary. In such a case, using a symmetric representation that considers all the edges between the different nodes is justified.

Even if the notion of exact fitting in a tree and a complete graph semantic structure are equivalent, this is not true anymore in the approximate fitting case. Indeed, the complete graph will provide a more robust notion of fitting as all pairs of atoms are considered, unlike with the tree representation. Hence, keeping all the edges despite the redundancy might be an advantage in the case where approximate structures are considered.
4 Problem formulation

In this work, we are interested in constructing a sparse recovery algorithm under structured sparsity prior and transformation invariance constraints. Let $\mathcal{D} = \{\phi, \gamma \in \mathcal{P}\}$ define some structured and geometric dictionary, where $\mathcal{P}$ is supposed to be finite of cardinality $K$ and $\phi_s \in \mathbb{R}^N$. If $\gamma_1, \ldots, \gamma_K$ denote the elements of $\mathcal{P}$, let $D$ be the $N \times K$ matrix that represents the atoms as columns:

$$D = [\phi_{\gamma_1} \ldots | \phi_{\gamma_K}].$$

In the rest of this report, $\gamma_1, \ldots, \gamma_K$ will always refer to the elements of $\mathcal{P}$ with the same order as the one used to construct $D$. For simplicity, we will sometimes use the term ‘atom’ to actually mean the parameter in $\mathcal{P}$ corresponding to the atom.

Let $s$ be a signal that admits a $S$-term expansion in $\mathcal{D}$:

$$s = Dx,$$

where $x \in \mathbb{R}^K$. Given any vector $u$ of length $K$, we define the geometric support of $u$ as the subset of $\mathcal{P}$ defined as follows:

$$\text{supp}_g(u) = \{\gamma_k : u_k \neq 0\}.$$ 

$\text{supp}_g$ thus denotes the geometric parameters corresponding to the non zero entries of a vector, as opposed to $\text{supp}$ that denotes the indices of the non zero entries.

Let $G_1, \ldots, G_R$ be $R$ semantic structures. We suppose that $\text{supp}_g(x)$ is decomposable into $G_1, \ldots, G_R$. The structured sparse recovery problem consists in finding $x$ from the observation of $s$, under the priors that $s$ is $S$-sparse and decomposable into $G_1, \ldots, G_R$. The structured sparse recovery problem SSR is formalized as follows:

$$\text{SSR}(\mathcal{D}, S, \{G_i\}_{i=1}^R) : \text{Find } x \in \mathbb{R}^K \text{ that satisfies: (i) } s = Dx, \text{ (ii) } \|x\|_0 \leq S, \text{ (iii) } \text{supp}_g(x) \text{ is decomposable into } G_1, \ldots, G_R.$$ 

Notice that the structured sparse recovery problem SSR is similar to the sparse recovery problem SR (section 2.1) with an additional constraint on the geometric support of the estimated vector. Indeed, the geometric support is constrained to form semantic structures.

We also propose a slightly modified version of the above problem definition, where the sparsity constraint is replaced with a constraint on the number of semantic structures that compose $s$.

$$\text{SSR}_2(\mathcal{D}, L, \{G_i\}_{i=1}^R) : \text{Find } x \in \mathbb{R}^K \text{ that satisfies: (i) } s = Dx, \text{ (ii) } \text{supp}_g(x) \text{ is } L\text{-decomposable into } G_1, \ldots, G_R.$$ 

Note that the sparsity constraint in SSR was dropped and replaced by a sparsity on the structure level. Unlike problem SSR, we put a constraint on the number of structures that compose the support of the signal. In order to have a well-posed problem, we suppose that problems SSR and SSR$_2$ admit a unique solution.

Similarly to the standard sparse approximation problem, we can define structured sparse approximation problems, where the signal $s$ is not exactly structured sparse, but structured compressible (i.e., well approximated by an exact structured $S$-sparse signal). In this case, we would simply replace condition (i) by the condition: $\|s - Dx\|_2$ is minimum. Notice that the approximation concerns the sparsity and not the structure. We consider for simplicity in all problem formulations that the signal is exactly decomposable into the prior structures.

When the structures are augmented with coefficients, the condition (iii) in SSR and (ii) in SSR$_2$ is replaced with the condition: $\{(\gamma_k, x_k) : \gamma_k \in \text{supp}_g(x)\}$ is (L)-decomposable into $G_1, \ldots, G_R$. In words, not only do we constrain the geometric support of the estimated solution to be decomposable into a union of structures, but the associated coefficients should also be proportional to the coefficients assigned to the prior structures (see Appendix A for the complete definition).

In the following sections, we will tackle the formulated problems SSR and SSR$_2$ (or equivalently, the approximation version if the signal is not exactly sparse) using iterative greedy algorithms. In section 5, we design an algorithm that solves SSR$_2$ by fitting entire structures to the image. In section 6, we propose another approach that finds suboptimal solutions for SSR based on the progressive construction of a structure promoting function. We suppose for simplicity in the sequel that all semantic structures have the same cardinality $r$. Note however that this condition could be relaxed with minor modifications.
5 Group orthogonal matching pursuit (GOMP)

We propose to construct an algorithm inspired from the greedy pursuit family in which we extend the Orthogonal Matching Pursuit algorithm to deal with semantic structure selection. This procedure, that we shall call Group orthogonal matching pursuit (GOMP)\(^2\) provides an approximation to the optimal solution of the structured sparse approximation problem (associated to \(SSR_2\)). In this approach, we adopt the tree definition of semantic structures. That is, we suppose that all semantic structures are defined according to Definition 3.

5.1 Intuition

As in the sparse recovery problem, the most difficult part in solving \(SSR_2\) is to correctly select the support of the vector \(x\). Based on our prior knowledge on the structure of the support, we know that it is constructed from the union of at most \(L\) sets which all fit into some structures \(G_1,\ldots,G_R\). Thus, one possible way to solve \(SSR_2\) would be to test every single signal whose support can be written as such. However, the search space becomes huge and hence, it is computationally infeasible.

We thus turn to a greedy iterative algorithm. The algorithm we propose runs in \(L\) iterations where \(L\) is the number of semantic structures to detect. Each iteration selects exactly one set of atoms that fits into one (or more) of the semantic structures. Notice that for a given semantic structure, the number of set of atoms that fit into this particular structure equals \(K\), where \(K\) denotes the number of atoms in the dictionary. Indeed, the parameters of any set of atoms that fit into a semantic structure \(G\) can be written as \(\{\eta, \eta \circ E_{1\to 2}, \ldots, \eta \circ E_{1\to r}\}\) for some element \(\eta\) in \(P\) (as always, \(E\) denote the edges of \(G\)). Thus by varying \(\eta\) to be in the whole space \(P\) of cardinality \(K\), we obtain exactly \(K\) sets of atoms that fit into a particular structure. Hence, there is at most \(RK\) sets of atoms that fit into either one of the structures in \(G = \{G_1,\ldots,G_R\}\). Thus, in our greedy approach, we have to examine at each iteration \(RK\) candidates and select the one achieving maximal correlation with the residual. As the number \(RK\) can be very large in practical situations, we cannot examine every possible set of atoms and thus we prune some of them. We do so by imposing the root atom of the selected structure to be energetic (i.e., it must have a high correlation with the residual). In other words, we consider only sets of atoms that can be written \(\{\eta, \eta \circ E_{1\to 2}, \ldots, \eta \circ E_{1\to r}\}\) with \(\phi_\gamma\) an energetic atom. By doing so, we reduce the search space from \(RK\) to \(Rw\) where \(w\) denotes the number of energetic atoms that are examined. This can be very beneficial in terms of computational complexity when \(w \ll K\).

We provide an intuitive justification of this pruning in the search space (from \(RK\) to \(Rw\)) by noting that for small structures (\(r\) small enough), it is unlikely to have a set of atoms which has a very high correlation with the residual while one of its atoms has a very low energy. Thus, by examining the energy of only one atom, we get a good idea on the energy of the whole structure, as the energy of the structure is the sum of the energies of the atoms. However, when structures typically have a large number of atoms, this is no more true since one atom does not represent enough the energy of the whole structure. As we shall see later, this intuition will be translated to the result of Proposition 5.

5.2 Description

The overall strategy of GOMP algorithm thus amounts to identifying sets of atoms fitting into structures one after another. A first greedy step is performed to determine the energetic atoms that can possibly be on the root of a structure. Then, in a second step, we determine the set of atoms that best matches with the residual. This set of atoms is searched in the space of sets that fit into some structure and whose root atom is selected in the first greedy step. The estimate of the image is formed by projecting the image on the space spanned by the selected atoms at all previous iterations and the residual is updated.

Formally, the GOMP algorithm begins by initializing the initial residual as \(r_0 = s\). At iteration \(1 \leq l \leq L\), for a fixed margin \(\delta > 0\), we calculate the set \(I\) of indices corresponding to the atoms having scalar product with the residual \(r_{l-1}\) higher than \(\frac{\|D^*r_{l-1}\|_\infty}{\delta}\):

\[
I = \left\{ k : |\langle \phi_{\gamma_k}, r_{l-1}\rangle| \geq \frac{\|D^*r_{l-1}\|_\infty}{\delta} \right\}.
\]

The next step is to fit one semantic structure whose root node is given by one of the energetic atoms listed in \(I\) to the image. For any parameter \(\eta\) and structure \(G\), let \(\Phi_{G,\eta}\) be the matrix of dimension \(N \times r\) defined as follows:

\[
\Phi_{G,\eta} = [\phi_1 | \phi_{\eta \circ E_{1\to 2}} | \ldots | \phi_{\eta \circ E_{1\to r}}].
\]

In words, \(\Phi_{G,\eta}\) stacks as columns the atoms corresponding to the set which fits into \(G\), and whose root node is fixed by \(\eta\). Our selection criterion is simple: we choose the pair of semantic structure and root node which

Note that an algorithm with the same name exists [24], but it is different from ours.
results in the best match with the residual $r_{l-1}$:

$$(\hat{G}, \hat{k}) = \arg\max_{G \in \mathcal{G}, k \in \mathcal{I}} \|\Phi_{\hat{\gamma}_k}^* r_{l-1}\|_2,$$  \hspace{1cm} (17)

where $(\cdot)^*$ denotes the transpose of a matrix and $\mathcal{G}$ denotes the set of structures $\{G_p\}_{p=1}^R$. Note that $k$ is constrained to be in $\mathcal{I}$ and not in the entire space $\{1, \ldots, K\}$.

As $(\hat{G}, \hat{\gamma}_k)$ completely characterizes a set of size $r$ which fits into $\hat{G}$, we look for the indices in the dictionary corresponding to the elements of this set. That is, we look for $q_1, \ldots, q_r$ in $\{1, \ldots, K\}$ such that:

$$\gamma_{q_1} = \gamma_k,$$
$$\gamma_{q_2} = \gamma_k \circ \hat{E}_{1 \rightarrow 2},$$
$$\ldots$$
$$\gamma_{q_r} = \gamma_k \circ \hat{E}_{1 \rightarrow r},$$

where $\hat{E}_{1 \rightarrow j}$ denote the edges of $\hat{G}$. Note that obviously, $q_1 = \hat{k}$.

Once the parameters are chosen, we update the support $\Omega_l$ and recalculate the residual:

$$\Omega_{l} \leftarrow \Omega_{l-1} \cup \{q_1, \ldots, q_r\}$$
$$\hat{x}_{l|\Omega_l} \leftarrow D_{[\Omega_l, s]}^\dagger \hat{x}_{l|\Omega_l} \leftarrow 0$$
$$r_l \leftarrow s - D \hat{x}_l.$$

We illustrate a comparative scheme between OMP and GOMP in Fig 10.

The algorithm is summarized in Algorithm 2.

**Algorithm 2: Group Orthogonal Matching Pursuit (GOMP)**

**Input:** signal $s$, number of semantic structures $L$ to detect, dictionary $D$, tree semantic structures $\mathcal{G} = \{G_1, \ldots, G_R\}$, margin $\delta$.

**Output:** estimated sparse vector $\hat{x}$.

1. Initialization: $r_0 \leftarrow s$, $\Omega_0 \leftarrow \emptyset$, $\hat{x}_0 \leftarrow 0$.

2. While $1 \leq l \leq L$, do:

   2.1 Compute the set $\mathcal{I}$:

   $$\mathcal{I} \leftarrow \left\{ k : |\langle \phi_{\gamma_k}, r_{l-1} \rangle| \geq \frac{\|D^* r_{l-1}\|_\infty}{\delta} \right\}$$

   2.2 Select the pair $(\hat{G}, \hat{k})$ as follows:

   $$(\hat{G}, \hat{k}) = \arg\max_{G \in \mathcal{G}, k \in \mathcal{I}} \|\Phi_{\hat{\gamma}_k}^* r_{l-1}\|_2$$

   2.3 Determine the indices of the set which fits into $G$ and whose root node is $\gamma_k$:

   $$\gamma_{q_1} = \gamma_k$$
   $$\gamma_{q_2} = \gamma_k \circ \hat{E}_{1 \rightarrow 2}$$
   $$\ldots$$
   $$\gamma_{q_r} = \gamma_k \circ \hat{E}_{1 \rightarrow r}$$

   2.4 Update the support and recompute the residual:

   $$\Omega_{l} \leftarrow \Omega_{l-1} \cup \{q_1, \ldots, q_r\}$$
   $$\hat{x}_{l|\Omega_l} \leftarrow D_{[\Omega_l, s]}^\dagger \hat{x}_{l|\Omega_l} \leftarrow 0$$
   $$r_l \leftarrow s - D \hat{x}_l.$$

3. Return the estimated sparse vector: $\hat{x} \leftarrow \hat{x}_L$.  

20
5.3 Computational complexity

In practice, instead of considering the set $I$ of atoms that have an energy higher than $\frac{\|D^*r_i\|_\infty}{\delta}$, we consider the set of atoms with cardinality $w$ achieving highest scalar product with the residual. By doing so, the cardinality of $I$ is kept constant through the iterations and the computational complexity is controlled. We detail as follows the complexity of each step in GOMP:

- Computation of the set $I$ (2.1): Every scalar product costs $N$ add and multiply operations. Thus, this step costs $O(NK)$ operations.

- Selection of the pair $(G, k)$ (2.2): For a given $G \in \mathcal{G}$ and $k \in I$, the generation of $\Phi_{G, \gamma_k}$ and the evaluation of $\|\Phi_{G, \gamma_k}r_i\|_2$ costs $O(rN)$ operations. Thus, the whole step costs $O(rRN)$ operations.

- Computation of $\{q_1, \ldots , q_r\}$ (2.3): This step needs only $O(r)$ operations since we can construct a hash table that maps every parameter in $P$ to its associated index in the dictionary. Note that this is possible since we suppose that the dictionary can be stored in memory.

- Update the support and computation of the residual (2.4): The update step is mainly governed by the calculation and inversion of the Gram Matrix $D^*_r D_I$. The calculation can be done from the previously computed matrix $D^*_r D_I$ and thus costs at most $O(rLN)$ operations. Then, the Gram matrix can be inverted with a complexity of roughly $O(r^2L^2)$ by reusing the computations of the last iteration. The exact cost depends however on the chosen method.

When $w$ (i.e., the cardinality of $I$) approaches $K$ (the dictionary size), step (2.2) becomes the bottleneck of the algorithm. The overall cost of GOMP is then governed by the cost of (2.2) which will be roughly equal to $O(rRKN)$. It is clear that for typical values of such parameters, this cost is extremely large. This provides a justification to the fact that we do not consider every atom in the dictionary in $I$ but only the most energetic ones. When $w$ is small enough, the complexity is governed by the selection and update steps (2.1) and (2.4).

5.4 Recovery conditions

Establishing recovery conditions consists in answering the following question: given a signal which is $L$-decomposable into the given semantic structures $\mathcal{G} = \{G_1, \ldots , G_R\}$, when does GOMP recovers the correct support of the optimal solution $x^*$ for the problem (SSR$_R$)? Let $\beta_1^*, \ldots , \beta_r^*$ and $\beta_1^L, \ldots , \beta_r^L$ denote respectively the decompositions of the optimal support and estimated support with GOMP (Definition 5). Perhaps the most natural way to proceed would be to determine a sufficient condition to guarantee that at each iteration $\hat{\beta}_i$ equals one of the $\beta_i^*$s. Indeed, as two sets are never chosen twice (because of the orthogonal projection), this would guarantee that the correct components of the support are detected which implies in turn that the support is correct. Unfortunately, we have not managed to determine such a condition. However, in this section we detail why such a recovery condition is difficult to obtain.

At each iteration of GOMP, we have two greedy steps where the algorithm may pick an incorrect $\hat{\beta}_i$. The first consists in the fact that we disregard all parameters that do not belong to $I$ to be on the root of semantic structures. The second one consists in selecting the semantic structure according to step 2.2 in Algorithm 2. For simplicity, let us work in ideal conditions where $\delta$ is sufficiently large to have $\frac{\|D^*r_i\|_\infty}{\delta} \approx 0$ at each iteration.
In other words, we suppose that the set $I$ contains the indices of all the dictionary elements and $|I| = K$. We focus on the suboptimality that step 2.2 may cause. As we supposed that $|Z| = K$, the maximization of step 2.2 occurs over $KR$ pairs of $(G, k) \in G \times I$. For each pair $(G, k)$, denote by $M_h$ the matrix : $M_h = \Phi_{G, \gamma_h}$, for $1 \leq h \leq KR$. We refer to $M_h$ as a block. As the ordering has no importance, let $M_1, \ldots, M_L$ denote the optimal blocks. That is, for any $1 \leq l \leq L$, $M_l$ is the matrix that stacks as columns the atoms corresponding to the set $\beta_l'$. Because the signal $s$ is decomposable, we can write:

$$s = \sum_{i=1}^{L} M_i x_i,$$

where $x_i$ denotes a vector of size $r$ containing the coefficients associated to the atoms in $M_i$. By determining a sufficient condition that all the chosen block indices are correct (i.e., they are in the set $\{1, \ldots, L\}$), we obtain a sufficient condition on the correctness of GOMP. The authors of [25] study the recovery conditions of Block Orthogonal Matching Pursuit (BOMP) algorithm, which is equivalent to ours under the condition that $\phi$ follows: let us consider the case where $\beta_l'$ is equal to one of these does not hold. Notice that in $\phi$ atom $\mu$ figures as one of the columns of $M_h$.

**Theorem 2** (Theorem 3 in [25]). Define the block-coherence $\mu_B$ and subcoherence $\nu$ as follows:

$$\nu = \max_{h \in \{1, \ldots, KR\}} \max_{\phi_j \in M_h, \phi_i \in M_h, \phi_j \neq \phi_i} |\phi_j^* \phi_i|,$$

$$\mu_B = \max_{a \in \{1, \ldots, KR\}, b \in \{1, \ldots, KR\}, b \neq a} \frac{1}{r} \|M_a^* M_b\|_{(2,2)},$$

where $\| \cdot \|_{(2,2)}$ is the spectral norm of a matrix [26]. If:

$$Lr < \frac{1}{2} \left( \mu_B^{-1} + r - (r - 1) \frac{\nu}{\mu_B} \right),$$

(18)

Then, the estimated block indices with GOMP/BOMP are correct.

Condition in Eq. (18) is never satisfied whenever two blocks overlap. That is, if there exists $\phi_j \in M_{i_0} \cap M_{j_0}$ with $i_0 \neq j_0$, then Eq. (18) yields: $L < 1$. As $L \in \mathbb{N}$, this necessarily means that $L = 0$ and the theorem thus gives no guarantee on the correct recovery of the algorithm. We give the justification as follows: let us consider the case where $\phi_j \in M_{i_0} \cap M_{j_0}$ with $i_0 \neq j_0$, then, the matrix $M_{i_0}^* M_{j_0}$ has an entry which equals 1. As the spectral norm of a matrix is lower bounded by the maximum element in the matrix [26], we have: $\|M_{i_0}^* M_{j_0}\|_{(2,2)} \geq 1$, which in turn implies $\mu_B \geq \frac{1}{K}$. By replacing this condition in Eq. (18), we obtain in the best case ($\mu_B = \frac{1}{K}$):

$$L < \frac{1}{2} (2 - (r - 1) \nu),$$

$$L < 1 - \frac{1}{2} (r - 1) \nu,$$

which implies $L = 0$. We give an example to show that this recovery condition cannot be improved to handle overlapping blocks: let $r = 2$ and let $M_1 = [\phi_{i_1} | \phi_{i_2}]$ and $M_2 = [\phi_{j_1} | \phi_{j_2}]$ be two blocks sharing a common atom $\phi_{i_1}$. Consider the signal $s = \phi_{i_1}$. There exist two distinct correct decompositions of $s$ into the blocks: $s = M_1[1]0^*$ and $s = M_2[1]0^*$. Thus, as soon as we have overlapping blocks, there exists signals whose supports are not uniquely decomposable into the blocks, and thus, it becomes difficult to define a ‘correct’ set of blocks. In other words, there exists a class of signals such that the decomposition $\beta_1', \ldots, \beta_L'$ is not unique, and thus, imposing that the estimated $\hat{\beta}_1$ with GOMP are equal to one of these does not hold. Notice that in the above example blocks $M_1$ and $M_2$ are equally correct (i.e., they lead to a correct support). This suggests to rather establish a condition on the error of the estimated solution obtained with GOMP, since the ‘correctness’ of the estimated blocks is an ambiguous term.

### 5.5 Approximation error

We propose to establish a bound on the error of the estimated signal in the ideal case where $|Z| = K$. That is, we give an upper bound on the energy of the residual $\|r_l\|_2$ at each iteration of the algorithm.
5.5.1 Intra-structure coherence

We first define the notion of intra-structure coherence as follows:

**Definition 7.** The intrastructure coherence $\mu_{IS}$ is given by:

$$\mu_{IS} = \max_{G\in \mathcal{G}} \max_{\gamma \in \Gamma_G} \sum_{\gamma' \in \Gamma_G, \gamma' \neq \gamma} |\langle \phi_\gamma, \phi_{\gamma'} \rangle|,$$

where $\Gamma_G$ denotes an arbitrary set which fits into $G$.

Since we take an arbitrary set which fits into $G$, we need to justify that $\mu_{IS}$ is independent of the choice of this set. We give a justification in $\mathcal{P} = SIM(2)$ where we prove that for any set $\Gamma$ and for any parameter $\eta \in \mathcal{P}$, $\langle \phi_\gamma, \phi_{\gamma'} \rangle = \langle \phi_{\eta \gamma}, \phi_{\eta \gamma'} \rangle$. The proof as well as a discussion on a technical assumption of this result can be found in Appendix B.1.

The intrastructure coherence provides a measure of the similarity of the different atoms in a structure. If atoms in one structure are very different from one another, we get $\mu_{IS} \approx 0$ whereas structures which contain atoms that are very similar will have their intrastructure coherence nearly equal to one.

5.5.2 Error in approximation

Let $s$ denote a signal that verifies: (i) $s = Dx \in \mathbb{R}^N$ and (ii) $\text{supp}_G(x)$ is L-decomposable. Thus, there exist matrices $M_1, \ldots, M_L$ of size $N \times r$ and coefficient vectors $x_1, \ldots, x_L$ of size $r$ such that:

$$s = Dx = M_1x_1 + \cdots + M_Lx_L.$$

Note that as we discussed in the previous section, such a decomposition is possibly non unique. We consider here any of these (possibly many) decompositions. We establish the following result on the norm of the residual in function of the iteration number $l$.

**Theorem 3** (Bound on the residual of GOMP). If, at each iteration of GOMP $\delta$ is chosen adequately to have $|I| = K$, then:

$$\|r_l\|_2 \leq \frac{\|x\|_{(2,1)}}{\sqrt{l} (1 - \mu_{IS})},$$

where $\|x\|_{(2,1)} = \sum_{k=1}^L \|x_k\|_2$.

As the proof of this theorem is quite technical, we give it in Appendix B.2.

This result is to be compared with [27][Theorem 3.7] which establishes a bound on the residual of OMP. It guarantees that for any signal $s = Dx$, the residuals of OMP will verify $\|r_l\|_2 \leq \frac{\|x\|_1}{\sqrt{l}}$. Hence, if the intrastructure coherence $\mu_{IS} \approx 0$, the established upper bound with GOMP goes faster to zero than the upper bound provided in [27]. Indeed, for any vector $x$ that can be written as a concatenation of smaller vectors $x_1, \ldots, x_L$, we have: $\|x\|_1 \geq \sum_{k=1}^L \|x_k\|_2$. This bound suggests that when the structures in $G$ are large enough, the GOMP residual will have a relatively fast decay which in turn implies that the obtained approximation after a finite number of iterations is of ‘good’ quality. However, when structures are chosen to be small (i.e., $r$ small), we get a poorer approximation for the same number of iterations. Note however that this result concerns only the upper bounds and we cannot conclude on whether GOMP provides a faster convergence to zero of the residuals than OMP. Fig 11 illustrates the upper bound on the residual for one signal when $L = 6$ (small blocks, OMP), $L = 3$ (medium blocks) and $L = 1$ (large blocks).

Thus, this theorem confirms our intuition that structures should be large enough in order to be beneficial. However, designing extremely large structures is not feasible as these structures will then be specific to a small class of signals. Thus, the value of $r$ should be chosen to optimize the compromise between flexibility and approximation error.

5.6 GOMP augmented with coefficients

5.6.1 Algorithm

When structures are augmented with the relative importance of their nodes, the decomposability gives us information not only on the support of the signal, but also on the coefficients. Indeed, the coefficients of the blocks participating in the signal are constrained to be proportional to the coefficients given as a prior. Hence,
one straightforward way to solve $(SSR_2)$ would be to use OMP with a new dictionary constructed from the blocks weighted with the prior coefficients. For any $G \in \mathcal{G}$, let $c^G$ define the following vector in $\mathbb{R}^r$:

$$c^G = \begin{pmatrix} E_{1 \rightarrow 2}(2) \\ \vdots \\ E_{1 \rightarrow r}(2) \end{pmatrix},$$

where $E_{1 \rightarrow j}(2)$ denotes the coefficient component in the edge (the first component being $T(\gamma_1, \gamma_j)$). As the root node of the structure is chosen systematically to be the one with maximal coefficient (Definition 9), all entries of $c^G$ are less than or equal one. Using this definition, we construct the new dictionary as follows:

$$\mathcal{D}' = \left\{ \frac{\Phi_G \cdot c^G}{\| \Phi_G \cdot c^G \|_2} : G \in \mathcal{G}, \gamma \in \mathcal{P} \right\}.$$

Dictionary $\mathcal{D}'$ contains $KR$ elements. When the number of structures $R$ is large, applying OMP with such a dictionary might be computationally demanding. Hence, we apply the same heuristic as in GOMP: we constrain the root node of the structure to be energetic. There are basically two differences between GOMP and GOMP with coefficients algorithms: (i) in the structure selection step, the energy function takes into account the coefficients and becomes $|r^*_l - 1|_2^2 = \sum_{G \in \mathcal{G}} \| \Phi_G \cdot c^G \|_2^2$ (recall that for GOMP, it was the squared sum of the scalar products between the residual and the atoms). (ii): in the update step, the new signal estimate is obtained by projecting $s$ onto the space spanned by the selected structures at the previous iterations. Thus, unlike GOMP without coefficients, the selected structure conserves the ratios between the coefficients of the different atoms. The procedure with the new structure model is shown in Algorithm 3.
Algorithm 3 Group Orthogonal Matching Pursuit (GOMP) with structures augmented with coefficients

**Input:** signal \( s \), number of semantic structures \( L \) to detect, dictionary \( D \), tree semantic structures \( G = \{ G_1, \ldots, G_R \} \), margin \( \delta \).

1. Initialization: \( r_0 \leftarrow s \), \( A_0 \leftarrow \emptyset \).
2. While \( 1 \leq l \leq L \), do:
   2.1 Compute the set \( \mathcal{I} \):
      \[
      \mathcal{I} \leftarrow \left\{ k : |\langle \phi_{\gamma_k}, r_{l-1} \rangle| \geq \frac{\| D^* r_{l-1} \|_\infty}{\delta} \right\}
      \]
   2.2 Select the pair \((\hat{G}, \hat{k})\) as follows:
      \[
      (\hat{G}, \hat{k}) = \arg\max_{G \in G, k \in \mathcal{I}} \frac{|r_{l-1}^T \Phi(G, \gamma_k) c_G^G|}{\| \Phi(G, \gamma_k) c_G^G \|_2}
      \]
   2.3 Update \( A_l \), find the best approximant lying in the column span of \( A_l \) and update the residual:
      \[
      A_l \leftarrow \left[ A_{l-1} | \frac{\Phi_{\gamma_k}^G c_G^\hat{G}}{\| \Phi(G, \gamma_k) c_G^\hat{G} \|_2} \right]
      \]
      \[
      r_l \leftarrow s - A_l A_l^T s
      \]

If we ignore the first greedy step (2.1), GOMP corresponds exactly to an OMP performed on dictionary \( D' \) (recall that this block dictionary contains as ‘atoms’ all the transformations of the structures).

Note that the only reason we assign the parameter with maximal coefficient to the root node (Definition 9) of the tree is to increase the chance of detecting the root node in the first greedy step (2.1).

### 5.6.2 Recovery conditions of GOMP with coefficients

Let \( s \) be a \( L \)-decomposable signal in the structures \( G_1, \ldots, G_R \). Equivalently, \( s \) admits an \( L \)-term expansion in \( D' \). We first study the behaviour of this algorithm in the ideal case where \( |\mathcal{I}| = K \) at each iteration. In this case, GOMP reduces to OMP with dictionary \( D' \). Thus, Tropp’s OMP exact recovery condition [8] applies:

**Theorem 4.** If \( \delta \) is sufficiently large (\(|\mathcal{I}| = K \) at each iteration), GOMP recovers every \( L \)-decomposable signal whenever:

\[
L < \frac{1}{2} \left( 1 + \frac{1}{\mu(D')} \right),
\]

where \( \mu(D') \) is the coherence [8] of \( D' \).

It remains to obtain an upper bound on the coherence of the dictionary \( D' \) in terms of the coherence of \( D \). The following proposition establishes such an upper bound:

**Proposition 4.** Let \( O \) define the maximum number of overlapping atoms in two different blocks:

\[
O = \max_{(G, \gamma) \neq (G', \gamma')} |\Phi(G, \gamma) \cap \Phi(G', \gamma')|.
\]

The coherence of the dictionary \( \mu(D') \) is bounded as follows:

\[
\mu(D') \leq \frac{1}{1 - \mu_{IS}} \left( \mu(D) r + \frac{O}{\min_{G \in G} \| c_G^G \|_2^2} \right)
\]

The proof of this proposition can be found in Appendix B.3.

This result shows that if the intracoherence of the structures is kept small \( \mu_{IS} \approx 0 \) and the different blocks of atoms do not have a large overlap with respect to the norm of the coefficients of the structures (i.e., \( \frac{O}{\min_{G \in G} \| c_G^G \|_2^2} \approx 0 \)), then the coherence \( \mu(D') \) is bounded by \( r \mu(D) \). In this case, the sufficient condition for perfect recovery from Theorem 4 becomes:

\[
L < \frac{1}{2} \left( 1 + \frac{1}{r \mu(D)} \right).
\]
In contrast, the sufficient condition of OMP tells us that the support of an \( rL \)-sparse vector is perfectly determined as soon as:

\[
 rL < \frac{1}{2} \left( 1 + \frac{1}{\mu(D)} \right).
\]  

(22)

Notice that the recovery condition of GOMP in Eq. (21) is at least as good as OMP in Eq. (22). Thus, if \( \mu \approx 0 \) and the number of overlapping atoms in the blocks remains low, applying GOMP with coefficients (or equivalently OMP with dictionary \( D' \)) can only be beneficial. The quantity \( \frac{O_{\min}}{O_{\min} \in \mathcal{G} \|c_G\|^2_2} \) captures the information on the dispersion of the coefficients across the different atoms of a structure. That is, when the coefficients are well shared across the different atoms, it is likely that \( \|c_G\|^2_2 \) becomes large and results in a quantity \( \frac{O_{\min}}{O_{\min} \in \mathcal{G} \|c_G\|^2_2} \approx 0 \). However, when the structure has its coefficients concentrated only on one or few atoms (the atom corresponding to the root node, since we supposed it to be the maximal one), this quantity becomes large which results in a large coherence of the dictionary \( D' \). To see why this happens, consider an example where coefficients of all the structures concentrate all on one atom (i.e., \( c_1 = 1, c_i = 0 \) for \( i \geq 2 \)). In this case, the coherence of dictionary \( D' \) equals one, since we have two atoms in the dictionary that are equal (consider two different ‘structures’ sharing the root node atom).

5.6.3 Lower bound on \( \delta \)

The next result provides a lower bound on \( \delta \) so that, at each iteration, the root atom of the block having maximal energy is selected in step 2.1.

**Proposition 5.** Let \( (G^*, k^*) = \arg\max_{G \in \mathcal{G}, k \in \{1, \ldots, K\}} \frac{r^*_l \Phi(G, \gamma_k)c^G}{\|\Phi(G, \gamma_k)c^G\|^2_2} \).

If \( \delta \) verifies:

\[
\delta \geq (3 - 2 \max_{G \in \mathcal{G}} \|c_G\|^1_1)^{-1},
\]

then, \( k^* \in \mathcal{I} \).

The proof can be found in Appendix B.4.

This bound gives a guarantee that by constraining the search space from \( \{1, \ldots, K\} \) to \( \mathcal{I} \) we do not remove away the highest energy block. Because this bound is too conservative, it should not be interpreted in a quantitative way. Rather, we will explain the main message behind this bound qualitatively. It simply shows that in order to guarantee that the root atom of the structure having maximal energy is selected in \( \mathcal{I} \), we need to make sure that the coefficients of the structures are well concentrated around their root node. In other words, we need to have \( c^G \approx [1, 0, \ldots, 0]^+ \) for all semantic structures \( G \) and hence \( \|c^G\|^1_1 \approx 1 \). In this case, the energies of the root atoms are good representatives of the energies of whole blocks, which leads to an accurate preselection step (2.1) that does not throw away high energy blocks. If blocks had their coefficients spread on all the atoms, observing the energy of only one atom in step (2.1) would not have been an accurate criteria for selection, hence yielding to a high \( \delta \) in order to guarantee the selection of the maximal energy block.

Hence, we are faced with two contradictory assumptions that we need to fulfill in order to guarantee a good performance of the GOMP algorithm. On the one hand, coefficients should be well spread among the atoms of the structure so that the coherence of the block dictionary remains low, and on the other hand, the root atom should be representative of the structure (which implies that coefficients should be concentrated).

We verify these qualitative results obtained from the above propositions by designing the following experiment: we generate random structures whose weights are given by \( c^G = |1, \mathcal{N}(0, \sigma), \ldots, \mathcal{N}(0, \sigma)| \) and generate random images created from some of the constructed structures (i.e., the images are constructed by taking a linear combination of the structures). Then, we run GOMP algorithm for different values of \( w \) (cardinality of \( \mathcal{I} \)) and show the success support recovery in function of the spread of the coefficients \( \sigma \) in Fig 12.
For small values of $\sigma$, the coefficients are concentrated on the root node, hence the block dictionary has a high coherence which explains the poor support recovery performance. When $\sigma$ becomes high, the performance drops whenever $w$ is taken small. This comes from the fact that the root atom is not representative anymore of the whole structure. When $w$ is taken large enough (equivalently $\delta$ large enough), the first greedy step does not affect much the performance of the algorithm.

In summary, the only difference between GOMP with coefficients and OMP on the block dictionary is the greedy step (2.1). The advantage of performing OMP on the block dictionary (or equivalently, GOMP with $w = K$) was highlighted in Proposition 4 as we obtained recovery conditions at least as good as OMP’s in some practical cases. The advantage of GOMP with coefficients over OMP on the block dictionary is that the computational complexity is controlled. We have established a bound in Proposition 5 in order to guarantee that step (2.1) does not ignore the most energetic block. However, this bound was only given to have a better understanding and cannot be of practical use. Hence, we do not have a quantitative result on the value of $\delta$ that should be chosen. In practice, we will see that even small values of $\delta$ (or $w$) lead to good experimental results.

### 5.7 Experimental results

#### 5.7.1 Synthetic images

We consider the two semantic structures ‘arrow’ and ‘rectangle’ illustrated in Fig 13. We build an image composed of 11 such semantic structures with different translations, rotations and scalings (Fig 14(a)). We run OMP and GOMP algorithm (without coefficients) to recover the support of the image. The success recovery rates are shown in Fig 14(b) and execution times are shown in Fig 14(c).

![Rectangle and Arrow Semantic Structures](image)

Figure 13: Rectangle and arrow semantic structures.

When $w$ is taken to be large enough (equivalently, $\delta$ is large enough), the support recovery success rate reaches 1 (perfect recovery) and our algorithm largely outperforms OMP. Note that we obtain perfect support with a value of $w = 50$ which is much less than the dictionary size ($K = 20000$). This provides an empirical justification to our first greedy step: using a value of $w = 20000$ would have been very costly in terms of execution time and we do not perform better by testing all possibilities, as the optimal solution is already obtained with $w = 50$. In terms of execution time, GOMP runs faster than OMP since it performs only $L = 11$ iterations as opposed to OMP, which performs $rL = 44$ iterations. Note that our algorithm is better in terms of execution time only when $w$ is small with respect to the dictionary size. When $w$ approaches $K$, the search space becomes of size $KR$ and the structure selection step (2.2) becomes the bottleneck of the algorithm.
5.7.2 Handwritten digits

We examine the performance of the GOMP algorithm on images constructed from handwritten digits for classification and denoising applications. The dataset is composed of several classes. Each class corresponds to a particular digit. For each class, the dataset is divided into a training set and a testing set. We limit the size of the training and testing set of each class to 100 images. We construct the prior structures $G$ in the following way: for each training image, we run OMP algorithm with a parameter $r = 15$. The recovered atoms are then used to construct the semantic structure $G \in G$. Thus, in total, we have $10 \times 100 = 1000$ semantic structures (100 per digit, 10 digits). Note that we consider many semantic structures per digit to capture the different handwritings. We then construct images by taking three random images from the testing set and place them in a big image with fixed positions, and random rotations. Fig. 15 illustrates some of the constructed images. The parameter $w$ (cardinality of $I$) is set fixed for all experiments equal to 50 and the dictionary is constructed from a Gaussian mother function.

We use the GOMP algorithm first for denoising. We corrupt the images with severe white gaussian noise and we denoise it by finding its sparse representation in the dictionary $\mathcal{D}$. We compare the denoising performance for or OMP, GOMP and GOMP with coefficients in Fig. 16.

Fig. 16 shows that GOMP outperforms standard OMP in this denoising application. Using OMP, we obtain images with scattered atoms because of the noise. On the other hand, GOMP constructs images from the set of structures (digits) which results always in an interpretable image. Note that GOMP determines sometimes wrong digits (in Fig 16(g) for example). The images obtained with GOMP using structures augmented with coefficients are of better quality than without coefficients since the recovered image respects the ratio on the coefficients of the structure, while GOMP adapts the coefficients to obtain an image which matches with the noisy one. Hence, taking into account the relative importance of the different atoms in a structure might improve the performance of GOMP, especially in denoising applications. Indeed, projecting the noisy image on the space spanned by the selected atoms might be a bad way to calculate the coefficients when the image is very noisy.

Since the semantic structures actually have a meaning, we can also use GOMP for classification/pattern recognition applications. In our simple example, we might want to automatically detect the digits composing the image. We do so by applying GOMP on the image: the detected digits are simply the ones corresponding to the estimated semantic structures. We then compare the digits estimated via GOMP with the true digits.

---

3Database downloadable at http://cs.nyu.edu/~roweis/data.html
The classification performance is computed by dividing the number of successfully detected digits over the number of digits in the image (3 in our case).

<table>
<thead>
<tr>
<th></th>
<th>GOMP</th>
<th>GOMP with coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean</td>
<td>0.70</td>
<td>0.78</td>
</tr>
<tr>
<td>Noisy</td>
<td>0.56</td>
<td>0.70</td>
</tr>
</tbody>
</table>

Table 2: Classification performance for GOMP and GOMP with coefficients for images composed of 3 digits. The classification performance is computed by dividing the number of successfully detected digits over the number of digits in the image (3 in our case). The performance is between 0 and 1 (perfect classifier). In each case, 60 random images were considered.

We show the performance of classification in Table 2. The results confirm that GOMP with coefficients is probably more robust to noise than GOMP (without coefficients). The achieved performance thus shows that GOMP can potentially be used in classification applications. Nevertheless, it is important to note that we needed to define a lot of semantic structures for each digit so that we capture the different variabilities of the handwritten digits. In practice, when we have more than 10 classes, it is impossible to do so and hence, we would have to design an algorithm that is robust to the variabilities of the semantic structures.

5.8 Drawbacks of GOMP

As we have seen through theoretical and experimental results, the GOMP algorithm is undoubtedly an interesting approach towards solving the structured sparse recovery problems. However, it suffers from various limitations:

- The number of semantic structures composing the image $L$ has to be known beforehand. In some applications, this might not be the case. For example, consider the case where the goal is to count the number of semantic structures detected in the image.
- We have not managed to obtain a practical bound on the margin $\delta$ that would quantify the number of atoms that should be considered in the first greedy step. Thus, the user needs to run the algorithm several times increasing $\delta$ till the algorithm performance is considered acceptable.
- It might be the case that the image is not solely composed of the prior semantic structures. It is a realistic scenario as we are unlikely to have a perfect knowledge of all the potential structures composing the image beforehand. However, GOMP fits at each iteration ‘by force’ one prior semantic structure thus
leading to an estimated image which is exactly composed of the prior structures. We might want to consider an algorithm that softly fits structures in the data.

- GOMP does not detect structures which are ‘approximately’ equal to one of the prior structures, where the approximation is defined in a ‘semantic way’. In the handwritten digits example, we needed to define a lot of similar semantic structures (i.e., similar digits) in order to have an algorithm which recognizes the digits with a good performance.

We give in the next section an alternative algorithm that tries to solve some of the mentioned shortcomings.
6 Progressive structure promoting algorithm using complete graphs

In this section, we propose an iterative algorithm that can be seen as a natural solution between OMP and GOMP. That is, we try to combine the benefits of both approaches in the proposed Progressive Structure Promoting (PSP) algorithm to solve problem SSR.

6.1 Intuition

6.1.1 Preliminaries

We propose an algorithm which iteratively selects an atom that fits well to the data and forms structures with the previously selected atoms. As the atoms are determined with no predefined ordering, we adopt the complete graph representation of semantic structures (see the discussion of section 3.5). The main difficulty in such an approach is to quantify how well an atom is likely to promote the ‘structure’ of the solution. That is, given a set of atoms, the number of incomplete structures is equal to \( r(r-1) \ldots (r-i+1) \). This number grows exponentially with the number of atoms \( i \) and thus leads to a computationally demanding solution. Note that pruning iteratively the structures that are incompatible might be a solution, but we have preferred to adopt here another approach.

6.1.2 Our approach

We consider here an alternative approach where the algorithm naturally evolves towards selecting a structure promoting atom, without keeping explicitly in memory all the potential structures. We describe the intuition behind the algorithm with a simple example where we have only one semantic structure (\( R = 1 \)) of size 3 (\( r = 3 \)) (Fig. 17(a)). Our method builds up progressively a structure promoting function \( f \) which takes its values in \( \mathcal{P} \) as follows: initially, \( f \) is set to 0, then, after the first atom \( \phi_1 \) is selected, every parameter in \( \mathcal{P} \) that is related to \( \eta_1 \) by some edge is promoted. That is, let \( \alpha_1, \ldots, \alpha_6 \) be defined by: \( \alpha_1 = \eta_1 \circ E_{1\rightarrow 2}, \alpha_2 = \eta_1 \circ E_{1\rightarrow 3}, \alpha_3 = \eta_1 \circ E_{2\rightarrow 1}, \alpha_4 = \eta_1 \circ E_{2\rightarrow 3}, \alpha_5 = \eta_1 \circ E_{3\rightarrow 1}, \alpha_6 = \eta_1 \circ E_{3\rightarrow 2} \). The reason why \( \alpha_i \)'s are promoted is that they ‘respect’ somehow the structure as they are related to \( \eta_1 \) with one of the edges. In the sequel, we refer to any parameter that can be written as \( \phi_1 \circ E_{a\rightarrow b} \) as a neighbour of \( \eta_1 \). Thus, at iteration 1, we update \( f \) by inducing 6 spikes at the positions \( \alpha_1, \ldots, \alpha_6 \). The function \( f \) at iteration 1 is schematically illustrated in Figure 17(b). Note that in the usual case where \( \mathcal{P} = SIM (2) \), this space is 4 dimensional. We suppose here that this space is 2D for visualization purposes. At the subsequent iteration, we select the atom based on a linear combination of the usual data term (i.e., \( |\langle r_i, \phi \rangle| \) where \( r_i \) is the residual) and the function \( f \), which gives a preference to the atoms that can be with \( \eta_1 \) in the semantic structure. Let \( \phi_2 \) be the selected atom at the second iteration. The same procedure followed in the first iteration applies: we calculate \( \beta_1, \ldots, \beta_6 \), where \( \beta_1 = \eta_2 \circ E_{1\rightarrow 2}, \ldots, \beta_6 = \eta_2 \circ E_{3\rightarrow 2} \) and add spikes at positions \( \beta_i \) to \( f \). If \( \eta_2 \) happens to be equal to one of the \( \alpha_i \)'s, spikes will add up and form a high peak with an amplitude equals to twice the amplitude of a single spike (Figure 17(c)). Hence, the structure promoting term will encourage an atom parametrized by the position of this peak, which leads to a full structure.

6.2 Algorithm

In this algorithm (PSP), we select exactly one atom per iteration, as in OMP. However, the selection step of PSP differs from OMP as we add a structure promoting term to the usual data fitting term. This function \( f : \mathcal{P} \rightarrow \mathbb{R} \) is constructed progressively using the previously selected atoms: the neighbours of the newly selected atom are promoted when no structure is detected, whereas the neighbours of the atoms in the detected structure are denoted when a structure is detected. The estimate of the image is then computed by projecting the image on the space spanned by the selected atoms at all previous iterations and the residual is updated.

Formally, PSP algorithm starts by initializing the residual to \( r_0 = s \). At iteration \( 1 \leq i \leq S \), the atom maximizing the combination of the data fitting term and the structure term is selected:

\[
k_i \leftarrow \arg \max_k \{ |\langle r_{i-1}, \phi_k \rangle| + \lambda f_{i-1}(\gamma_k) \},
\]
where $\lambda$ dictates the relative importance between the data fitting and the structure term.

The second step consists in detecting whether the newly selected atom permits to form a structure. This is done by checking whether there exists some subset of the previously determined parameters and involving $\gamma_{k_i}$ that exactly fits into a structure. We put as a separate procedure this function that returns a subset of the previously selected parameters that fits into one (or more) structures. For readability, we put the pseudocode of this function in Appendix 5.

$$S \leftarrow \text{DETECTSTRUCTURE}([\gamma_{k_1}, \ldots, \gamma_{k_{i-1}}, \gamma_{k_i}], G).$$

If $S$ is non empty, then there exists a subset of $\{\gamma_{k_1}, \ldots, \gamma_{k_{i-1}}, \gamma_{k_i}\}$ and involving $\gamma_{k_i}$ that fits into a structure. We ‘clean’ $f$ by removing the spikes in $f$ that correspond to the neighbours of the parameters in the detected structure.

$$g \leftarrow \text{SPIKESNEIGHBOURS}(S, G),$$
$$f_i \leftarrow f_{i-1} - g.$$ 

where $\text{SPIKESNEIGHBOURS}$ is a function that constructs a structure promoting function which promotes every neighbour of each parameter in $S$. Recall that the neighbours of $\eta \in P$ are calculated as follows:

$$\eta \circ E^m_{a \rightarrow b},$$

for all $(a, b) \in \{1, \ldots, r\}^2$ s.t. $a \neq b$ and $m \in \{1, \ldots, R\}$ and $E^m_{a \rightarrow b}$ denotes the edge from nodes $a$ to $b$ in structure $m$. The pseudocode of function $\text{SPIKESNEIGHBOURS}$ can be found in Appendix 5.

When no structure is detected (i.e., $S$ is empty), we promote the neighbours of $\gamma_{k_i}$:

$$g \leftarrow \text{SPIKESNEIGHBOURS}(\gamma_{k_i}, G),$$
$$f_i \leftarrow f_{i-1} + g.$$
Then, the estimate of the signal is computed and the residual updated as in standard OMP:

\[
\begin{align*}
\Omega_i & \leftarrow \Omega_{i-1} \cup \{k_i\} \\
\hat{x}_i|_{\Omega_i} & \leftarrow D_{\Omega_i}^\dagger s, \hat{x}_i|_{\Omega_i} \leftarrow 0 \\
r_i & \leftarrow s - D\hat{x}_i
\end{align*}
\]

A scheme block of the PSP algorithm that highlights its differences with OMP is illustrated in Fig 18.

Figure 18: Scheme blocks of OMP and PSP algorithms. We suppose that the signal is \(L\) decomposable, and that each structure has \(r\) atoms. OMP and PSP both need to perform \(rL\) iterations in order to recover the support. The same notations as in Algorithm 4 are used. \(a_i\) denotes the orthogonal projection of \(s\) onto the space spanned by atoms indexed with \(\Omega_i\).

The detailed procedure of PSP algorithm can be found in Algorithm 4.

Note that at each iteration, we only need to check for structures that involve the atom selected at the current iteration, as structures not involving this atom would have been detected at previous iterations. Note also that the algorithm reduces to OMP when \(\lambda = 0\).

6.3 Toy examples

We give several simple examples to illustrate how the algorithm works in practice. We consider a two-dimensional space \(P\) parametrizing the position in the plane of the generating function. The mother function is taken to be equal to a square of size 3x3 pixels. We consider an image which is composed of a single cross (Fig.(19(a))) and we assume that this cross is exactly the prior semantic structure that we have (this semantic structure is represented using a complete graph of 12 nodes). Using the proposed algorithm, we determine the correct position of the cross.
Algorithm 4 Progressive structure promoting algorithm

Input: signal \( s \), sparsity \( S \), dictionary \( D \), complete graph semantic structures \( \mathcal{G} = \{G_1, \ldots, G_R\} \), amplitude parameter \( \lambda \)

Output: sparse representation \( \hat{x} \)

1. Initialization: \( r_0 \leftarrow s \), \( \hat{x}_0 \leftarrow 0 \), \( f_0 \) a zero-valued function from \( \mathcal{P} \rightarrow \mathbb{R} \), \( \Omega_0 \leftarrow \emptyset \).
2. While \( 1 \leq i \leq S \), do:
   2.1 Selection step:
   
   \[ k_i \leftarrow \arg \max_k \{ |\langle r_{i-1}, \phi_{\gamma_k} \rangle| + \lambda f_{i-1} (\gamma_k) \} \]

   2.2 Detect structures involving \( \gamma_k \): \( \mathcal{S} \leftarrow \text{DETECTSTRUCTURE} (\{\gamma_k, \ldots, \gamma_{k-1}, \gamma_k\}, \mathcal{G}) \)

   2.3 If \( \mathcal{S} \) is nonempty, compute the spikes at the neighbours of \( \mathcal{S} \): \( g \leftarrow \text{SPIKESNEIGHBOURS} (\mathcal{S}, \mathcal{G}) \) and remove them from the structure function \( f_i \):
   
   \[ f_i \leftarrow f_{i-1} - g \]

   2.4 Else, generate the spikes at the neighbours of \( \gamma_k \): \( g \leftarrow \text{SPIKESNEIGHBOURS} (\gamma_k, \mathcal{G}) \) and add them to the structure function \( f_i \):
   
   \[ f_i \leftarrow f_{i-1} + g \]

2.5 Update step:

   \[ \Omega_i \leftarrow \Omega_{i-1} \cup \{k_i\} \]

   \[ \hat{x}_{i|\Omega} \leftarrow D^\dagger_{\Omega} s, \hat{x}_{i|\Omega^c} \leftarrow 0 \]

   \[ r_i \leftarrow s - D \hat{x}_i \]

3. Return the sparse representation \( \hat{x} \leftarrow \hat{x}_S \)

---

Figure 19: Estimated image and shape of the structure promoting function for the first iterations. \( \lambda = 0.4 \)
Fig. 19 shows the shape of the structure promoting function $f$ as well as the recovered image, for the first 3 iterations. As the algorithm progresses, the shape of the structure promoting function naturally evolves towards a map which encourages the correct position of the structure in the image. Initially, the map contains a lot of low amplitude spikes (Fig. 19(c)). At the second iteration, newly created spikes coincide with some of those created at the first iteration (as a correct atom is chosen) and results in higher peaks at these positions (Fig. 19(e)). The positions of those peaks correspond to more ‘reliable’ atoms and thus are further promoted than others. Note that still after the second iteration, two cross positions are equally likely. The third atom however favors the correct position and leads to global maximas for the correct structure (Fig. 19(g)).

We illustrate in Fig. 20 the result of denoising an image constituted from 3 crosses with unknown positions and corrupted with white gaussian noise. We run the algorithm for different values of $\lambda$. When $\lambda = 0$ (OMP), we get incomplete and damaged structures as the noise managed to fool OMP in selecting energetic but incorrect dictionary elements. Increasing $\lambda$ leads to more accurate cross structures. This is confirmed by the calculation of the Mean Square Error in Fig. 20(h) which decreases till $\lambda = 0.5$. When $\lambda$ exceeds 0.5, the structure term gains the upper hand over the data term and the algorithm will give more importance in respecting the structure than fitting the atom to the noisy data. This can be problematic when wrong atoms are detected at the early stages of the algorithm, which leads to complete ‘wrong’ structures. In this example, when $\lambda = 0.65$, the leftmost cross is ‘constructed’ at a wrong position that is 1 pixel away from the correct one. This leads to a sharp increase in the MSE and explains the behaviour of Fig. 20(h).

This example suggests that the parameter $\lambda$ should be chosen carefully (neither too high because the algorithm will not respect the data, nor too small because the structure will not be taken into account) in order to obtain satisfactory results. This choice should obviously depend on the number of atoms in a semantic structure. A good choice perhaps would be to take $\lambda r$ of the same order of magnitude as the data term, because it leads to peaks which have an amplitude comparable to the data term whenever $r$ spikes build up together.

Figure 20: Toy example with 3 semantic structures in a noisy image. The denoising was done using the proposed algorithm with different values of peak amplitudes $\lambda$.

### 6.4 Computational complexity

Algorithm 4 involves 5 steps. We provide here the computational complexity of each of these steps:

- **Selection step (2.1):** The selection step involves the calculation of $K$ inner products of vectors belonging to $\mathbb{R}^N$ and the evaluation of function $f$ $K$ times. The computational complexity of this step is dominated by the computation of the inner products, which costs $O(KN)$ operations.

- **Detection of structures involving $\gamma_k$ (2.2):** This step costs at most $O(r^2R)$ iterations, since every possible pair of nodes in any complete structure is considered.

- **Calculation of the positions of the spikes to remove, when a structure is detected (2.3):** As the cardinality of the set $\mathcal{S}$ is upper bounded by $\hat{S}$, and we loop through all edges in any semantic structure, this step costs $O(r^2RS)$ operations.

- **Generation of the spikes associated to the neighbours of $\gamma_k$ (2.4):** This step costs $O(r^2R)$ operations.

- **Update step (2.5):** The update step is mainly governed by the calculation and inversion of the Gram Matrix $D^*_{\Omega_2}D_{\Omega_3}$. The calculation can be done from the previously computed matrix $D^*_{\Omega_2}D_{\Omega_3}$, and thus
costs at most $O(SN)$ operations. Then, the Gram matrix can be inverted with a complexity of roughly $O(S^2)$ by reusing the computations of the last iteration. The exact cost depends however on the chosen method.

Note that step (2.3) is only executed whenever a structure is detected. In most cases, this will not happen at each iteration, but only a small number of times with respect to the sparsity $S$. If we don’t take into account this step, then the bottleneck of the algorithm would be either at the selection step or at steps (2.2) and (2.4). If $r^2R \ll KN$, then, the algorithm will roughly have the same computational complexity as OMP. However, in case of many large structures, the complexity would be controlled by the $r^2R$ term. In practical cases, the value of $r$ should remain small (typically less than 10) in order to have semantic groups which are flexible with respect to the image. Thus, if the number $R$ is small with respect to $\frac{KN}{100}$, the complexity of our algorithm is roughly the same as OMP.

### 6.5 Example and comparison with OMP and GOMP

We provide a simple denoising example on handwritten digits to illustrate the benefits of PSP over OMP and GOMP. We define two semantic structures: ‘0’ digit and ‘1’ digit. As in section 5.7.2, we generate the semantic structures from the images representing the digits by applying OMP with a fixed sparsity. Note that in this example, we only have 2 semantic structures (i.e., 2 digits and we consider only one style of handwriting).

Suppose now that for some reason, the image is not exactly decomposable into those structures. Instead, we have an image composed of one ‘0’, one ‘1’, and one ‘3’ (note that semantic structure ‘3’ was not predefined beforehand) (Fig 21(a)). Our task is to denoise the image in Fig 21(b). We apply successively OMP, GOMP (without coefficients) and PSP on the noisy image and we obtain respectively the results shown in Fig 21(c), Fig 21(d) and Fig 21(e).

![Figure 21: Denoising using OMP, GOMP (without coefficients) and PSP. OMP: $S = 30$, GOMP: $L = 3, w = 50$, PSP: $S = 30, \lambda = 0.05$](image)

Without any prior on the structures, OMP estimates a scattered image because of the noise. GOMP, on the contrary fits by force the structures into the image. As structure ‘3’ was not predefined, it results in fitting a rotated ‘1’. In between, PSP softly tries to fit into the image the prior structures. The algorithm manages to fit the ‘0’ and ‘1’ but did not manage to fit any structure onto the ‘3’. Hence, in this case, the structure term is neglected and the algorithm follows the data term which results in a recognizable shape of the digit ‘3’. This can also be beneficial in classification applications since GOMP detects one ‘0’ and two ‘1’ whereas PSP only detects two structures: one ‘0’ and one ‘1’ (the ‘3’ is not detected as a particular structure) which can be seen as a more accurate solution in some applications.

This example illustrates that PSP is more robust than GOMP to a model mismatch. The fact that we do not select a structure at each iteration but rather select atoms and try to form structures with these makes the algorithm more flexible. This is extremely important as we are unlikely to know all structures composing a natural image beforehand. Hence, softly constructing a structure in the image is more reasonable than trying to fit structures by force. Note however that if the image exactly respects the model, GOMP with large enough might perform better than PSP especially for small values of $\lambda$.

### 6.6 Discussion and extension to approximate structures

The first limitation of this algorithm concerns the extra parameters $\lambda$ that needs to be correctly set. The algorithm is indeed sensitive to this parameter and performance deteriorates quickly with a wrong configuration. If $\lambda$ is too small, the algorithm will take no account of the structure and thus reduces to traditional OMP. However, if $\lambda$ is too large, the algorithm gives priority to the structure over the data fitting, which may lead to recovered images which are quite far from the original one.

Another drawback in the algorithm is the use of only one structure function $f$ for all semantic structures. If the number of prior semantic structures $R$ is large, spikes of different semantic structures might interfere and thus promote structures that are not correct. One way to avoid interference between the semantic structures is the use of a different structure function for each of these. That is, let $f_i^m$ be the structure function at iteration...
i corresponding to semantic structure \( m \). Spikes corresponding only to edges in \( G_m \) would be added to the function \( f_1^m \). The structure function \( f_i \) would then be equal to the pointwise maximum of \( f_1^i \), ..., \( f_R^i \).

Besides, it would be interesting to investigate the extension of this algorithm to approximate structures. In other words, the algorithm would not only promote the exact structures given as a prior but also approximate structures. This might be interesting since we are unlikely to have the exact structures composing the image ahead of time. Introducing variability in the structures is undoubtedly an important and challenging point in order to apply such techniques on real images. As we have mentioned in the problem formulation, notions of approximate structures and approximate fitting are not rigorously defined and thus are outside the scope of this report. It is indeed necessary to first define clearly a metric on the structures in order to correctly capture the meaning of approximation. We provide here some ideas to extending this algorithm to ‘approximate’ structures.

The basic intuition would be to replace the spikes of the above algorithm with a shape which promotes not only a particular position in the space \( \mathcal{P} \), but also its neighbours. To do so, we can use a Gaussian function. Depending on the variance (or covariance matrix) of the Gaussian, the neighbourhood would be more or less promoted. Note that spikes form a particular case where the variance equals 0. As expected, the major difficulty in such an approach would be to define when a structure is detected. That is, how far can it be from the original structure in order to be classified as ‘detected’. This is indeed a key step in our algorithm since it can perturb all subsequent iterations: if a complete structure is not correctly detected, the ‘spikes’ (or Gaussians) will not be cleared and the algorithm might end up choosing wrong atoms. One way to define approximate detection is to relax the equality constraint in our structure detection technique to a proximity constraint. Formally, we would say that a structure \( G_m \) involving \( \gamma_k \) is detected in \( \{\gamma_1, \ldots, \gamma_k\} \) if there exists a set \( \epsilon = \{\epsilon_1, \ldots, \epsilon_r\} \subset \mathcal{P} \) and \( y \in \{1, \ldots, r\} \) such that:

\[
\forall z \in \{1, \ldots, r\}, \gamma_k \circ E_{\gamma_{y\rightarrow z}}^m \text{ is inside the ellipsoid centered at } \epsilon_z \text{ and of radii } (a_1, \ldots, a_d). \tag{23}
\]

where \( d \) is the dimension of the parameter space \( \mathcal{P} \) and \( a_1, \ldots, a_d \) are parameters which control the notion of proximity. This detection criterion is illustrated in Fig 22. The parameters \( a_1, \ldots, a_d \) defining the proximity are related to the variances \( \sigma_1, \ldots, \sigma_d \) of the Gaussian function. Indeed, if \( \sigma \) is nonzero in a particular direction, adding a local maxima at \( \gamma \) implies that we promote \( \gamma \) as well as its neighbourhood of size \( \sigma \) in this direction, which in turn means that variations of structures are allowed up to a factor \( \sigma \) following this direction. Hence, it is natural to take into account the same parameters in the promotion of the atoms as well as in the detection of the structures. This notion of approximate detection is however not the finest since the proximity takes into account only \( r - 1 \) edges. Furthermore, using such an approach introduces yet another parameter (the covariance matrix) to be set by the user. As a future work, it might be interesting to precisely define approximate fitting and adapt the above ideas to handle the chosen characterization of approximate structures.
7 Conclusion and future work

In this project, we have introduced a new framework for modeling transformation-invariant structures of geometric atoms. The main motivation behind the definition of such structures stems from image processing: in this context, structures should naturally obey transformation-invariance constraints since the transformed versions of meaningful objects or patterns refer to the same semantic information. In particular, we studied in detail the case where structures are invariant under translation, rotation and scaling. The basic intuition behind our proposed transformation-invariant structures follows from the observation that when objects are transformed, the cohesion (i.e., relative transformations) between the constituents of the object remains the same. Following this intuition, we formalized the definition of such structures using graphs and we proposed two alternative representations (tree graph and complete graph). We illustrated the benefits of this framework by proposing two algorithms that solve the structured sparse recovery problem and we have shown illustrative applications in image denoising and pattern recognition. Moreover, we have highlighted the fact that under structured sparsity priors, the proposed algorithms outperform OMP that only imposes priors on sparsity.

There is still plenty of work to be done in this direction: first of all, we can extend the proposed framework to include the notions of approximate structures. In real scenarios, we are unlikely to have structures in the image that obey exactly the ones given as priors. In order to do so, we need to design a norm that efficiently captures the notions of distance between structures. Ideally, this distance should have a ‘semantic’ flavour that encodes how much the ‘meaning’ of one structure differs from another one. Armed with this notion of approximation, the proposed algorithms will need to be adapted (or new ones considered) to take into account such a notion. One major drawback of the proposed iterative greedy algorithms is that the sparsity/block sparsity should be known beforehand. Instead, we might want to consider a relaxation of the original problem in order to solve the problem directly. Another point that could be interesting to consider is to study the structured sparse recovery problem without discretization of the dictionary. The dictionary would be in some sense learnt under the prior that the dictionary is of parametric nature. Finally, the performance of a structured sparse recovery algorithm heavily depends on the prior structures. Hence, an efficient strategy for learning transformation invariant structures is a crucial step towards obtaining effective structured sparse recovery algorithms.

8 Acknowledgments

I would like to thank Prof. Pascal Frossard for his guidance and assistance during the project. I also wish to thank Sofia Karygianni for the fruitful discussions and Hemant Tyagi for his help in linear algebra.
A Structure model augmented with coefficients

Definition 8 (Complete graph definition of semantic structures augmented with coefficients). Let $\Gamma = \{ \gamma_1, \ldots, \gamma_r \}$ be a subset of $P$ and $\{ c_1, \ldots, c_r \}$ be a subset of $\mathbb{R}$. The complete graph semantic structure of $\{ (\gamma_1, c_1), \ldots, (\gamma_r, c_r) \}$ is the complete graph $G = (V, E)$ where:

- $V = \{1, \ldots, r\}$
- $E = \{ E_{i\rightarrow j}, (i, j) \in \{1, \ldots, r\}^2 \}$ with $E_{i\rightarrow j} = \left( T(\gamma_i, \gamma_j), \frac{c_j}{c_i} \right)$ and $i$ and $j$ are the labels of the nodes.

Definition 9 (Tree definition of semantic structures augmented with coefficients). Let $\Gamma = \{ \gamma_1, \ldots, \gamma_r \}$ be a subset of $P$ and $\{ c_1, \ldots, c_r \}$ be a subset of $\mathbb{R}$. We consider without loss of generality that $c_1 = \max_{i \in \{1, \ldots, r\}} c_i$ (if this is not the case, relabel the $\gamma_i$s and $c_i$s). The tree semantic structure of $\{ (\gamma_1, c_1), \ldots, (\gamma_r, c_r) \}$ is the tree $G = (V, E)$ where:

- $V = \{1, \ldots, r\}$
- $E = \{ E_{1\rightarrow j}, j \in \{1, \ldots, r\} \}$ with $E_{1\rightarrow j} = \left( T(\gamma_1, \gamma_j), \frac{c_j}{c_1} \right)$ where 1 and $j$ denote the labels of the nodes.

The reason behind considering the root node as the node with maximal coefficient will become clear in section 5.6.

Definition 10 (Fitting between a set and a semantic structure augmented with coefficients). Let $\{ (\gamma_1, c_1), \ldots, (\gamma_r, c_r) \} \subset P \times \mathbb{R}$ and $G$ be a semantic structure generated from $\{ (\gamma_1', c_1'), \ldots, (\gamma_r', c_r') \}$. We say that $\{ (\gamma_1, c_1), \ldots, (\gamma_r, c_r) \}$ fits to $G$ if $\{ (\gamma_1, c_1), \ldots, (\gamma_r, c_r) \}$ and $\{ (\gamma_1', c_1'), \ldots, (\gamma_r', c_r') \}$ are semantically similar. Equivalently,

- If $G = (V, E)$ is a complete graph semantic structure, then $\{ (\gamma_1, c_1), \ldots, (\gamma_r, c_r) \}$ fits to $G$ if there exists a permutation $\chi$ of $\{1, \ldots, r\}$ that satisfies the following equalities for all $i, j$ in $\{1, \ldots, r\}$:

$$T(\gamma_i, \gamma_j) = E_{\chi(i)\rightarrow\chi(j)}(1)$$

$$\frac{c_j}{c_i} = E_{\chi(i)\rightarrow\chi(j)}(2)$$

where $E(1)$ and $E(2)$ denote respectively the first and second components of the edge augmented with the coefficients.

- If $G = (V, E)$ is a tree semantic structure, then $(\Gamma, c)$ fits to $G$ if there exists a permutation $\chi$ of $\{1, \ldots, r\}$ that satisfies the following equalities for all $j$ in $\{2, \ldots, r\}$:

$$T(\gamma_{\chi(1)}, \gamma_{\chi(j)}) = E_{1\rightarrow j}(1)$$

$$\frac{c_{\chi(j)}}{c_{\chi(1)}} = E_{1\rightarrow j}(2)$$

Definition 11 (Decomposability with semantic structures augmented with coefficients). We say that $\{ (\gamma_1, c_1), \ldots, (\gamma_r, c_r) \}$ is L-decomposable into $G_1, \ldots, G_R$ if there exists $\beta_1, \ldots, \beta_L$ subsets of $P \times \mathbb{R}$ such that the following conditions are satisfied:

1. $\{ \beta_1, \ldots, \beta_L \}$ forms a partition of $\{ (\gamma_1, c_1), \ldots, (\gamma_r, c_r) \}$.
2. For all $i$, $\beta_i$ fits at least into one of the semantic structures $G_1, \ldots, G_R$. 
B Proofs of the GOMP algorithm

B.1 Intrastructure coherence

We prove that if \( P = SIM(2) \), we have \( \langle \phi_{\eta\gamma}, \phi_{\eta\gamma'} \rangle = \langle \phi_{\gamma}, \phi_{\gamma'} \rangle \) for any \( \gamma, \gamma', \eta \) in \( P \). We suppose that the considered dictionary contains atoms which vary slowly so that:

\[
\langle \phi_{\gamma}, \phi_{\gamma'} \rangle \approx \int \int \phi_{\gamma}(x, y) \phi_{\gamma'}(x, y) dx dy.
\]

In other words, we discard the effect of discretizing the atoms. In the sequel we consider that the above approximation is an exact equality.

If \( \eta = (\hat{b}, a, \theta) \), the function \( \phi_{\eta\gamma} \) can be expressed in terms of \( \phi_{\gamma} \) in the following way:

\[
\phi_{\eta\gamma} = \frac{1}{a} \phi_{\gamma} \left( \frac{r_{\theta}}{a} [x - b_x, y - b_y]^* \right).
\]

The term \( \frac{1}{a} \) is a normalizing constant. The same formula holds for \( \phi_{\eta\gamma'} \). Hence, we have:

\[
(\phi_{\eta\gamma}, \phi_{\eta\gamma'}) = \frac{1}{a^2} \int \int \phi_{\gamma}(F(x, y)) \phi_{\gamma'}(F(x, y)) \det(J_F(x, y)) dx dy.
\]

Let be \( F \) be defined as follows: \( \mathbb{R}^2 \rightarrow \mathbb{R}^2, F(x, y) = \frac{r_{\theta}}{a} [x - b_x, y - b_y]^* \). It is easy to check that the Jacobian matrix of \( F \) is equal to

\[
J_F(x, y) = \frac{r_{\theta}}{a}.
\]

Thus, \( |\det(J_F(x, y))| = \frac{1}{a} \) and we are now ready to perform the change of variable:

\[
(\phi_{\gamma}, \phi_{\gamma'}) = \int \int \phi_{\gamma}(x, y) \phi_{\gamma'}(x, y)
= \int \int \phi_{\gamma}(F(x, y)) \phi_{\gamma'}(F(x, y)) |\det(J_F(x, y))| dx dy
= \frac{1}{a^2} \int \int \phi_{\gamma}(\frac{r_{\theta}}{a} [x - b_x, y - b_y]^*) \phi_{\gamma'}(\frac{r_{\theta}}{a} [x - b_x, y - b_y]^*) dx dy
= (\phi_{\eta\gamma}, \phi_{\eta\gamma'})
\]

We thus deduce that \( \mu_{BS} \) is independent of the considered set \( \Gamma_C \) which fits into \( G \). Note that the intrastructure coherence also equals:

\[
\mu_{BS} = \max_{h \in \{1, ..., KR\}} \max_{\phi_{\gamma} \in M_h, \phi_{\gamma} \neq \phi_{\gamma'}} \sum_{\phi_{\gamma'} \in M_h, \phi_{\gamma'} \neq \phi_{\gamma}} |\langle \phi_{\gamma}, \phi_{\gamma'} \rangle|.
\]

B.2 Proof of theorem 3

The proof of this theorem is inspired from the proof of theorem 3.7 in [27]. We divide it into two steps:

**Step 1.** Let \( h_l \) be the index of the estimated block at iteration \( l \) (i.e., \( 1 \leq h_l \leq KR \)). Let \( a_l \) be the orthogonal projection of \( s \) onto the space spanned by atoms indexed with \( \Omega_l \). That is, \( a_l = D\hat{x}_l \). Hence, for any \( l \), \( \|r_l\|_2 \) is less than \( \|s - u\|_2 \) where \( u \) is any vector in the span of atoms indexed by \( \Omega_l \). Thus, in particular, we have:

\[
\|r_l\|^2 \leq \|s - a_l - M_{h_l}M_{h_l}^* r_l - r_l\|^2
= \|r_l - M_{h_l}M_{h_l}^* r_l\|^2,
\]

as the atoms in the columns of \( M_{h_l} \) are included in the atoms indexed by \( \Omega_l \).

Expanding this last expression, we have:

\[
\|r_l - M_{h_l}M_{h_l}^* r_l\|^2 = \|r_l - M_{h_l}r_l\|^2 + \|M_{h_l}M_{h_l}^* r_l\|^2 - 2\|M_{h_l}r_l\|^2
\leq \|r_l - M_{h_l}r_l\|^2 - 2\|M_{h_l}r_l\|^2
\leq (\|r_l\|^2 - 2\|M_{h_l}r_l\|^2),
\]

where \( \|\cdot\|_{(2,2)} \) denotes the spectral norm of a matrix. We know that for any matrix \( A, \|A\|^2_{(2,2)} = \lambda_{\max}(A^*A) \).

Using Gershgorin’s disk theorem on the matrix \( M_{h_l}^*M_{h_l} \), we know that the eigenvalues of this matrix verify:

\[
|1 - \lambda| \leq \max_{\phi_{\gamma} \in M_{h_l}, \phi_{\gamma} \neq \phi_{\gamma'}} \sum_{\phi_{\gamma'} \in M_{h_l}, \phi_{\gamma'} \neq \phi_{\gamma}} |\langle \phi_{\gamma}, \phi_{\gamma'} \rangle|
\leq \mu_{BS},
\]

40
as the matrix contains 1 on the diagonals (since atoms are normalized) and the scalar products between the atoms in $M_{h_l}$ at the other entries. Thus, $\lambda \leq 1 + \mu_{IS}$ and since it is valid for any eigenvalue $\lambda$, $\|M_{h_l}\|_{(2,2)}^2 \leq 1 + \mu_{IS}$.

Hence, we obtain the following bound:

$$
\|r_{l-1} - M_{h_l}^* M_{h_l} r_{l-1}\|_2^2 \leq \|r_{l-1}\|_2^2 - 2\|M_{h_l}^* r_{l-1}\|_2^2 + (1 + \mu_{IS})\|M_{h_l}^* r_{l-1}\|_2^2
$$

$$
\leq \|r_{l-1}\|_2^2 - \|M_{h_l}^* r_{l-1}\|_2^2(1 - \mu_{IS}).
$$

We finally have:

$$
\|r_l\|_2^2 \leq \|r_{l-1}\|_2^2 - \|M_{h_l}^* r_{l-1}\|_2^2(1 - \mu_{IS}).
$$

Thus, at each iteration, the norm of the residual decreases at least by $\|M_{h_l}^* r_{l-1}\|_2^2(1 - \mu_{IS})$.

**Step 2.**

We suppose in the rest that $\mu_{IS} < 1$. Thus, $1 - \mu_{IS} > 0$ and we thus wish to obtain a lower bound on $\|M_{h_l}^* r_{l-1}\|_2^2$.

We do so by writing this series of equations:

$$
\|r_{l-1}\|_2^2 = \langle r_{l-1}, r_{l-1} \rangle = \langle s - a_{l-1}, r_{l-1} \rangle = \langle s, r_{l-1} \rangle = \left\langle \sum_{k=1}^{L} M_k x_k, r_{l-1} \right\rangle
$$

$$
= \sum_{k=1}^{L} \langle M_k x_k, r_{l-1} \rangle
$$

$$
= \sum_{k=1}^{L} x_k^* M_k^* r_{l-1}
$$

$$
= \sum_{k=1}^{L} \langle x_k, M_k^* r_{l-1} \rangle
$$

$$
\leq \frac{1}{\|x\|_2^2} \sum_{k=1}^{L} \|x_k\|_2 \|M_k^* r_{l-1}\|_2
$$

$$
\leq \max_{k \in \{1, \ldots, L\}} \|M_k^* r_{l-1}\|_2 \leq \max_{k \in \{1, \ldots, K R\}} \|M_k^* r_{l-1}\|_2 \|x\|_2
$$

$$
\leq \|M_{h_l}^* r_{l-1}\|_2 \|x\|_2
$$

(a) is obtained by using the fact that $a_{l-1}$ is orthogonal to $r_{l-1}$. (b) is obtained using Cauchy Schwarz. (c) is obtained using the definition of $h_l$ (i.e., at each iteration we choose the index $h_l$ of maximal energy). Thus, $\|M_{h_l}^* r_{l-1}\|_2^2 \geq \|r_{l-1}\|_2^2$. As we supposed that $1 - \mu_{IS} > 0$, we have:

$$
\|r_l\|_2^2 \leq \|r_{l-1}\|_2^2 - \frac{|r_{l-1}|^4}{\|x\|_2^2} (1 - \mu_{IS})
$$

$$
= \|r_{l-1}\|_2^2 \left( 1 - \frac{|r_{l-1}|^2}{\|x\|_2^2} (1 - \mu_{IS}) \right).
$$

We verify the condition in Lemma 3.4 from [27] with $A = \frac{\|x\|_2^4}{1 - \mu_{IS}}$. We have to verify indeed that: $\|r_0\|^2 = \|s\|^2 \leq \frac{\|s\|^2_2}{1 - \mu_{IS}}$. We do it as follows:

$$
\|s\|_2^2 = \left| \sum_{k=1}^{L} M_k x_k \right|_2 \leq \sum_{k=1}^{L} \|M_k x_k\|_2 \leq \sum_{k=1}^{L} \|M_k\|_{(2,2)} \|x_k\|_2
$$

$$
\leq \max_{k \in \{1, \ldots, L\}} \|M_k\|_{(2,2)} \sum_{k=1}^{L} \|x_k\|_2
$$

$$
\leq \max_{k \in \{1, \ldots, L\}} \|M_k\|_{(2,2)} \|x\|_{(2,1)}.
$$
Similarly as before, we have: \( \|M_k\|_{(2,2)} \leq \sqrt{1 + \mu_{IS}} \). Hence, \( \|s\|_2^2 \leq (1 + \mu_{IS}) \|x\|_{(2,1)}^2 \leq \frac{\|x\|_{(2,1)}^2}{1 - \mu_{IS}} \) as \( 0 \leq \mu_{IS} < 1 \). Applying Lemma 3.4 from [27], we finally obtain:

\[
\|r_1\|_2 \leq \frac{\|x\|_{(2,1)}}{\sqrt{(1 - \mu_{IS})}}.
\]

### B.3 Proof of Proposition 4

The coherence of \( \mathcal{D}' \) is equal to:

\[
\mu(\mathcal{D}') = \max_{(G, \gamma), (G', \gamma') \neq (G, \gamma)} \frac{\| (c_G^G)^* \Phi_{(G', \gamma')} \Phi_{(G, \gamma)} c_G^G \|_{2} \| \Phi_{(G', \gamma')} c_G^G \|_{2}}{\| \Phi_{(G, \gamma)} c_G^G \|_{2} \| \Phi_{(G', \gamma')} c_G^G \|_{2}}.
\]

Let \((G, \gamma)\) and \((G', \gamma')\) be two arbitrary elements in \( \mathcal{G} \times \mathcal{P} \), and let \( \{\eta_1, \ldots, \eta_r\} \) and \( \{\eta'_1, \ldots, \eta'_r\} \) denote respectively the parameters associated to \((G, \gamma)\) and \((G', \gamma')\). In other words, \( \eta_1 = \gamma, \eta_2 = \gamma \circ E_{1 \rightarrow 2}(1) \) and \( \eta'_1 = \gamma', \eta'_2 = \gamma' \circ E'_{1 \rightarrow 2}(1) \). Thus, by replacing this expression in \( \mathcal{D} \) denote respectively the edges of \( G \) and \( G' \).

\[
\| (c_G^G)^* \Phi_{(G', \gamma')} \Phi_{(G, \gamma)} c_G^G \|_{2} \| \Phi_{(G', \gamma')} c_G^G \|_{2} \leq \sum_{k=1}^{r} \sum_{j=1}^{r} c_k^G c_j^G \left( \phi_{\eta_k}, \phi_{\eta'_j} \right) \| \Phi_{(G, \gamma)} c_G^G \|_{2} \| \Phi_{(G', \gamma')} c_G^G \|_{2}.
\]

Let us begin with the second term:

\[
\sum_{(k,j) \in \{1, \ldots, r\}^2, \eta_k \neq \eta'_j} \left| c_k^G c_j^G \left( \phi_{\eta_k}, \phi_{\eta'_j} \right) \right| \| \Phi_{(G, \gamma)} c_G^G \|_{2} \| \Phi_{(G', \gamma')} c_G^G \|_{2} \leq \mu(\mathcal{D}) \sum_{k=1}^{r} \sum_{j=1}^{r} c_k^G \left| c_j^G \right| \| \Phi_{(G, \gamma)} c_G^G \|_{2} \| \Phi_{(G', \gamma')} c_G^G \|_{2}.
\]

Using Gershgorin’s disk theorem, we can prove that the squared singular values of \( \Phi_{(G, \gamma)} \) exceed \( 1 - \mu_{IS} \). For a complete proof, see for example Lemma 2.3 in [8]. Besides, we have the following general inequality for any matrix \( A \) and any \( x \):

\[
\sqrt{\lambda_{\min}(A^*A)} \|x\|_2 \leq \|Ax\|_2 \leq \sqrt{\lambda_{\max}(A^*A)} \|x\|_2.
\]

Using the left inequality with \( A = \Phi_{(G, \gamma)} \) and \( x = c_G^G \) we have:

\[
\|c_G^G\|_2 \leq \|\Phi_{(G, \gamma)} c_G^G\|_2 \sqrt{1 - \mu_{IS}}.
\]

Hence, \( \|c_G^G\|_2 \leq \frac{1}{\sqrt{1 - \mu_{IS}}} \) and we obtain a similar result for \((\gamma', G')\). Thus, by replacing this expression in Eq.(24) we obtain:

\[
\sum_{(k,j) \in \{1, \ldots, r\}^2, \eta_k \neq \eta'_j} \left| c_k^G c_j^G \left( \phi_{\eta_k}, \phi_{\eta'_j} \right) \right| \| \Phi_{(G, \gamma)} c_G^G \|_{2} \| \Phi_{(G', \gamma')} c_G^G \|_{2} \leq \mu(\mathcal{D}) \frac{r}{1 - \mu_{IS}}.
\]
We now bound the first term:

\[
\sum_{(k,j) \in \{1, \ldots, r\}^2 : \eta_k = \eta_j'} \frac{|c_k^G c_j^G \langle \phi_{\eta_k}, \phi_{\eta_j'} \rangle|}{\|\Phi(\Gamma, \gamma) c^G \|_2 \|\Phi(\Gamma', \gamma') c^G \|_2} = \sum_{(k,j) \in \{1, \ldots, r\}^2 : \eta_k = \eta_j'} \frac{|c_k^G c_j^G|}{\|\Phi(\Gamma, \gamma) c^G \|_2 \|\Phi(\Gamma', \gamma') c^G \|_2} \\
\leq \sum_{(k,j) \in \{1, \ldots, r\}^2 : \eta_k = \eta_j'} \frac{1}{\|\Phi(\Gamma, \gamma) c^G \|_2 \|\Phi(\Gamma', \gamma') c^G \|_2} \\
\leq \frac{1}{(1 - \mu s) \min_{G \in \mathcal{G}} \|c^G \|_2^2},
\]

where the last inequality is obtained by applying the same techniques as in the other term. As these step are valid for any \((\Gamma, \gamma)\) and \((\Gamma', \gamma')\), we obtain the desired result.

### B.4 Proof of Proposition 5

Let \((G^*, k^*) = \arg \max_{G \in \mathcal{G}, k \in \{1, \ldots, K\}} r_{i-1} \Phi(\Gamma, \gamma) c^G \|_{\|G^i \|_2^2}^2 \) and let \(k_{\text{max}}\) be the atom maximizing the scalar product with \(r_{i-1}\). Note that \(G^*\) and \(k^*\) has nothing to do with the transpose of matrix. In fact, \(G^*\) and \(k^*\) are respectively a graph and a number.

\[
k_{\text{max}} = \arg \max_{k \in \{1, \ldots, K\}} (r_{i-1}, \phi_{\gamma_k}).
\]

For readability, let \(y = r_{i-1}\) and let \(\gamma_{\text{max}} = \gamma_{k_{\text{max}}}\) and let \(\gamma^* = \gamma_{k^*}\). Because \((G^*, k^*)\) is optimal, we have:

\[
\left| \frac{\langle y, \Phi(G^*, \gamma^*) c^G \rangle}{\|\Phi(G^*, \gamma^*) c^G \|_2} \right| > \left| \frac{\langle y, \Phi(G^*, \gamma_{\text{max}}) c^G \rangle}{\|\Phi(G^*, \gamma_{\text{max}}) c^G \|_2} \right|
\]

We shall now prove that:

\[
\|\Phi(G^*, \gamma^*) c^G \|_2 = \|\Phi(G^*, \gamma_{\text{max}}) c^G \|_2.
\]

This is true by definition, the set \(\Gamma^*\) which fits into \(G^*\) and whose root atom is \(\gamma^*\) is semantically similar to the set \(\Gamma_{\text{max}}\) which fits into \(G^\ast\) and whose root atom is \(\gamma_{\text{max}}\). Thus, thanks to the conservation of scalar products by semantic similarity (Appendix B.1), we deduce that \(\Phi(G^*, \gamma^*) \Phi(G^*, \gamma^*) = \Phi(G^*, \gamma_{\text{max}}) \Phi(G^*, \gamma_{\text{max}})\).

Indeed, for any \(a\) and \(b\) in \(\{1, \ldots, r\}\):

\[
\langle \phi_{\gamma^*} \circ E_{1-a}, \phi_{\gamma^*} \circ E_{1-a} \rangle = \langle \phi_{\gamma_{\text{max}} \circ \gamma^* \circ E_{1-a}}, \phi_{\gamma_{\text{max}} \circ \gamma^* \circ E_{1-a}} \rangle = \langle \phi_{\gamma_{\text{max}} \circ \gamma^* \circ E_{1-a}}, \phi_{\gamma_{\text{max}} \circ \gamma^* \circ E_{1-a}} \rangle,
\]

where \(E\) denotes the first component of the edges of \(G^*\) (i.e., \(T(\gamma, \gamma')\)). Hence:

\[
\|\Phi(G^*, \gamma^*) c^G \|^2_2 = \|c^G\|^2 \Phi(G^*, \gamma^*) \Phi(G^*, \gamma^*) c^G = \|c^G\|^2 \Phi(G^*, \gamma_{\text{max}}) \Phi(G^*, \gamma_{\text{max}}) c^G = \|\Phi(G^*, \gamma_{\text{max}}) c^G \|^2_2.
\]

Thus, we have:

\[
\left| \langle y, \Phi(G^*, \gamma^*) c^G \rangle \right| \geq \left| \langle y, \Phi(G^*, \gamma_{\text{max}}) c^G \rangle \right|.
\]

We take the left hand side and bound it as follows:

\[
\left| \langle y, \Phi(G^*, \gamma^*) c^G \rangle \right| \leq |\langle y, \phi_{\gamma^*} \rangle| + \sum_{i=2}^r |\langle y, c_i^G \phi_{\gamma^*} \circ E_{1-i} \rangle| \\
\leq |\langle y, \phi_{\gamma^*} \rangle| + \sum_{i=2}^r |\langle y, c_i^G \phi_{\gamma^*} \circ E_{1-i} \rangle| \\
\leq |\langle y, \phi_{\gamma^*} \rangle| + |\langle y, \phi_{\gamma_{\text{max}}} \rangle| \|c^G\|_1 - 1).
\]

Similarly, we show that:

\[
\left| \langle y, \Phi(G^*, \gamma_{\text{max}}) c^G \rangle \right| \geq |\langle y, \phi_{\gamma_{\text{max}}} \rangle| - |\langle y, \phi_{\gamma^*} \rangle| \|c^G\|_1 - 1).
\]

By combining the LHS and RHS, we get:

\[
|\langle y, \phi_{\gamma^*} \rangle| \geq |\langle y, \phi_{\gamma_{\text{max}}} \rangle| (1 - 2\|c^G\|_1 - 1))
\]

Hence, by choosing \(\delta\) greater than: \((3 - 2 \max_{G \in \mathcal{G}} \|c^G\|_1)^{-1}\), we guarantee that \(k^*\) is in \(\mathcal{I}\).
C  Pseudocode of DETECT_STRUCTURE and SPIKES_NEIGHBOURS functions

The procedures of the different functions are detailed as follows:

Algorithm 5 DETECT_STRUCTURE
Input: Parameters of atoms $\eta_1, \ldots, \eta_l$, structures $G_1 = (V^1, E^1), \ldots, G_R = (V^R, E^R)$
Output: Parameters $S$ of the atoms in the detected structures

$S \leftarrow \emptyset$
for $i = 1 \rightarrow R$
do
  for all starting nodes $a \in V^i$
do
    $\text{foundStructure} \leftarrow \text{true}$
    $M \leftarrow \emptyset$
    for all node $b \in V^i, b \neq a$
do
      if $\eta_l \circ E^i_{a \rightarrow b} \notin \{\eta_1, \ldots, \eta_{l-1}\}$ then
        $\text{foundStructure} \leftarrow \text{false}$ and break
      else
        $M \leftarrow M \cup \{(\eta_l \circ E^i_{a \rightarrow b}) \cap \{\eta_1, \ldots, \eta_{l-1}\}\}$
      end if
    end for
  if $\text{foundStructure}$ then
    $S \leftarrow S \cup M$
  end if
end for

Algorithm 6 SPIKES_NEIGHBOURS
Input: Parameters of atoms $\eta_1, \ldots, \eta_l$, structures $G_1, \ldots, G_R$
Output: Function $g : \mathcal{P} \rightarrow \mathbb{R}$.

$g_0 \leftarrow 0_{\mathcal{P} \rightarrow \mathbb{R}}$
for $k = 1 \rightarrow l$
do
  $g_k \leftarrow g_{k-1}$
  for $i = 1 \rightarrow R$
do
    for all edges $E$ in $G_i$ do
      $g_k(\eta_k \circ E) \leftarrow g_{k-1}(\eta_k \circ E) + 1$ \{Put a spike at position $\eta_k \circ E$\}
    end for
    $i \leftarrow i + 1$
  end for
  $k \leftarrow k + 1$
end for
$g \leftarrow g_l$

44
References


