



**HAL**  
open science

# Fast orthogonal sparse approximation algorithms over local dictionaries

Boris Mailhé, Rémi Gribonval, Pierre Vandergheynst, Frédéric Bimbot

► **To cite this version:**

Boris Mailhé, Rémi Gribonval, Pierre Vandergheynst, Frédéric Bimbot. Fast orthogonal sparse approximation algorithms over local dictionaries. 2010. hal-00460558v1

**HAL Id: hal-00460558**

**<https://inria.hal.science/hal-00460558v1>**

Submitted on 1 Mar 2010 (v1), last revised 26 Apr 2011 (v3)

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Fast orthogonal sparse approximation algorithms over local dictionaries

Boris Mailhé<sup>a,b,\*</sup>, Rémi Gribonval<sup>c</sup>, Pierre Vandergheynst<sup>d</sup>, Frédéric Bimbot<sup>a,e</sup>

<sup>a</sup>IRISA, Rennes, France

<sup>b</sup>Université de Rennes 1, France

<sup>c</sup>Centre de recherche INRIA Rennes Bretagne Atlantique, France

<sup>d</sup>École Polytechnique Fédérale de Lausanne, Switzerland

<sup>e</sup>Centre National de la Recherche Scientifique, France

---

## Abstract

In this work we present a new greedy algorithm for sparse approximation called LocOMP. LocOMP is meant to be run on local dictionaries made of atoms with much shorter supports than the signal length. This notably encompasses shift-invariant dictionaries and time-frequency dictionaries, be they monoscale or multiscale. In this case, very fast implementations of Matching Pursuit are already available. LocOMP is almost as fast as Matching Pursuit while approaching the signal almost as well as the much slower Orthogonal Matching Pursuit.

*Keywords:* sparse approximation, greedy algorithms, shift invariance, Orthogonal Matching Pursuit

---

## 1. Introduction

One basis is not enough to represent certain kinds of signals. For example, music is known to be well represented by time-frequency decompositions, but which window length to choose? There are longer and shorter notes, and inside a note played by a free oscillation instrument such as a guitar or piano there is a short attack followed by a long relaxation. If one can find a proper basis to represent each of those phenomena, then one would like to use the union of all

---

\*Corresponding author

these bases to represent the whole signal. This union of bases is not a basis itself but a *redundant dictionary*  $\Phi \in \mathbf{R}^{N \times D}$  with  $D > N$ : given a signal  $s \in \mathbf{R}^N$ , there are infinitely many possible choices of coefficients  $x \in \mathbf{R}^D$  that decompose  $s$  as  $s = \Phi x$ . Sparsity has been proposed as a way to solve the ambiguity of such redundant models by selecting the decomposition that contains the fewest non-zero coefficients.

*Local dictionaries.* The length of the signal is chosen by the user and only limited by the devices sensing and recording it, but the observed phenomena can have their own characteristic durations. So one can commonly find signals composed of several events that are each much shorter than the whole signal. This is typically the case in music, where a single signal will cover a whole piece made of much shorter notes. Good dictionaries to model such a signal will also be composed of short atoms, each atom or a few of them trying to match one of the phenomena.

**Definition 1.** Let  $\Phi$  be a dictionary of size  $D \times L$ .  $\Phi$  is said to be local if all its atoms  $\varphi$  are null outside of a support interval  $\text{supp}(\varphi)$  of length  $L \ll N$ .

Common dictionaries such as the Gabor dictionary associated with Short Term Fourier Transform (STFT) are local dictionaries. Shift-invariant dictionaries are the most employed class of local dictionaries: those are made of a few patterns of length  $L$  that are repeated all over the signal support.

The problems that require local dictionaries typically involve large dimensions: the signal can contain several millions of samples and the dictionary even more atoms. Most known sparse approximation algorithms are too complex to be applied to such large problems.

*Scope of the paper.* This article explores how the locality hypothesis can be exploited to accelerate existing sparse approximation algorithms or to propose new ones. It is focused on Matching Pursuit and Orthogonal Matching Pursuit. A fast implementation of MP is already available for shift-invariant dictionaries [12] but MP has been observed to compute significantly worse approximations than OMP. The aim of this work is to try to provide an implementation of

OMP as close as possible to the complexity of MP. As structural properties of OMP make it impossible to reach the complexity of MP on local dictionaries, we propose a new algorithm called LocOMP that is only slightly less efficient than OMP but can run almost as fast as MP.

*Organisation.* Section 2 reviews existing sparse approximation algorithms with a strong emphasis on greedy algorithm, their detailed complexity and fast implementation with local dictionaries. Section AppendixD presents possible accelerations to OMP implementation in the local case. Even with those accelerations OMP remains much more expensive than MP.

Section 4 introduces the main contribution of this paper: the new algorithm LocOMP. The behaviour of LocOMP is close to OMP but it enables the same accelerations as MP. Section 6 presents experimental evaluations of Matlab implementations of the presented algorithms on music signals. These experiments show that LocOMP can achieve the same performance as OMP within the same order of computation time as MP. Section 7 presents future possible extensions and theoretical developments of this work.

## 2. State of the art: sparse approximation algorithms

In this section we briefly review some of the most common greedy algorithms with a strong emphasis on their complexity. Let  $s$  be a signal of length  $N$  and  $\Phi$  a dictionary of size  $N \times D$ .  $\Phi$  is redundant if  $D > N$  and we define its redundancy factor  $\alpha = \frac{D}{N}$ . A signal  $s$  is said to be sparse over  $\Phi$  if  $s$  can be approached closely by an decomposition  $\Phi x$  where  $x$  contains few non-zero coefficients:

$$s = \Phi x + r \tag{1}$$

$$\|r\|_2 \ll \|s\|_2 \tag{2}$$

$$\|x\|_0 \ll N \tag{3}$$

with  $\|x\|_0 = \sum_{d=1}^D x_d^0$  the number of non-zero coefficients in  $x$ . The columns of  $\Phi$  are called *atoms*. The problem is to find the sparse decomposition among all

the possible ones.

**Problem 1.** *Given a signal  $s$  and a dictionary  $\Phi$  and an allowed sparsity level  $K$ , find the coefficients  $x$  that minimize*

$$\hat{x} = \operatorname{argmin}_{\|x\|_0 \leq K} \|s - \Phi x\|_2^2 \quad (4)$$

The sparse approximation problem (4) is proved to be NP-Hard [1]. Yet many suboptimal algorithms have been proposed which can compute a close approximation within reasonable time.

### 2.1. $\ell_p$ minimization

The  $\ell_0$  sparsity measure has very bad properties for numerical optimization: it is not convex, not differentiable and piecewise constant.  $\ell_p$  minimization algorithms replace the  $\ell_0$  measure by other constraints that are easier to optimize, often  $\ell_p$  pseudo norms with  $0 < p \leq 1$ .

$\ell_p$  minimization is easier to compute than  $\ell_0$  from a mathematical point of view.  $\ell_p$  pseudo-norms are continuous and piecewise differentiable so variational approaches can lead to a local minimum. In the special case of  $p = 1$ , the norm is even convex, so there is unique local minimum and there are algorithms to find it. However those approaches remain costly, especially when dealing with large data. In fact, even if the final result is sparse, intermediate iterations of  $\ell_p$  minimization algorithms involve computations with a non-sparse  $x$ .

### 2.2. Greedy algorithms

Greedy algorithms reduce the complexity of the sparse approximation problem (4) by ensuring that the current support is always sparse during the execution of the algorithm.

#### 2.2.1. Hard Thresholding (HT)

*Principle.* One can hardly think of a simpler algorithm. It only consists in selecting the  $K$  atoms with the highest correlations  $\|\langle s, \varphi \rangle\|$  with the signal  $s$ . The coefficient amplitudes  $x_K$  can then be obtained by projecting the signal  $s$

on the selected sub-dictionary  $\Phi_K$ :

$$\Phi_K^+ = (\Phi_K^* \Phi_K)^{-1} \Phi_K^* \quad (5)$$

$$x_K = \Phi_K^+ s \quad (6)$$

$$r = s - \Phi_K x_K \quad (7)$$

*Weaknesses.* This algorithm does not recover the closest approximation as soon as the dictionary contains atoms with high cross-correlation. This comes from the fact that as all atoms are selected simultaneously, several atoms can model the same component of the signal.

### 2.2.2. Matching Pursuit (MP)

*Principle.* MP replaces simultaneous atom selection with sequential atom selection [8]. Only one atom is selected and removed from the signal at each iteration. Algorithm 1 details the process.

---

**Algorithm 1**  $x = \text{MP}(s, \Phi), K$

---

```

 $r^{(0)} = s$ 
 $\Phi^{(0)} = \emptyset$ 
 $x^{(0)} = 0$ 
for  $i = 1$  to  $I$  do
   $\varphi^{(i)} = \text{argmax}_{\varphi \in \Phi} |\langle r^{(i-1)}, \varphi \rangle|$  {best atom selection}
   $x^{(i)} = x^{(i-1)} + \langle r^{(i-1)}, \varphi^{(i)} \rangle \delta_{\varphi^{(i)}}$  {coefficient update  $\varphi^{(i)}$ }
   $r^{(i)} = r^{(i-1)} - \langle r^{(i-1)}, \varphi^{(i)} \rangle \varphi^{(i)}$  {residual update}
end for
return  $x^{(I)}$ 

```

---

*Weaknesses.* The residual  $r^{(i)}$  is the projection of the previous residual  $r^{(i-1)}$  orthogonally to the selected atom  $\varphi^{(i)}$ . So we have  $r^{(i)} \perp \varphi^{(i)}$  for any  $i$ . If  $\Phi$  is not orthogonal, then the subtraction of an atom can bring back a correlation with a previously selected atom. So MP can select the same atom several times. Even in the noiseless case, it can take an infinite number of iterations to reach a null residual even though the complete support has already been recovered.

### 2.2.3. Orthogonal Matching Pursuit (OMP)

*Principle.* OMP prevents selecting the atom twice by ensuring that the residual  $r^{(i)}$  remains orthogonal to *all* the atoms selected so far. Let the sub-dictionary  $\Phi^{(i)} = (\varphi_j)_{1 \leq j \leq i}$ . The residual  $r^{(i)}$  is computed by projecting  $s$  (or  $r^{(i-1)}$ , which gives the same result) orthogonally to  $\Phi^{(i)}$  [9]. Algorithm 2 details the process.

---

**Algorithm 2**  $x = \text{OMP}(s, \Phi)$

---

```

 $r^{(0)} = s$ 
 $\Phi^{(0)} = \emptyset$ 
 $x^{(0)} = 0$ 
for  $i = 1$  to  $I$  do
   $\varphi^{(i)} = \operatorname{argmax}_{\varphi \in \Phi} |\langle r^{(i-1)}, \varphi \rangle|$  {best atom selection}
   $\Phi^{(i)} = [\Phi^{(i-1)}, \varphi^{(i)}]$ 
   $\chi^{(i)} = (\Phi^{(i)*} \Phi^{(i)})^{-1} \Phi^{(i)*} r^{(i-1)}$  {projection computation}
   $x^{(i)} = x^{(i-1)} + \chi^{(i)}$  {coefficient update}
   $r^{(i)} = r^{(i-1)} - \Phi^{(i)} \chi^{(i)}$  {residual update}
end for
return  $x^{(I)}$ 

```

---

*Weaknesses.* The selections of suboptimal atoms OMP makes are mostly due to its greedy nature. An atom is selected based only on the residual known at the current iteration. Once it has been selected, there is no way to remove it. This can lead to early selections of suboptimal atoms that will never be corrected. Recent algorithms such as CoSaMP [10] and Iterative Hard Thresholding [7] attempt to fix this by adding a backtracking reaction to the pure greedy scheme: slightly more atoms than  $K$  are selected, then the decomposition is pruned from the smallest coefficients.

### 2.2.4. Gradient Pursuit (GP)

GP replaces the projection step of OMP with a single gradient descent, which leads to an approximate but faster quasi-OMP [11]. The coefficient update  $\chi^{(i)}$  is given by

$$\chi^{(i)} = -\rho^{(i)} \nabla x^{(i)} \quad (8)$$

where  $\nabla x^{(i)} = -2\Phi^{(i)*}r^{(i-1)}$  is the gradient of the residual error to minimize in Equation (4) and  $\rho^{(i)}$  the optimal step of this descent:

$$\rho^{(i)} = \operatorname{argmin}_{\rho \in \mathbf{R}} \left\| r^{(i-1)} - \rho \Phi^{(i)} \nabla x^{(i)} \right\|_2^2 = -\frac{\|\Phi^{(i)*}r^{(i-1)}\|_2^2}{2\|\Phi^{(i)}\Phi^{(i)*}r^{(i-1)}\|_2^2} \quad (9)$$

### 3. Complexity of existing greedy algorithms

This section summarizes the complexity of greedy algorithms both with arbitrary dictionaries and with structured dictionaries that allow fast transforms. More detailed explanations can be found in Appendix A. The complexity depends on the signal length  $N$ , the number of atoms in the dictionary  $D$ , and the number of non-zero coefficients  $K$ . Under the redundant dictionary and sparse decomposition hypotheses, we have  $K \ll N < D$ .

#### 3.1. Using general dictionaries

The cost of each step of HT, MP, OMP and GP is summarized in Table 1. The main observations one can draw from these figures are that:

- the Gram matrix computation is surprisingly more expensive than its inversion;
- the main cost remains the correlation computation in  $O(DKN)$ .

*Gram matrix inversion.* The low cost for inverting the Gram matrix comes from the sparsity hypothesis  $K \ll N$ . An inversion seems more difficult than a simple matrix/vector product, but the size the Gram matrix  $G^{(i)}$  to invert is bounded by  $K \times K$  whereas the sub-dictionary  $\Phi^{(i)}$  can be as large as  $K \times N$ . The Gram matrix is small, but its computation involves scalar products of long vectors.

*Comparison of MP and OMP.* MP is strictly cheaper than OMP because its selection step is the same and its projection step is simpler. However, in the general case, both MP and OMP projection steps are cheaper than the selection step. So both MP and OMP end up in the same complexity class. As GP complexity lies between GP and OMP, it does not appear to bring any significant gain over OMP in that case.



Table 1: Detailed complexity of greedy algorithms in the general case, depending on the signal length  $N$ , number of atoms  $D$  and sparsity  $K$ , under the hypotheses  $K \ll N < D$ .

Algorithm	Thresholding	MP	OMP	GP
$\lambda = \Phi^* r$	$DN$	$DN$	$DN$	$DN$
$\operatorname{argmax}  \lambda $	$D \log D$	$D$	$D$	$D$
$G^{(i)} = \Phi^{(i)*} \Phi^{(i)}$	$K^2 N$	$\emptyset$	$iN$	$iN$
$\chi = G^{-1} \lambda^{(i)}$	$K^2$	$\emptyset$	$i^2$	$i$
$r^{(i)} = r^{(i-1)} - \Phi^{(i)} \chi$	$KN$	$N$	$iN$	$iN$
Cost per iteration	$DN + K^2 N$	$DN$	$DN$	$DN$
Number of iterations	1	$\gtrsim K$	$K$	$\gtrsim K$
TOTAL	$DN + K^2 N$	$DKN$	$DKN$	$DKN$

Table 2: Detailed complexity for greedy algorithms using a fast dictionary, depending on the signal length  $N$ , number of atoms  $D$  and sparsity  $K$ , under the hypotheses  $K \ll N < D$ .

Algorithm	HT	MP	OMP	GP
$\lambda = \Phi^* r$	$D \log N$	$D \log N$	$D \log N$	$D \log N$
$\operatorname{argmax}  \lambda $	$D \log D$	$D$	$D$	$D$
$G = \Phi^{(i)*} \Phi^{(i)}$	$K \log N$	$\emptyset$	$i \log N$	$i \log N$
$\chi = G^{-1} \lambda^{(i)}$	$K^2$	$\emptyset$	$i^2$	$i$
$r^{(i)} = r^{(i-1)} - \Phi^{(i)} \chi$	$D \log N$	$N$	$D \log N$	$D \log N$
Cost per iteration	$D \log D + K^2$	$D \log N$	$D \log N + i^2$	$D \log N$
Number of iterations	1	$\gtrsim K$	$K$	$\gtrsim K$
TOTAL	$D \log D + K^2$	$DK \log N$	$DK \log N + K^3$	$DK \log N$

### 3.2. Complexity using fast dictionaries

The application of  $\Phi^*$  during the selection step might be the most expensive step of greedy algorithms in the general case, but in practice this cost is often avoided. There are many known bases that allow fast analysis (application of  $\Phi^*$ ) and synthesis (application of  $\Phi$ ). For example the Fast Fourier Transform can compute both of those operations in  $O(N \log N)$  for a Fourier basis of size  $N$ .

Let the redundant dictionary  $\Phi$  be a union of such fast bases. Let  $\alpha$  be the number of bases. Then  $D = \alpha N$  and the global cost for analysis and synthesis is  $O(\alpha N \log N) = O(D \log N)$ . This makes the selection step, thus MP, much faster. As the cost remains the same for the OMP projection, it becomes more costly than the selection cost if  $K > \sqrt{D \log N}$ , which might be reached or not depending on the considered data. These results are summarized in Table 2.

### 3.3. MPTK: fast MP implementation for shift-invariant dictionaries

A shift-invariant dictionary is a dictionary made of atoms of size  $L \ll N$  shifted at different positions in the signal. In that case, a scalar product only costs  $O(L)$  to compute. If the dictionary is a union of fast “bases” of  $L$  atoms that perform an analysis in  $O(L \log L)$ , then the cost for the whole dictionary analysis is  $O(D \log L)$ .

Moreover, the residual only changes on an interval of length  $L$  between two consecutive MP iterations. Let  $\text{supp}(\varphi) = [t_{\min}(\varphi), t_{\max}(\varphi)]$  be the support of an atom. Then  $r^{(i)} = r^{(i)}$  outside of  $\text{supp}(\varphi^{(i)})$ . So any atom with a disjoint support from  $\text{supp}(\varphi^{(i)})$  has the same correlation with  $r^{(i)}$  and  $r^{(i-1)}$ . So at the selection step of iteration  $i + 1$ , only the correlations of atoms which support overlaps  $\text{supp}(\varphi^{(i)})$  have to be computed again, the other ones being the same as during the previous iteration  $i$ . **The update of the residual is local, which makes the update of the correlations also local.** If the atoms of  $\Phi$  have a uniform distribution in time, there are about  $D \frac{L}{N}$  correlations to recompute. So the complete analysis is only performed on the first iteration. After that, only a partial analysis in  $O(L \log L)$  is required. This acceleration is presented in [8].

The highest correlation is also easier to find: if 2 correlations have not changed, then their comparison has not changed either. Comparisons can be stored in a tournament tree. This enables to find the best atom in  $O(\log D)$  and also decreases the storage cost..

These improvement are the core of the MPTK library<sup>1</sup> [12] that decreases the complexity of MP from  $O(DK \log L)$  to  $O((D + KL) \log L)$ .

### 3.4. Conclusion

The speed gap between MP and OMP gets even wider when working with local dictionaries, as MP can be implemented very efficiently. This article presents a new algorithm called LocOMP that achieves the same approximation quality

---

<sup>1</sup><http://mptk.irisa.fr>

as OMP while remaining in the complexity class of MPTK when working with local dictionaries. As OMP has structural properties that prevent a fast implementation (see AppendixD for more detail), LocOMP only approaches the behaviour of OMP.

#### 4. LocOMP: Local Orthogonal Matching Pursuit

##### 4.1. General description

The speed gap between MP and OMP for local dictionaries comes from the different costs to compute the correlations  $\Phi^*r$  (see AppendixD for more detail). MP complexity is low because the residual only changes on a short interval of length  $L$  at each iteration. As the cost of an iteration is linked to the length of this interval, this length is the parameter we need to control to obtain a fast algorithm.

LocOMP does so by projecting the residual on a subset of  $\Phi^{(i)}$  that only contains atoms “close” to the last selected atom  $\varphi^{(i)}$  in the time domain. Algorithm 3 describes the simplified process (without the accelerations previously described in Sections 3.3 and AppendixD).

The algorithm uses a function **neighbour** that computes a *sub-dictionary*  $\Psi^{(i)} \subset \Phi^{(i)}$  on which the residual is to be projected.  $\Psi^{(i)}$  of course contains the last selected atom  $\varphi^{(i)}$ . Different possible choices for neighbour define a gradual progression from MP to OMP:

- MP is given by the choice  $\Psi^{(i)} = \varphi^{(i)}$ ;
- OMP is given by the choice  $\Psi^{(i)} = \Phi^{(i)}$ .

The sub-dictionary  $\Psi^{(i)}$  will generally not contain all the atoms of  $\Phi^{(i-1)}$  that would need a coefficient update in *OMP*, so LocOMP is only an approximation of OMP.

##### 4.2. LocGP algorithm

LocGP selects a sub-dictionary as LocOMP does, then updates the coefficients with a single gradient descent as GP does instead of a complete least-square minimization. It is detailed in Algorithm 4.

---

**Algorithm 3**  $x = \text{LocOMP}(s, \Phi)$ 

---

$r_0 = s$   
 $\Phi_0 = \emptyset$   
 $x_0 = 0$   
**for**  $i = 1$  to  $I$  **do**  
     $\varphi^{(i)} = \text{argmax}_{\varphi \in \Phi} |\langle r^{(i-1)}, \varphi \rangle|$  {best atom selection}  
     $\Phi^{(i)} = \Phi^{(i-1)} \cup \varphi^{(i)}$   
     $\Psi^{(i)} = \text{neighbour}(\Phi^{(i)}, \varphi^{(i)})$  {sub-dictionary selection}  
     $\chi^{(i)} = (\Psi^{(i)*} \Psi^{(i)})^{-1} \Psi^{(i)*} r^{(i-1)}$  {projection}  
     $x^{(i)} = x^{(i-1)} + \chi^{(i)}$  {coefficient update}  
     $r^{(i)} = r^{(i-1)} - \Psi^{(i)} \chi^{(i)}$  {residual update}  
**end for**  
**return**  $x^{(i)}$

---

---

**Algorithm 4**  $x = \text{LocGP}(s, \Phi)$ 

---

$r_0 = s$   
 $\Phi_0 = \emptyset$   
 $x_0 = 0$   
**for**  $i = 1$  to  $I$  **do**  
     $\varphi^{(i)} = \text{argmax}_{\varphi \in \Phi} |\langle r^{(i-1)}, \varphi \rangle|$  {best atom selection}  
     $\Phi^{(i)} = \Phi^{(i-1)} \cup \varphi^{(i)}$   
     $\Psi^{(i)} = \text{neighbour}(\Phi^{(i)}, \varphi^{(i)})$  {sub-dictionary selection}  
     $\chi^{(i)} = \frac{\|\Psi^{(i)*} r^{(i-1)}\|^2}{\|\Psi^{(i)} \Psi^{(i)*} r^{(i-1)}\|^2} \Psi^{(i)*} r^{(i-1)}$  {Gradient computation}  
     $x^{(i)} = x^{(i-1)} + \chi^{(i)}$  {coefficient update}  
     $r^{(i)} = r^{(i-1)} - \Psi^{(i)} \chi^{(i)}$  {residual update}  
**end for**  
**return**  $x^{(i)}$

---

As a side effect of the neighbourhood selection, the Gram matrix to invert at each iteration is very small, only a  $O(\frac{iL}{N})$  square. Thus there is little speed gain to expect in the application of a faster projection. However we will see in Section 7.1 that LocGP has other appealing properties for practical implementation.

### 4.3. Complexity

As for MPTK the first iteration is expensive: it requires a full correlation computation in  $O(D \log L)$ . Let  $T \geq L$  be the length of the interval over which the residual changes at each iteration. For any iteration  $i > 1$ , the selection step only requires the correlation computation of  $r^{(i-1)}$  with  $\alpha(T + 2L - 2)$  atoms of  $\Phi$ , which can be performed in  $O(\alpha T \log L)$ . Then the search for the highest correlation is done as in MPTK and the Gram matrix computation  $G^{(i)}$  as in OMP (a faster computation for local dictionaries is described in AppendixD). The sub-matrix  $\Psi^{(i)*}\Psi^{(i)}$  is then extracted from  $G^{(i)}$ . It contains about  $i\frac{T}{N}$  atoms. We chose to use a complete conjugate gradient descent to solve the projection problem for a cost in  $O\left(i^2\frac{T^2}{N^2}\right)^2$ . Finally the residual update can be performed in  $O(T \log L)$ .

If the algorithm is only run for a few iterations, then the main cost is the cost  $O(D \log L)$  of the first iteration, as for MPTK. If run for a large number of iterations, the main cost becomes the correlation computation time in  $O(T \log L)$  per iteration.  **$T$  has to remain close to  $L$  to ensure that LocOMP complexity remains close to that of MP.** Those results are summarized in Table 3.

## 5. Selection of the sub-dictionary $\Psi^{(i)}$

The choice of  $\Psi^{(i)}$  controls both the quality of the approximation and the cost of LocOMP. It has to contain many atoms to provide a good approximation while keeping the length of the residual change  $T = |\text{supp}(\Psi^{(i)})|$  small.

---

<sup>2</sup>Pati's inversion method cannot be applied because the selected sub-dictionary  $\Psi^{(i)}$  changes at each iteration.

Table 3: Detailed complexity of MP, OMP and LocOMP with fast local dictionaries, depending on the signal length  $N$ , number of atoms  $D$  and sparsity  $K$ , under the hypotheses  $K \ll N < D$  and  $L \ll N$ .

Algorithm	MPTK	OMP	LocOMP/LocGP
$\lambda(1) = \Phi^* s$	$D \log L$	$D \log L$	$D \log L$
$m^{(1)} = \operatorname{argmax}  \lambda^{(1)} $	$D$	$D$	$D$
$\lambda^{(i)} = \Phi^* r^{i-1}$	$\alpha L \log L$	$D \log L$	$\alpha T \log L$
$m^{(i)} = \operatorname{argmax}  \lambda^{(i)} $	$\log(D)$	$D$	$\log(D)$
$\Phi^{(i)} = \Phi^{(i-1)} \cup \{\varphi^{(i)}\}$	1	$\log i$	$\log i$
$G^{(i)} = \Phi^{(i)*} \Phi^{(i)}$	0	$L \log L$	$L \log L$
$\Psi^{(i)}$ selection	0	0	$\log i$
$\chi^{(i)} = \Psi^{(i)+} r^{(i-1)}$	0	$\frac{i^2 L}{N}$	$\frac{i^2 T^2}{N^2}$
$r^{(i)} = r^{(i-1)} - \Psi^{(i)} \chi^{(i)}$	$L$	$D \log L$	$T \log L$
Cost per iteration	$\alpha L \log L$	$D \log L + \frac{i^2 L}{N}$	$T \log L$
TOTAL	$\alpha(N + KL) \log L$	$DK \log L + \frac{K^3 L}{N}$	$\alpha(N + TK) \log L$

### 5.1. Choosing $\operatorname{supp}(\Psi^{(i)})$ is choosing $\Psi^{(i)}$

As the cost of LocOMP mostly depends on  $T$ , the objective of the sub-dictionary selection is to select as many atoms as possible in a given residual change interval. **If the residual change interval is fixed to be  $I = [t_{\min}, t_{\max}]$ , then the best possible sub-dictionary is the exhaustive sub-dictionary  $\hat{\Psi}^{(i)}$  that contains all the atoms of  $\Phi^{(i)}$  whose support is included in  $I$ .** Any other admissible sub-dictionary  $\Psi$  contains less atoms than  $\hat{\Psi}^{(i)}$ , so it provides a worse approximation for an equivalent computation cost. So one only has to select an time interval  $I$  around the last selected atom. Then we define the sub-dictionary

$$\Psi^{(i)} = \left\{ \varphi \in \Phi^{(i)} \mid \operatorname{supp}(\varphi) \subseteq I \right\} \quad (10)$$

### 5.2. Choice of $I$

*Monoscale case.* The selected sub-dictionary  $\Psi^{(i)}$  should at least contain all the atoms of  $\Phi^{(i-1)}$  correlated with the last atom  $\varphi^{(i)}$  to ensure a behaviour close to that of OMP. If this is not the case, it could happen that two correlated

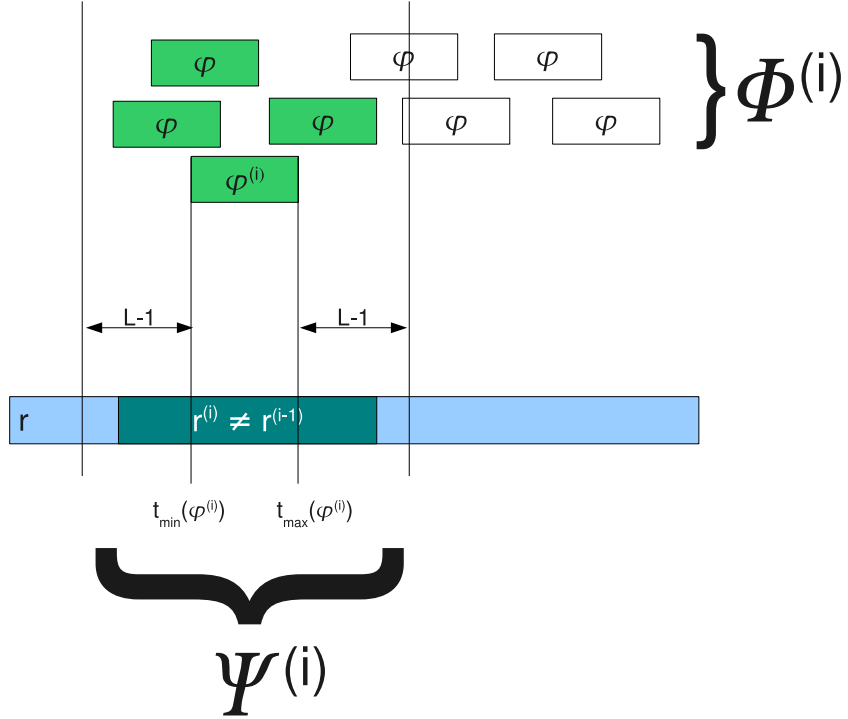


Figure 1: Selection of the sub-dictionary  $\Psi^{(i)}$ . All the atoms of  $\Phi^{(i)}$  that overlap the last atom  $\varphi^{(i)}$  are kept. The residual only changes on an interval of length  $T \leq 3L - 2$ .

atoms are never selected together, so their correlation is neglected for the whole algorithm.

In this work, we chose the smallest choice of  $I$  that ensures this property:  $I = [t_{\min}(\varphi^{(i)}) - L + 1, t_{\max}(\varphi^{(i)}) + L - 1]$ . Then the sub-dictionary  $\Psi^{(i)}$  is the set of all atoms belonging to  $\Phi^{(i-1)}$  that overlap  $\varphi^{(i)}$ , plus  $\varphi^{(i)}$  itself, as shown in Figure 1. The maximal length of the residual change is  $T = 3L - 2$ . This choice should lead to LocOMP being  $\frac{5}{3}$  more costly than MP.

*Multiscale case.* So far we have considered that all atoms have the same support length  $L$ . However sparsity is commonly used as a regularization criterion for multiscale models. For example the dictionary can be a union of Gabor bases

with different window length  $L_1, L_2, \dots$ . These models are frequently used in music processing when one needs fine frequency definition for the stationary parts and fine temporal definition for the transient parts.

With a multiscale dictionary, when a large scale atom is selected, then all short scale atoms that are inside its support must be selected too even if they do not overlap the last selected atom  $\varphi^{(i)}$ . This makes the computation of optimal boundaries for  $I$  more difficult. It requires to:

- find the beginning of the earliest atom that overlaps with  $\varphi^{(i)}$

$$t_{\min}^{\sim} = \min\{t_{\min}(\varphi) | \varphi \in \Phi^{(i)} \wedge t_{\max}(\varphi) > t_{\min}(\varphi^{(i)})\}$$

- find the end of the latest atom that overlaps with  $\varphi^{(i)}$

$$t_{\max}^{\sim} = \max\{t_{\max}(\varphi) | \varphi \in \Phi^{(i)} \wedge t_{\min}(\varphi) < t_{\max}(\varphi^{(i)})\}$$

- extract all atoms of  $\Phi^{(i)}$  whose support is included in  $[t_{\min}^{\sim}, t_{\max}^{\sim}]$ .  $\Psi^{(i)}$  can contain short atoms that do not overlap with  $\varphi^{(i)}$ , as shown in Figure 2.

In this work, we rather chose simple, data-independent boundaries that are looser but easier to compute. Let  $L$  be the largest scale in the dictionary. Then we define  $I$  as:

$$I = [t_{\min}(\varphi^{(i)}) - L + 1, t_{\max}(\varphi^{(i)}) + L - 1] \quad (11)$$

$$\supseteq [t_{\min}^{\sim}, t_{\max}^{\sim}] \quad (12)$$

The two intervals are equal if there are  $L$ -scale atoms in  $\Phi^{(i-1)}$  that only overlap with  $\varphi^{(i)}$  for 1 sample. If there are no short-scale atoms close to  $t_{\min}^{\sim}$  or  $t_{\max}^{\sim}$ , then the intervals can also select the same sub-dictionary  $\Psi^{(i)}$  even though they are different.

## 6. Experimental results

We evaluated LocOMP and LocGP to support our asymptotic complexity evaluations to measure their approximation quality compared to MP and OMP.



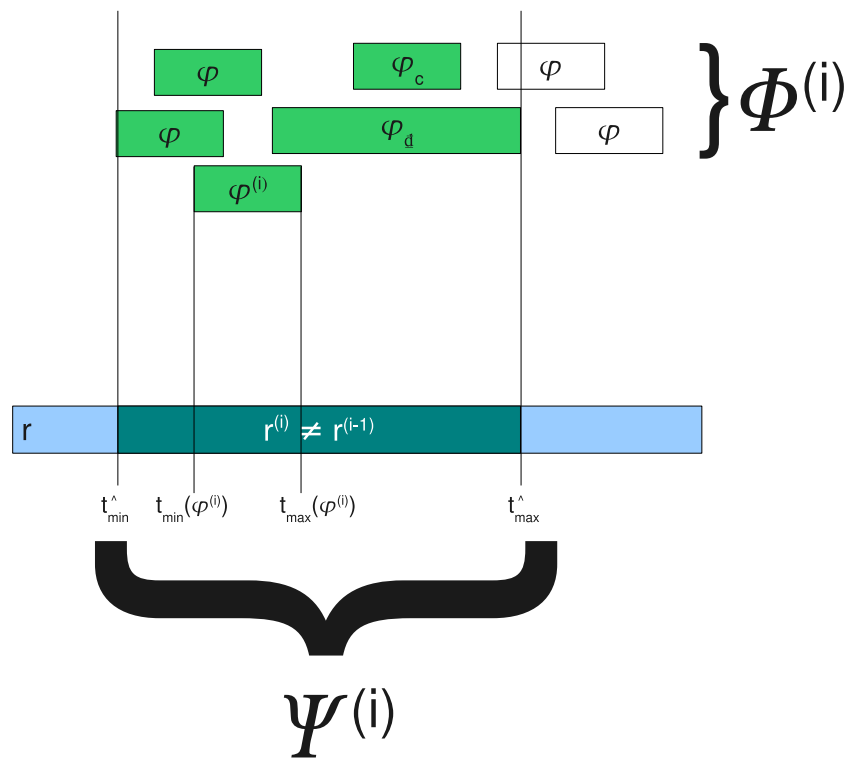


Figure 2: Selection of the sub-dictionary  $\Psi^{(i)}$  with a multiscale dictionary. The short atom  $\varphi^{(c)}$  is uncorrelated with the last atom  $\varphi^{(i)}$ , but it is kept anyway because its support is inside the support of the large kept atom  $\varphi^{(d)}$ .

We compared MATLAB implementations of the algorithms MP, LocGP, LocOMP, GP and OMP. We chose to recode all the algorithms instead of using the much faster MPTK to set all the algorithms on an equal footing.

### 6.1. Protocol

The great cost of OMP restricts the dimensions that can be handled. We chose a 1 minute music extract from the RWC base [13]. This extract was downsampled to 8000Hz, which makes a signal length of  $N = 480000$  samples. The dictionary was a fully shift-invariant MDCT dictionary of scale  $L = 32$ . It contains about  $D = 15 \cdot 10^6$  atoms.

All algorithms were run for  $I = 20000$  iterations.

### 6.2. Effective computation time

Figure 3 plots the cumulated computation time needed by each algorithm depending on the number of iterations. One can clearly see that OMP and GP are much slower than MP, LocGP and LocOMP. The figures had to be plotted in log – log scale so that all curves fit in the same plot.

It took about 5 days to global projection algorithms (OMP et GP) to complete the 20000 iterations whereas MP finished its run in only 10 minutes and both LocOMP and LocGP only required about 15 minutes. This confirms numerically that **LocOMP remains in the same order of cost as MP**.

The slope of the curves towards the first iteration also provides interesting insight. The curves of local algorithms start horizontally because the first iteration is much more expensive than the other ones. So the cost to run one iteration or a few ones is almost the same.

For global projection algorithms the slope towards the first iteration shows a linear behaviour in the log / log coordinates: at the beginning of the algorithm, OMP or GP have almost nothing else to do than recomputing the correlations again and again. When  $i$  grows, the projection step becomes more noticeable and the curves drift above their initial tangent. This tangent is a lower bound for the cost of global projection algorithms: even if one could compute the

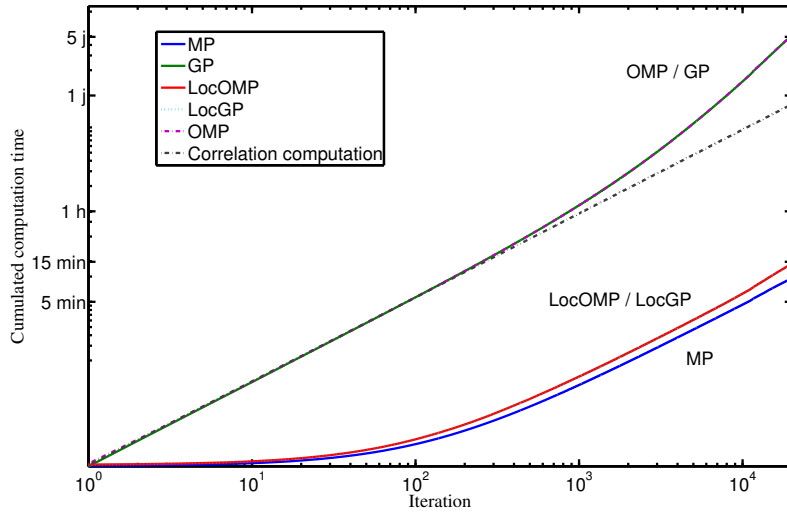


Figure 3: Cumulated computation time spent by different algorithms depending on the iteration. Local algorithms (MP, LocOMP and LocGP) are much faster than global ones (OMP and GP). The LocOMP and LocGP curves cannot be told apart on this plot, neither can OMP and GP. The dashed tangent is the lower bound for any algorithm that computes the correlations at each iteration.

projection at no cost, a global algorithm could not cost less. This confirms the interest of local updates.

### 6.3. Approximation quality

Figures 4 show the approximation quality defined as

$$SNR^{(i)} = -10 \log \frac{\|r^{(i)}\|_2^2}{\|s\|_2^2} \quad (13)$$

depending on the iteration.

The different curves are hard to distinguish on the original curve (left). Only MP seems to provide significantly lower quality, all other curves are mixed.

To get a closer look, we used OMP, that is presumably the best performing algorithm under this experiment, as a reference. The right curve shows the difference between the SNR achieved by OMP and the SNR achieved by each algorithm. One can see that the final loss of MP is equal to 0.6dB. LocOMP

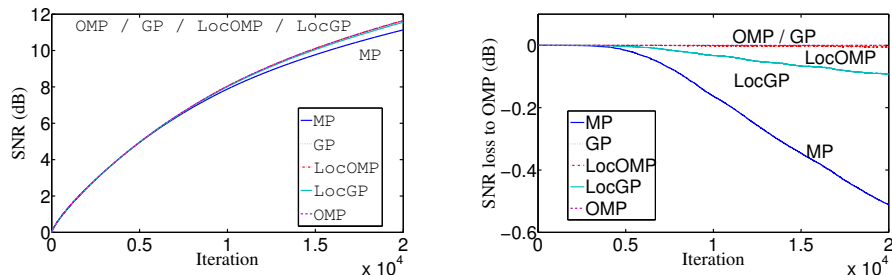


Figure 4: Approximation SNR obtained by several algorithms depending on the iteration. All the algorithms perform similarly apart from MP. The right plot shows the SNR loss compared to OMP.

ends up 0.01dB lower than OMP and LocGP 0.09dB lower. OMP and GP achieved the same quality on this experiment.

This confirms that the local update strategy, which only *approximates* OMP, can provide almost as good performance as the much more expensive, complete OMP, while remaining in the order of complexity of MP. The higher quality loss of LocGP compared to LocOMP is still not explained. The algorithmic link is the same between OMP and GP on one side and LocOMP and LocGP on the other side. As OMP and GP share the same behaviour, one could expect LocOMP and LocGP to do the same.

On these experiments, the overall approximation quality of all algorithms, including OMP, is limited, with only 11dB reached after 20000 iterations. The quality difference between MP and OMP is accordingly small. This is mainly due to the choice of a small, short-scale dictionary. This choice was driven by the will to provide a comparison with OMP, so the dictionary had to be small enough so that we could actually afford to run OMP and GP.

More promising, although still preliminary, results are displayed in the next section with larger dictionaries. They show that LocGP provides a substantial quality gain over MP.

## 7. Perspectives

### 7.1. MPTK implementation

An implementation of LocGP in the MPTK library is currently under development. A prototype is already running, but it is still much slower than expected.

We chose LocGP for software engineering reasons. As MPTK uses fast dictionaries and is programmed in C++, it currently does not use any matrix computations nor does it not have native access to. The simple expression of the gradient in LocGP makes it possible to implement it without having to link MPTK with an external matrix library. As LocOMP seems to achieve significantly better quality, its implementation is also targeted in the long term.

*Obtained quality.* We compared our prototype implementation of LocGP with the MP implementation in MPTK. Only these two algorithms could be compared since C++ implementations of other algorithms were not available (and not worth developing) for OMP and GP. We still used 1 minute music signals downsampled to 8 kHz but this time we used a multiscale MDCT dictionary with scales  $L_1 = 32$  and  $L_2 = 1024$ , which amount to 4ms and 128ms. These scales roughly correspond to the time windows used for AAC audio compression. Both algorithms were run for 20000 iterations.

Figure 5 shows the SNR depending on the iteration. We observe that **LocGP brings a average gain of 2dB over MP**, which looks promising.

*Computation time.* The obtained computation times are, however, disappointing. MP finished the whole computation in less than 10 minutes, whereas it almost took one day to run LocGP. Profiling of the LocGP program hints at a great time loss during residual update.

MPTK uses fast dictionaries. Correlations are computed using an FFT-based fast analysis algorithm. In MP, an atom that is selected at iteration  $i$  is only used at this iteration when it is removed from the residual, and also maybe if it is selected again later. Because of that the residual update has never

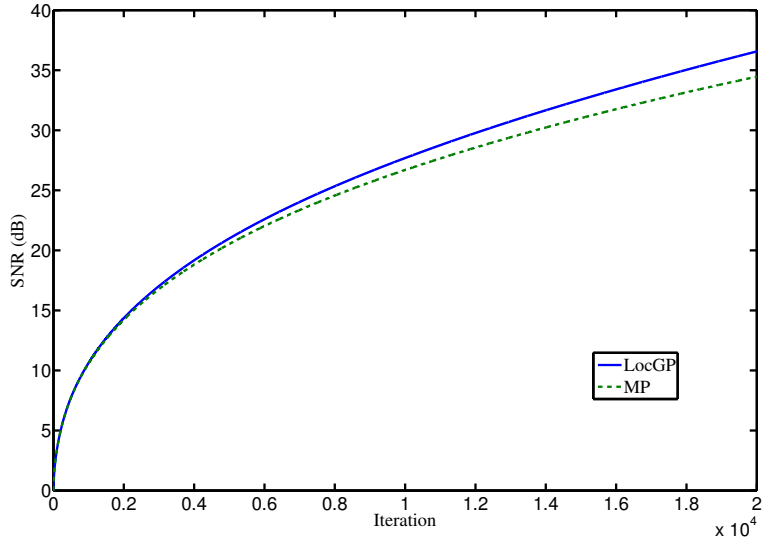


Figure 5: Average approximation obtained by MP and LocGP depending on the iteration. The averages have been computed over 10 piano signals downsampled to 8 kHz. The decompositions used MDCT dictionaries with scale 32 and 1024. LocGP provides an average gain of 2dB over MP.

been a problem for MPTK and fast synthesis has not been implemented in the blocks. To subtract an atom from the residual, its waveform is synthesized from its analytical definition and the subtraction is carried sample-wise. On the contrary, LocGP subtracts every atom of the sub-dictionary  $\Psi^{(i)}$  at each iteration. The synthesis of the waveforms over and over again seems to be the cause of the poor speed: LocGP spends its time computing cosines to generate oscillating waveforms. A caching mechanism is under development.

### 7.2. Extension to multidimensional signals

LocOMP as described only applies to temporal series. However, local or shift-invariant dictionaries are also used in signal processing. It would be interesting to extend LocOMP to this case.

To do so, one needs to define the sub-dictionary  $\Psi^{(i)}$ . The notion of support overlapping is not specific to unidimensional signals: two atoms overlap if they have at least one atom in common. So the criteria to define  $\Psi^{(i)}$  are still valid.

However  $\Psi^{(i)}$  might be harder to extract from  $\Phi^{(i)}$ . In the unidimensional case, the fast extraction is based on sorting the atoms (see Appendix AppendixC for more detail). If the dimension of the signal grows, the location of the atoms is not provided by a single instant anymore but by a vector of coordinates. As **the multidimensional coordinate spaces have no total ordering that preserves locality**, we will need to find another way to extract  $\Psi^{(i)}$ .

### 7.3. Theoretical guarantees

LocOMP was designed to ensure that its complexity remains within MP range, and its quality should lie somewhere between MP and OMP. However we currently have no theoretical guarantee about that and we do not know when the quality will be closer to MP than OMP, even if experimental results are promising.

This is a difficult problem that is probably wider than LocOMP, as the cases where MP and OMP diverge are not well studied yet. Many theoretical guarantees on greedy algorithms apply to both MP and OMP. For example, Tropp’s Exact Recovery Condition [14] give a sufficient condition on the dictionary for greedy algorithms to recover the best decomposition possible. The ERC is valid for any greedy algorithm that:

- selects the atom with highest correlation to the residual,
- computes the new residual by subtracting a linear combination of previously selected atoms.

So the ERC is also valid for LocOMP, but this will not be helpful in practice. We would be more interested in a criterion to predict that OMP will succeed whereas MP will fail.

Moreover, most theoretical results are based on a measure of the orthogonality of the dictionary. Even the most naive algorithm, hard thresholding, works if the dictionary is an orthonormal basis. Some algorithms are proved to recover the sparsest decomposition provided the dictionary is close enough to an orthonormal basis. One of the orthogonality measures used by Tropp is the

coherence  $\mu$  defined as

$$\mu = \operatorname{argmax}_{(\varphi_1, \varphi_2) \in \Phi^2} |\langle \varphi_1, \varphi_2 \rangle| \quad (14)$$

The coherence is 0 for an orthonormal basis and dictionaries with low coherence enough satisfy the ERC [14]. However, shift-invariant dictionaries are notably bad according to this criterion. Any redundant dictionary needs to have a coherence higher than 0. As coherence is defined as a maximum, low coherence dictionaries need to spread out the cross-correlations to decrease the maximum correlation. A good low coherence dictionary is a dictionary where all cross-correlations are non-zero, but all are low. Shift-invariant dictionaries offer a completely different geometry. As seen in Section AppendixD.2, **every atom of a shift-invariant dictionary is orthogonal to almost every other atom, but the few non-zero correlations can be high**. So even if the known theoretical results can apply to LocOMP, the dictionaries that LocOMP typically uses will not satisfy the hypotheses.

## 8. Conclusion

Sparse approximation over local dictionaries requires adapted because of the large signal and dictionary dimensions that one wants to handle. MP was already known to be suitable for fast implementation over local dictionaries.

*Contributions.* We proposed several algorithmic accelerations that enable a faster OMP implementation without changing the behaviour of the algorithm. All those accelerations still do not make OMP tractable because the correlations  $\lambda = \Phi^* r$  have to be computed at each iteration.

We proposed a new algorithm, LocOMP, whose behaviour is close to OMP for a complexity that stays in the same class as MP. The key idea was to select a sub-dictionary  $\Psi^{(i)}$  containing only atoms located close to the last selected atom  $\varphi^{(i)}$ . We also provided a fast way to extract  $\Psi^{(i)}$  from  $\Phi^{(i)}$  by using a sorted index of  $\Phi^{(i)}$ .

LocOMP has shown experimentally that the approximations it computes can be as good as OMP.



*Perspectives.* We have proposed possible extensions of LocOMP, but for now the most urgent task to address is the optimization of the MPTK implementation. Theoretical complexity and experimental results show that LocOMP fills the necessary speed and quality requirements to replace MP as a tractable approximation algorithm, but it still lacks a high-performance implementation to reach its theoretical speed on real size data.

The link between the choice of  $\Psi^{(i)}$  and the obtained quality has also yet to be fully understood. Larger sub-dictionaries lead to better approximations (on one iteration at least). Quantification of this evolution might help design better sub-dictionary selection heuristics as the one used here. This problem is linked to the theoretical study hinted at in Section 7.3, as both rely on theoretical bounding of the approximation error. As we chose  $\Psi^{(i)}$  as small as possible here and still got almost as good results as OMP in this work, there might be no need for larger sub-dictionaries.

#### **Appendix A. Detailed explanations on the complexity of greedy algorithms with general dictionaries**

*Hard Thresholding.* HT requires a matrix-vector product  $\Phi^*s$  that is performed in  $O(DN)$ , then the search for the  $K$  highest correlations is lower than  $O(D \log D)$  (which is the cost to sort them all), finally the amplitude computation can be performed in  $O(NK)$ . As  $D > N \gg K$ , the main cost is the cost in  $O(DN)$  to compute the correlations.

*MP.* One MP iteration is a thresholding with  $K = 1$ , so the correlation computation in  $O(DN)$  stays the most expensive part. The number of iterations to run to recover  $K$  different atoms is unknown. If one assumes that it stays in the range of  $K$ , then the global algorithm cost is in  $O(DKN)$ .

*OMP.* OMP performs the same computations as MP, plus an orthogonal projection, at each iteration. This projection consists in computing the Gram matrix  $G^{(i)} = \Phi^{(i)*}\Phi^{(i)}$ , invert it and apply the inverse to a correlation vector that is already known from the selection step. Pati proposed a way to compute the new

inverse  $G^{(i)-1}$  from the previous one  $G^{(i-1)-1}$  [9]. The main cost in  $O(iN)$  is spent computing the new line of the Gram matrix  $\Phi^{(i-1)*}\varphi^{(i)}$ . Then the previous inverse has to be applied once, which is done in  $O(i^2)$ . This leads to a total cost in  $O(K^2N)$ , that is no more than the cost to compute only the projection on the final dictionary  $\Phi^{(K)*}\Phi^{(K)}$ .

*GP.* The gradient  $\nabla x^{(i)}$  is composed of correlations that are already known from the selection step. The computation of  $\rho$  requires one more synthesis in  $O(D \log N)$ . Then one just has to change the coefficients, which is done in  $O(i)$ . Compared to OMP, the cost in  $O(i^2)$  has been decreased to  $O(i)$ .

### Appendix B. Average proportion of overlapping atoms

When updating the Gram matrix, how many atoms are there  $\Phi^{(i-1)}$  that overlap  $\varphi^{(i)}$ ? Let us assume that  $t_{\min}(\varphi^{(i)})$  belongs to the interval  $J = [L - 1, N - 2L + 1]$ . Then

$$I = [t_{\min}(\varphi^{(i)}) - L + 1, t_{\min}(\varphi^{(i)}) + L - 1] \quad (\text{B.1})$$

and the length of  $I$  is  $2L - 1$ . If the  $t_{\min}$  of the other atoms of  $\Phi^{(i-1)}$  are supposed independent and uniformly distributed over the interval  $[0, N - L]$ , then the probability  $p$  for an atom to fall in the neighbourhood  $I$  is given by

$$p = \frac{2L - 1}{N - L + 1} \quad (\text{B.2})$$

and as there  $i - 1$  atoms in  $\Phi^{(i-1)}$  the average number of selected atoms equals

$$(i - 1)p = \frac{(i - 1)(2L - 1)}{N - L + 1} = O\left(\frac{iL}{N}\right) \quad (\text{B.3})$$

### Appendix C. Data structures for fast access to selected atoms

*Fast recovery of the connected component of  $\varphi^{(i)}$ .* The atoms of  $\Phi^{(i-1)}$  that overlap the last atom  $\varphi^{(i)}$  can be found without traversing the whole sub-dictionary  $\Phi^{(i-1)}$  by maintaining a sorted index of the atoms. If atoms are ordered with increasing  $t_{\min}$ , then one just has to find the atoms with the smallest and the

largest admissible  $t_{\min}$  and select every atom between those two. Moreover the sorted index is dynamic as a new atom is added each iteration. So the index needs to be fast at:

- inserting a new element,
- extracting a sub-index between 2 given boundaries,
- browsing that sub-index.

These criteria are fulfilled by sorted set implementation based on Red-Black trees, such as the ones used in the Java <sup>3</sup> and C++ <sup>4</sup> standard libraries. The insertion of  $t_{\min}(\varphi^{(i)})$  is performed in  $O(\log i)$  as well as the search for the boundaries, and the browsing of a sub-index containing  $Q$  elements is performed in  $O(Q)$ . This leaves a global cost for the Gram matrix computation step equal to

$$O\left(\log i + \frac{iL^2}{N}\right) = O(\kappa^2 iN) \quad (\text{C.1})$$

*Double indexing.* The Gram matrix  $G^{(i)} = \Phi^{(i)*}\Phi^{(i)}$  is a dynamic matrix that grows by one line and column per iteration. To be able to store this matrix without having to move elements, one would like the new line and column to be added at the end of the matrix. This means that two different indexes have to be maintained for  $\Phi^{(i)}$ : the increasing  $t_{\min}$  index and the increasing  $i$  index. The first index is used to find the correlated atoms with  $\varphi^{(i)}$  and the second one to access coefficients in  $G^{(i)}$ .

#### *Appendix C.1. MPTK implementation*

It has already been noted in Section Appendix C that the selection of  $\Psi^{(i)}$  can be performed efficiently thanks to a sorted time index but that the order in which atoms are added to the book must also be stored. MPTK implementation adds another constraint to the book data structure.

---

<sup>3</sup><http://java.sun.com/javase/6/docs/api/java/util/TreeSet.html>

<sup>4</sup><http://gcc.gnu.org/onlinedocs/libstdc++/libstdc++-html-USERS-4.4/a00528.html>

In MPTK, the dictionary is a collection of *blocks*. Each block is a filter bank that implements fast analysis for a given family of atoms. For example, there is one block per scale when using a multiscale STFT dictionary.

LocGP needs to know the correlations  $\Psi^{(i)*}r^{(i-1)}$  during the projection step. Theoretically, these have been computed in the previous selection step. However, most of them are not stored in MPTK as they are not useful for MP. So the useful correlations have to be detected and saved during the selection step before they are forgotten. This requires the extraction of all the atoms produced by a given block.

To do so we chose a hierarchical structure described in Figure C.6. The atoms are first sorted according to their block, then their  $t_{\min}$ , then other parameters (the frequency for STFT atoms).

This structure slows the extraction of  $\Psi^{(i)}$  a little. If there are  $B$  blocks, then the extraction has to be performed  $B$  times for a total cost of  $O(B \log \frac{i}{B})$  instead of  $O(\log i)$ . However the number of blocks is usually small. Browse and insertion times do not change significantly.

#### AppendixD. “Fast” exact OMP implementation for local dictionaries

The shift-invariant structure could be used to provide a faster implementation of OMP as it was done for MP in MPTK. We show that even with these improvements, OMP would remain much slower than MP. This will point which part of the algorithm is the most costly and worth replacing with a faster, suboptimal step.

##### AppendixD.1. Correlation computation

As for MP, the cost of a fast analysis is reduced to  $O(D \log L)$  but contrary to MP, the residual change  $r^{(i)} - r^{(i-1)}$  is global. As detailed in Algorithm 2, at iteration  $i - 1$  the coefficient update  $\chi^{(i-1)} = x^{(i-1)} - x^{(i-2)}$  is equal to

$$\chi^{(i-1)} = \left( \Phi^{(i-1)*} \Phi^{(i-1)} \right)^{-1} \Phi^{(i-1)*} r^{(i-2)} \quad (\text{D.1})$$

Some of the  $\chi^{(i-1)}$  can be null. First,  $r^{(i-2)}$  is orthogonal to  $\Phi^{(i-2)}$  so  $\Phi^{(i-1)*} r^{(i-2)} = \langle \varphi^{(i-1)}, r^{(i-2)} \rangle \delta_{\varphi, \varphi^{(i-1)}}$ . Then, if the Gram matrix has the

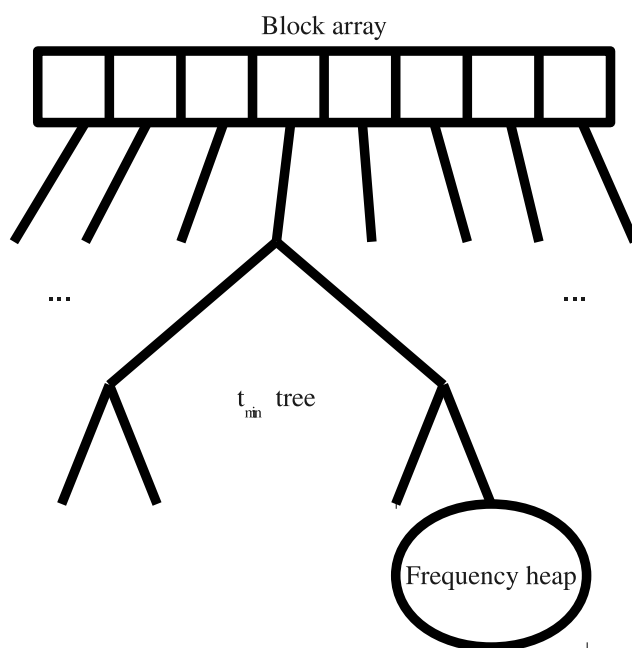


Figure C.6: MPTK data structure for fast access to the atoms. Atoms belonging to the same block and  $t_{\min}$  are in a single heap, then the heaps of the same block are sorted by growing  $t_{\min}$ , then those trees are stored in a block array.

following block structure

$$\Phi^{(i-1)*}\Phi^{(i-1)} = \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix} \quad (\text{D.2})$$

then the inverse is equal to

$$\left(\Phi^{(i-1)*}\Phi^{(i-1)}\right)^{-1} = \begin{pmatrix} A^{-1} & 0 \\ 0 & B^{-1} \end{pmatrix} \quad (\text{D.3})$$

In this case, only the coefficients of  $\chi^{(i-1)}$  belonging to the block that contains the last atom  $\varphi^{(i-1)}$  can be non-zero. Let  $\Gamma$  be the undirected graph that has the atoms of  $\Phi^{(i-1)}$  as vertexes and a link between two atoms if their correlation is non-zero. Then the blocks of the Gram matrix are the connected components of  $\Gamma$ .

Local dictionaries are more likely to provide such a block structure for the Gram matrix of the sub-dictionary  $\Phi^{(i-1)}$  (to a permutation): two atoms whose supports do not overlap are not correlated. So if there is a time  $t$  in the signal that is not inside the support of any atom of  $\Phi^{(i-1)}$ , then all the atoms either end before  $t$  or start after  $t$ . So there are at least 2 connected components in  $\Gamma$ , and only residual can only change over the time support of the component that contains the last atom  $\varphi^{(i)}$ .

But do such  $t$  always exist? If enough atoms are selected their supports can cover the whole support of  $s$ . Then  $\Gamma$  is connected and the residual changes over its whole support. This behaviour seems likely at the end of OMP.

#### *Appendix D.2. Gram matrix computation*

*Principle.* The Gram matrix  $G^{(i)} = \Phi^{(i)*}\Phi^{(i)}$  can be computed faster if the dictionary is local. First, as atoms only have a support length  $L$ , a cross-correlation can be computed in  $O(L)$  instead of  $O(N)$ . Second, less correlations are required. As two atoms whose supports do not overlap are uncorrelated, the Gram matrix is sparse and the position of its zero coefficients can be predicted.

The prediction consists in a simple on the support begin  $t_{\min}$  and end  $t_{\max}$ :

$$\text{supp}(\varphi) \cap \text{supp}(\varphi^{(i)}) \neq \emptyset \Leftrightarrow \begin{cases} t_{\max}(\varphi) \geq t_{\min}(\varphi^{(i)}) \\ t_{\min}(\varphi) \leq t_{\max}(\varphi^{(i)}) \end{cases} \quad (\text{D.4})$$

$$\Leftrightarrow \begin{cases} t_{\min}(\varphi) + L - 1 \geq t_{\min}(\varphi^{(i)}) \\ t_{\min}(\varphi) \leq t_{\min}(\varphi^{(i)}) + L - 1 \end{cases} \quad (\text{D.5})$$

$$\Leftrightarrow t_{\min}(\varphi^{(i)}) - L < t_{\min}(\varphi) < t_{\min}(\varphi^{(i)}) + L \quad (\text{D.6})$$

So an atom  $\varphi$  overlaps the last atom  $\varphi^{(i)}$  if  $t_{\min}(\varphi)$  belongs to the interval

$$I = [\max\{t_{\min}(\varphi^{(i)}) - L + 1, 0\}, \min\{t_{\min}(\varphi^{(i)}) + L - 1, N - L\}]$$

*Naive implementation.* How to find the atoms of  $\Phi^{(i-1)}$  that overlap the last atom  $\varphi^{(i)}$ ? The direct way would be to browse  $\Phi^{(i-1)}$  and to compute the scalar product for atoms that satisfy the constraint (D.6). It can be assumed that there are  $O\left(\frac{iL}{N}\right)$  atoms to be selected (see Appendix AppendixB for justification). Then the cost for this step would be

$$O\left(|\Phi^{(i-1)}| + LE^{(i)}\right) = O\left((i-1) + \frac{L(i-1)(2L-1)}{N-L+1}\right) \quad (\text{D.7})$$

$$= O\left(i + \frac{iL^2}{N}\right) \quad (\text{D.8})$$

This cost encompasses the  $O(i)$  cost of the traversal of  $\Phi^{(i-1)}$  and the computation time for selected atoms. One can see that the computation time has been reduced by a factor  $\left(\frac{L}{N}\right)^2$ . The first  $\frac{L}{N}$  factor comes from the fastest computation of one scalar product and the second from the smaller amount of scalar products to be computed. Yet, the traversal cost is not guaranteed to be smaller than the effective computation time. In Appendix AppendixC we show that this cost can be avoided by using a sorted data structure.

*Fast dictionaries.* Correlations between atoms can also computed using FFT if the dictionary enables it. FFT decreases the costs by computing the correlations with a basis of  $L$  atoms in a single step. For many fast local dictionaries such as Short Term Fourier Transform, locality is defined for a basis: all the

atoms that are processed together by the FFT share the same support. Those dictionaries can combine the fast correlation algorithm and our fast detection of zero correlations for a cost in  $O(L \log L)$ : one only needs to compute correlations between  $\varphi^{(i)}$  and the few bases that overlap it. This implementation is not always interesting: when computing the Gram matrix one only needs the atoms that:

- overlap  $\varphi^{(i)}$
- belong to  $\Phi^{(i-1)}$

The FFT will compute the correlations for a whole basis but because of the second criterion only some of them are useful. For the FFT to be faster than naive scalar product computation, one needs

$$L \log L < \frac{iL^2}{N} \quad (\text{D.9})$$

$$i > N \frac{\log L}{L} \quad (\text{D.10})$$

*Gram matrix inversion.* As seen before, the use of short, local atoms makes  $G^{(i)}$  sparse. The number of atoms that overlap the last atom  $\varphi^{(i)}$  given in Equation (B.3) is also the average number of non-zero coefficients on each line of  $G^{(i)}$ . So there are totally about  $O\left(\frac{i^2 L}{N}\right)$  non-zero coefficients in  $G^{(i)}$ . This is also the cost to invert  $G^{(i)}$  using Pati's method for example [9].

So the total cost of an OMP iteration is reduced to

$$O\left(D \log L + \frac{L \log L}{N} + \frac{i^2 L}{N}\right) \quad (\text{D.11})$$

$$= O\left(N \log L \left(\alpha + \frac{L}{N} + \frac{i^2 L}{N^2 \log L}\right)\right) \quad (\text{D.12})$$

This cost is expressed using the FFT implementation to compute the Gram matrix. As we suppose  $L \ll N$ , the computation of the Gram matrix is much faster than the best atom selection. The Gram matrix inversion cost only becomes relevant when

$$i \gtrsim N \sqrt{\frac{\alpha \log L}{L}} \quad (\text{D.13})$$



Choosing for example  $N \approx 10^6$ ,  $\alpha = 2$  and  $L = 1024$ , the inversion cost would become relevant  $i \gtrsim 140000$ , which does not seem highly sparse. So the main term in this complexity is due to the best atom selection in  $O(D \log L)$ , although the Gram matrix inversion might become more expensive for large iterations.

#### *Appendix D.2.1. Conclusion*

In the general case, the best atom selection step is the same for MP and OMP. The only difference is the projection step. Working with local dictionaries decreases the projection step for OMP and the selection step for MP. This still leaves a speed gap between MP and OMP, but **the speed difference between MP and OMP is mostly due to the selection step.**

This invalidates previous attempts at fast approximate OMP such as GP. Those approaches tried to decrease the cost of the projection step, but in our case it is not the most expensive one. We need an algorithm that can decrease the cost of the best atom selection compared to OMP. This is what LocOMP does.

## AppendixE. Notations

Vectors	$\varphi$ atom $s$ signal $r$ residual $x$ decomposition coefficients $\lambda$ correlations
Matrices	$\Phi$ dictionary $\Psi$ sub-dictionary $G$ Gram matrix
Indexing	$X_i$ $i^{\text{th}}$ column of matrix $X$ $x_i$ $i^{\text{th}}$ coefficient of vector $x$ $x^{(i)}$ variable $x$ at iteration $i$
Dimensions	$N$ signal length $D$ number of atoms in the dictionary $K$ sparsity level $L$ atom support length $T$ residual change support length $I$ total number of iterations $\alpha$ redundancy factor
Norms	$\ r\ _2$ euclidean norm $\ x\ _0$ number of non-zero coefficients in $x$
Misc.	$\langle \varphi_1, \varphi_2 \rangle$ scalar product $\operatorname{argmin}_E f(e)$ element $e$ of $E$ that minimizes $f(e)$ $M^*$ adjoint of $M$

- [1] G. Davis, S. Mallat, M. Avellaneda, Adaptive greedy approximations, *Constr. Approx.* 13 (1) (1997) 57–98.
- [2] K. Kreutz-delgado, B. Rao, K. Engan, T. w. Lee, T. Sejnowski, Convex/schur-convex (csc) log-priors and sparse coding, in: in 6th Joint Symposium on Neural Computation, 1999, pp. 65–71.

- [3] S. S. Chen, D. L. Donoho, M. A. Saunders, Atomic decomposition by basis pursuit, *SIAM Journal on Scientific Computing* 20 (1996) 33–61.
- [4] I. Daubechies, R. DeVore, M. Fornasier, S. Gunturk, Iteratively re-weighted least squares minimization for sparse recovery, *Commun. Pure Appl. Math* To appear.
- [5] T. Blumensath, M. Davies, Sparse and shift-invariant representations of music, *Audio, Speech, and Language Processing, IEEE Transactions on* [see also *Speech and Audio Processing, IEEE Transactions on*] 14 (1) (Jan. 2006) 50–57. doi:10.1109/TSA.2005.860346.
- [6] I. Daubechies, M. Defrise, C. D. Mol, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, *Comm. Pure Appl. Math* 57 (11) (2004) 1413–1457.
- [7] T. Blumensath, M. Davies, Iterative hard thresholding for compressed sensing, *Applied and Computational Harmonic Analysis*.
- [8] S. Mallat, Z. Zhang, Matching pursuits with time-frequency dictionaries, *Signal Processing, IEEE Transactions on* 41 (12) (1993) 3397–3415. doi:10.1109/78.258082.
- [9] Y. C. Pati, R. Rezaifar, Y. C. P. R. Rezaifar, P. S. Krishnaprasad, Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition, in: *Proceedings of the 27 th Annual Asilomar Conference on Signals, Systems, and Computers, 1993*, pp. 40–44.
- [10] D. Needell, J. Tropp, Cosamp: Iterative signal recovery from incomplete and inaccurate samples, *Applied and Computational Harmonic Analysis*.
- [11] T. Blumensath, M. Davies, In greedy pursuit of new directions: (nearly) orthogonal matching pursuit by directional optimisation, in: *Proc. EU-SIPCO'08, 2008*.

- [12] S. Krstulovic, R. Gribonval, MPTK: Matching Pursuit made tractable, in: Proc. Int. Conf. Acoust. Speech Signal Process. (ICASSP'06), Vol. 3, Toulouse, France, 2006, pp. 496–499. doi:10.1109/ICASSP.2006.1660699.  
URL [http://www.irisa.fr/metiss/gribonval/Conf/2006/ICASSP06%/2006\\_ICASSP\\_KrstulovicGribonval\\_MPTK.pdf](http://www.irisa.fr/metiss/gribonval/Conf/2006/ICASSP06%/2006_ICASSP_KrstulovicGribonval_MPTK.pdf)
- [13] G. Masataka, Development of the rwc music database, in: Proceedings of the 18th International Congress on Acoustics (ICA 2004), 2004, pp. I-553–556.
- [14] J. Tropp, Greed is good: algorithmic results for sparse approximation, IEEE Transactions on Information Theory.