Fast orthogonal sparse approximation algorithms over local dictionaries

Boris Mailhé, Rémi Gribonval, Pierre Vandergheynst, Frédéric Bimbot

Fast orthogonal sparse approximation algorithms over local dictionaries☆

Boris Mailhé[a,b,*], Rémi Gribonval[c], Pierre Vandergheynst[d], Frédéric Bimbot[a,e]

[a] *IRISA, Rennes, France*
[b] *Université de Rennes 1, France*
[c] *Centre de recherche INRIA Rennes Bretagne Atlantique, France*
[d] *École Polytechnique Fédérale de Lausanne, Switzerland*
[e] *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

☆This work was supported in part by Agence Nationale de la Recherche (ANR), project ECHANGE (ANR-08-EMER-006) and by the EU FET-Open project FP7-ICT-225913-SMALL.

*Corresponding author

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 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 that 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 Φ be a dictionary of size $D \times L$. Φ is said to be* local *if all its atoms φ are null outside of a support interval* $\mathrm{supp}(\varphi)$ *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

[1] 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 8 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 Φ a dictionary of size $N \times D$. Φ is redundant if $D > N$ and we define its redundancy factor $\alpha = \frac{D}{N}$. A signal s is said to be sparse over Φ if s can be approached closely by an decomposition Φx 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 Φ 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 Φ and an allowed sparsity level K, find the coefficients x that minimize*

$$\hat{x} = \text{argmin}_{\|x\|_0 \leq K} \|s - \Phi x\|_2^2 \tag{4}$$

The sparse approximation problem (4) is proved to be NP-Hard [2]. Yet many suboptimal algorithms have been proposed which can compute a close approximation within reasonable time.

2.1. ℓ_p minimization

The ℓ_0 sparsity measure has very bad properties for numerical optimization: it is not convex, not differentiable and piecewise constant. ℓ_p minimization algorithms replace the ℓ_0 measure by other constraints that are easier to optimize, often ℓ_p pseudo norms with $0 < p \leq 1$.

ℓ_p minimization is easier to compute than ℓ_0 from a mathematical point of view. ℓ_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 ℓ_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 Φ_K:

$$\Phi_K^+ = \left(\Phi_K^* \Phi_k\right)^{-1} \Phi_k^* \tag{5}$$

$$x_K = \Phi_K^+ s \tag{6}$$

$$r = s - \Phi_K x_K \tag{7}$$

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 [3]. Only one atom is selected and removed from the signal at each iteration. Algorithm 1 details the process.

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

$r^{(0)} = s$
$\Phi^{(0)} = \emptyset$
$x^{(0)} = 0$
for $i = 1$ to I **do**
  $\varphi^{(i)} = \mathrm{argmax}_{\varphi \in \Phi} \left|\left\langle r^{(i-1)}, \varphi \right\rangle\right|$ {best atom selection}
  $x^{(i)} = x^{(i-1)} + \left\langle r^{(i-1)}, \varphi^{(i)} \right\rangle \delta_{\varphi^{(i)}}$ {coefficient update $\varphi^{(i)}$}
  $r^{(i)} = r^{(i-1)} - \left\langle r^{(i-1)}, \varphi^{(i)} \right\rangle \varphi^{(i)}$ {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 Φ 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 $\Phi^{(i)}$ [4]. 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**
  $\varphi^{(i)} = \text{argmax}_{\varphi \in \Phi} \left| \left\langle r^{(i-1)}, \varphi \right\rangle \right|$ {best atom selection}
  $\Phi^{(i)} = \left[\Phi^{(i-1)}, \varphi^{(i)}\right]$
  $\chi^{(i)} = \left(\Phi^{(i)*}\Phi^{(i)}\right)^{-1} \Phi^{(i)*} r^{(i-1)}$ {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 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.

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 [5]. The coefficient update $\chi^{(i)}$ is given by

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

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:

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

2.2.5. More recent algorithms

With the ongoing trend of compressed sensing, several other sparse approximation algorithms have been recently proposed, with a strong emphasis on theoretical guarantees regarding the stable recovery of sparse vectors x_{opt} from the observation $s \approx \Phi x_{opt}$. They are mentionned here for the sake of completeness, but they can hardly be compared with the algorithm contributed in this paper, since the considered objectives (recovery guarantees rather than speed) and typical data dimensions are somewhat different. These algorithms extend the greedy paradigm with two main features: the addition of backtracking opportunities (in the Pursuit framework, this means being able to remove previously selected atoms from the support), and the selection of multiple atoms at a each iteration, to tentatively improve the resolution of close atoms (instead of taking a poorly informed early decision based only on correlations, one can keep all the good candidate atoms and wait until after the projection to see which one fits the decomposition the best).

CoSaMP, and Subspace Pursuit. The CoSaMP algorithm selects $2K$ atoms at each iteration, then projects the signal over the overall $2K$ large selection and only keeps the K atoms with the highest amplitude [6]. This is very close to the Subspace Pursuit (SP) algorithm [7], the main differences being that SP only adds K new atoms per iterations and that it updates the projection coefficients a second time after removing the low amplitude atoms. CoSaMP has been proven to recover the right support if the dictionary has a quasi-orthonormality property called Restricted Isometry Property (RIP). Each iteration requires an orthogonal projection on a $2K$ dimension subspace. In theory, the performance guarantees are robust when the orthogonal projection is replaced by a few gradient descent steps, but it has been observed in practice that this can seriously degrade the performance [8]. Between two iterations up to half of the considered atoms can change. It is not clear whether a fast implementation is possible or not, because of the apparent need to repeatedly perform (approximate) orthogonal projections over potentially large collections of atoms.

Iterative Hard Thresholding, and GraDeS. As its name suggests, the Iterative Hard Thresholding (IHT) algorithm [9, 8] performs several successive HT steps. After thresholding, the correlations with the obtained residual are added to the estimated amplitudes of all atoms, then a new iteration begins and the obtained amplitudes are thresholded again. IHT provides the same kind of theoretical guarantees as CoSaMP. The GraDeS (for Gradient Descent with Sparsification) algorithm proposes to relax IHT by adding only a fraction of the correlations at each iteration [10].

3. Complexity of existing greedy algorithms

This section summarizes the complexity of the greedy algorithms described in Section 2.2 (excluding those briefly described in subsection 2.2.5) both with arbitrary dictionaries and with structured dictionaries that allow fast transforms. More detailed explanations can be found in AppendixA. 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.

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
$\operatorname{argmax} \lvert\lambda\rvert$	$D \log D$	D	D	D
$G^{(i)} = \Phi^{(i)*}\Phi^{(i)}$	K^2N	$\emptyset$	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	$DN + K^2N$	DN	DN	DN
Number of iterations	1	$\gtrsim K$	K	$\gtrsim K$
TOTAL	$DN + K^2N$	DKN	DKN	DKN

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.

3.2. Complexity using fast dictionaries

The application of Φ^* 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 Φ^*) and synthesis (application of Φ). 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 Φ be a union of such fast bases. Let α 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.

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
$\lambda = \Phi^* r$	$D \log N$	$D \log N$	$D \log N$	$D \log N$
$\operatorname{argmax} \lvert\lambda\rvert$	$D \log D$	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	$D \log D + K^2$	$DK \log N$	$DK \log N + K^3$	$DK \log N$

3.3. 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 $\operatorname{supp}(\varphi) = [t_{\min}(\varphi), t_{\max}(\varphi)]$ be the support of an atom. Then $r^{(i)} = r^{(i)}$ outside of $\operatorname{supp}(\varphi^{(i)})$. So any atom with a disjoint support from $\operatorname{supp}(\varphi^{(i)})$ 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 $\operatorname{supp}(\varphi^{(i)})$ 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 Φ 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 [3].

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[1] [1] that decreases the complexity of MP from $O(DK \log L)$ to $O\left((D + KL) \log L\right)$.

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 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 $\Psi^{(i)} = \varphi^{(i)}$;

[1] `http://mptk.irisa.fr`

- 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.

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

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

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

$r_0 = s$
$\Phi_0 = \emptyset$
$x_0 = 0$
for $i = 1$ to I **do**
  $\varphi^{(i)} = \text{argmax}_{\varphi \in \Phi} \left|\left\langle r^{(i-1)}, \varphi \right\rangle\right|$ {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 8.1 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 Φ, 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 $T = |\operatorname{supp}(\Psi^{(i)})|$ small.

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 resid-

[2]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
$\lambda(1) = \Phi^* s$	$D \log L$	$D \log L$	$D \log L$
$m^{(1)} = \operatorname{argmax} \lvert\lambda^{(1)}\rvert$	D	D	D
$\lambda^{(i)} = \Phi^* r^{i-1}$	$\alpha L \log L$	$D \log L$	$\alpha T \log L$
$m^{(i)} = \operatorname{argmax} \lvert\lambda^{i}\rvert$	$\log(D)$	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	$\alpha(N + KL) \log L$	$DK \log L + \frac{K^3 L}{N}$	$\alpha(N + TK) \log L$

ual change interval. **If the residual change interval is fixed to be $I = [\hat{t_{\min}}, \hat{t_{\max}}]$, 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 Ψ 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

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

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 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 subdictionary $\Psi^{(i)}$ is the

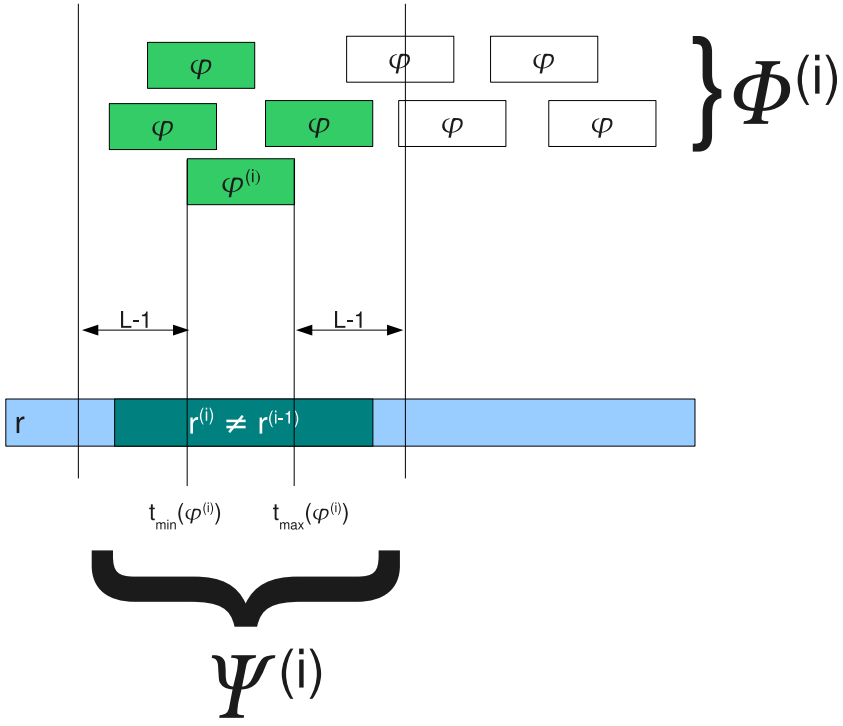

Figure 1: Selection of the sub-dictionary $\Psi^{(i)}$. All the atoms of $\Phi^{(i)}$ that overlap the last atom $\varphi^{(i)}$ are kept. The residual only changes on an interval of length $T <=\leq 3L-2$.

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,... 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 $\varphi^{(i)}$

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

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

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

- extract all atoms of $\Phi^{(i)}$ whose support is included in $[\hat{t_{\min}}, \hat{t_{\max}}]$. $\Psi^{(i)}$ 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:

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

$$\supseteq [\tilde{t_{\min}}, \tilde{t_{\max}}] \tag{12}$$

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 $\tilde{t_{\min}}$ or $\tilde{t_{\max}}$, 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. 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.

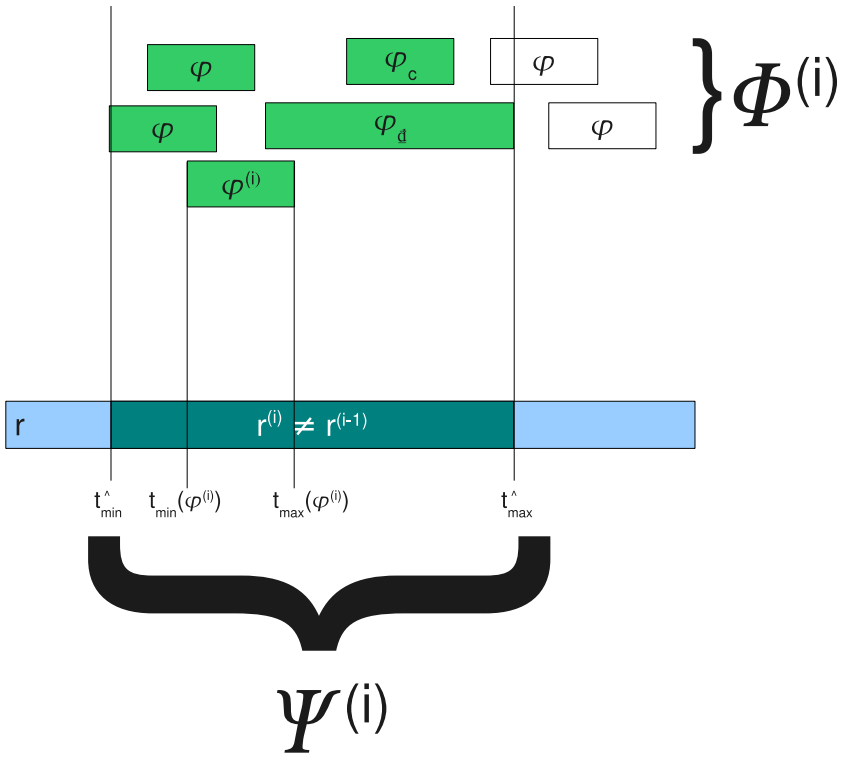

Figure 2: Selection of the sub-dictionary $\Psi^{(i)}$ with a multiscale dictionary. The short atom $\varphi^{(c)}$ is uncorrelated with the last atom $\varphi^{(i)}$, but it is kept anyway because its support its inside the support of the large kept atom $\varphi^{(d)}$.

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 [11]. 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 $D = 15.10^6$ 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 projection at no cost, a global algorithm could not cost less. This confirms the interest of local updates.

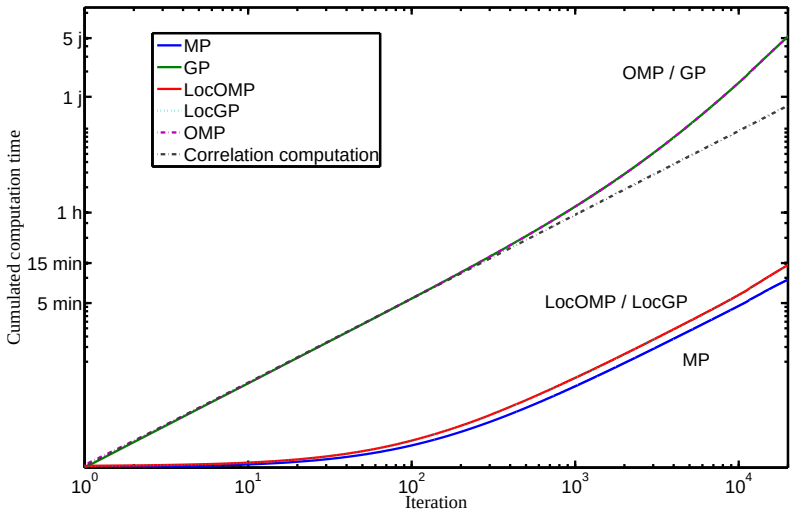

Figure 3: Cumulated computation time spent by different algorithms depending on the iteration. Local algorithms (MP, LocOMP and LocGP) are much faster than global ones (OMP and GP). The LocOMP and LocGP curves cannot be told apart on this plot, neither can OMP and GP. The dashed tangent is the lower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

6.3. Approximation quality

Figures 4 show the approximation quality defined as

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

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 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

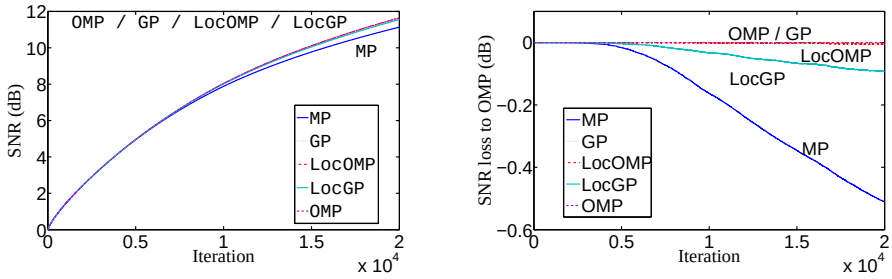

Figure 4: Approximation SNR obtained by several algorithms depending on the iteration. All the algorithms perform similarly apart from MP. The right plot shows the SNR loss compared to OMP.

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. Theoretical study

LocOMP was designed to ensure that its complexity remains within that of MP, and its quality should lie somewhere between MP and OMP. In this section we discuss which known theoretical guarantees that apply to both MP and OMP are also valid for LocOMP (resp. LocGP).

7.1. General MP, General Strong MP

The results presented in this section are based on the work of Tropp [12] and Gribonval and Vandergheynst [13]. Tropp provided results for OMP, and Gribonval and Vandergheynst pointed out that some of these results are valid for a wider class of algorithms they labelled General MP. A General MP algorithm is an algorithm that at iteration i:

- selects the atom with highest correlation to the residual,
- computes an approximant that lies in the span of all previously computed atoms $\Phi^{(i)}$.

One can easily see that LocOMP belongs to General MP.

However, not all the results extend because the General MP class is too wide: it contains obviously non-functional algorithms such as selecting the best atom then *adding* it (typos happen...) to the residual instead of subtracting it. In this paper we define a smaller class of algorithms that we call General Strong MP. This class intuitively corresponds to algorithms at least as good as MP. A General Strong MP algorithm xMP is an algorithm that:

- belongs to General MP;
- ensures that from any given residual s, any dictionary Φ and after any number of xMP iterations i, one more iteration of xMP decreases the residual energy at least as much as one iteration of MP.

Lemma 1. *LocOMP and OMP belong to General Strong MP.*

Proof. At iteration i, both MP, OMP and LocOMP update the residual with an orthogonal projection. MP projects the residual on the space S_{MP} orthogonal to the last atom $\varphi^{(i)}$. OMP projects it on the space S_{OMP} orthogonal to the set of selected atoms $\Phi^{(i)}$. LocOMP projects it on the space S_{LocOMP} orthogonal to the selected subdictionary $\Psi^{(i)}$. We conclude using the fact that $\varphi^{(i)} \in \Psi^{(i)} \subset \Phi^{(i)}$ so $S_{OMP} \subset S_{LocOMP} \subset S_{MP}$. □

Lemma 2. *GP and LocGP do not belong to General Strong MP.*

Proof. One can build a counter-example where an iteration of MP would get an exact decomposition (yielding a zero residual), but not the corresponding

iteration of GP. This example needs at least three iterations: GP and MP are always identical over the first two iterations.

Consider the dictionary Φ made of the three atoms $\varphi_1 = \begin{pmatrix}1 & 0 & 0\end{pmatrix}$, $\varphi_2 = \frac{1}{\sqrt{5}}\begin{pmatrix}1 & 2 & 0\end{pmatrix}$ and $\varphi_3 = \frac{1}{\sqrt{6}}\begin{pmatrix}2 & -1 & 1\end{pmatrix}$. Let $s = 12\varphi_1 + 2\varphi_2\sqrt{5} - \varphi_3\sqrt{6} = \begin{pmatrix}12 & 5 & -1\end{pmatrix}$. The first iteration of GP (or LocGP, since they share the same behaviour if the dictionary is not local) selects φ_1 and leads to $r^{(1)} = \begin{pmatrix}0 & 5 & -1\end{pmatrix}$. The second iteration selects φ_2 and we let the reader check that GP leads to $r^{(2)} = \begin{pmatrix}-2 & 1 & -1\end{pmatrix} = -\varphi_3\sqrt{6}$. The third iteration selects φ_3. An MP residual update would lead to $r^{(3)} = 0$. However, the gradient is proportional to $\Phi^* r^{(2)} = (-2, 0, -\sqrt{6})$ which is not in the direction of φ_3 so GP leads to a non-zero residual and does not decrease the energy of the residual as much as a step of MP would. □

This observation is consistent with the convergence rate for GP proven by Blumensath and Davies [5], that is slower than MP in the worst case.

7.2. Recovery of exactly sparse vectors

Assume that the signal s is exactly k-sparse, *i.e.* there exists a K-sparse vector x_{opt} such that $s = \Phi x_{opt}$. In that case, a natural question is whether the algorithm can retrieve x_{opt}. Let Φ_{opt} be the subdictionary of Φ associated to the nonzero entries of x_{opt}. Tropp provided a sufficient Exact Recovery Condition (ERC) on the dictionary Φ for OMP to recover x_{opt} [12]:

Theorem 1 (Tropp)**.** *Denote $\bar{\Phi}_{opt} = \Phi \setminus \Phi_{opt}$ and assume that*

$$\max_{\varphi \in \bar{\Phi}_{opt}} \left\| \Phi_{opt}^{+} \varphi \right\|_1 < 1. \tag{14}$$

Then, for any signal $s = \Phi_{opt} x_{opt}$, OMP recovers x_{opt} in $K = \|x_{opt}\|_0$ iterations.

Tropp only proves that under the ERC (14) OMP can only select atoms of Φ_{opt}. The exact recovery comes form the fact that OMP can never select the same atom twice, so if it keeps selecting optimal atoms it has to select them all. Gribonval and Vandergheynst pointed that the first part of the proof is also valid for General MP. As LocOMP belongs to General MP, the following theorem holds:

Theorem 2 (Gribonval/Vandergheynst)**.** *With the same notations and assumptions as in Theorem 1, for any signal $s = \Phi_{opt} x_{opt}$, all the atoms selected by LocOMP belong to Φ_{opt}.*

7.3. Convergence speed for exactly sparse signals

There are also results for MP that guarantee a fix decay rate per iteration, thus an overall exponential decay. Indeed, if one can prove that for some $0 < \eta < 1$, $\left\|r^{(i)}\right\|_2^2 \leq \eta \left\|r^{(i-1)}\right\|_2^2$, then $\left\|r^{(i)}\right\|_2^2 \leq \eta^i \|s\|_2^2$.

Mallat and Zhang proposed a geometrical bound [3] for η. In finite dimension, if the dictionary is complete, then there is a $\rho > 0$ such for any unitary vector s, there at least one atom $\varphi \in \Phi$ such that $|\langle s, \varphi \rangle| \geq \rho$. Then, at iteration i, for any General Strong MP algorithm, we have

$$\begin{aligned}
|\left\langle r^{(i-1)}, \varphi^{(i)} \right\rangle| &\geq \rho \left\| r^{(i-1)} \right\|_2 \\
\left\| r^{(i)} \right\|_2^2 &\leq \left\| r^{(i-1)} - \left\langle r^{(i-1)}, \varphi^{(i)} \right\rangle \varphi^{(i)} \right\|_2^2 \\
&\leq \left\| r^{(i-1)} \right\|_2^2 - \left\langle r^{(i-1)}, \varphi^{(i)} \right\rangle^2 \\
&\leq \left(1 - \rho^2\right) \left\| r^{(i-1)} \right\|_2^2
\end{aligned}$$

so $\eta = 1 - \rho^2$ is a lower bound for the decay rate. However this bound is pessimistic, especially in high dimension. For example, if the dictionary Φ is an orthonormal basis in dimension N, then the best possible ρ is $\frac{1}{\sqrt{N}}$ and the corresponding η is $1 - \frac{1}{N}$, which tends towards 1 when the dimension N increases.

Gribonval and Vandergheynst proposed another bound in the case of quasi-incoherent dictionaries. The cumulative coherence μ_1 of a dictionary Φ is a function of K defined as

$$\mu_1(K) = \max_{\varphi_0 \in \Phi, (\varphi_k)_{1 \leq k \leq K} \in (\Phi \setminus \{\varphi_0\})^K} \sum_{k=1}^{K} |\langle \varphi_0, \varphi_k \rangle| \tag{15}$$

This function measures how close to orthogonal the dictionary is: if it was orthogonal, then $\mu_1(K)$ would be 0 for any $K \leq N$. If the cumulative coherence increases slowly with K, then the dictionary is called quasi-incoherent. In that case, the following theorem holds:

Theorem 3. *Let Φ be a dictionary of cumulative coherence μ_1. Let K be such that $\mu_1(K) + \mu_1(K-1) < 1$ and let $\Phi_{opt} \subset \Phi$ be a sub-dictionary containing K atoms. Then the ERC (14) holds for Φ_{opt}, and for any General Strong MP*

algorithm, if $s = \Phi_{opt} x_{opt}$ *then*

$$\forall i > 0, \left\| r^{(i)} \right\|_2^2 \leq \left(1 - \frac{1 - \mu_1(K-1)}{K} \right)^i \|s\|_2^2 \tag{16}$$

Proof. The reader can refer to the proof provided by Gribonval and Vandergheynst for both MP and OMP. The proof for OMP only adds the fact that OMP decreases the error more than MP on one iteration, so it is actually valid for the whole General Strong MP class, including LocOMP. □

7.4. Stable recovery of sparse vectors in the presence of noise

Natural signals are usually not exactly sparse, either because the sparse model is only a simplified approach or because the measurements were noisy. The signal model $s = \Phi_{opt} x_{opt} + \epsilon$ is therefore often more realistic than the exact sparse model $s = \Phi_{opt} x_{opt} + \epsilon$. Tropp proved that with quasi-incoherent dictionaries, OMP manages to retrieve atoms belonging to Φ_{opt} until the residual error gets small enough. Gribonval and Vandergheynst pointed that Tropp's proof also holds for any General MP algorithm, including LocOMP and LocGP.

Theorem 4. *Let* Φ *be a dictionary of cumulative coherence* μ_1*. Let* K *be such that* $\mu_1(K) + \mu_1(K-1) < 1$ *and let* $\Phi_{opt} \subset \Phi$ *be a sub-dictionary containing* K *atoms. Let* $s = \Phi_{opt} x_{opt} + \epsilon$*: LocOMP only selects atoms of* Φ_{opt} *until the following error threshold is reached:*

$$\left\| r^{(i)} \right\|_2^2 \leq \left(1 + \frac{K\left(1 - \mu_1(K-1)\right)}{\left(1 - \mu_1(K-1) - \mu_1(K)\right)^2} \right) \|s - \Phi_{opt} x_{opt}\|_2^2 \tag{17}$$

7.5. Convergence speed in the presence of noise

Gribonval and Vandergheynst provided an upper bound for the number of iterations it takes MP to reach the error threshold of Theorem 4. This result can be generalized to General Strong MP, hence LocOMP.

Theorem 5. *With all the hypotheses of Theorem 4 still holding, let* σ_K^2 *be the residual energy of the best* K*-term approximant to* s*. If* $\sigma_K^2 \leq 3\frac{\sigma_1^2}{K}$*, then the threshold of Theorem 4 is reached within at most*

$$I = 2 + \frac{K}{1 - \mu_1(K-1)} \log \frac{3\sigma_1^2}{K\sigma_K^2} \tag{18}$$

iterations. If not, then the signal is too noisy to guarantee the recovery of atoms from Φ_{opt}*.*

Proof. The reader can follow the proof by Gribonval and Vandergheynst. The only change needed to extend the proof to General Strong MP is in the proof of their Lemma 3. They use the fact that for MP,

$$\left\|r^{(i)}\right\|_2^2 - \left\|r^{(i+1)}\right\|_2^2 = \left\langle r^{(i)}, \varphi^{(i+1)} \right\rangle^2 \tag{19}$$

To extend the proof to General Strong MP, replace this equality with

$$\left\|r^{(i)}\right\|_2^2 - \left\|r^{(i+1)}\right\|_2^2 \geq \left\langle r^{(i)}, \varphi^{(i+1)} \right\rangle^2 \tag{20}$$

□

8. Perspectives

8.1. 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. MPTK currently does not use any matrix computations thanks to fast dictionaries. We would like to keep it that way because it is programmed and C++ so the access to linear algebras libraries is not native. 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.

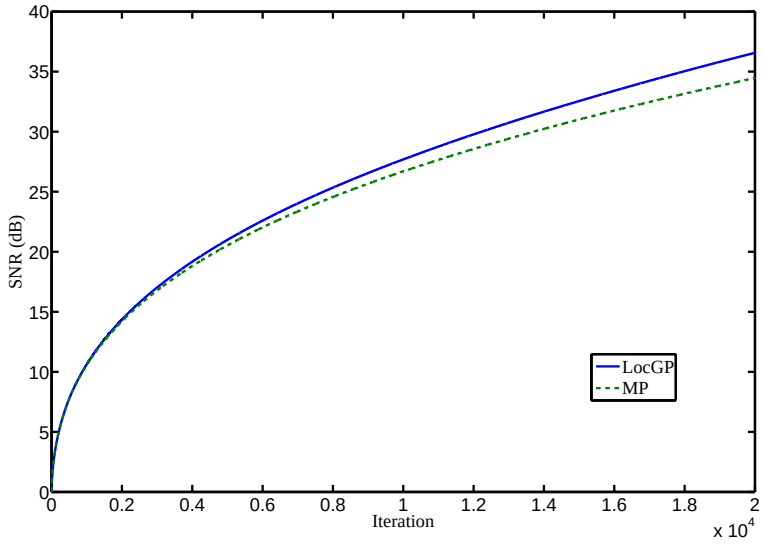

Figure 5: Average approximation obtained by MP and LocGP depending on the iteration. The averages have been computed over 10 piano signals downsampled to 8 kHz. The decompositions used MDCT dictionaries with scale 32 and 1024. LocGP provides an average gain of 2dB over MP.

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 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. We are implementing a fast synthesis method to solve this problem. As the residual update is local, several atoms of the neighbourhood $\Psi^{(i)}$ should fall within the same few frames, which is what fast methods need to provide a gain over naive implementations.

8.2. Extension to multidimensional signals

LocOMP as described in this paper only applies to temporal series. However, local or shift-invariant dictionaries are also used in image 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 AppendixC for more details). 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 $\Psi^{(i)}$.

9. Conclusion

Sparse approximation over local dictionaries requires specific algorithms 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, 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.

Acknowledgements

The authors would like to thank the reviewers for their useful comments.

AppendixA. 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 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 $G^{(i-1)-1}$ [4]. 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 th 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 ρ requires one more synthesis in $O(D \log N)$. Then one just have to change the coefficients, which is done in $O(i)$. Compared to OMP, the cost in $O(i^2)$ has been decreased to $O(i)$.

AppendixB. 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

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

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

$$p = \frac{2L - 1}{N - L + 1} \tag{B.2}$$

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

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

AppendixC. 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 have 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 [3] and C++ [4] 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

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

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

[3] `http://java.sun.com/javase/6/docs/api/java/util/TreeSet.html`
[4] `http://gcc.gnu.org/onlinedocs/libstdc++/libstdc++-html-USERS-4.4/a00528.html`

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 $G^{(i)}$.

AppendixC.1. MPTK implementation

It has already been noted in Section AppendixC 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 indexing of selected atoms.

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.

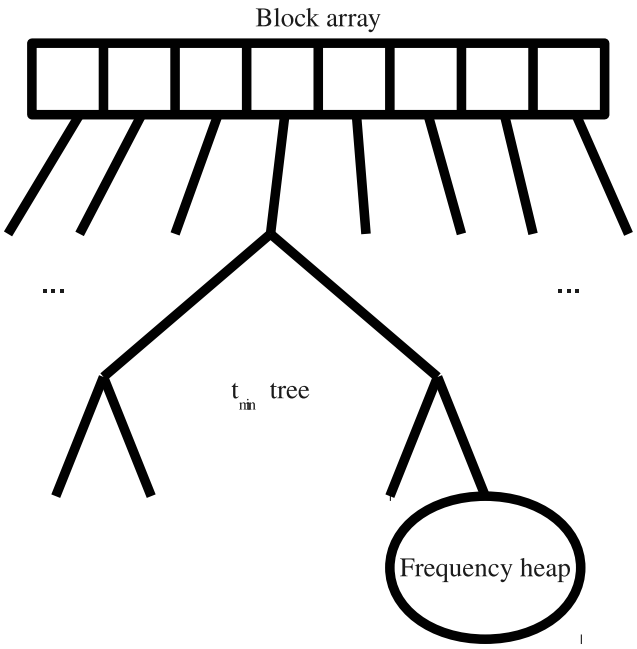

Figure C.6: MPTK data structure for fast access to the atoms. Atoms belonging to the same block and $t_{\min}$ are in a single heap, then the heaps of the same block are sorted by growing $t_{\min}$, then those trees are stored in a block array.

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

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

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

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

then the inverse is equal to

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

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 Γ 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 Γ.

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 distinct connected components in Γ, and the 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 Γ is connected and the residual changes over its whole support. This behaviour seems likely at the end of OMP.

AppendixD.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 test on the support begin $t_{\min}$ and end $t_{\max}$:

$$\begin{aligned}
\operatorname{supp}(\varphi) \cap \operatorname{supp}(\varphi^{(i)}) \neq \emptyset \Leftrightarrow & \begin{cases} t_{\max}(\varphi) \geq t_{\min}(\varphi^{(i)}) \\ t_{\min}(\varphi) \leq t_{\max}(\varphi^{(i)}) \end{cases} && \text{(D.4)} \\
\Leftrightarrow & \begin{cases} t_{\min}(\varphi) + L - 1 \geq t_{\min}(\varphi^{(i)}) \\ t_{\min}(\varphi) \leq t_{\min}(\varphi^{(i)}) + L - 1 \end{cases} && \text{(D.5)} \\
\Leftrightarrow & \; t_{\min}(\varphi^{(i)}) - L < t_{\min}(\varphi) < t_{\min}(\varphi^{(i)}) + L && \text{(D.6)}
\end{aligned}$$

So an atom φ 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 AppendixB for justification). Then the cost for this step would be

$$\begin{aligned}
O\left(|\Phi^{(i-1)}| + LE^{(i)}\right) &= O\left((i-1) + \frac{L(i-1)(2L-1)}{N-L+1}\right) && \text{(D.7)} \\
&= O\left(i + \frac{iL^2}{N}\right) && \text{(D.8)}
\end{aligned}$$

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 AppendixC we show that this cost can be avoided by using a sorted data structure.

Fast dictionaries. Correlations between atoms can also be computed using FFT or other fast algorithms 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

$$L \log L < \frac{iL^2}{N} \tag{D.9}$$

$$i > N \frac{\log L}{L} \tag{D.10}$$

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 [4].

So the total cost of an OMP iteration is reduced to

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

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

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

$$i \gtrsim N\sqrt{\frac{\alpha \log L}{L}} \tag{D.13}$$

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.

AppendixD.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	φ	atom
	s	signal
	r	residual
	x	decomposition coefficients
	λ	correlations
Matrices	Φ	dictionary
	Ψ	sub-dictionary
	G	Gram matrix
Indexing	X_i	i^{th} column of matrix X
	x_i	i^{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
	α	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] 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`

[2] G. Davis, S. Mallat, M. Avellaneda, Adaptive greedy approximations, Constr. Approx. 13 (1) (1997) 57–98.

[3] 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.

[4] Y. C. Pati, R. Rezaiifar, Y. C. P. R. Rezaiifar, 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.

[5] T. Blumensath, M. Davies, In greedy pursuit of new directions: (nearly) orthogonal matching pursuit by directional optimisation, in: Proc. EUSIPCO'08, 2008.

[6] D. Needell, J. Tropp, Cosamp: Iterative signal recovery from incomplete and inaccurate samples, Applied and Computational Harmonic Analysis.

[7] W. Dai, O. Milenkovic, Subspace pursuit for compressive sensing signal reconstruction, Information Theory, IEEE Transactions on 55 (5) (2009) 2230 –2249. doi:10.1109/TIT.2009.2016006.

[8] T. Blumensath, M. E. Davies, Normalised iterative hard thresholding; guaranteed stability and performance, Tech. rep. (feb 2009).

[9] T. Blumensath, M. Davies, Iterative hard thresholding for compressed sensing, Applied and Computational Harmonic Analysis.

[10] R. Garg, R. Khandekar, Gradient descent with sparsification: an iterative algorithm for sparse recovery with restricted isometry property, in: ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, New York, NY, USA, 2009, pp. 337–344. doi:http://doi.acm.org/10.1145/1553374.1553417.

[11] G. Masataka, Development of the rwc music database, in: Proceedings of the 18th International Congress on Acoustics (ICA 2004), 2004, pp. I–553–556.

[12] J. Tropp, Greed is good: algorithmic results for sparse approximation, IEEE Transactions on Information Theory.

[13] R. Gribonval, P. Vandergheynst, On the exponential convergence of matching pursuits in quasi-incoherent dictionaries, Information Theory, IEEE Transactions on 52 (1) (2006) 255–261.