A LOW COMPLEXITY ORTHOGONAL MATCHING PURSUIT FOR SPARSE SIGNAL APPROXIMATION WITH SHIFT-INVARIANT DICTIONARIES

Boris Mailhé, Rémi Gribonval, Frédéric Bimbot

Projet METISS
Centre de Recherche INRIA Rennes - Bretagne Atlantique
IRISA, Campus de Beaulieu, F-35042 Rennes Cedex, France
E-mail: firstname.lastname@irisa.fr

Pierre Vandergheynst

Signal Processing Laboratories (LTS)
School of Engineering, EPFL
Station 11, CH - 1015 Lausanne, Switzerland
E-mail: firstname.lastname@epfl.ch

ABSTRACT

We propose a variant of Orthogonal Matching Pursuit (OMP), called LoCOMP, for scalable sparse signal approximation. The algorithm is designed for shift-invariant signal dictionaries with localized atoms, such as time-frequency dictionaries, and achieves approximation performance comparable to OMP at a computational cost similar to Matching Pursuit. Numerical experiments with a large audio signal show that, compared to OMP and Gradient Pursuit, the proposed algorithm runs in over 500 less time while leaving the approximation error almost unchanged.

***Index Terms*—** sparse approximation, greedy algorithms, shift-invariance, orthogonal matching pursuit

1. INTRODUCTION

Finding a sparse linear decomposition or a sparse approximation of a given signal is a key issue in various domains such as compression, under-determined source separation, and more recently compressed sensing. Many algorithms have been proposed to obtain good sparse approximations in polynomial time, but it remains quite challenging to design algorithms that combine good approximation performance and scalability for large signal dimensions, and the general problem of finding the best m-term approximation is non polynomial (NP-Complete).

Today' s most popular approaches are ℓ^1 minimization, on the one hand, which is tackled with specialised convex optimization iterative techniques, and greedy algorithms, on the other hand, which iteratively decrease the approximation error by relaxing the sparsity constraint. In this paper we focus on the latter class, which includes Matching Pursuit (MP) [1], Orthogonal Matching Pursuit (OMP) [1, 2], as well as several variants such as Gradient Pursuit (GP) [3] or Relaxed Greedy Algorithm [4]. Roughly speaking, MP is fast but can yield subtantially poorer approximation performance than OMP and GP, which however typically have substantially larger running times for large data.

In this paper we propose an algorithm, called LoCOMP, which goal is to behave like OMP and GP in terms of approximation performance, with essentially the cost of MP. Since there is no free lunch, the proposed algorithm is not valid for general dictionaries but is rather dedicated to dictionaries that display some level of "shift-invariance" and are made of atoms with localized supports, such as the time-frequency dictionaries used, e.g., in audio coding. We illustrate the properties of the proposed algorithm by comparing it with MP, OMP and GP on a high dimensional audio signal.

2. GREEDY ALGORITHMS

Let $\mathcal{H}$ be an Hilbert space of finite dimension N. A dictionary $\mathbf{\Phi}$ is a set of unit norm vectors φ_k of $\mathcal{H}$ called atoms. We will also use the notation $\mathbf{\Phi}$ for the matrix that admits the atoms φ_k as columns. A sparse approximation of a signal s over a dictionary $\mathbf{\Phi}$ is a vector x with small approximation error $\|s - \mathbf{\Phi}x\|_2$ under a constraint on the number of nonzero coefficients $\sharp\{k, x_k \neq 0\}$, usually denoted with the ℓ^0 "norm" $\|x\|_0$. Finding the best approximation with $\|x\|_0 \leq K$ is an NP-hard problem, and greedy algorithms are sub-optimal iterative algorithms that attempt to solve this problem by successively adding new atoms into a sparse approximation $\mathbf{\Phi}_i x_i$ with the objective of minimizing the new residual $r_i = s - \mathbf{\Phi}_i x_i$. Each iteration i of a greedy algorithm is composed of two successive steps:

1. **selection**: find the atom that has the highest scalar product with the residual $\varphi_i = \text{argmax}_{\varphi \in \mathbf{\Phi}} |\langle r_{i-1}, \varphi \rangle|$ and add it to the selected atoms $\mathbf{\Phi}_i = \mathbf{\Phi}_{i-1} \cup \varphi_i$;
2. **update** the coefficients x_i (and the residual r_i) , trying to minimize the new approximation error $\|r_i\|_2$.

Several update rules have been proposed, among which

1. MP : $r_i = r_{i-1} - \langle r_{i-1}, \varphi_i \rangle \varphi_i$;
2. OMP: $r_i = r_{i-1} - \mathbf{\Phi}_i(\mathbf{\Phi}_i^T \mathbf{\Phi}_i)^{-1} \mathbf{\Phi}_i^T r_{i-1}$;
3. GP : $r_i = r_{i-1} - \frac{\|\mathbf{\Phi}_i^T r_{i-1}\|_2^2}{\|\mathbf{\Phi}_i \mathbf{\Phi}_i^T r_{i-1}\|_2^2} \mathbf{\Phi}_i \mathbf{\Phi}_i^T r_{i-1}$.

MP is the fastest of the above described algorithms, because it only attempts to optimize the coefficient of the last selected atom to minimize the new approximation error. OMP optimizes all coefficients to obtain the minimum error with the set of selected atoms. This can provide a much smaller error, but to the price of significantly more computations. GP essentially attempts to reduce the cost of OMP by performing the first step of a conjugate gradient descent to approximate the full projection OMP would perform.

Table 1 indicates the complexity order of each step for MP,OMP and GP with a general dictionary. The main quantities driving the complexity are N (the dimension of the signal space), $\alpha \geq 1$ (the redundancy of the dictionary that contains αN atoms), and i (the iteration number, which indicates that i atoms have been selected). Since the goal is to obtain a sparse approximation of the signal, the iteration number i is always lower than the signal dimension N.

The selection step involve two substeps: the computation of the αN *correlations* $\langle r_{i-1}, \varphi \rangle$, $\varphi \in \mathbf{\Phi}$, each of them costing of the order of N multiply-add; the search for the atom with *maximum* correlation, which requires $\alpha N - 1$ comparisons. The update step for OMP involves computing the *Gram matrix* $\mathbf{\Phi}_i^T \mathbf{\Phi}_i$, which can be updated from the previously computed $\mathbf{\Phi}_{i-1}^T \mathbf{\Phi}_{i-1}$ but requires the computation of the scalar product of the last selected atom φ_i with the $i-1$ previous ones $\mathbf{\Phi}_{i-1}$. Then, the new *coefficients* x_i in OMP can be computed by inverting the Gram matrix with a cost roughly quadratic in the size i of the matrix by reusing the computations done in previous iterations. The exact cost will depend on the chosen inversion method. A more complete study about it, as well as the explanations for GP complexity, can be found in [3]. Eventually, all methods require updating the *residual*, which involves an $N \times i$ matrix multiply $\mathbf{\Phi}_i x_i$ and/or updating the N entries of the vector.

Table 1. Complexity order of a given iteration of several greedy algorithms in the general case

	Step	MP	OMP	GP
selection	correlations	αN^2	αN^2	αN^2
	maximum	αN	αN	αN
update	Gram matrix	0	iN	0
	coefficients	0	i^2	iN
	residual	N	iN	N

3. SHIFT-INVARIANT LOCALIZED DICTIONARIES

A shift-invariant localized dictionary is a dictionary formed of time shifts $\psi_{\ell,n}(t) := \psi_\ell(t-n)$ of a collection of short patterns ψ_ℓ of length $L \ll N$. The structure of such dictionaries can be exploited to substantially decrease the algorithms' complexity through the following tricks:

- faster correlation computations can be achieved through FFT-based methods. This decreases the full correlation computation cost from $\alpha N L$ down to $\alpha N \log L$.
- with the MP update, between two consecutive iterations, the residual only changes on the support $\texttt{support}(\varphi_i) := \{t, \varphi_i(t) \neq 0\}$ of the last selected atom. This leaves only about αL correlations to recompute instead of αN, since all the other ones are still relevant. This drives down the cost to $\alpha L \log L$.
- previous comparisons between the correlations can also be stored in a tournament tree to make the search of the maximum faster.

These tricks are implemented in the Matching Pursuit ToolKit (MPTK[1]) C++ library [5] and enable a speedup of up to a 100 times in decomposition, allowing $I = 1.5 \cdot 10^6$ iterations of MP on a twenty-minute 44kHz audio signal ($N = 60 \cdot 10^6$ samples) to be performed in less than twenty minutes of computation time on a standard PC, with an $\alpha = 3$ times overcomplete dictionary of Gabor atoms of length $L = 1024$.

The key property that makes such speedup possible is that *the residual update is local at each step*: atoms have a limited time support, and the residual outside the support of the last selected atom is the same before and after the update.

This locality property could certainly be used to speed up the first iterations of OMP or GP, however in OMP/GP the updated support size will grow with the number of iterations until the whole signal support is spanned by selected atoms. In the worst case, this can happen with $\lceil \frac{N}{L} \rceil$ atoms in the representation. Table 2 summarizes the complexity order of each step (except the first one which involves computing all correlations) of MP, OMP and GP with a general shift-invariant dictionary. The cost of the update step is estimated by taking

Table 2. Complexity order of a given iteration (after the first one) of several greedy algorithms in the shift-invariant case

	Step	MP	OMP	GP
selection	correlations	$\alpha L \log L$	$\alpha N \log L$	$\alpha N \log L$
	maximum	αL	αN	αN
update	Gram matrix	0	$i\frac{L^2}{N}$	0
	coefficients	0	$i^2\frac{L}{N}$	iL
	residual	L	iL	N

into account the fact that *the Gram matrix* $\mathbf{\Phi}_i^T \mathbf{\Phi}_i$ *is sparse*, since all the atoms which support do not overlap are orthogonal. If the repartition of the selected atoms is uniform along the time axis, then at each iteration there are only about $\frac{iL}{N}$ atoms that are not orthogonal to the last one. Keeping the selected atoms $\mathbf{\Phi}_{i-1}$ sorted by increasing time index allows to select all the atoms $\varphi \in \mathbf{\Phi}_{i-1}$ non-orthogonal to φ_i with only two searches for the first and last time indices of the set. All these book-keeping operations can be performed within logarithmic complexity using binary search trees (*e.g.* the red/black trees used in current implementations of C++ STL and Java sorted sets).

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

4. LocOMP ALGORITHM

As described above, in shift-invariant dictionarie, simple tricks allow to significantly reduce the computational complexity of MP compared to a naive implementations. However, the cost of OMP and GP remains quite high, calling for modified algorithms to handle real-world large-scale signals, where the aimed number of atoms I is somewhat lower than the signal size N, but the latter is large enough to discourage naive computation (*e.g.* for one minute of music sampled at 8 kHz, we already have $N \approx 5 \cdot 10^5$).

The prohibitive costs for OMP and GP are the ones with strongest dependency in N: as shown in Table 2 the most costly steps are the correlation computation and maximum search, which have linear dependency in N. This linear dependency has disappeared in MP by exploiting the locality of the changes in the residual. This is why we propose an algorithm that only slightly loosens this locality property. To our knowledge, all approaches to decrease OMP complexity emphasize the reduction in the cost of the update step (e.g., by replacing full matrix inversion by conjugate gradient descent as in [3]), not the selection step.

The main idea of the proposed LoCOMP algorithm is to select a sub-dictionary $\mathbf{\Psi}_i \subset \mathbf{\Phi}_i$ containing the last selected atom φ_i and to orthogonalize the decomposition only on this sub-dictionary. The algorithm is described in Algorithm 1, and the key element that determines the behaviour of the algorithm is the `neighbour()` function that performs the sub-dictionary selection:

- MP corresponds to $\texttt{neighbour}(\mathbf{\Phi}_i, \varphi_i) := \varphi_i$;
- OMP corresponds to $\texttt{neighbour}(\mathbf{\Phi}_i, \varphi_i) := \mathbf{\Phi}_i$;

To decrease the computational cost with respect to OMP, it is crucial to ensure that the support of $\mathbf{\Psi}\chi_i$ is small so that the update of the residual remains localized. In LoCOMP, $\texttt{neighbour}(\mathbf{\Phi}_i, \varphi_i)$ contains exactly all the atoms $\varphi \in \mathbf{\Phi}_i$ which support intersects with the support of φ_i. This choice was mainly led by the observation that, as explained in Section 3, this set is already the one that has to be searched for when updating the Gram matrix. Selecting it as the atom's neighbourhood spares another search.

5. EXPERIMENTAL RESULTS

LoCOMP has been tested and compared to MP, OMP and GP on an excerpt from the RWC base[2]. It is a one-minute mono-channel jazz guitar audio signal downsampled to 8kHz ($N \approx 5 \cdot 10^5$). Given the high cost of running OMP and GP for comparison (the total running time for each of these algorithms in the first experiment below was roughly $5 \cdot 10^5$ seconds, ≈ 5.7 *days*), it was not possible to run experiments on more than one signal, and this was also the largest signal dimension we could test. In comparison, the computation time of LoCOMP was 854 seconds (≈ 15 *minutes*).

[2]`http://staff.aist.go.jp/m.goto/RWC-MDB/`

Algorithm 1 x = LoCOMP(s, $\mathbf{\Phi}$)

```
$r_0 = s$
$\mathbf{\Phi}_0 = \emptyset$
$x_0 = 0$
**for** $i = 1$ to $I$ **do**
    $\varphi_i = \mathrm{argmax}_{\varphi \in \mathbf{\Phi}} |\langle r_{i-1}, \varphi \rangle|$ {selection}
    $\mathbf{\Phi}_i = \mathbf{\Phi}_{i-1} \cup \varphi_i$
    $\mathbf{\Psi}_i = \texttt{neighbour}(\mathbf{\Phi}_i, \varphi_i)$ {sub-dictionary selection}
    $\chi_i = (\mathbf{\Psi}_i^* \mathbf{\Psi}_i)^{-1} \mathbf{\Psi}_i^* r_{i-1}$ {coefficients of projection on sub-dictionary}
    $x_i = x_{i-1} + \chi_i$ {update coefficients}
    $r_i = r_{i-1} - \mathbf{\Psi}_i \chi_i$ {update residual}
**end for**
**return** $x_I$
```

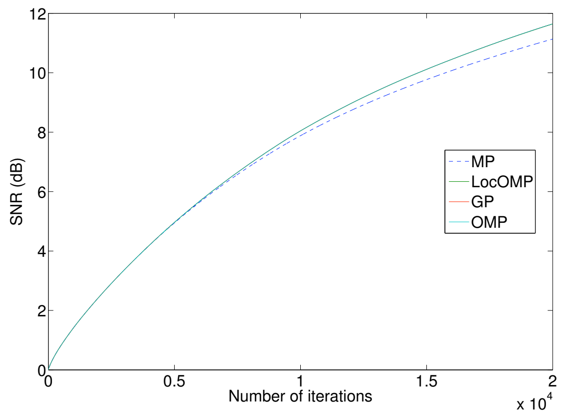

Fig. 1. SNR depending on the iteration

5.1. SNR and computation time

In a first experiment, OMP, GP, LoCOMP and MP were run for $I = 20000$ iterations[3] to decompose the signal on a fully shift-invariant MDCT dictionary of scale $L = 32$ (therefore with redundancy factor $\alpha = 32$) containing $\alpha N \approx 1.5 \cdot 10^7$ atoms. The scale roughly corresponds to the smallest scale of the windows used in AAC encoding on 44.1 kHz signals, while remaining small enough to make it possible to actually run OMP and GP.

Figure 1 shows the SNR reached by each algorithm at each iteration. OMP, GP and LoCOMP cannot be distinguished on this plot. The final SNR for LoCOMP after 20000 iterations is actually only 0.01dB lower than for OMP and GP, while the final SNR for MP is 0.6dB lower.

[3]The iterations of the different algorithms were interleaved on the same process to ensure a similar environment for all. The Matlab® code was compiled and run on a standard PC (2.33 Ghz, 4 Go RAM). It was not fully optimized for runtime speed, especially for the first OMP and GP iterations, but this should not affect the observed time magnitudes.

Table 3. CPU time per iteration (s)

Iteration	MP	LoCOMP	GP	OMP
First ($i = 0$)	3.4	3.4	3.4	3.5
Begin ($i \approx 1$)	0.028	0.033	3.4	3.4
End ($i \approx I$)	0.028	0.050	40.5	41
Total time	571	854	$4.50 \cdot 10^5$	$4.52 \cdot 10^5$

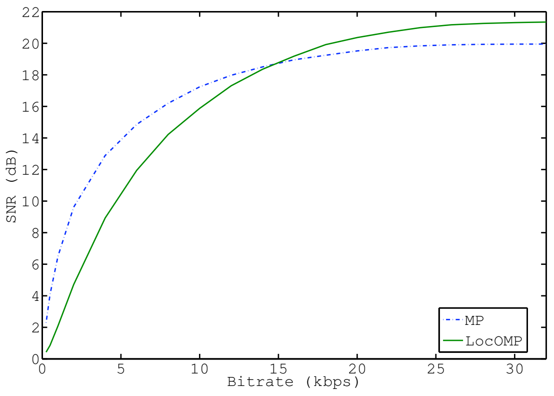

Fig. 2. SNR depending on the decoding bit rate

The CPU times per iteration evolved linearly for each algorithm. Table 3 shows their value for the first iteration (which is relatively costly for every algorithm because it involves computing inner products with all atoms of the dictionary), the next beginning iterations, the last iterations and finally the total duration of the complete execution.

The algorithms clearly split into two groups. The cost drop after the first iteration for MP shows that most of the first iteration was spent computing the correlations, and both MP and LoCOMP iterations remain much cheaper after the first iteration. To the opposite, the cost of GP and OMP iterations grows substantially with the iteration index and reaches up to 1500 (resp. 800) times than that of MP (resp. LoCOMP) iterations. On this example, LoCOMP almost reached the same level of approximation error as OMP/GP, with a total computation cost only 1.5 times that of MP and 500 times smaller than that of OMP/GP .

5.2. Preliminary application to audio coding

In a second experiment, we investigated the potential use of LoCOMP in the scalable coding framework proposed by Ravelli and Daudet [6]. The 8 kHz signal was decomposed on a two-scale fully shift-invariant MDCT dictionary with scales $L_1 = 32$ and $L_2 = 256$, roughly corresponding at 8kHz to the scales used in AAC encoding at 44.1kHz.

Figure 2 shows the rate/distortion curve of this coding scheme using MP and LoCOMP as a transform. At high rates, LoCOMP coding leads to less distortion than MP coding, with a final gain of 1.4dB. However, LoCOMP also brings a degradation at lower rates. Thios might be partly due to the choice of a much smaller dictionary than the eight-scale dictionary used in [6].

6. CONCLUSION

We proposed a greedy algorithm called LoCOMP for computationally tractable sparse approximation of long signals with large shift-invariant dictionaries. We have shown on an example that its approximation performance is similar to that of OMP/GP, with a gain of 0.6 dB over MP, while the computational cost remains 500 times lower than that of OMP. We expect the approximation gain of LoCOMP over MP to be more significant for dictionaries more adapted to the decomposed signal (e.g., L rather of the order of 256, the largest scale used in AAC codecs), however for such scales it no longer seems possible to compare the proposed algorithm with OMP/GP, because of the computational complexity of the latter.

Current work consists in implementing LoCOMP as well as a localized version of Gradient Pursuit in MPTK [5] to benefit from all other speedup tricks briefly described in this paper, and we believe this will open the door to large scale experiments and applications of sparse approximation that so far seemed unachievable.

7. ACKNOWLEDGEMENTS

The authors would like to thank Emmanuel Ravelli and Laurent Daudet from the LAM team at University Paris 6 for their help with the audio coding experiments.

8. REFERENCES

[1] S. Mallat and Z. Zhang, “Matching pursuit with time-frequency dictionaries,” *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, Dec 1993.

[2] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad, “Orthonormal matching pursuit : recursive function approximation with applications to wavelet decomposition,” in *Proc.* 27^{th} *Annual Asilomar Conf. on Signals, Systems and Computers*, Nov. 1993.

[3] T. Blumensath and M.E. Davies, “In greedy pursuit of new directions: (nearly) orthogonal matching pursuit by directional optimisation,” in *Proc. EUropean SIgnal Processing COnference (EUSIPCO’08)*, Lausanne, August 2008.

[4] Andrew R. Barron, Albert Cohen, Wolfgang Dahmen, and Ronald A. DeVore, “Approximation and learning by greedy algorithms,” *Annals of statistics*, vol. 36, no. 1, pp. 64–94, 2008.

[5] Sacha Krstulovic and Rémi Gribonval, “MPTK: Matching Pursuit made tractable,” in *Proc. Int. Conf. Acoust. Speech Signal Process. (ICASSP’06)*, Toulouse, France, May 2006, vol. 3, pp. 496–499.

[6] E. Ravelli, G. Richard, and L. Daudet, “Extending fine-grain scalable audio coding to very low bitrates using overcomplete dictionaries,” in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA’07)*, 2007, pp. 195–198.