Audio sparse decompositions in parallel
Let the greed be shared !

Laurent Daudet, *Member, IEEE*

***Abstract*—Greedy methods are often the only practical way of solving very large sparse approximation problems. Among such methods, Matching Pursuit (MP) is undoubtedly one of the most widely used, due to its simplicity and relatively low overhead. Since MP works sequentially, however, it is not straightforward to formulate it as a parallel algorithm, to take advantage of multi-core platforms for real-time processing. In this paper, we investigate how a slight modification of MP makes it possible to break down the decomposition into multiple local tasks, while avoiding blocking effects. Our simulations on audio signals indicate that this Parallel Local Matching Pursuit (PLoMP) gives results comparable to the original MP algorithm, but could potentially run in a fraction of the time — on-the-fly sparse approximations of high-dimensional signals should soon become a reality.**

The last two decades have witnessed the advent of *sparsity* as a major paradigm in many areas of signal processing. Sparsity is the key to success for most of state-of-the-art multimedia compression schemes, such as still image coding (for instance JPEG-2000 [1]), and audio coding (MPEG-2/4 Advanced Audio Coding (AAC) [2]). Basically, sparsity exploits the fact that there exist bases in which, for most natural signals, only a few of the transform coefficients are sufficient to provide a good approximation. To be more precise, given a signal, by sorting its transform coefficients by absolute decaying order, one observes a fast decay, typically a power law with some large negative exponent. This ability to concentrate most of the energy of the signals into only a few of the transform coefficients naturally leads to an increased coding efficiency. For instance, in the JPEG-2000 image coder based on an orthogonal 2-D dyadic wavelet transform, only portions of the image that correspond to sharp transitions (at objects' edges for instance) will lead to large wavelet coefficients : most of the bit budget is spent in these regions. Similarly, in the Modified Discrete Cosine Transform (MDCT) domain, i.e. the cosine-based filterbank of AAC, the large coefficients represent the perceptually dominant sinusoidal harmonics of the musical content. With a smart quantization of these few large transform coefficients, and an efficient indexing of their parameters, these coders achieve a high compression ratio at virtually no loss of perceptual quality (typically at 1:20 or more for JPEG-2000, and 1:6 for MPEG-2/4 AAC).

But why does it work – where does this energy compaction come from? This is basically due to the fact that the transform basis elements "look like" elementary components of the analyzed signals: 2-D wavelets look like the edges of objects in images, discrete cosines look like the harmonics of musical notes. Only a few of these elementary building blocks are thus sufficient to well approximate the signals. It should be emphasized that the corresponding algorithms have a relatively low complexity: in the orthonormal bases described above (discrete wavelets / MDCT), selecting the set of significant coefficients involves a simple thresholding.

However, with all these nice properties also comes a major flaw: orthogonal bases are usually too rigid to accommodate even basic invariance properties of our signals. For instance, standard wavelet image codecs do not have shift- nor rotation-invariance: if the object pictured is slightly moved and/or tilted then its transform representation is fundamentally different. Similarly, in the audio domain, the MDCT is not shift-invariant: depending on the exact position of the signal with respect to the analysis frames, the transform coefficients may be radically different and so is the compression efficiency. Furthermore, the single frame length of the MDCT is inappropriate to simultaneously represent both the very sharp attack transients at the onset of percussive notes (where very short windows are desirable), and the long harmonics of tones (where a high frequency resolution is needed, hence long frame sizes).

To achieve higher sparsity, the key is to use decomposition spaces that have more basis vectors than orthonormal bases, and thereby more flexibility. These extended bases are called overcomplete, or redundant bases. Would you like time-shift invariance in your audio transform? The discrete Gabor Transform, as known for 1-D signals as Short-Time Fourier Transform, is nearly shift-invariant at the cost of (at least) doubling the size of the basis. Would you like shift-invariance in your image coder? The dual-tree complex wavelet [3] offers you this (approximately), but it is now 4-times overcomplete. With such overcomplete bases, sparsity is improved: basically, the larger the basis (the more redundant) the more likely it is that, for every local feature of the signal, there will be one basis vector that nearly fits. Overcompleteness brings flexibility and generality in the class of signals that are well, sparsely, represented. Recently, prototype codecs have been developed in many fields of multimedia, for example image [4], audio [5], [6] or video [7]. At very low bitrates (i.e., very high compression ratio), these new codecs outperform standard codecs based on orthogonal transforms. Besides coding, there are also many applications that benefit from this sparse energy compaction property [8], for instance information extraction [9], [10], source localization [11] or source separation [12], [13].

So why are these sparse overcomplete transforms not used in

Laurent Daudet is Professor at the Université Paris Diderot - Paris 7, Institut Langevin "Waves and Images" (LOA), 10 rue Vauquelin, 75005 Paris, France. This research was supported by the French GIP ANR under contract ANR-06-JCJC-0027-01 "DESAM". Part of this research was done during an Audio research residency at the Banff Centre, Canada. Email: laurent.daudet@espci.fr

widespread, standardized codecs? This is primarily due to their computational cost. Most modern multimedia applications now require on-the-fly encoding, and here the processing time for audio in [5] is typically 100 times slower than real-time, or the codec of [4] needs up to one hour for just one still image! As we shall see, as soon as the transform basis becomes overcomplete, selecting the coefficients is not as easy as the simple thresholding used in the orthogonal case: now the coefficients are no more mutually independent, and thus selecting one coefficient affects all the others. The delicate art of sparse approximation is a constant struggle against combinatorial complexity, as described in the next section.

The study presented in this paper is a proof of concept that, in the framework of sparse overcomplete decompositions, real-time processing of large streams of multimedia data can potentially be achieved with very simple parallel algorithms based on the simple and effective Matching Pursuit algorithm. The paper is organized as follows. Section I presents a short overview of the main complexity issues behind the sparse approximation problem, which can be skipped by the reader already familiar with sparse representations. Section II introduces the class of greedy methods for sparse decompositions, and discuss why these are good candidates for parallelization. In section III, we introduce different strategies for local Matching Pursuits. Preliminary results, described in section IV, illustrate the potential of our new Parallel Local Matching Pursuit (PLoMP) algorithm. Finally, section V contains concluding remarks.

I. Sparse overcomplete decompositions : a massive combinatorial problem

Let us introduce some notation : let $\mathbf{x} \in \mathbb{R}^N$ be a length-N real signal. We would like to find an optimal (sparsest) decomposition of $\mathbf{x}$, as a linear combination of few elementary "atoms"[1] $\mathbf{g}_\gamma$ that belong to a large, overcomplete set, called dictionary $\mathcal{D} = \{\mathbf{g}_\gamma, \gamma \in \Gamma\}$. In general, dictionaries are collections of elementary waveforms, that are chosen to represent the local characteristics of the class of signals under study. In the simplest case, dictionaries are concatenations of orthonormal bases (e.g. Fourier sinusoids + Dirac impulses, or Local Fourier bases with different window sizes). Dictionaries can also be learned from a set of signals [15].

Here, we want to approximate $\mathbf{x}$ as a weighted sum of a known number K of atoms, for instance corresponding to our bit budget:

$$\mathbf{x} \approx \sum_{k=0}^{K-1} \alpha_{\gamma_k} \mathbf{g}_{\gamma_k} \tag{1}$$

This is illustrated on Fig. 1, with a signal represented by a linear combination of 6 atoms from the dictionary: the problem consists of finding the optimal scale - time - frequency parameters for each atom in this set, and the corresponding weights α.

[1]The term *atom* is a slight abuse of language, as it literally implies that the decomposition is unique, which in general is not true with overcomplete dictionaries. However, this analogy with physics dates back from the 1947 pioneering work of D. Gabor on the *acoustical quanta* [14] and is still prevailing today.

Now, we can formulate our

Direct sparse approximation problem
With dictionary $\mathcal{D} = \{\mathbf{g}_\gamma, \gamma \in \Gamma\}$ and K fixed, find the set of indices $\{\gamma_k\}_{k=0\cdots K-1}$ and corresponding coefficients $\{\alpha_{\gamma_k}\}_{k=0\cdots K-1}$ that minimize $\|\mathbf{x} - \sum_{k=0}^{K-1} \alpha_{\gamma_k} \mathbf{g}_{\gamma_k}\|_2$.

Unfortunately, this problem cannot be solved exactly for real-size problems, because of its combinatorial nature. Indeed, if M is the total number of atoms in the dictionary $\mathcal{D}$, there are $\binom{M}{K}$ combinations of indices $\{\gamma_k\}_{k=0\cdots K-1}$ to test, and for each of these finding the least-squares optimal set of coefficients $\{\alpha_{\gamma_k}\}_{k=0\cdots K-1}$ is an orthogonal projection problem, requiring (in general) the inversion of a $K \times K$ matrix.

To circumvent this problem, there are two different options. The first one is to design a *similar problem* that can be solved exactly. The other approach looks for an *approximate solution* of the original problem, obtained with a tractable algorithm. In both cases, one is left to trust that the obtained solution is not far from the optimal one.

As the rest of the paper is devoted to the latter solution, let us quickly review the first option. Instead of having an exact sparsity constraint (number of non-zero elements bounded by some number K), we can look for signals that have few large coefficients and a lot of very small ones. Amongst all measures for this "relaxed sparsity problem", it is common practice to use the ℓ_1-norm $\|\mathbf{x}\|_1 = \sum_{k=0}^{N-1} |x(k)|$. Now, as the ℓ_1-norm has good convexity properties, it is possible to jointly optimize the data fidelity and the ℓ_1-sparsity of the set of coefficients, by standard quadratic programming [16], interior point methods [16], or a modification of least angle regression (LARS) technique [17]. There are also similar problems that can be solved more efficiently, such as the Dantzig Selector [18] that only requires a linear-time program and is hence applicable to large datasets. It should be noted that these methods have received renewed attention lately in the framework of compressive sensing [19], [20], which involves similar types of optimization problems. In this context, there is currently a large activity to adapt these sparse solvers on multicore or GPUs [21], [22], [23].

However efficient these methods may be, we have chosen not to use them in this study. The main reason is that our long-term goal is to design a "real-time" (though with delay) sparse decomposition algorithm, applicable to an incoming stream of data. In general, the previously described algorithms deal with the signal as a whole, and are therefore more suited to an offline scenario. The simplest greedy methods described in the next section, however, can very easily be modified into a "local" implementation, amenable to on-the-fly processing. Another advantage of greedy methods is that they are natively scalable in complexity, hence usable on any hardware architecture, and virtually any signal size. Finally, they provide an intuitive view of the involved mechanisms. Accordingly, in the rest of this paper, we will only consider greedy methods.

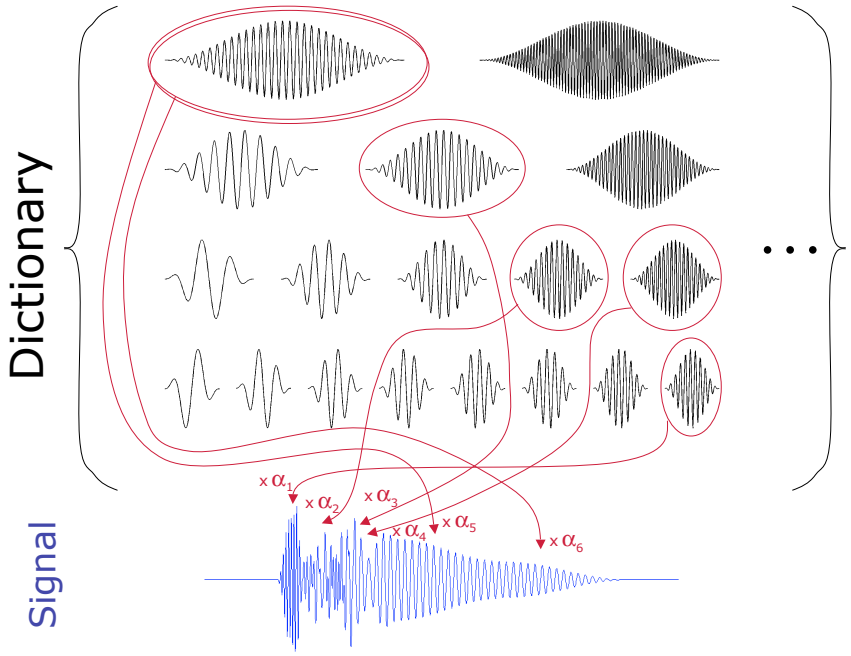

Fig. 1. Finding the best sparse decomposition of a signal, in a large dictionary of elementary waveforms, is a hard optimization problem that in general cannot be solved by brute force.

II. Greedy methods for sparse approximations

Greedy methods [24] are based on a simple divide-and-conquer principle: they select one atom, subtract its contribution, and iterate on the residual. The efficiency of these methods arise from the fact that, in order to select only one atom ($K = 1$), the Direct sparse approximation problem is easily solved: the parameter γ_{opt} and the corresponding coefficient $\alpha_{\gamma_{opt}}$ that minimize $\|\mathbf{x} - \alpha_\gamma \mathbf{g}_\gamma\|_2$ are obtained by selecting the best orthogonal projection on individual atoms, i.e., by simple scalar products (correlation) between the signal and the atoms: $\gamma_{opt} = \text{argmax}_{\gamma\in\Gamma}|\langle \mathbf{x}, \mathbf{g}_\gamma\rangle|$, and $\alpha_{\gamma_{opt}} = \langle \mathbf{x}, \mathbf{g}_{\gamma_{opt}}\rangle$. This is the basis for the Matching Pursuit (MP) algorithm [25], whose pseudo-code is given in Algorithm 1. Basically, it can be described as follows : at each iteration, find in the dictionary $\mathcal{D}$ the unit-norm atom best correlated with the signal (with the correlation computed as a scalar product, *atom selection stage*), subtract its contribution by (least-squares) orthogonal projection, and reiterate. There are more elaborate strategies such as Orthogonal Matching Pursuit [26], Low Complexity Orthogonal Matching Pursuit [27], Relaxed Greedy Algorithm [28], or Gradient Pursuit [29], etc), that find a better minimizer of the error by considering the whole set of already selected atoms, from previous iterations. For the sake of simplicity, we do not consider them in this study.

Besides giving good approximate solutions of the sparse decomposition problem, MP is very appealing: first, its design is extremely simple, offering flexibility to adapt to the problem at hand; and second, it is scalable in complexity: at any given time, every subsequent iteration adds a vector and reduces the approximation error. The algorithm can be stopped at any time, depending on the available resources and/or the target precision. In a basic, most general implementation, two steps can potentially be computational bottlenecks: the computation of the scalar products, and the atom selection stage (respectively marked with (*) and (**) in Algorithm 1).

Algorithm 1 Matching Pursuit (MP)

Require: signal $\mathbf{x} \in \mathbb{R}^N$, dictionary $\mathcal{D} = \{\mathbf{g}_\gamma, \gamma \in \Gamma\}$, maximum number of iterations N_0
Ensure: $\{\alpha_\gamma, \gamma \in \Gamma\}$ set of coefficients
  $n \leftarrow 0$ index of iteration
  $\mathbf{r}^0 \leftarrow x$ residual at initialization
  $\alpha_\gamma \leftarrow 0, \forall \gamma \in \Gamma$
  repeat
    $c_\gamma = \langle \mathbf{r}^n, \mathbf{g}_\gamma\rangle, \forall \gamma \in \Gamma$ *scalar products computation (*)*
    $\gamma_{opt} \leftarrow \text{argmax}_{\gamma\in\Gamma}|c_\gamma|$ *atom selection stage (**)*
    $\mathbf{r}^{n+1} \leftarrow \mathbf{r}^n - c_{\gamma_{opt}}\, \mathbf{g}_{\gamma_{opt}}$ *residual update*
    $\alpha_{\gamma_{opt}} \leftarrow \alpha_{\gamma_{opt}} + c_{\gamma_{opt}}$
    $n \leftarrow n + 1$
  until $n = N_0$ iterations performed or required precision reached
  Return $\{\alpha_\gamma, \gamma \in \Gamma\}$ set of decomposition coefficients, such that $\mathbf{x} = \sum_{\gamma\in\Gamma} \alpha_\gamma \mathbf{g}_\gamma + \mathbf{r}^n$

As described above, these greedy iterative methods are essentially sequential. In the most general case, atoms have the same length as the signal itself, and any iteration is based on the residual of the previous one. Therefore, the potential

gain in parallelizing is weak, and limited to sub-tasks such as the update of the scalar products. In practice, atoms with arbitrary shape and support are barely used, as the cost of updating the scalar products is often prohibitive (updating the scalar product of the length-N signal with P atoms requires in general $N \times P$ multiplications) : one prefers fast algorithms such as the (local) FFT or the Mallat's pyramidal algorithm for the discrete wavelet transform. When such structured time-localized atoms are used, only a local update of the scalar products must be performed, considerably speeding-up the process. For 1-D signals, there are now a number of flexible, optimized packages for Matching Pursuit, such as the open-source Matching Pursuit ToolKit (MPTK) [30]. However, for large signals, highly redundant dictionaries and/or high precision (i.e., large number of iterations), the decomposition time can still be large. As reported above, computation times are in the order of 1 hour for the processing of one image in [4], and of typically 100 times the duration of the audio piece in [5].

The goal of this research is to show how such principles can be generalized to "parallelize" MP – or more generally any greedy pursuit algorithm. The main idea is to break down the problem into a number of threads that could be handled by different processor cores working on different portions of the same signal. It should be emphasized that our approach is fundamentally different from the previously published work on parallel MP [31], [32], [33], that present efficient multi-core *implementations* of MP by optimally distributing the computationally intensive steps of MP (atom selection stage, update of the scalar products), hence minimizing the message passing between subtasks. Instead, in the current work, we present a "sub-optimal" version of MP that allows a straightforward parallelization of the MP algorithm : different threads work on different portions of the signal. The first benefit is that there is no message passing between tasks – which may be a limiting factor when scaled to a very large number of tasks, and may lead to a performance strongly dependent on the architecture. Our approach is fully scalable, in the way that is adapts to the processing power at hand (basically, dividing the load by the number of cores, with a negligible master process), and does not require any platform-dependent implementation or parameter optimization. The second benefit is that there is no need to know, or load into memory, the entire signal. For a typical 5-minutes musical song at CD-quality with more than 10^7 samples, there is no strong penalty in working locally on time frames whose size match the largest coherent "objects" in our signals (e.g., musical notes), with a typical duration of 0.1 s. This opens a perspective on "real-time" processing of an incoming stream of audio data – though with a delay that can become significant. The penalty to pay is that we are no longer guaranteed to make optimal choices at every iteration as in the plain MP. We shall see that, for typical signals, a smart algorithm design can not only reduce this "penalty", but even take advantage of it, at high number of iterations. Throughout this article, all the examples will be audio signals, as 1-D signals are the easiest way to present such algorithms, however, similar principles could potentially apply to other signals with similar scaling structures.

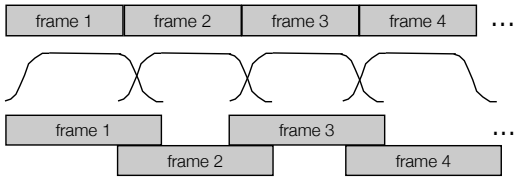

Fig. 2. Parallel processing with (top) fixed frame-based segmentation, or (bottom) fixed smooth overlapping windows.

III. Fixed vs. sliding local matching pursuit

As a first approach, we divide the computational burden of MP by working locally on fixed adjacent segments ("frames") of the signal (see Fig. 2). The simplest strategy is to use block-based frames : the signal is simply divided into equal-length frames (top plot of fig. 2) and a local MP is applied on each of them. Although this strategy may be acceptable for some signals, for the audio signals studied here it provides sharp blocking effects : transients at edges, varying quality across edges.

An alternate strategy is to use overlapping frames with smooth windows (bottom plot of Fig. 2). It is important to mention that the window is not applied on the signal but is a re-weighting of the scalar products at the atom selection stage (step (**) in Algorithm 1), reducing the likeliness of choosing atoms at the side of the frames. This Weighted Matching Pursuit (WMP) is described in Algorithm 2.

Algorithm 2 Weighted Matching Pursuit (WMP)

Require: signal $\mathbf{x} \in \mathbb{R}^N$, dictionary $\mathcal{D} = \{\mathbf{g}_\gamma, \gamma \in \Gamma\}$, maximum number of iterations N_0, **set of weights** $\{w_\gamma, \gamma \in \Gamma\}$.
Ensure: $\{\alpha_\gamma, \gamma \in \Gamma\}$ set of coefficients
  $n \leftarrow 0$ index of iteration
  $\mathbf{r}^0 \leftarrow x$ residual at initialization
  $\alpha_\gamma \leftarrow 0, \forall \gamma \in \Gamma$
  repeat
    $c_\gamma = \langle \mathbf{r}^n, \mathbf{g}_\gamma \rangle, \forall \gamma \in \Gamma$ *scalar products computation*
    $\gamma_{opt} \leftarrow \text{argmax}_{\gamma \in \Gamma} |w_\gamma \, c_\gamma|$ ***weighted*** *atom selection stage*
    $\mathbf{r}^{n+1} \leftarrow \mathbf{r}^n - c_{\gamma_{opt}} \, \mathbf{g}_{\gamma_{opt}}$ *residual update*
    $\alpha_{\gamma_{opt}} \leftarrow \alpha_{\gamma_{opt}} + c_{\gamma_{opt}}$
    $n \leftarrow n + 1$
  until $n = N_0$ iterations performed or required precision reached
  Return $\{\alpha_\gamma, \gamma \in \Gamma\}$ set of decomposition coefficients, such that $\mathbf{x} = \sum_{\gamma \in \Gamma} \alpha_\gamma \mathbf{g}_\gamma + \mathbf{r}^n$

The weights w_γ are chosen according to the center times of the atoms : large weights for atoms centered around the middle of the frame, small weights on the side. More precisely, if t_γ is the centre time of the atom $\mathbf{g}_\gamma$, then $w_\gamma = w(t_\gamma / L)$, where L is the frame size and w any smooth tapering window defined on $[0, 1]$ such that $w(0) = w(1) = 0$, for instance a Hanning window (preliminary experiments indicate that the regularity of w is important but its exact design has little

influence on the final performance). With this modification, the boundary effects are significantly reduced, but, as seen on the bottom plot of Fig. 2, adjacent overlapping windows share a portion of the signal. If an atom is selected in this zone, then a message has to be passed to its neighbor, indicating that the signal there has been updated locally. Due to the tapering window, these events should be relatively rare, but their existence impose stringent parallel programming constraints with message passing and synchronization.

The simplest and more efficient strategy that we propose is to use instead sliding frames, such as the one displayed in Fig. 3: a given core acts on a (windowed) frame of the signal, that after a number of iterations moves forward. An alternate view of this problem is shown on Fig. 4, where multiple cores act on adjacent frames : after a number N_0 of iterations, the signal is shifted by h samples, with h a fraction of the frame size L. It is important to note that now, there is no overlap between adjacent frames, and therefore no need for message passing. This way, it is possible to use multiple core processors performing with the highest precision while guaranteeing real-time processing of the data - though with significant delay. If f_s is the signal sampling frequency, and $N_{iter/core}$ the guaranteed number of MP iterations per core per second (after initialization, the computational cost per iteration remains roughly constant), N_0 is simply given by $N_0 = (N_{iter/core} h)/(L_f f_s)$.

Algorithm 3 Parallel Local Matching Pursuit (PLoMP)

Require: incoming signal $\mathbf{x}$, frame length L, size-L local dictionary $\mathcal{D} = \{\mathbf{g}_\gamma, \gamma \in \Gamma\}$, set of weights $\{w_\gamma, \gamma \in \Gamma\}$, frame hop size h, number of cores K, number of iterations per core N_0.

Ensure: $\{\alpha_\gamma, \gamma \in \Gamma\}$ set of coefficients

$\mathbf{x}_{local} \leftarrow$ first $K * L$ coefficients of $\mathbf{x}$

repeat

equally divide $\mathbf{x}_{local}$ into K non-overlapping frames

parallel process each frame with WMP, N_0 iterations

store results in $\{\alpha_\gamma, \gamma \in \Gamma\}$

load next h samples of $\mathbf{x}$ and shift $\mathbf{x}_{local}$

until no more incoming signal

Return $\{\alpha_\gamma, \gamma \in \Gamma\}$ set of decomposition coefficients, such that $\mathbf{x} = \sum_{\gamma \in \Gamma} \alpha_\gamma \mathbf{g}_\gamma + \mathbf{r}^n$

The pseudo-code for this Parallel Local Matching Pursuit (PLoMP) algorithm is given in Algorithm 3. Apart from file input/output, PLoMP perfectly spreads the computational load onto the available computational resource, by making at each iteration K *independent* calls to the Weighted Matching Pursuit.

IV. Results

We have simulated these approaches for representing audio signals on a redundant dictionary of local cosines. We first used a test signal of a sum of 3 constant-amplitude sinusoids with well-separated frequencies, a signal considered as very sparse, with added white noise. The energy of the residual

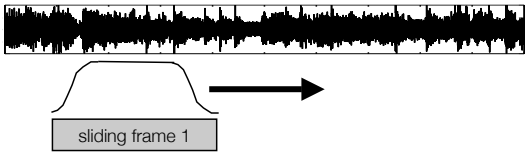

Fig. 3. One sliding frame with a smooth tapering window.

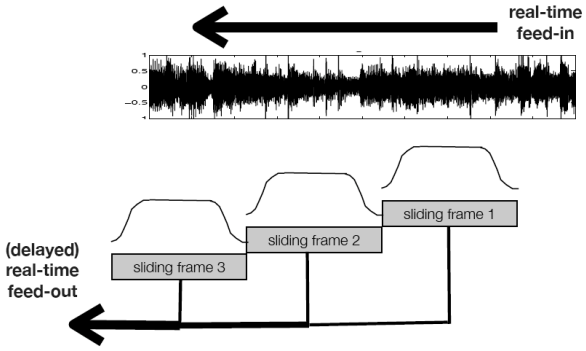

Fig. 4. Parallel processing PLoMP with adjacent smooth windows. The incoming signal is fed through the successive local processors, each one making just as many operations as to guarantee real-time-processing

is plotted on Fig. 5, as a function of the total number of iterations. Four different PLoMP configurations were tested, and compared to the standard MP algorithm :

1) one-pass PLoMP (one core, working locally on the signal)
2) a two-pass PLoMP ($K = 2$ cores working locally on the signal)
3) two one-pass PLoMP, while the second time the signal is entered in a time-reversed fashion – attempting to reduce the effect of time asymmetry in PLoMP.
4) a four-pass PLoMP ($K = 4$ cores).

Interestingly, for a small number of iterations per core, all parallel or sequential strategies are equivalent, corresponding to "good" choices: selected atoms remove energy only from the sinusoids. For a range of subsequent iterations, parallel strategies sometimes make "mistakes", choosing atoms in the noise or side lobes. However, at high precision, all the sinusoidal components have been removed and the performance of all strategies are similar. Note that in this regime, the parallel local MP strategies even obtain a slightly better asymptotic behavior than global MP (see inset of Fig. 5) ! It should also be noted that strategy (3), using two passes in alternate directions, results in slightly better results in the intermediate regime than the $K = 2$ PLoMP (strategy 2) (it can be guessed that some of the "bad choices" are a consequence of the time asymmetry of the local MP algorithm), but does not lead to any gain in the high-precision, asymptotic regime.

Then, we performed similar simulations on a real audio excerpt chosen for its large dynamics (jazz trio with loud piano notes / quiet double-bass + drums), and compared the standard MP to a $K = 5$ cores PLoMP, for a total number of iterations where the sound quality was deemed acceptable. For the same number of total iterations, the global SNR of the standard,

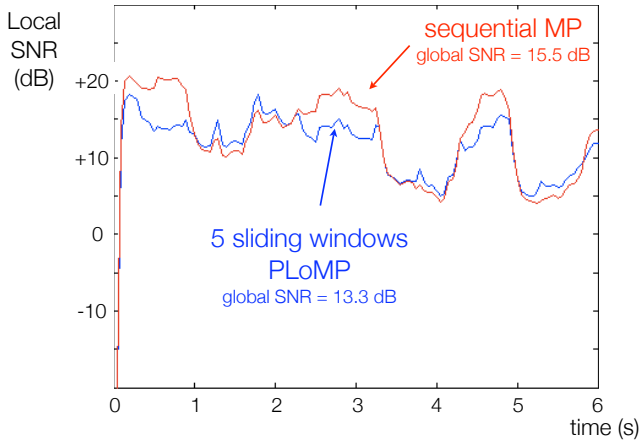

Fig. 6. Parallel vs. sequential processing on a real audio signal. Local Signal-To-Noise ratio (SNR) for sequential MP (red) and PLoMP (blue), for the same number of iterations.

sequential MP was 15.5 dB; while PLoMP resulted in 13.3 dB SNR.

However, looking at the global SNR may not be the only criteria to look at. Figure 6 shows the local SNR (computed on sliding windows of length 16384 samples, i.e. about 370 ms at 44.1 sampling rate), for the same total number of iterations, between the global MP and PLoMP. Although the average SNR is higher in the case of the global MP, the situation can be the opposite locally: PLoMP has a more steady local SNR. This can be beneficial from a perceptual point of view, and this is confirmed by listening to the soundfiles[2]: in PLoMP, the bass has significantly more presence, while it has almost disappeared in the global, sequential MP.

V. Conclusion

The widely-used MP algorithm is intrinsically a sequential algorithm. We have shown that it can be modified to work locally, for carefully chosen frame duration and window shape, and is therefore particularly well suited to multicore processing. Simulations show that this comes at a usually small penalty in performance, if any. For signals with large dynamics, it may even provide a better adaption to the specificities of the signal. Although still at a preliminary stage, this study paves the way to what is so far considered as intrinsically impossible : an "on-line" sparse solver for live multimedia continuous data streams. For high-quality audio, this would typically require tens of cores !

Acknowledgment

The author would like to thank Bob L. Sturm for proof-reading, and the anonymous reviewers for their insightful comments.

[2]Corresponding sound files can be downloaded at `http://old.lam.jussieu.fr/src/Membres/Daudet/SPM/`

References

[1] M. Adams, "The JPEG-2000 still image compression standard," *ISO/IEC JTC 1/SC 29/WG 1*, vol. 2412, 2001.
[2] M. Bosi, K. Brandenburg, S. Quackenbush, L. Fielder, K. Akagiri, H. Fuchs, M. Dietz, J. Herre, G. Davidson, and Y. Oikawa, "ISO/IEC MPEG-2 advanced audio coding," *Journal of the Audio engineering society*, vol. 45, no. 10, pp. 789–814, 1997.
[3] I. Selesnick, R. Baraniuk, and N. Kingsbury, "The dual-tree complex wavelet transform," *IEEE Signal Processing Magazine*, vol. 22, no. 6, pp. 123–151, 2005.
[4] L. Peotta, L. Granai, and P. Vandergheynst, "Image compression using an edge adapted redundant dictionary and wavelets," *Signal Processing*, vol. 86, no. 3, 2006.
[5] E. Ravelli, G. Richard, and L. Daudet, "Union of MDCT Bases for Audio Coding," *IEEE Trans. Audio Speech Lang. Proc.*, vol. 16, no. 8, pp. 1361–1372, 2008.
[6] R. Pichevar, H. Najaf-Zadeh, and L. Thibault, "A biologically-inspired low-bit-rate universal audio coder," in *Audio Engineering Society Convention, Vienna, Austria*, 2007.
[7] R. Neff and A. Zakhor, "Matching pursuit video coding. I. Dictionary approximation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 1, pp. 13–26, 2002.
[8] M. Plumbley, T. Blumensath, L. Daudet, R. Gribonval, and M. Davies, "Sparse representations in audio and music: From coding to source separation," *Proceedings of the IEEE*, 2009 (to appear).
[9] P. Leveau, E. Vincent, G. Richard, and L. Daudet, "Instrument-specific harmonic atoms for mid-level music representation," *IEEE Transactions on Audio Speech and Language Processing*, vol. 16, no. 1, p. 116, 2008.
[10] E. Ravelli, G. Richard, and L. Daudet, "Audio signal representations for indexing in the transform domain," *IEEE Transactions on Audio Speech and Language Processing*, to appear.
[11] D. Malioutov, M. Cetin, and A. Willsky, "A sparse signal reconstruction perspective for source localization with sensor arrays," *IEEE Transactions on Signal Processing*, vol. 53, no. 8 Part 2, pp. 3010–3022, 2005.
[12] R. Gribonval, "Sparse decomposition of stereo signals with matching pursuit and application to blind separation of more than two sources from a stereo mixture," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2002.
[13] A. Bronstein, M. Bronstein, M. Zibulevsky, and Y. Zeevi, "Sparse ICA for blind separation of transmitted and reflected images," *International Journal of Imaging Systems and Technology*, vol. 15, no. 1, pp. 84–91, 2005.
[14] D. Gabor, "Acoustical quanta and the theory of hearing," *Nature*, vol. 159, no. 4044, pp. 591–594, 1947.
[15] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, vol. 54, no. 11, p. 4311, 2006.
[16] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM review*, pp. 129–159, 2001.
[17] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Annals of statistics*, pp. 407–451, 2004.
[18] E. Candes and T. Tao, "The Dantzig selector: statistical estimation when p is much larger than n," *Annals of Statistics*, vol. 35, no. 6, pp. 2313–2351, 2007.
[19] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on information theory*, vol. 52, no. 2, pp. 489–509, 2006.
[20] D. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.
[21] A. Borghi, J. Darbon, S. Peyronnet, T. Chan, and S. Osher, "A simple compressive sensing algorithm for parallel many-core architectures," *CAM Report*, pp. 08–64, 2008.
[22] S. Lee and S. Wright, "Implementing Algorithms for Signal and Image Reconstruction on Graphical Processing Units," *submitted*, 2009.
[23] M. Andrecut, "Sparse Approximation of Computational Time Reversal Imaging," *Arxiv preprint arXiv:0904.3396*, 2009.
[24] J. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. Inf. Th.*, vol. 50, no. 10, 2004.
[25] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Sig. Proc.*, vol. 41, no. 12, pp. 3397–3415, 1993.
[26] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation withapplications to wavelet decomposition," in *27th Asilomar Conference on Signals, Systems and Computers*, 1993, pp. 40–44.

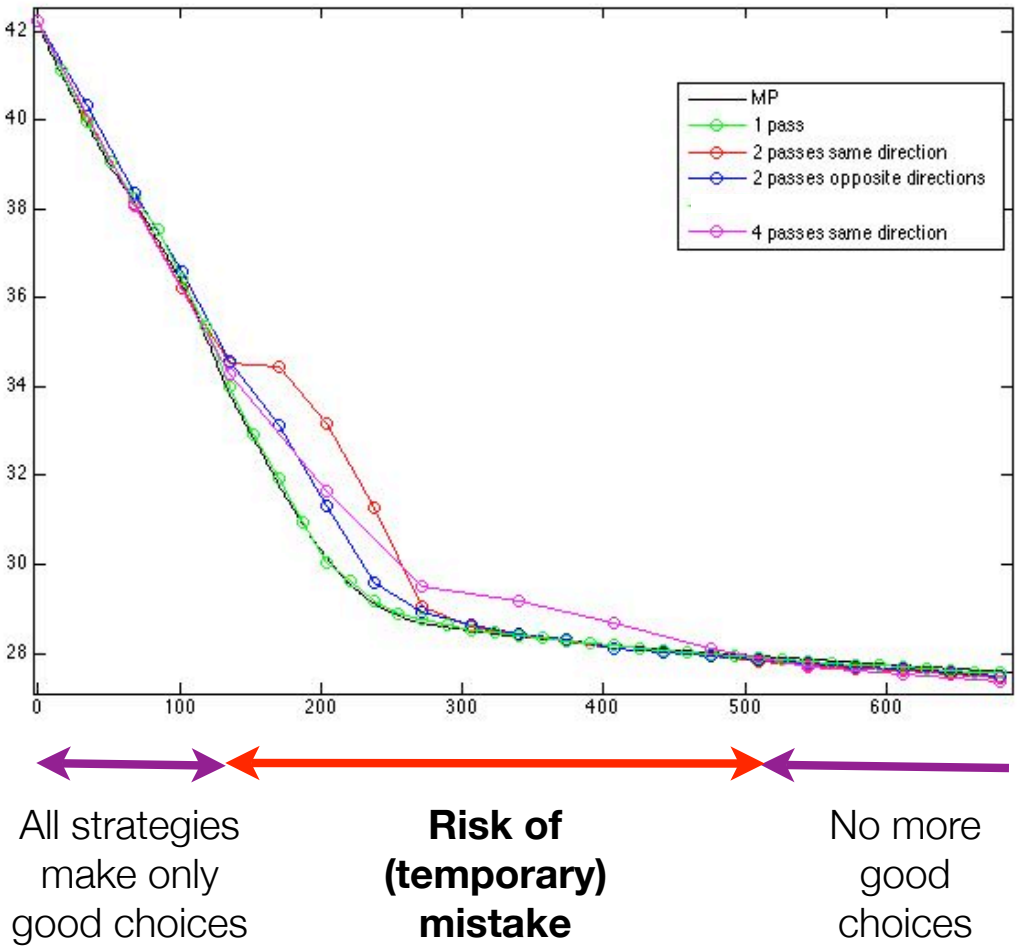

Fig. 5. Decay of the energy of the residual as a function of the number of iterations, for a signal being a sum of 3 constant sinusoids and the dictionary a union of local cosines with different scales. The plain black line is the reference global MP, other colors are local PLoMP with different strategies (different number of windows / number of iterations at a given position).

[27] B. Mailhé, R. Gribonval, F. Bimbot, and P. Vandergheynst, "A low complexity orthogonal matching pursuit for sparse signal approximation with shift-invariant dictionaries," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2009.

[28] A. Barron, A. Cohen, W. Dahmen, and R. DeVore, "Approximation and learning by greedy algorithms," *Annals of statistics*, vol. 36, no. 1, pp. 64–94, 2008.

[29] T. Blumensath and M. Davies, "Gradient pursuits," *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2370–2382, 2008.

[30] S. Krstulovic and R. Gribonval, "MPTK: Matching pursuit made tractable," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2006.

[31] G. Dodero, V. Gianuzzi, M. Moscati, and M. Corvi, "A scalable parallel algorithm for Matching Pursuit decomposition," in *Proc. HPCN*, 1998, pp. 458–466.

[32] A. Bultan and O. Arikan, "A parallelized matching pursuit algorithm for the four-parameter chirplet decomposition," in *Proc. Int. Symp. Time-freq. Time-scale Anal., Pittsburgh, PA,*, 1998, pp. 421–424.

[33] H. Feichtinger, A. Turk, and T. Strohmer, "Hierarchical parallel matching pursuit," in *Proc. SPIE Int. Soc. Opt. Eng.*, vol. 2302, 1994, pp. 222–232.