Akademie věd České republiky
Ústav teorie informace a automatizace

Academy of Sciences of the Czech Republic
Institute of Information Theory and Automation

RESEARCH REPORT

J. Hoskovec, P. Tichavský:

Experimental Comparison of Sparse Signal Recovery Algorithms

No. 2306 September 2011

ÚTIA AV ČR, P. O. Box 18, 182 08 Prague, Czech Republic
E-mail: utia@utia.cas.cz

Contents

Chapter 1

Introduction

This report presents an experimental comparison of some of the newest and/or most common algorithms that are used for solving the sparse recovery problem. In particular, the noiseless case is studied. No theoretical explanation of the problem's background will be provided here, since this has been already covered in many papers (for example [3] or, in Czech language, [8]) - the emphasis will be on the experimental results and the conditions having influence upon them.

The Compressed Sensing (CS) domain become to evolve rapidly since about 2005. This expansion is mostly associated with the names of Emmanuel Candès, Justin Romberg and Terence Tao ([10],[2],[9]). However, the theoretical background for the sparse recovery problems is over a decade older, mostly linked to the work of D.L. Donoho ([4],[5]), and originally concerned rather the field of statistics than that of signal processing.

Note: Throughout this report, the number of non-zero coefficients of a signal/vector is called *sparsity* or *degree of sparsity*, even though it is linguistically a nonsense (a 1-sparse signal is, by common sense, much *sparser* than a 15-sparse one!).

Chapter 2

Algorithms' Implementation

There are packages dedicated to the spare signal recovery available on the Internet, such as the SparseLab [1] package, developed by David Donoho and team. The following items from this package were tested: the greedy algorithms Matching Pursuit (MP), Orthogonal Matching Pursuit (OMP) and the standard convex relaxation algorithm called Basis Pursuit (BP).

Reweighted l_1 - optimisation (RL1)

The reweighted *l1*-optimisation algorithm was based on the paper by Candès et al.[11]. The weighted minimisation problem can be formulated as follows:

$$\mathbf{x}^{(k)} = arg\,min \|W^{(k)}\mathbf{x}\|_{l_1}, \;\; subject\,to \;\; \mathbf{y} = A\mathbf{x}, \tag{2.1}$$

where $W^{(k)}$ is a diagonal matrix with the weighting coefficients on the diagonal and zeros elsewhere ($W^{(k)}_{i,j} = w^{(k)}_i$ if $j = i$, 0 otherwise). At each step (k), the new approximation is calculated via a linear programming solver (such as, here, the SparseLab SolveBP function). An equivalent formulation of the problem (2.1), more suitable for the solver, is:

$$\mathbf{z}^{(k)} = arg\,min \|\mathbf{z}\|_{l_1}, \;\; subject\,to \;\; \mathbf{y} = A(W^{(k)})^{-1}\mathbf{z}, \tag{2.2}$$

which leads to the $\mathbf{x}^{(k)}$ signal estimate via $\mathbf{x} = W^{-1}\mathbf{z}$. Candès [11] proposed the following weights: $w^{(k)}_i$ are initialised at 1, $\forall i$ before the first iteration and then updated at each time as:

$$w^{(k+1)}_i = \frac{1}{|x^{(k)}_i| + \epsilon}, \tag{2.3}$$

where ϵ is a user-defined parameter. Here, it was chosen constant (independent of k).

The stopping condition for the algorithm (as well as for RL2) was chosen as follows. After each iteration, the last and the previous obtained approximation of $\boldsymbol{x}$ were compared - if the $\|\cdot\|_{l_\infty}$ norm of their difference was smaller than a user-defined lower-bound (here, 10^{-4}), the algorithm was stopped.

Reweighted l_2 - optimisation (RL2)

The RL2 algorithm is based on the FOCUSS algorithm invented by Gorodnitsky and Rao [6]. An analytical solution exists for the *(k+1)*th iteration of a reweighted l_2 minimization :

$$\mathbf{x}^{(k+1)} = arg\,min \|W^{(k)}\mathbf{x}\|_{l_2}, \quad subject\,to \;\; \mathbf{y} = A\mathbf{x}, \tag{2.4}$$

$$\mathbf{x}^{(k+1)} = \tilde{W}^{(k)} A^T (\lambda I + A\tilde{W}^{(k)} A^T)^{-1}\mathbf{y}, \tag{2.5}$$

where $W^{(k)}$ is a diagonal weighting matrix with the ith diagonal element $w_i^{(k)}$, $\tilde{W}^{(k)}$ a diagonal weighting matrix with the ith element $1/w_i^{(k)}$ and $w_i^{(k)}$ a weighting function depending on the previous iteration. (Since the noiseless case was treated in this report, λ was set to zero.) The weights were chosen as suggested by Wipf [12]:

$$w_i^{(k+1)} = [(x_i^{(k+1)})^2 + \epsilon |x_i^{(k+1)}|]^{-1}. \tag{2.6}$$

The stopping condition used was the same as for the RL1 algorithm.

A*OMP

Recently, a new extension of the OMP algorithm was proposed by Karahanoğlu and Erdoğan [7], combining the OMP with a tree-search technique. The algorithm was tested using the code given by the authors[1]. Default (author-suggested) parameters were used, more precisely:

$I = 3$; (number of initial paths in the search stack, integer, $I \geq 1$)
$B = 2$; (number of branches added to the search stack at each iteration)
$P = 200$; (maximum number of paths in the search stack)
$\alpha = 0.75$; (parameter for multiplicative auxiliary function, $0 < \alpha \leq 1$)
AuxMode = 'Mul' (we were using the multiplicative cost function) .

[1]The corresponding Matlab code has been made available at Karahanoğlu's personal website, *http://myweb.sabanciuniv.edu/karahanoglu/*.

Chapter 3

Experimental Results

3.1 First Experiment: $\mathbf{A} \in \mathbb{R}^{50\times 250}$

Experimental Set-up

The coding (dictionary) matrix $\mathbf{A} \in \mathbb{R}^{50\times 250}$ was randomly generated with normally distributed elements (with mean equal to 0 and variance to 1), then its columns were normalised to 1 in the l_2 norm.

The input signal $\mathbf{x} \in \mathbb{R}^{250}$ was also generated randomly, but with a user-chosen number of non-zero elements. For each algorithm, two cases were studied:

a) The case where non-zero coefficients of $\mathbf{x}$ have unit amplitudes (1 or -1 with equal probabilities), with the degree of sparsity k going from 1 to 20, or

b) the case of non-zero coefficients of $\mathbf{x}$ being normally distributed (with zero mean and variance 1), the sparsity going from 1 to 25.

Each test was performed 1000 times (for each algorithm, degree of sparsity and non-zero element distribution). To evaluate and to compare the algorithms' performance, we compared the following two sets:

- $I = \{\, i, |\mathbf{x}_i| > \epsilon \,\}$ and
- $J = \{\, i, |\hat{\mathbf{x}}_i| > \epsilon \,\}$,

where $\mathbf{x}$ is the original sparse vector, $\hat{\mathbf{x}}$ its reconstruction and ϵ a user-defined 'zero-threshold' - a value below which a number is considered negligible. Here, we used $\epsilon = 0.01$.

If the two sets were identical, the recovery attempt was classified as successful (the number of successful attempts divided by the number of all attempts giving the exact recovery rate). Moreover, two other criteria were observed: *err*, the number of 'lost' non-zero coefficients (its coordinates appear in I, but not in J) and *ext*, the number of 'false positives' - positions appearing in J, but not in I. Formally, $err = I \setminus J$ and $ext = J \setminus I$.

Experimental Results

3.1.1 Case of Unit Amplitudes

Figure 3.1: Exact reconstruction rate versus sparsity, non-zero elements have unit amplitudes.

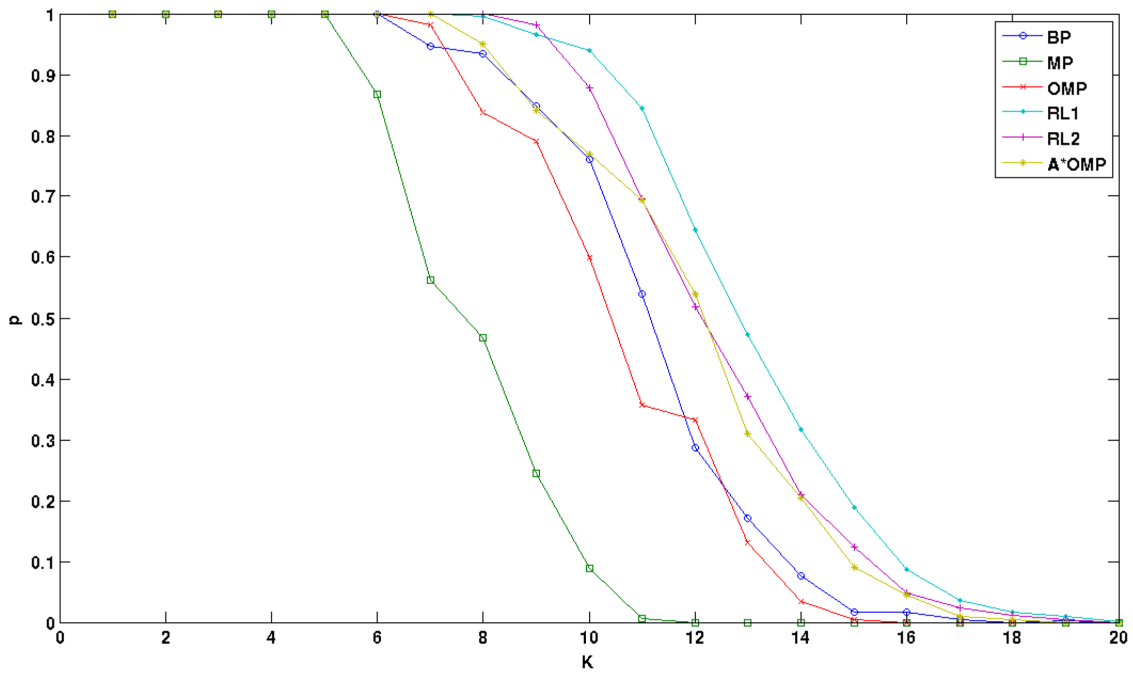


Figure 3.2: Average number of lost non-zero coefficients versus sparsity, non-zero elements have unit amplitudes.

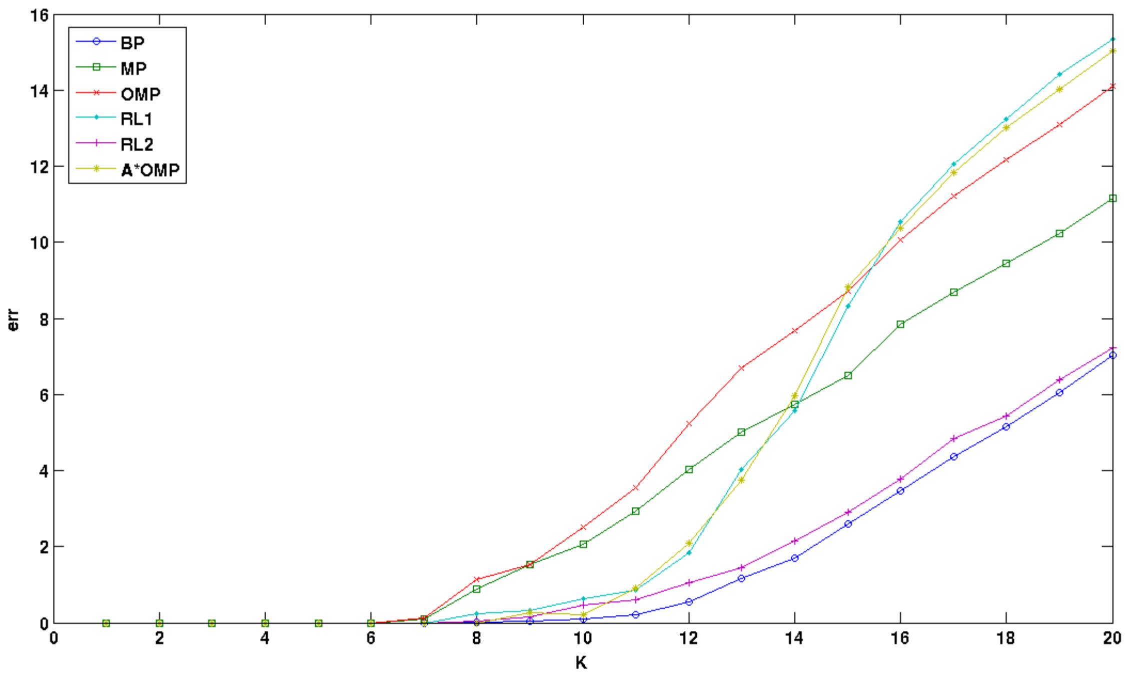
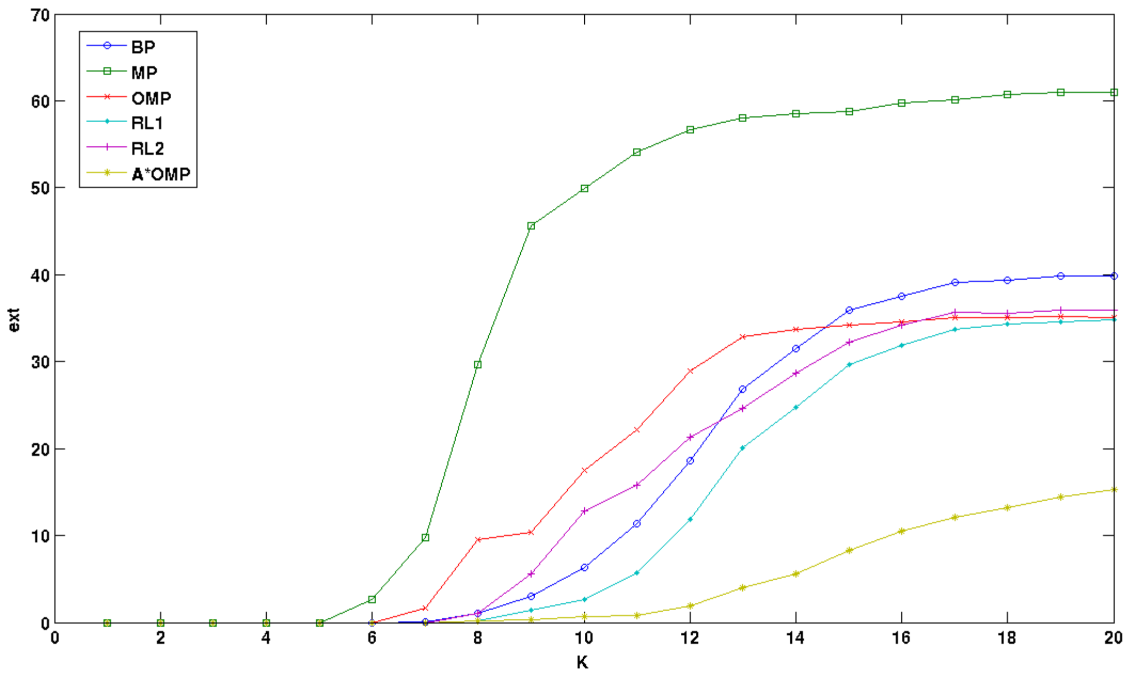


Figure 3.3: Average number of false non-zero coefficients versus sparsity, non-zero elements have unit amplitudes.



3.1.2 Case of Normally Distributed Elements

Figure 3.4: Exact reconstruction rate versus sparsity, non-zero elements are normally distributed.

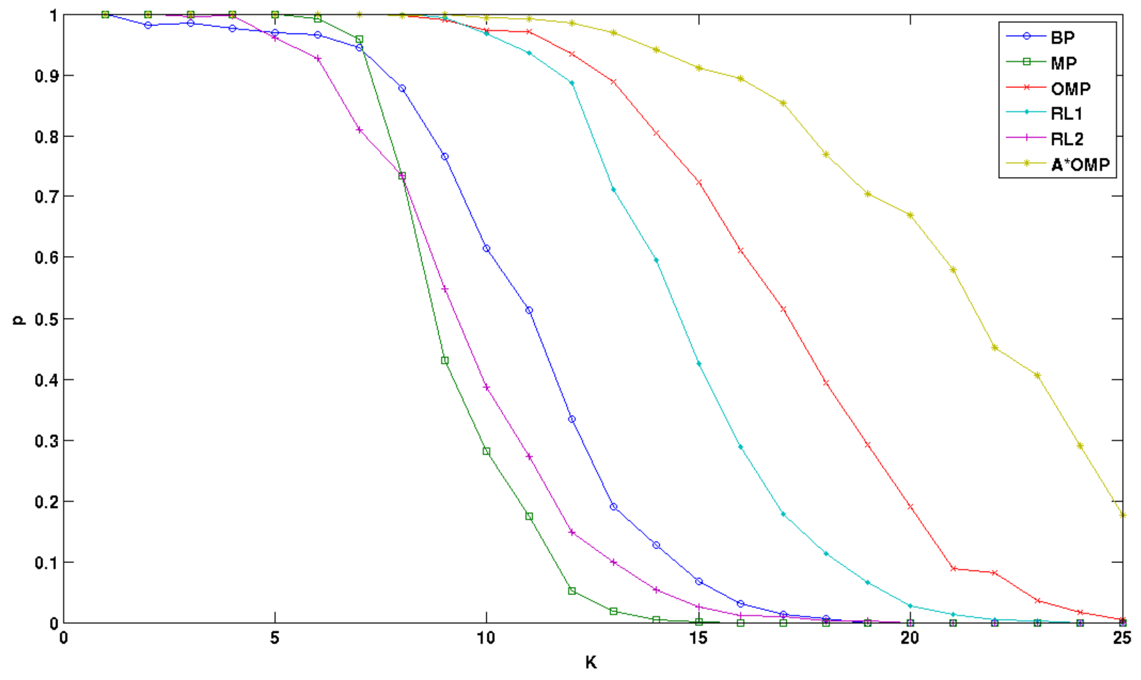


Figure 3.5: Average number of lost non-zero coefficients versus sparsity, non-zero elements are normally distributed.

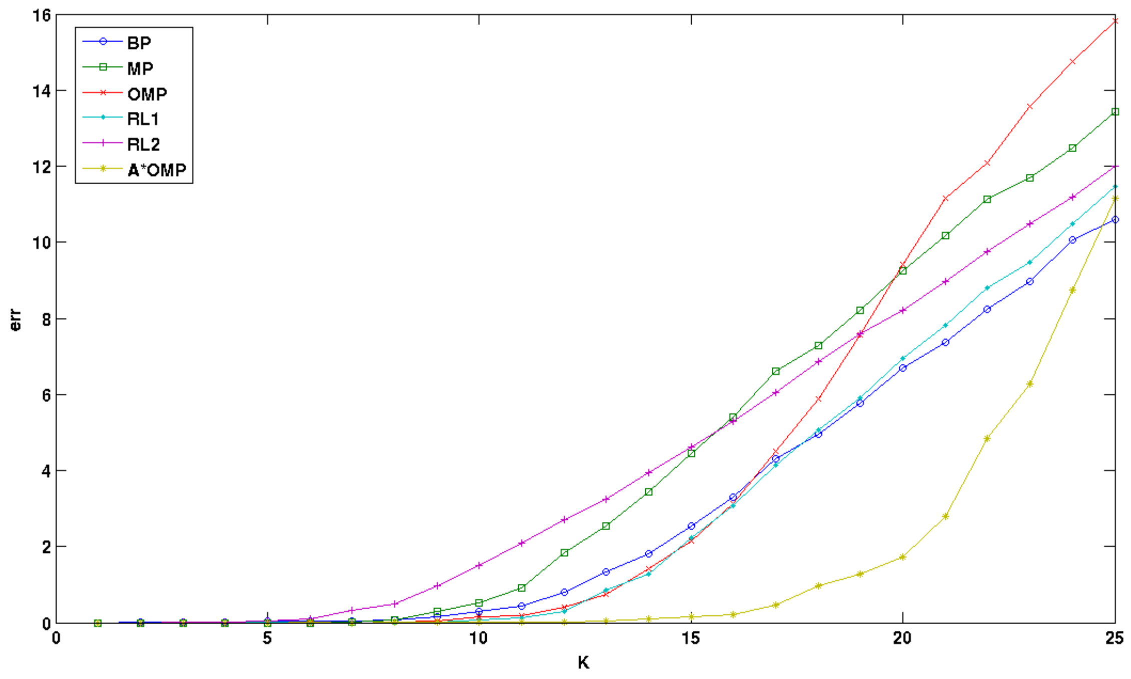
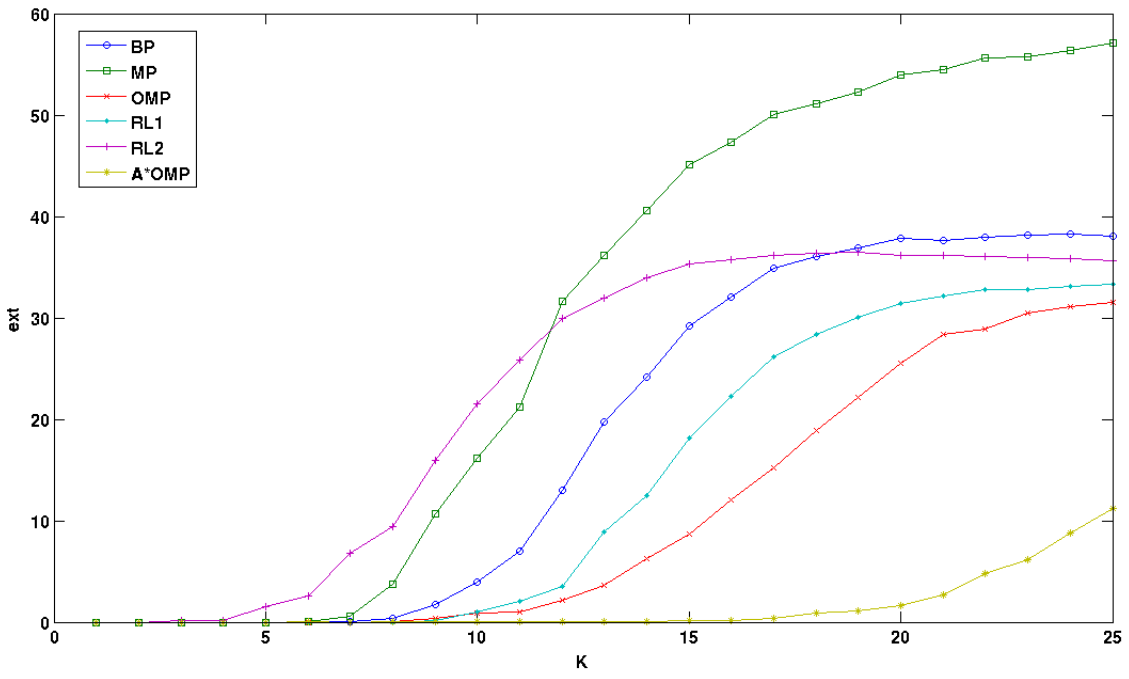


Figure 3.6: Average number of false non-zero coefficients versus sparsity, non-zero elements are normally distributed.



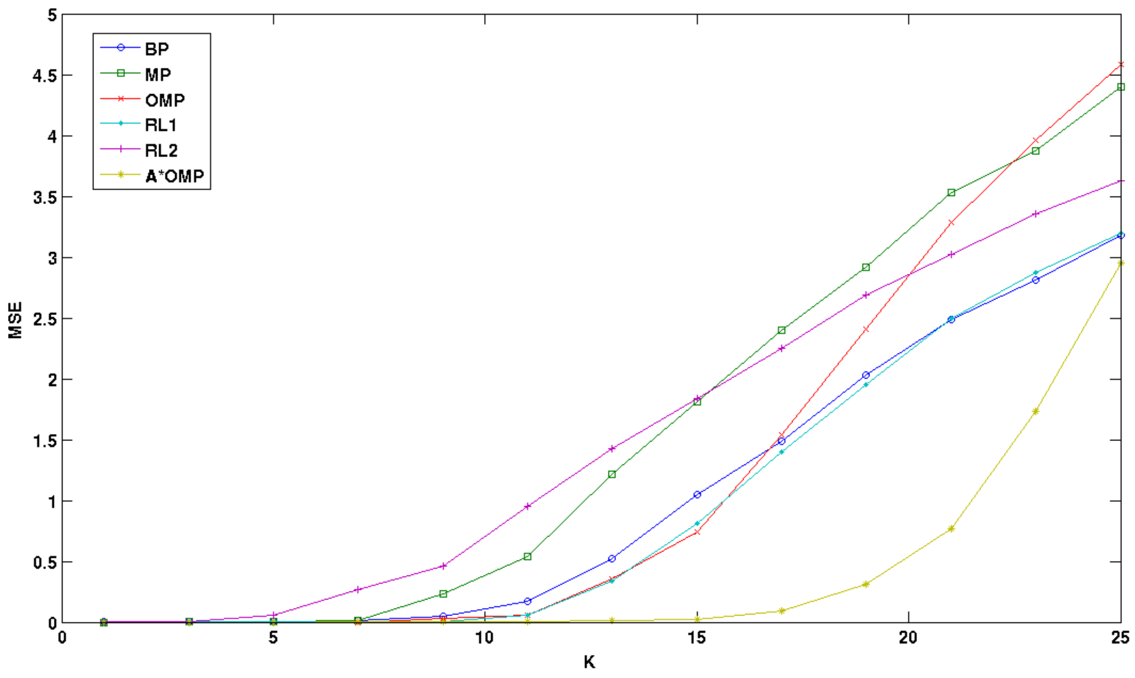
Mean square error in the case of Gaussian elements.

For normally distributed non-zero elements the exact reconstruction criterion, as introduced above, might be misleading. Indeed, the distinction between 'small' and 'big' elements is much less obvious than in case of unit amplitudes. There was a concern about the possibility of wrong conclusions due to this. Let us consider the following situation: In the original sparse vector, a coefficient was equal to 0.011. The reconstruction finds 0.009 instead, thus falling below the $\epsilon = 0.01$ threshold. Consequently, the coefficient is considered lost and the reconstruction unsuccessful, even though the final error between the original and the reconstructed signal might be acceptable. Therefore, the mean square error of the reconstructed signal was added as a new criterion:

$$MSE = \|\mathbf{x} - \hat{\mathbf{x}}\|_{l_2}. \tag{3.1}$$

This was evaluated for all the algorithms in the survey (BP, MP, OMP, RL1, RL2, A*OMP), with the sparsity k going from 1 to 25. To save some computation time, only odd degrees of k were taken into account. Also, even though for the first five algorithms the measure was averaged over 1000 trials, only 500 were carried out in the case of A*OMP.

Figure 3.7: Mean square reconstruction error versus sparsity, non-zero elements are normally distributed.



The comparison between the figures (3.7) and (3.5) shows an analogical behaviour of the algorithms in these two cases. That implies that every lost coefficient induces about the same l_2 error in the approximation, no matter which sparsity degree or which algorithm is being treated.

3.1.3 Computation Time Study

Experimental Set-up

As in the previous section, the coding matrix $\mathbf{A}$ has the size 50×250 and the input signals $\mathbf{x} \in \mathbb{R}^{250}$ are generated randomly with varying sparsity. Since some of the algorithms were time-consuming (especially A*OMP for large values of k, but also RL1, BP and, to a lesser extent, RL2), only 50 trials were carried out for each sparsity, algorithm and non-zero element distribution. The experiment took place on an Intel Core2 Duo 32-bit CPU running at 2.1 GHz with 1.9 gigabytes of RAM. The CPU time consumption was measured using internal Matlab functions and averaged over the 50 trials.

Figure 3.8: CPU consumption versus sparsity, non-zero coefficients have unit amplitudes

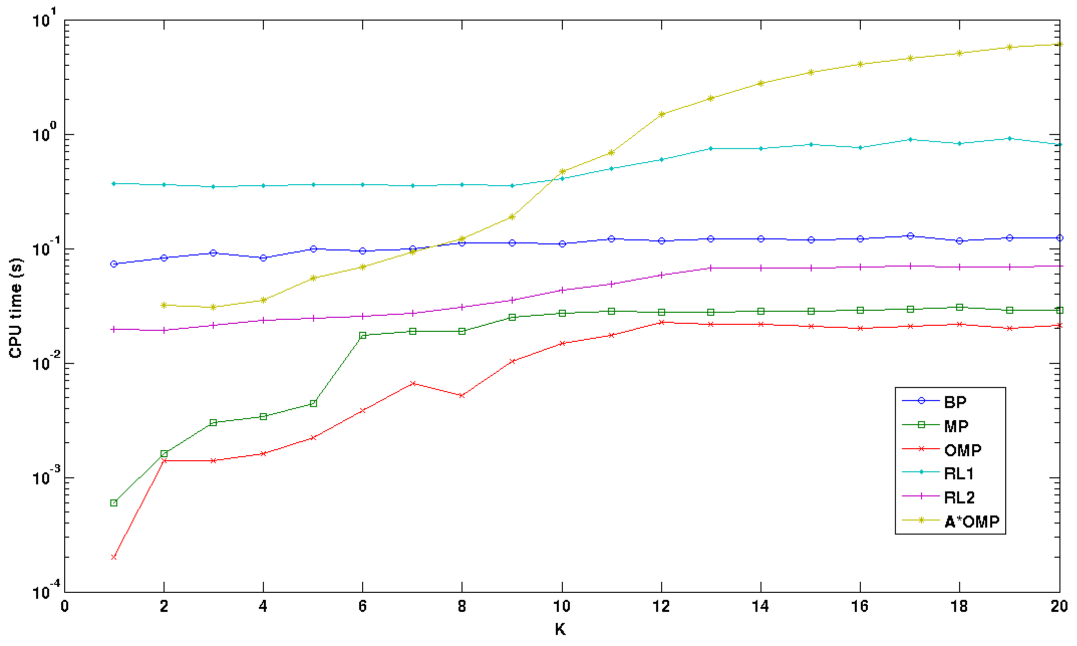
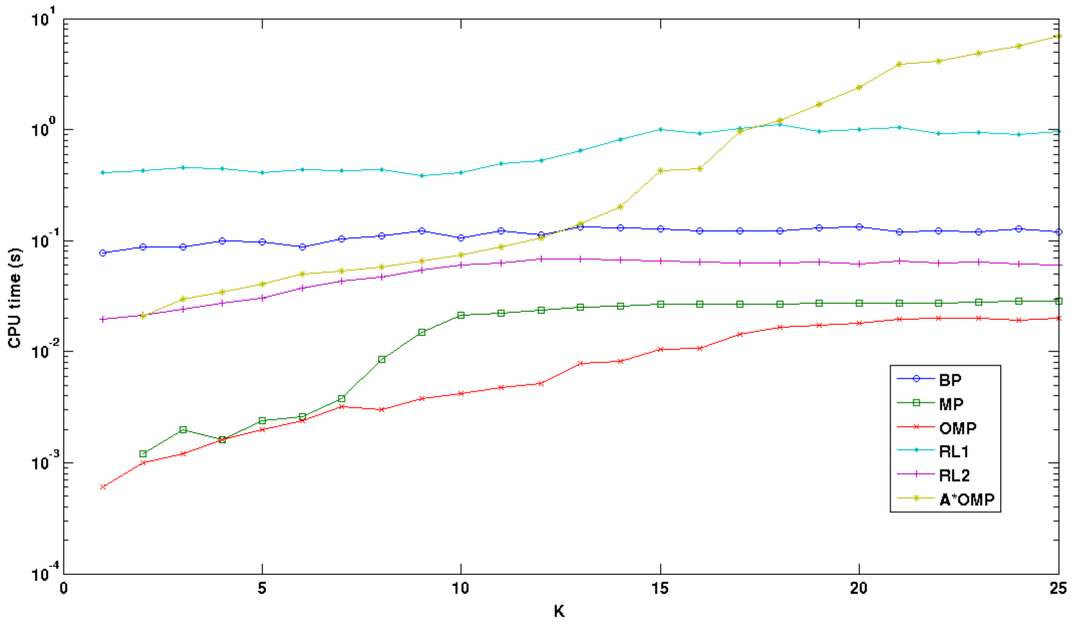


Figure 3.9: CPU consumption versus sparsity, non-zero coefficients are normally distributed



3.2 Big Data Experiment: $\mathbf{A} \in \mathbb{R}^{500\times2500}$

Experimental Set-up

As in the previous case, the coding matrix $\mathbf{A}$ was generated pseudo-randomly with normally distributed elements and its columns were normalised to 1 in the l_2 norm. This time, the dimensions of the problem were ten times greater: $\mathbf{A} \in \mathbb{R}^{500\times2500}$. The vectors $\mathbf{x} \in \mathbb{R}^{2500}$ were generated with Gaussian-distributed non-zero elements (zero mean, variance 1) in the first time and with unit amplitudes in the second time. The degree of sparsity was varying from 5 to 200 with a step of 5.

Since the problem's dimension lead to a serious concern about the computation time, the maximum number of trials to run at each value of k was set to 100 instead of 1000. Moreover, at each degree of sparsity k, the experiment was not allowed to take more than 200 seconds of CPU time (at the end of each reconstruction, the loop would stop if the time consumed since its beginning had become superior to 200 seconds). This lead sometimes to a number of reconstructions as low as 5 (RL2) or even 3 (A*OMP). But for the purpose of overall comparison of the algorithms in question, even such a small number of trials was enough to provide some interesting results.

The algorithms used were the same as in the previous case, with two exceptions:

- The BP and RL1 algorithms were omitted, since they are generally more time-consuming than all the others.
- The A*OMP algorithm's parameters were changed, so that the algorithm becomes faster: The number of initial paths I was set to 2 (instead of 3) and the maximum number of paths on the stack P was limited to 10 (instead of 200).

The values observed were the exact reconstruction rate and the mean square error of the reconstruction $\hat{\mathbf{x}}$ obtained ($MSE = \|\mathbf{x} - \hat{\mathbf{x}}\|_{l_2}$, averaged over the number of trials having run). To obtain information on the algorithms' speed, the number of trials having taken place within the time given and the average duration of each reconstruction were also measured. However, those two quantities are presented in another form (table of values rather than a plot).

3.2.1 Gaussian Non-Zero Elements

Figure 3.10: Exact reconstruction rate versus sparsity

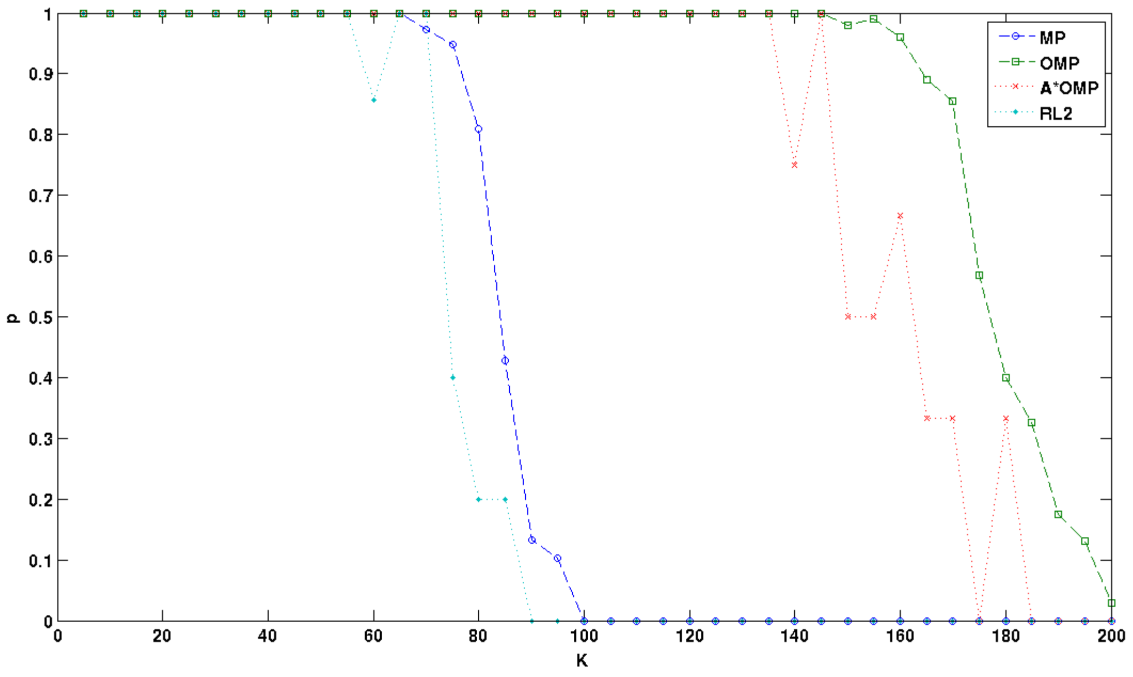


Figure 3.11: Mean square error of the reconstruction versus sparsity

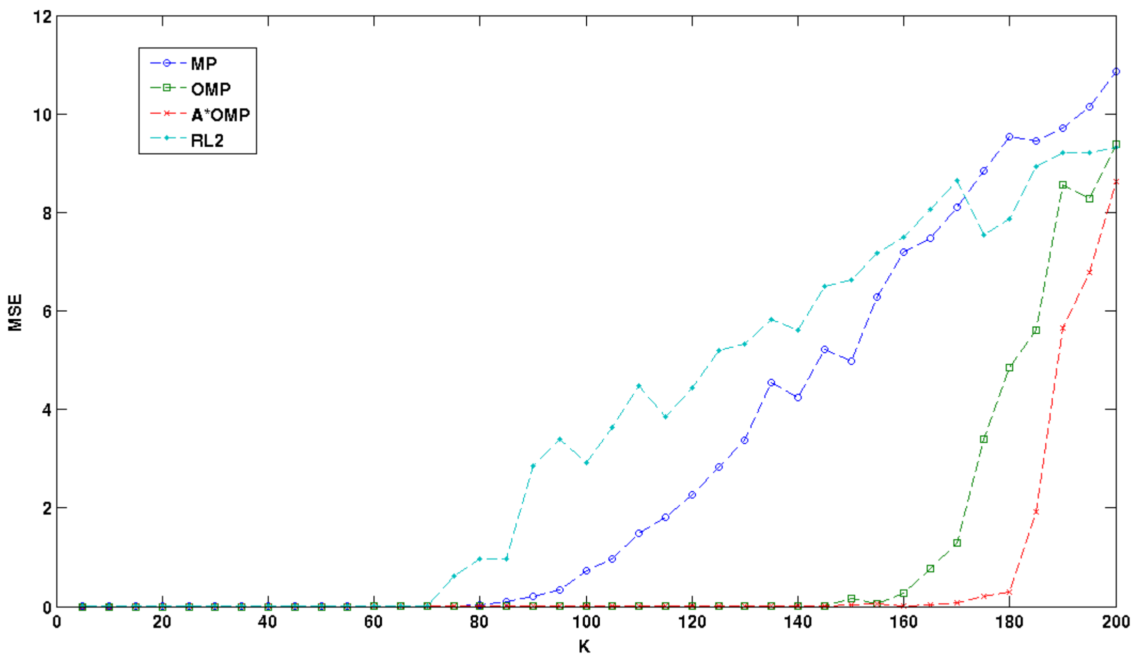


Table 3.1: Number of trials carried out within the time limit (max 200 seconds or 100 reconstructions)

k	*5*	*25*	*50*	*75*	*100*	*125*	*150*	*175*	*200*
MP	100	100	100	58	26	23	20	18	19
OMP	100	100	100	100	100	100	100	58	35
A*OMP	100	37	14	9	6	5	4	3	3
RL2	18	13	9	6	5	5	5	5	5

Table 3.2: Average CPU time consumed by a single reconstruction (in seconds)

k	*5*	*50*	*100*	*150*	*200*
MP	0.064	1.283	7.738	10.039	10.695
OMP	0.028	0.275	0.574	1.176	5.800
A*OMP	0.080	14.844	34.658	60.410	94.550
RL2	11.1917	23.737	40.944	41.952	42.220

3.2.2 Unit Amplitude Non-Zero Coefficients

Figure 3.12: Exact reconstruction rate versus sparsity

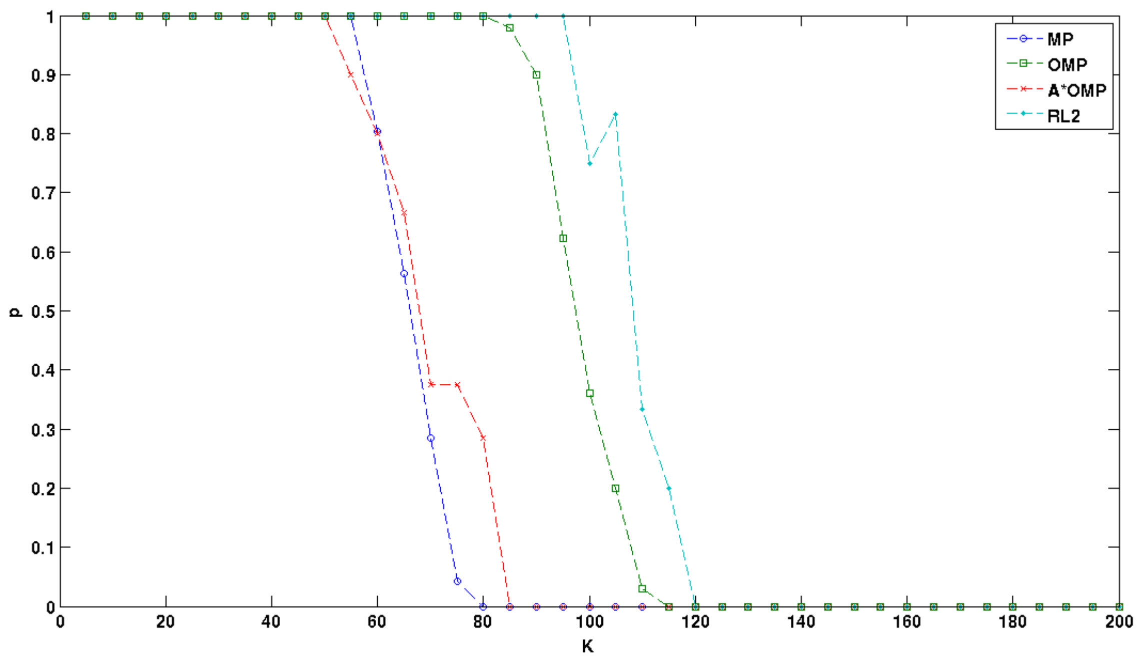


Figure 3.13: Mean square error of the reconstruction versus sparsity

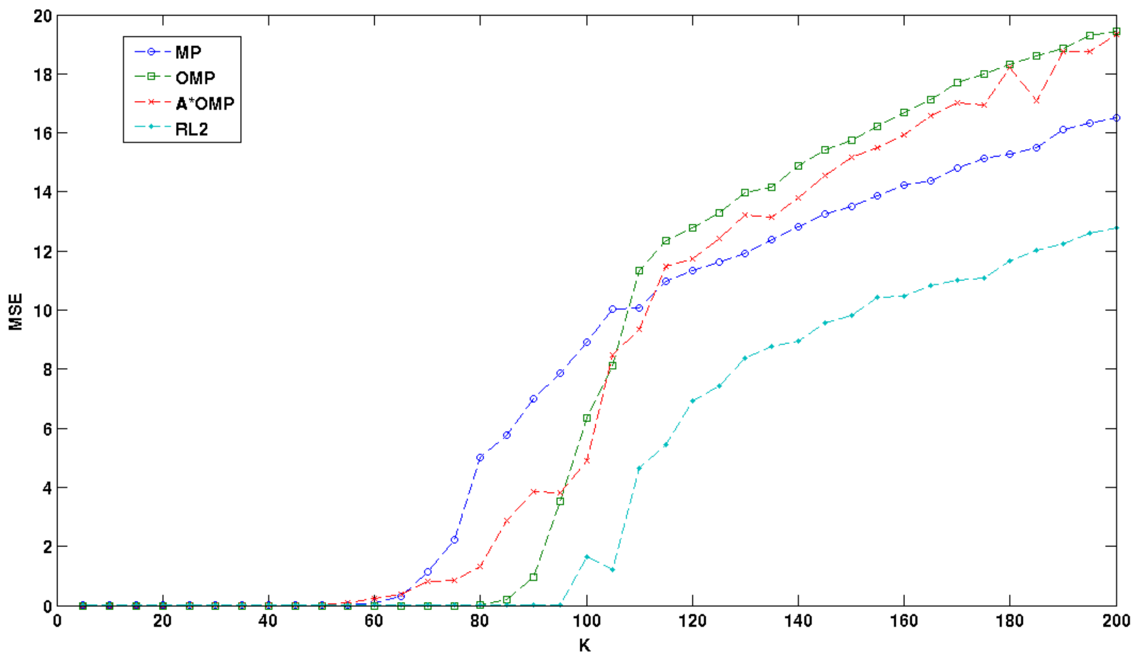


Table 3.3: Number of trials carried out within the time limit (max 200 seconds or 100 reconstructions)

k	*5*	*25*	*50*	*75*	*100*	*125*	*150*	*175*	*200*
MP	100	100	100	24	21	20	19	20	20
OMP	100	100	100	100	50	34	33	33	33
A*OMP	100	27	12	8	6	4	4	3	3
RL2	20	20	17	13	8	5	5	5	5

Table 3.4: Average CPU time consumed by a single reconstruction (in seconds)

k	*5*	*50*	*100*	*150*	*200*
MP	0.0823	1.5549	9.6248	10.6279	10.2085
OMP	0.0245	0.2516	4.0062	6.0670	6.1167
A*OMP	0.8759	17.7358	39.5517	62.6750	88.5600
RL2	10.1315	12.4100	29.8600	42.8060	43.7520

Chapter 4

Discussion

4.1 Individual Algorithm Survey

Matching Pursuit As mentioned in [13], the original Matching Pursuit algorithm is nowadays overcome. Indeed, its accuracy ranks regularly the last (see figures **3.1** and **3.4**), while not even being the fastest (fig. **3.8**, **3.9**). The importance of this algorithm is rather historical, as the very popular OMP algorithm derives directly from it.

Orthogonal Matching Pursuit The OMP algorithm is clearly the fastest among the algorithms studied here (fig. **3.8**, **3.9**), and that even for large data (tables **3.1**, **3.2**). We could have said 'by far the fastest' if there was not for the MP algorithm; however, the latter suffers from great inaccuracy. In terms of exact reconstruction rate, OMP preforms quite well, while being heavily influenced by the distribution of non-zero coefficients. Its performance is very good in the case of Gaussian amplitudes (fig.**3.4**) and mediocre - however not far behind the others - in the case of unit amplitudes (fig. **3.1**). The algorithm's major drawback is a relatively high number of coefficients lost in case of failure (fig. **3.2**, **3.5**), hence quite a high mean square reconstruction error in some cases - and that even when the exact reconstruction rate is being a success (fig. **3.7**, **3.13**).

Basis Pursuit Among the algorithms compared in this report, BP seems to be average in every way, with the exception of the number of errors made in case of failure, which is outstandingly small. It is a popular algorithm for comparison studies - its performance, both in terms of reconstruction exactitude and computing time, may often be considered a boundary between success and failure.

Reweighted l_1 Optimisation The time consumption of RL1 depends little on the sparsity degree. Its exact reconstruction rate is about the same regardless of distribution of non-zero coefficients, which makes it the best performing algorithm in case of unit amplitudes. However, the algorithm is several times slower than BP (fig. **3.8**, **3.9**), as it consists of repetitive calls of the latter.

Reweighted l_2 Optimisation The RL2 algorithm is faster than RL1 (and BP), while conserving its property of time-consumption stability as the degree of sparsity varies. Its performance is yet quite ambiguous - above average (with the exception of the number of false non-zero coefficients) in case of unit amplitudes, but rather poor with the Gaussian coefficients. When used for reconstructing large vectors, it becomes very slow (see table **3.2**, **3.4**). Still, in the case of non-zero coefficients having unit amplitudes, this algorithm outperforms all the greedy or semi-greedy competition (fig. **3.12**, **3.13**).

A* Orthogonal Matching Pursuit The most recent (and most complex) A*OMP algorithm presents by far the best accuracy when the non-zero elements are normally distributed and stays among the best performing in the case of unit amplitudes. This will make it an excellent choice at each time where the focus is mostly on the reconstruction accuracy. However, due to its path-finding nature[7], the algorithm becomes extremely slow for higher degrees of sparsity (the evolution being almost exponential between $k = 10$ and $k = 20$ and with the Gaussian amplitudes, see fig. **3.9**). For the purpose of reconstructing vectors of higher sparsity degrees, a parameters' change can help with accelerating the algorithm, but 1) this acceleration is still very limited and doesn't counter the rapid evolution of time consumption with the growing sparsity degree (see table **3.2**) and 2) this leads necessarily to a loss of precision. Finally, the accelerated A*OMP does even have an exact reconstruction rate inferior to that of the 'standard' OMP (fig. **3.10**), but outperforms the latter in terms of mean square reconstruction error (fig. **3.11**). In the longer vector cases, it also seems that the accelerated A*OMP algorithm is more heavily influenced by the non-zero element distribution as its performance is much weaker than that of both RL2 and OMP (fig. **3.12**).

4.2 Overall Comparison

Generally, we can say that the best-performing algorithm among those studied above is the most recent and most complicated one, A*OMP. However, it does have a major drawback - the computing time growing exponentially with the sparsity degree. Thus, in some cases, the use of other algorithms (such as the 'basic' OMP for longer vectors or RL2 for shorter ones with higher sparsity degrees) could be preferred instead. Moreover, attempts to accelerate the A*OMP algorithm lead to a loss of performance, which can make the algorithm lose its superiority regarding exact reconstruction rate. On the other hand, even in the accelerated version, the algorithm produces remarkably low reconstruction errors.

The RL1, RL2 and BP algorithms merit from the greatest CPU-consumption stability regarding the sparsity degree. This can possibly act in favour of RL2 in domains where the application demands an upper-bound on the computation time, without the possibility to predict the sparsity of the signals treated. On the other hand, those algorithms are not suitable for reconstructing greater-dimension problems, where the difference in speed between them and the greedy (or semi-greedy) algorithms becomes way too important.

The nature of the non-zero element distribution does have an influence upon the algorithms' performance. Generally, the normal distribution of non-zero algorithms suits well the (semi-)greedy OMP and A*OMP algorithms, while the Bernoulli distribution with unit amplitudes can be a problem for them. For the RL2 algorithm, the situation is exactly the opposite. This can imply that (semi-)greedy algorithms are better adapted for reconstructing 'natural-looking' vectors, while convex relaxation algorithms are promising for the reconstruction of 'logical-like' vectors.

Finally, we can say that for lower problem dimensions, the A*OMP algorithm stays beyond any competition in case of Gaussian (or 'natural-looking') amplitudes, but RL2 could be the best choice when reconstructing 'logical-like' vectors. Still, the OMP algorithm may, in many cases, provide the best overall performance by being not much weaker than A*OMP but much, much faster. Yet finding the minimum of the imaginary '*performance-by-speed*' product depends on the nature of an individual problem or application.

Acknowledgements

This work was supported by Ministry of Education, Youth and Sports of the Czech Republic through the project 1M0572 and by Grant Agency of the Czech Republic through the project 102/09/1278.

Bibliography

[1] Sparselab - seeking sparse solutions to linear systems of equations. *http://sparselab.stanford.edu*, last consulted on 08/25/2011.

[2] E. Candès and T. Tao. Near optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. Inf. Theory, vol. 4, no. 2, pp. 317-329*, April 2006.

[3] E. Candès and M. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine, vol. 21*, 2008.

[4] D.L. Donoho and B.F. Logan. Signal recovery and the large sieve. *SIAM J. Appl. Math., vol. 2, pp. 577-591*, 1992.

[5] D.L. Donoho and P.B.Stark. Uncertainty principles and signal recovery. *SIAM J. Appl. Math., vol. 49, pp. 906-931*, 1989.

[6] Irina Gorodnitsky and Bhaskar D. Rao. Sparse signal reconstruction from limited data using focuss: A re-weighted minimum norm algorithm. *IEEE Transactions on Signal Processing, vol. 45, no. 3, pp. 600-616*, 1997.

[7] N. B. Karahanoglu and H. Erdogan. A* orthogonal matching pursuit: Best-first search for compressed sensing signal recovery. *Submitted to Digital Signal Processing, available for download from http://arxiv.org/abs/1009.0396*, 2010.

[8] P. Rajmic and J. Špiřík. Řídké reprezentace signálů: Úvod do problematiky. *Elektrorevue*, 2011.

[9] E. Candès J. Romberg and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inf. Theory, vol. 52, no. 2, pp. 489-509*, 2006.

[10] E. Candès J. Romberg and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Commun. Pure Appl. Math., vol. 59, no. 8, pp. 1207-1233*, 2006.

[11] E. Candès M. Wakin and S. Boyd. Enhancing sparsity by reweighted l_1 minimization. *J. Fourier Anal. App., vol. 14, pp. 877-905*, 2008.

[12] D. Wipf and S. Nagarajan. Iterative reweighted *l1* and *l2* methods for finding sparse solutions. *IEEE Journal of Selected Topics in Signal Processing, vol. 4, n. 2, pp. 317-329*, April 2010.

[13] J. Špiřík. Performance analysis of matching pursuit algorithm modifications. *Proceedings of the 13th International Conference on Research in Telecommunication Technologies*, 2011.

Jan Hoskovec, INSA de Lyon, département Génie Électrique, 20 avenue Albert Einstein, 69621 Villeurbanne, France
e-mail: jan.hoskovec@insa-lyon.fr

Petr Tichavský, Institute of Information Theory and Automation - Academy of Sciences of the Czech Republic, Pod Vodárenskou věží 4, 182 08 Praha 8, Czech Republic
e-mail: tichavsk@utia.cas.cz