algorithms

MDPI

Article

Feasibility Pump Algorithm for Sparse Representation under Gaussian Noise

Florin Ilarion Miertoiu and Bogdan Dumitrescu *

Department of Automatic Control and Computers, University Politehnica of Bucharest, 313 Spl. Independenţei, 060042 Bucharest, Romania; miertoiu.florin21@gmail.com
* Correspondence: bogdan.dumitrescu@acse.pub.ro

Received: 12 February 2020; Accepted: 6 April 2020; Published: 9 April 2020

Abstract: In this paper, the Feasibility Pump is adapted for the problem of sparse representations of signals affected by Gaussian noise. This adaptation is tested and then compared to Orthogonal Matching Pursuit (OMP) and the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA). The feasibility pump recovers the true support much better than the other two algorithms and, as the SNR decreases and the support size increases, it has a smaller recovery and representation error when compared with its competitors. It is observed that, in order for the algorithm to be efficient, a regularization parameter and a weight term for the error are needed.

Keywords: sparse representations; mixed integer programming; feasibility pump; Gaussian noise; regularization; weight term

1. Problem Formulation

The sparse representation of a signal $y \in \mathbb{R}^m$ is the solution $x \in \mathbb{R}^n$ with the smallest number of nonzero elements to the under-determined system $y = Dx$, where $D \in \mathbb{R}^{m \times n}$, $m < n$, is a given matrix named dictionary. Since in most cases noise is involved, the data misfit measure is minimized. Also, rather than attempting to find the sparsest solution, it is easier to bound the number of nonzero elements by a given threshold K and so the sparse representation can be found by solving the optimization problem

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & \|y - Dx\|_2 \\ \text{subject to} \quad & \|x\|_0 \leq K. \end{aligned} \tag{1}$$

Since the 2-norm is involved, the assumption is that the noise is Gaussian. This is still an NP-hard problem.

A common way to treat (1) is to replace the l_0 norm with the l_1 norm, thus relaxing the problem to a convex one. Transferring also the constraint into the objective, the result is a lasso style problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \|y - Dx\|_2^2 + \lambda \|x\|_1. \tag{2}$$

A less used approach takes into account the fact that the number of non-zero coefficients in (1) is bounded using the l_0 norm, hence Mixed Integer Programming (MIP) algorithms can be considered to solve the problem as it is posed, using either an integer variable for the number of coefficients or the binary decision whether a coefficient is used for the representation or not.

The Feasibility Pump (FP), proposed in References [1,2], is an MIP algorithm that alternates between solving the problem with relaxed integer constraints and the one satisfying the integer requirements. This is done until a point is reached that satisfies all the constraints, even though it might be not optimal, or for a prescribed number of iterations. The Feasibility Pump begins at an initial solution and then proceeds through several iterations to minimize the difference between the solutions of the alternating problems. Several modifications and improvements [3–10] have been proposed for this algorithm. See the bibliography in Reference [11] for a more extensive image. The case in which the l_1 norm is used for the representation error was analyzed and a modification of the Feasibility Pump algorithm for this problem was presented in Reference [12].

We propose to combine the MIP approach with the lasso problem (2). The binary variable $b \in \{0,1\}^n$ is introduced to perform the role of an indicator that shows which atom of the dictionary D is used for the representation of y. We then combine (1) with (2) to obtain the problem

$$\begin{aligned} \underset{x\in\mathbb{R}^n, b\in\{0,1\}^n}{\text{minimize}} \quad & \|y - Dx\|_2^2 + \lambda\|x\|_1 \\ \text{subject to} \quad & 1_n^T b \le K \\ & -Mb \le x \le Mb, \end{aligned} \tag{3}$$

where 1_n is a vector of length n whose elements are all equal to 1. The l_1 regularization enforces the sparsity condition on the representation x, as the l_1 norm favors sparsity, and helps the convergence of the Feasibility Pump algorithm, which will be used as an (approximate) MIP solver. As in Reference [13], a pioneer of MIP techniques for sparse representations, the big-M trick is employed in (3), where M is a preset parameter chosen as $M = 1.1\|D^T y\|_\infty / \|D\|_2^2$. This is used to bound the size of the representation coefficients. Note that if $b = 0$, then $x = 0$; if $b = 1$, then x is practically free, since the constraint $|x| \le M$ is of no consequence due to the large value of M.

Finally, to implement problem (3), an auxiliary variable $w \in \mathbb{R}^n$ is introduced for the regularization term, resulting in

$$\begin{aligned} \underset{x\in\mathbb{R}^n, w\in\mathbb{R}^n, b\in\{0,1\}^n}{\text{minimize}} \quad & x^T D^T D x - 2y^T D x + \lambda 1_n^T w \\ \text{subject to} \quad & 1_n^T b \le K \\ & -w \le x \le w \\ & w \le Mb. \end{aligned} \tag{4}$$

In this paper, the focus is on the implementation of the Feasibility Pump of problem (4). Our solution is based on the Objective FP, originated in Reference [3]. Section 2 presents our Feasibility Pump algorithm and the implementation details. Section 3 is dedicated to experimental results showing the behavior of our algorithm and comparisons with the Orthogonal Matching Pursuit (OMP) and Fast Iterative Shrinkage-Thresholding (FISTA) algorithms. We show that our regularized Feasibility Pump algorithm is consistently better when compared to the other algorithms. Section 4 presents the conclusions and future ideas for research.

2. Algorithm

Algorithm 1 solves the reformulation (4) of problem (1). It is similar in structure to that in Reference [12], although the underlying optimization problem and some important details are different. We will point out the differences when they are presented.

The Feasibility Pump algorithm starts with the computation of the solution for the relaxed version of the initial problem (4), where b takes values in the $[0,1]^n$ interval, instead of being a binary variable. The problem becomes convex; more precisely, it belongs to quadratic programming; the MATLAB function `quadprog` is used to solve it, based on an interior-point algorithm; other solvers or packages could be used as well.

The real solution b is rounded to the vector $\widetilde{b} \in \{0,1\}^n$. The largest K elements of b are rounded upwards to 1, while the others are rounded downwards to 0. Indirectly, a sparse solution is obtained with exactly K elements.

Algorithm 1: Modified Feasibility Pump.

```
Data: Dictionary $D \in \mathbb{R}^{m \times n}$, signal to represent $y \in \mathbb{R}^m$, number of non-zero coefficients used
      for the representation $K \in \mathbb{Z}$, maximum number of iterations $Iter$, weight parameters $\alpha$,
      $\lambda$, $\gamma$
Result: a feasible solution $x \in \mathbb{R}^n$
Solve relaxed (4) with $b \in [0,1]^n$. The vectors $x$ and $b$ are obtained.
Use rounding procedure to obtain vector $\widetilde{b}$.
while number of iterations $\leq Iter$ do
    Solve problem (5). The vectors $x$ and $b$ are obtained.
    if $b$ is integer then
        return $x$;
    end
    Use rounding procedure to obtain vector $\widetilde{b}$.
    if cycle is detected then
        Use perturbation on $\widetilde{b}$.
    end
    Update the value of $\alpha$ using (6).
end
Use Least Squares Method to optimize the error for the found support.
```

In each iteration of the Feasibility Pump, the vector b and the tentative solution x are updated by solving

$$\begin{array}{lll} \underset{x \in \mathbb{R}^n, w \in \mathbb{R}^n, b \in [0,1]^n}{\text{minimize}} & (1-\alpha)\triangle(b,\widetilde{b}) + \alpha \left[\frac{K}{err_{init}^2}(x^T D^T D x - 2y^T D x) + \lambda 1_n^T w \right] & \\ \text{subject to} & 1_n^T b \leq K & (5) \\ & -w \leq x \leq w & \\ & w \leq Mb, & \end{array}$$

where $\triangle(b,\widetilde{b}) = \|b - \widetilde{b}\|_1$ and err_{init} is the representation error at the initial Feasibility Pump step (4). This quadratic programming problem is also solved with `quadprog` in MATLAB. The iteration step has an objective that combines the representation error $\|y - Dx\|_2$ with a term that enforces the new solution b to be near from the current integer vector $\widetilde{b}$, with the aim of making the solution b nearer from a binary vector. This kind of modification, named Objective FP, is proposed in Reference [3] and is essential in our case for finding good values of the objective; feasibility is easy to obtain, since rounding always gives a K-sparse solution, and so the original FP [1] would always converge in one iteration, possibly to a poor solution.

After each iteration, the parameter α is reduced and the integer condition will weigh more than the error objective. The reduction of α is done by multiplication with a value $\gamma \in (0,1)$:

$$\alpha \leftarrow \gamma\alpha. \quad (6)$$

A large γ will give the smallest error, but the execution time is longer, while a smaller γ offers faster results, but with a larger error.

During our tests, it was observed that the addition of the factor $\frac{K}{err_{init}^2}$ in (5) is necessary to increase the influence of the error in the optimization process because the error is much smaller than the $\triangle(b,\widetilde{b})$ and $\|x\|_1$ terms and it needs an additional weight parameter. The division with the square of the error removes the difference of the orders in magnitude between the $\triangle(b,\widetilde{b})$ term, regularization term and the error term during the first iteration steps. With each iteration the importance of the error terms decays. Without the weight, the importance of the error term decays too fast and has little effect in the optimization process; the weight is used so that the error term plays an important role for more iterations of the algorithm. The factor K is added such that the error term is as important as the $\triangle(b, \widetilde{b})$ term for more time, as α decreases the importance of the error term. It also helps the choice of λ. As $0 \le \triangle(b, \widetilde{b}) \le 2K$, with $2K$ being nearly impossible to attain, the multiplication with K puts both the error and the feasibility term on similar weights; without the K value, the square error would only normalize the error term, while the magnitude of the $\triangle(b, \widetilde{b})$ term would act as a weight that forces the algorithm to focus more on getting a sparse solution, than on reducing the representation error. We note that directly using the algorithm from Reference [12], without the factor $\frac{K}{err^2_{init}}$, leads to poor results. Balancing the terms of the objective, like we did in (5), is also done in Reference [9], but with different means.

The regularization term $\|x\|_1$ has a double role. It helps enforce the desired sparsity and also helps the algorithm to converge when the dictionary is ill conditioned. The lack of the regularization term increases the running time of the algorithm, sometimes not even reaching convergence. The λ tuning parameter is very important as it influences heavily the performance of the model, as shown in Reference [14]. Note that at the end of Algorithm 1, when the support is settled, the regularization term is removed and the (optimal) least squares solution corresponding to that sparse support is computed; such a feature was not present in Reference [12].

The perturbation strategy used when cycles occur in the Feasibility Pump iterative process is similar to the one used in Reference [12]; it belongs to the category of strong perturbations, as all elements of $\widetilde{b}$ are perturbed. Inspired by Reference [6], we consider that a cycle appears when one of the following conditions is met (we use index t for the current iteration and $t-1$ for the previous one): (i) $\widetilde{b}$ is the same as in the previous iteration and $|\alpha_t - \alpha_{t-1}| < 10^{-3}$ (note that even if $\widetilde{b}$ is the same, different values of α lead to different solutions in (5)), (ii) $\|b_t - b_{t-1}\| < 10^{-4}$, (iii) $\|b_t - b_{t-1}\| \ge 0.9\|b_{t-1} - b_{t-2}\|$; the last two conditions take into account both the absolute value of $\|b_t - b_{t-1}\|$ and its relative change with respect to the previous iteration.

The algorithm described above is named Sparse Quadratic Feasibility Pump (SQFP). Unlike the Branch and Bound algorithms from Reference [13], SQFP may not attain the optimal solution, but it has a running time that is comparable with those of popular algorithms for sparse representation. In contrast, the MIP algorithms from Reference [13] may be extremely slow for some problems.

3. Results

In order to obtain numerical results, the testing scheme from Reference [12] is used, with the significant distinction that noise is now Gaussian.

In a first test, for which we report extensive results, we use randomly generated dictionaries with condition numbers of 100, 1000, 10,000 and 100,000. For each condition number, 160 dictionaries of size 80×200 are generated. The test signals are obtained with $y = Dx_{\text{true}} + u$, where the solutions x_{true} have $K \in \{5, 7, 9, 11\}$ nonzero coefficients generated randomly following a Gaussian distribution, in random positions; the noise u is Gaussian and its variance is chosen such that the signal to noise ratios have values $10, 20, 30, \infty$.

For the computation of the representation error, the relative error

$$e = \frac{\|Dx - y\|_2}{\|y\|_2} \tag{7}$$

is used (where now x is the computed solution), in accordance with the formulation of the initial problem (1).

We have implemented SQFP as shown in Algorithm 1. The initial weight α is set to 1 and is multiplied by an update factor $\gamma = 0.9$ at each iteration; these values seem to provide a good compromise between convergence speed and representation error. The number of iterations *Iter* is set to 1000. We run SQFP with 50 equally spaced values of the regularization parameter λ from 0 to 1. The value for which the mean representation error is the smallest is considered the best choice for λ.

Several types of algorithms have been proposed [15] to find the solution of (1) or its relaxed version (2). The OMP (Orthogonal Matchmaking Pursuit) [16] and the FISTA (Fast Iterative Shrinkage-Thresholding Algorithm) [17] algorithms where chosen for comparison as they are some of the most commonly used.

Both FISTA and SQFP use the regularization parameter λ. FISTA is tested with 500 equally spaced values between 0 to 0.05.

The algorithms are implemented in MATLAB and tested on a computer with a 6-core 3.4 GHz processor and 32 GB of RAM.

The variation of the error produced by SQFP depending on λ is represented in Figure 1 for $K = 11$ and condition number 1000. It can be seen that for the lower SNR values, the error has a relatively well defined minimum around $\lambda = 0.5$ for $SNR = 10$ and $\lambda = 0.18$ for $SNR = 20$. From the minimum point the error increases as λ increases. For the larger SNR values, the error is very small in the beginning, with small variations as λ increases from zero; after a certain value the error increases a lot. In these cases regularization offers very small benefits.

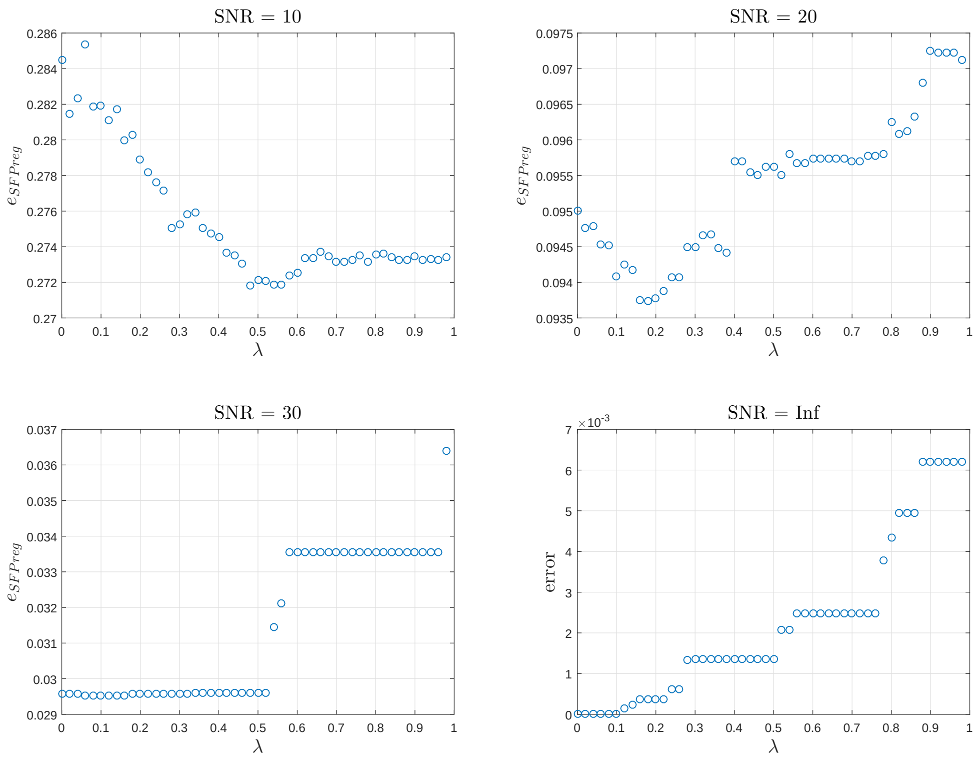

Figure 1. Mean representation error depending on λ for $K = 11$, for matrix conditioning of 1000.

An example of relative errors (7) is given in Figure 2, where SQFP shows the ability to consistently produce solutions whose errors are at about the SNR level. In the same conditions, the recovery error $\|x - x_{\text{true}}\| / \|x_{\text{true}}\|$ has the values shown in Figure 3; although with more variability, the errors decrease nicely with the SNR for all values of K.

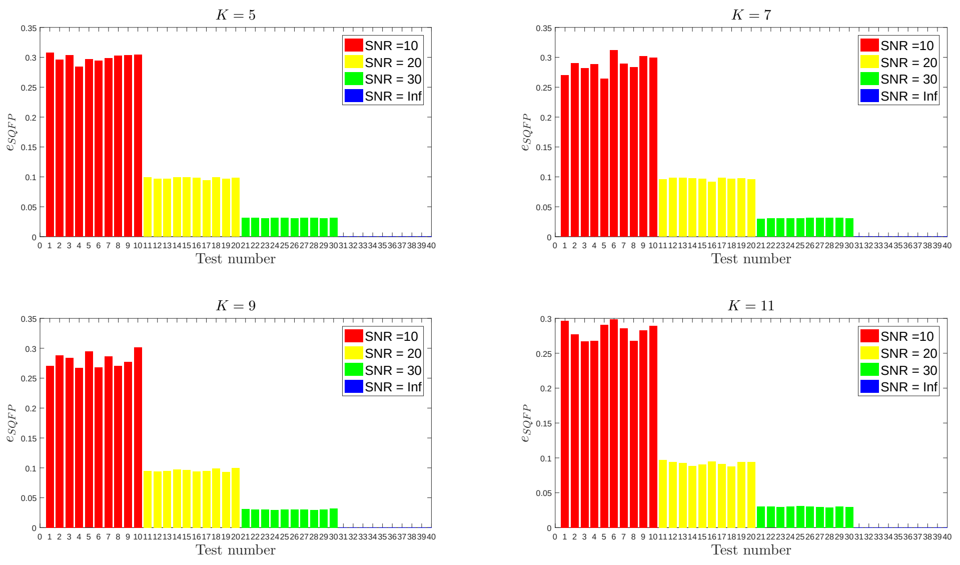

Figure 2. Relative errors for Sparse Quadratic Feasibility Pump (SQFP) for matrix conditioning of 10,000.

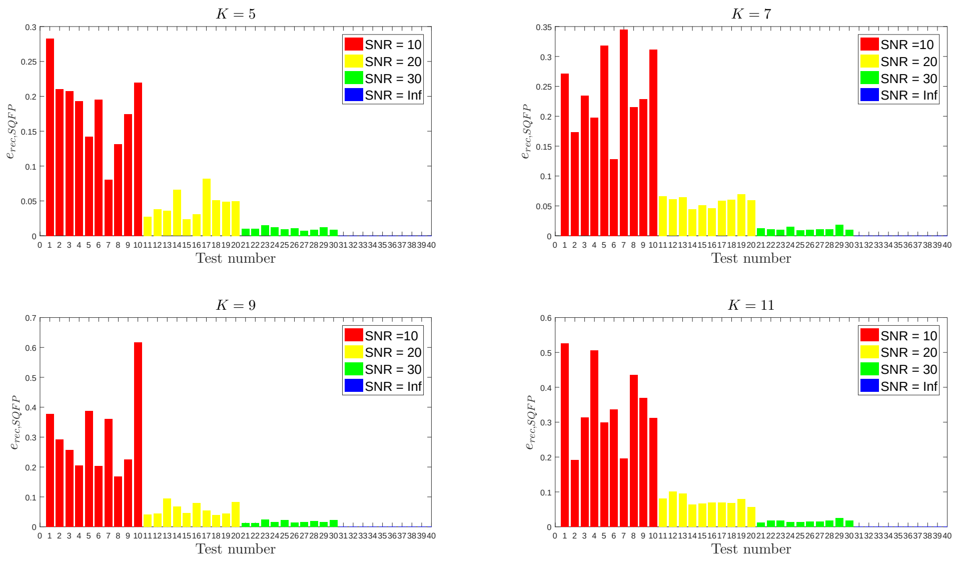

Figure 3. Recovery errors for SQFP for matrix conditioning of 10,000.

The mean errors obtained by running the tests are displayed in Figures 4–7. The first (red) bar in each cell corresponds to the relative error of FISTA, the second (green) is for SQFP and the last (blue) is for OMP.

It can be seen that as the sparsity level K increases, the SQFP algorithm has a much smaller representation error than FISTA and, only for some condition numbers, than OMP. For $K = 5$ and $K = 7$, the difference between the algorithms is very small. For larger K, SQFP is clearly better.

To evaluate the complexity of the algorithms, we note that the mean number of iterations is 43.33 for SQFP and 726.56 for FISTA. The FISTA iterations are much less complex and time consuming. The average running time for SQFP is 1.09 s, for FISTA is 0.43 s and for OMP is 0.0025 s.

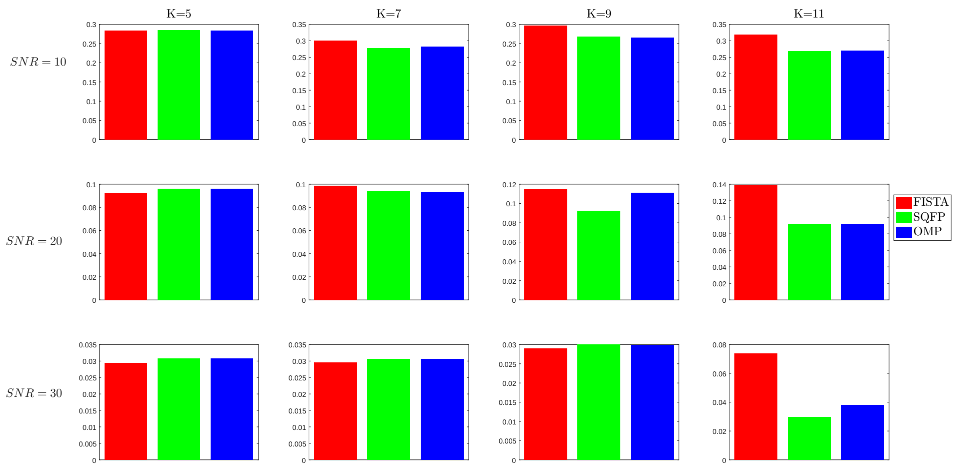

Figure 4. Mean errors for all algorithms, dictionary size 80 $\times$ 200, condition number 100.

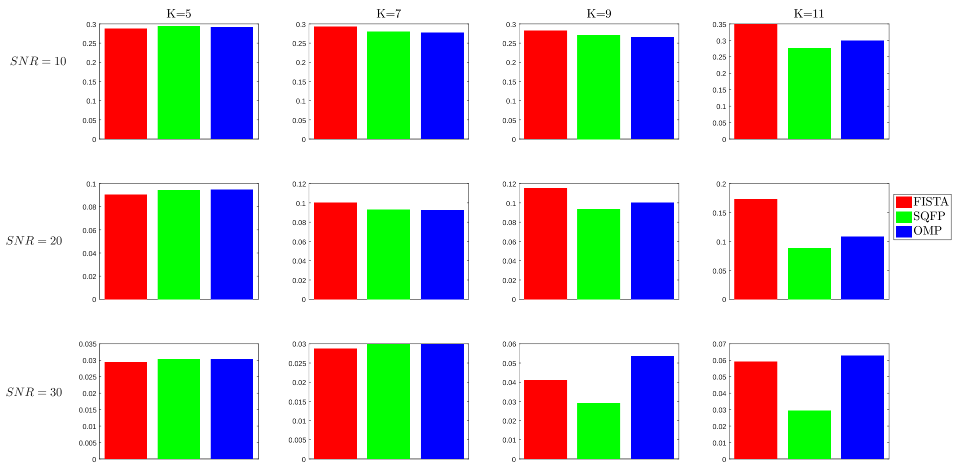

Figure 5. Mean errors for all algorithms, dictionary size 80 $\times$ 200, condition number 1000.

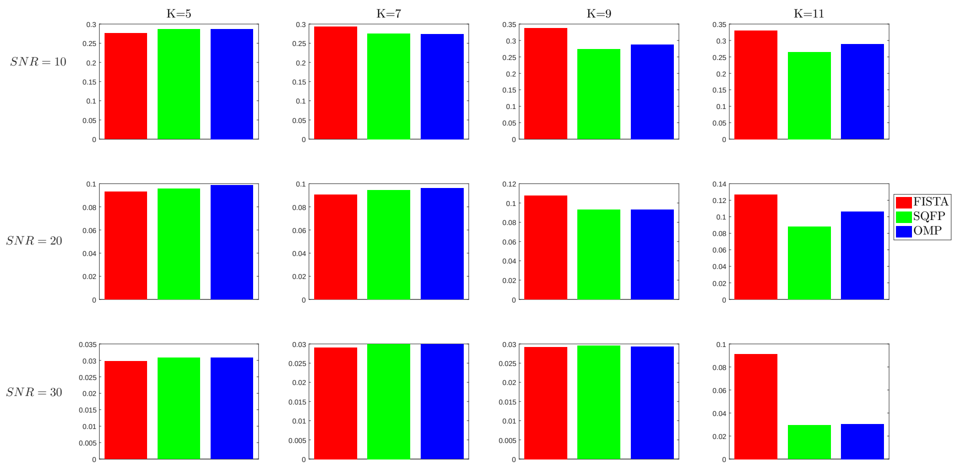

Figure 6. Mean errors for all algorithms, dictionary size 80 $\times$ 200, condition number 10,000.

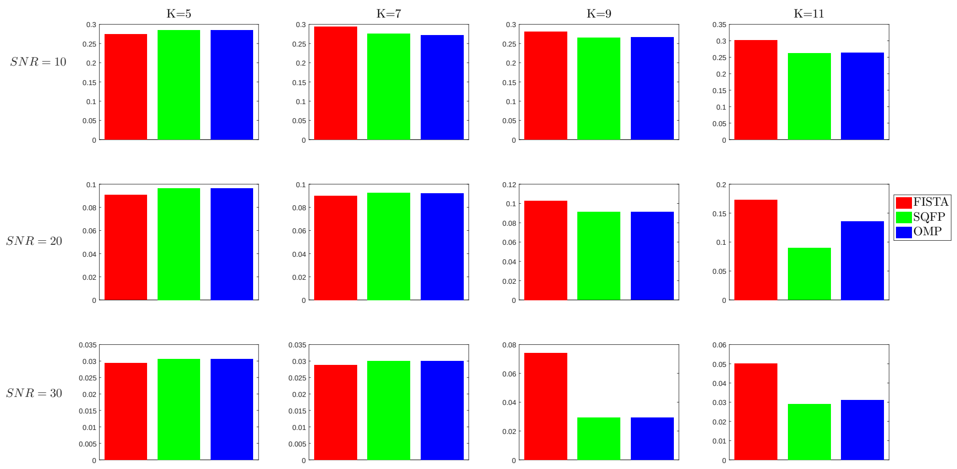

Figure 7. Mean errors for all algorithms, dictionary size 80 × 200, condition number 100,000.

To evaluate the quality of support recovery, we note that, out of the 640 tests, false negatives appear in 254 (39%) cases for OMP, in 225 (35%) cases for SQFP and in 281 (44%) cases for FISTA. While SQFP needs the longest running time, it recovers the support more precisely than the two other methods. FISTA shows a false positive in 609 (95%) cases. False negatives indicate when an atom is missing from the support of the true solution. False positives appear when the support of the computed solution contains atoms outside the true support.

The second experiment is made with the same parameters as before, but now the dictionary size is 80 × 400, the condition number is 1000 and $K \in \{9, 11, 13, 15\}$. The overcompleteness factor is larger than in the first experiment, hence the properties of the dictionary are less favorable to sparse recovery. Also, higher sparsity levels are considered, hence the problem becomes more difficult. The mean errors are shown in Figure 8. Now the results of SQFP are even clearly better than those of OMP and FISTA, confirming its ability to work well in more difficult conditions.

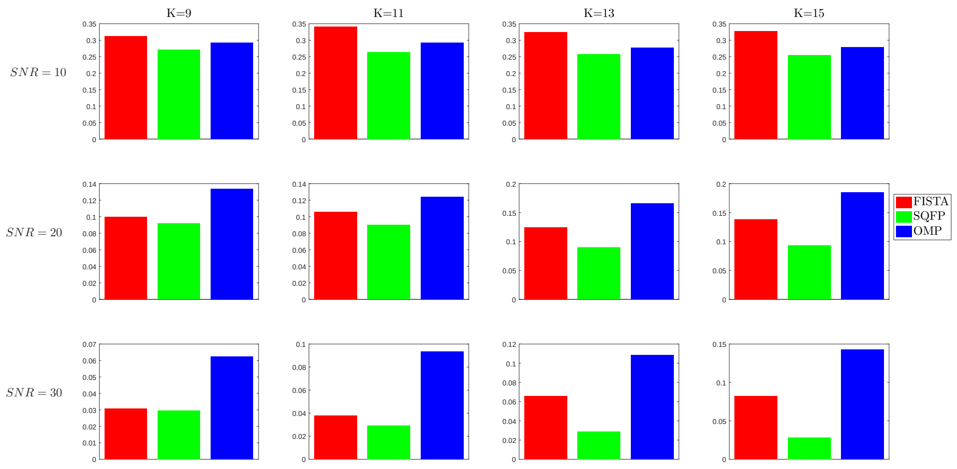

Figure 8. Mean errors for all algorithms, dictionary size 80 × 400, condition number 1000.

4. Conclusions

In this paper we have presented a version of the Feasibility Pump algorithm adapted for the sparse representation problem with l_2 norm of the error. Our tests show that the Feasibility Pump gives a better solution when compared with FISTA and OMP, especially at higher sparsity levels. The addition of the weight term for the error proves to be a very important factor for the performance of the algorithm, as it forces the representation error to be smaller. The regularization and the big-M trick limit the magnitude of the values of the coefficients and thus allow the use of this algorithm for ill-conditioned problems. Future lines of research will focus on improving the randomization step, using different regularization terms, treating other norms of the error, implementing recent modifications of FP, like those from [5,9], and adapting the algorithm for other sparse problems reformulations and dictionary learning.

Author Contributions: Conceptualization, B.D. and F.I.M.; Methodology, B.D. and F.I.M.; Software, F.I.M.; Validation, F.I.M. and B.D.; Formal analysis, B.D. and F.I.M.; Investigation, F.I.M.; Data curation, F.I.M.; Writing—original draft preparation, F.I.M.; Writing—review and editing, B.D.; Visualization, F.I.M.; Supervision, B.D.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fischetti, M.; Glover, F.; Lodi, A. The feasibility pump. *Math. Program.* **2005**, *104*, 91–104. [CrossRef]
2. Bertacco, L.; Fischetti, M.; Lodi, A. A feasibility pump heuristic for general mixed-integer problems. *Discret. Optim.* **2007**, *4*, 63–76. [CrossRef]
3. Achterberg, T.; Berthold, T. Improving the feasibility pump. *Discret. Optim.* **2007**, *4*, 77–86. [CrossRef]
4. Huang, K.; Mehrotra, S. An empirical evaluation of walk-and-round heuristics for mixed integer linear programs. *Comput. Optim. Appl.* **2013**, *55*, 545–570. [CrossRef]
5. De Santis, M.; Lucidi, S.; Rinaldi, F. A new class of functions for measuring solution integrality in the Feasibility Pump approach. *SIAM J. Optim.* **2013**, *23*, 1575–1606. [CrossRef]
6. De Santis, M.; Lucidi, S.; Rinaldi, F. Feasibility pump-like heuristics for mixed integer problems. *Discret. Appl. Math.* **2014**, *165*, 152–167. [CrossRef]
7. Boland, N.; Eberhard, A.; Engineer, F.; Fischetti, M.; Savelsbergh, M.; Tsoukalas, A. Boosting the feasibility pump. *Math. Program. Comput.* **2014**, *6*, 255–279. [CrossRef]
8. Dey, S.; Iroume, A.; Molinaro, M.; Salvagnin, D. Exploiting sparsity of MILPs by improving the randomization step in feasibility pump. *SIAM J. Optim.* **2016**, *28*, 355–378. [CrossRef]
9. Geißler, B.; Morsi, A.; Schewe, L.; Schmidt, M. Penalty alternating direction methods for mixed-integer optimization: A new view on feasibility pumps. *SIAM J. Optim.* **2017**, *27*, 1611–1636. [CrossRef]
10. Dey, S.; Iroume, A.; Molinaro, M.; Salvagnin, D. Improving the randomization step in feasibility pump. *SIAM J. Optim.* **2018**, *28*, 355–378. [CrossRef]
11. Berthold, T.; Lodi, A.; Salvagnin, D. Ten years of feasibility pump, and counting. *EURO J. Comput. Optim.* **2019**, *7*, 1–14. [CrossRef]
12. Miertoiu, F.I.; Dumitrescu, B. Feasibility Pump Algorithm for Sparse Representation under Laplacian Noise. *Math. Probl. Eng.* **2019**, *2019*, 5615243. [CrossRef]
13. Bourguignon, S.; Ninin, J.; Carfantan, H.; Mongeau, M. Exact sparse approximation problems via mixed-integer programming: Formulations and computational performance. *IEEE Trans. Signal Process.* **2016**, *64*, 1405–1419. [CrossRef]
14. Kirkland, L.A.; Kanfer, F.; Millard, S. LASSO tuning parameter selection. *Annu. Proc. S. Afr. Stat. Assoc. Conf.* **2015**, *57*, 49–56.
15. Zhang, Z.; Xu, Y.; Yang, J.; Li, X.; Zhang, D. A survey of sparse representation: Algorithms and applications. *IEEE Access* **2015**, *3*, 490–530. [CrossRef]
16. Tropp, J.A. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. Inf. Theory* **2004**, *50*, 2231–2242. [CrossRef]
17. Beck, A.; Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.* **2009**, *2*, 183–202. [CrossRef]