MATHEMATICS OF COMPUTATION
Volume 00, Number 0, Pages 000–000
S 0025-5718(XX)0000-0

FAST SPARSE RECONSTRUCTION: GREEDY INVERSE SCALE SPACE FLOWS

MICHAEL MOELLER AND XIAOQUN ZHANG

Abstract. In this paper we analyze the connection between the recently proposed adaptive inverse scale space methods for basis pursuit [6] and the well known orthogonal matching pursuit method for the recovery of sparse solutions [28, 22, 31] to underdetermined linear systems. Furthermore, we propose a new greedy sparse recovery method, which approximates ℓ^1 minimization more closely. A variant of our new approach can increase the support of the current iterate by many indices at once, resulting in an extremely efficient algorithm. Our new method has the advantage that there is a simple criterion to determine a-posteriori if an ℓ^1 minimizer was found. Numerical comparisons with orthogonal matching pursuit, weak orthogonal matching pursuit [30, 17], hard thresholding pursuit [16] and compressive sampling matching pursuit [24] underline that our methods indeed inherits some advantageous properties from the inverse scale space flow.

1. Introduction

Compressed sensing and techniques exploiting sparsity in data analysis, image processing, and inverse problems recently gained enormous interest. Representing unknowns for (underdetermined) systems of linear equations in appropriate bases or dictionaries can be reformulated as finding the sparsest solution of a linear system, i.e., solving

$$\min_u |u|_0 \qquad \text{such that } Au = f, \tag{1.1}$$

where $|\cdot|$ is the so called ℓ^0 norm, which denotes the number of nonzero elements and is not a norm in the mathematical sense. The ℓ^0 norm is the natural measure for sparsity. The quantity $f \in \mathbb{R}^m$ is the given (or measured) data and $A \in \mathbb{R}^{m\times n}$ is the sensing matrix, usually with m much smaller than n. Unfortunately, minimizing the ℓ^0 norm is a highly nonconvex problem. Solving (1.1) exactly has been shown to be NP-hard [29], such that it quickly becomes infeasible in dimensions interesting for practical applications. There are two different general strategies for providing an approximation of the sparsest solution.

2010 *Mathematics Subject Classification.* Primary 65K10, 90C59, 90C26; Secondary 92C55.

This work was supported by the DFG grant "Sparsity constrained inversion with Tomographic Applications". X.Zhang was additionally supported by the National Science Foundation of China (grant numbers NSFC91330102, NSFC11101277 and NSFC11161130004) and by the Shanghai Pujiang Talent program (grant number 11PJ1405900).

The first is to use the convex relaxation by minimizing the ℓ^1- instead of the ℓ^0-norm,

$$\min_u \|u\|_1 \qquad \text{such that } Au = f. \tag{1.2}$$

Various important results have been obtained on the equivalence of ℓ^0 and ℓ^1-minimization under different conditions, we refer e.g. to [10, 9, 13, 12].

The second approach is to develop greedy methods to approximate the ℓ^0 minimizing solution. One of the most popular greedy methods is the orthogonal matching pursuit (OMP) as proposed in [28, 22, 31]. While OMP (like all greedy algorithms) is based on heuristic assumptions, one can show that under certain conditions on the matrix A, OMP recovers the ℓ^0 minimizing solution exactly [30].

Comparing the two approaches, the ℓ^1 minimization allows more analysis and more general theoretical exact recovery results. Furthermore, ℓ^1 minimization often yields better recovery results in numerical experiments, particularly for ill-posed matrices A, which we will confirm in our numerical results section. However, solving the minimization problem (1.2) requires the solution of a non-differentiable constrained optimization problem, such that its numerical solution remains a challenge. Although many very efficient numerical methods for ℓ^1 minimization have been proposed (cf. [26, 18, 2, 7, 35, 8, 36], or the references on the websites http://dsp.rice.edu/cs or http://nuit-blanche.blogspot.com), greedy approaches often significantly outperform convex optimization methods in terms of speed such that in applications that deal with extremely high dimensional problems or require fast processing, greedy methods are often preferable (cf. [21, 14, 19]).

The goal of this paper is to close some parts of the gap between the greedy and convex approach to sparse reconstruction. Our motivation will be to start from the recently proposed adaptive inverse scale space method (aISS) [6] and see that a small tweak in the method leads to a greedy method, which on the one hand is similar to OMP but on the other hand approximates the ℓ^1 minimization idea much more closely. We will analyze the proposed method and prove that it has similar exact recovery guarantees as OMP while it additionally provides the opportunity to check a-posteriori if the solution we obtain is the ℓ^1 minimizing solution. The latter only depends on the coincidence of certain signs and, of course, does not require the knowledge of the ℓ^1 minimizing solution itself.

Furthermore, we will look into methods that increase their support more quickly: Note that OMP always adds a single component to the support at each iteration, and, while the aISS method can theoretically change its support arbitrarily, it has been observed in [6] that the support often only changes by a single or at least only very few indices. For very high dimensional problems (e.g. with a one million dimensional u), even a very sparse true solution will have several thousand nonzero entries. For these kinds of problems the OMP strategy of adding only a single index per iteration does not seem to be ideal since the algorithm will need at least several thousand iterations - even in the best case where the method only acts on the support of the true solution. We will therefore consider including multiple indices in the support at each iteration similar to the idea of weak orthogonal matching pursuit (WOMP) [30, 17].

We will compare our proposed method to the aISS method, OMP, WOMP, (due to their similarity) as well as to hard thresholding pursuit (HTP) [16] and

compressive sampling matching pursuit (CoSaMP) [24] as a recently proposed state of the art method, which immediately have a support of desired size.

The rest of the paper is organized as follows. In the next Section we will give an (incomplete) summary of sparse reconstruction methods. While extremely many greedy reconstruction methods as well as ℓ^1 minimization techniques have been proposed, we will particularly focus on OMP, WOMP, HTP and CoSaMP among the greedy methods, and recall the aISS method as an ℓ^1 minimization technique. The reason that we choose aISS as the compared ℓ^1 minimization method is that in [33], detailed comparison of several ℓ^1 minimization techniques (including aISS) have been made for different compressive sensing settings and it has been observed that aISS is the fastest or among the fastest with highest accuracy. In Section 3 we will see how similar OMP and the aISS method are and propose a small change in the aISS method to obtain a new greedy method, which is related to OMP but approximates ℓ^1 minimization more closely. We will see that the ideas of WOMP can also be adapted to our new method to obtain quicker, more greedy results. Furthermore, we will prove that our method allows to see a posteriori, if the result is also the ℓ^1 minimizing solution. In Section 4 we will compare the results of all algorithms in a series of numerical experiments and see that our newly proposed algorithm indeed inherits many desirable properties from the aISS method, while being as fast as OMP or WOMP. Finally, in Section 5 we will draw conclusions and point out future areas of research.

2. State of the Art

2.1. **Greedy Methods for Sparse Recovery.** Many greedy methods for sparse recovery have been proposed, like for instance iterative hard thresholding (IHT) [4], compressive sampling matching pursuit (CoSaMP) [24], Subspace Pursuit (SP) [11], iterative thresholding with inversion (ITI) [20], hard thresholding pursuit (HTP) [16] and many others. Due to the huge number of greedy sparse recovery methods we will neither summarize nor compare our proposed method to all of them. Instead, we will focus on the most related methods, OMP and WOMP, and include the recently proposed HTP and CoSaMP methods in our comparison.

2.1.1. *Orthogonal Matching Pursuit.* OMP, as proposed in [28, 22, 31], is a classical and popular greedy recovery method. It iteratively adds components to the support of the approximation u^k whose correlation to the current residual is maximal.

To be more precise the pseudo code for OMP is given in algorithm 1 below. We can see that OMP has three main iterative steps:

(1) Add an index i to the current index set I_k, such that the correlation between the i column of A and the current residual r_k is maximal.
(2) Solve a linear least squares problem on the index set I_k to obtain the next approximation u_k. We used P_{I_k} to denote the projection of u onto the index set I_k.
(3) Update the residual.

Clearly, at the k-th iteration, u^k is k-sparse such that the algorithm will converge after at most m steps for $A \in \mathbb{R}^{m\times n}$. OMP is fast and easy to implement.

Algorithm 1 OMP

Parameters: $A,\ f,\ \text{threshold} > 0$
Initialization: $r_0 = f$, $I_0 = \emptyset$
while $\|r_k\| > \text{threshold}$ **do**

1. Compute $I_k = I_{k-1} \cup i$ with i such that $|(A^T r_k)_i| = \|A^T r_k\|_\infty$
2. Compute $u_k = \arg\min_u \left\{\|AP_{I_k} u - f\|^2\right\}$ and set $(u_k)_i = 0$ for $i \notin I_k$
3. Update $r_{k+1} = f - Au_k$

end while
return u_k

For theoretical guarantees, Tropp considered the following quantity in [30]:

Definition 2.1. We say that a u_{opt} with $Au_{\text{opt}} = f$ and A having normalized columns satisfies an exact recovery condition (ERC) of order α if

$$\max_{i \notin I} \|(AP_I)^\dagger a_i\|_1 < \alpha, \tag{2.1}$$

where I is the index set of the support of u_{opt}, P_I is the projection onto the index set I, a_i is the i-th column of A and the superscript $\dagger$ denotes the pseudo inverse of the matrix.

Theorem 2.2 (from [30])**.** *If a solution u with $Au = f$ satisfies an ERC of order $\alpha \leq 1$, then OMP recovers u exactly.*

While OMP shows nice recovery properties under the assumption of an ERC, it seems to yield rather weak recovery results under the popular restricted isometry property (RIP) [19], which is commonly used as a criterion for measuring up to which sparsity ℓ^1 minimization or greedy approaches can recover a u_{opt} exactly.

2.1.2. *Weak orthogonal matching pursuit.* WOMP is a faster and more greedy version of OMP [30, 17]. It converges much faster since it does not only add a single index to the support at each iteration, but all whose correlation to the currently residual is large enough. The WOMP method is given as algorithm 2 below.

Algorithm 2 WOMP

Parameters: $A,\ f,\ \rho \leq 1,\ \text{threshold} > 0$,
Initialization: $r_0 = f$, $I_0 = \emptyset$
while $\|r_k\| > \text{threshold}$ **do**

1. Compute $I_k = I_{k-1} \cup \{i \mid |(A^T r_k)_i| \geq \rho \|A^T r_k\|_\infty\}$
2. Compute $u_k = \arg\min_u \left\{\|AP_{I_k} u - f\|^2\right\}$ and set $(u_k)_i = 0$ for $i \notin I_k$
3. Update $r_{k+1} = f - Au_k$

end while
return u_k

We can see that the only difference to OMP lies in the first step, where all indices for which the absolute value of $A^T r^k$ is close enough to the maximal value,

are included in the support. The parameter ρ controls how much more greedy WOMP is in comparison to OMP. While WOMP comes with the advantage of faster convergence, one pays for the additional speed by a reduced exact recovery guarantee as shown in [30].

Theorem 2.3 (from [30]). *If a solution u with $Au = f$ satisfies an ERC of order ρ, then WOMP recovers u exactly.*

2.1.3. *Compressive Sampling Matching Pursuit.* CoSaMP has been proposed by Needell and Tropp in 2009 as a very efficient greedy recovery algorithm. Instead of iteratively increasing the support of the solution, the desired support size is an input parameter of the algorithm. In this sense CoSaMP can rather be seen as an approximation for solving

$$\min_u \|Au - f\|^2 \qquad \text{such that } |u|_0 \leq s,$$

for a given sparsity level s. Although this formulation is very similar to OMP if s is given, it seems that in general a stopping criterion on the residual $\|Au - f\|$ is more intuitive than the (unknown) sparsity level.

In general, CoSaMP consists of five iterative steps: In the first step the $2s$ indices for which the correlation between columns of the sensing matrix and the current residual is maximal are determined. In the second step, these components are merged with the previous support. As the third step CoSaMP determines the least squares solution with the increased support (of size $3s$). Next, the least squares solution is pruned such that only the s largest components in magnitude are kept, or, in other words, hard thresholding is performed. Finally, the residual is updated based on the pruned signal from the previous step. Pseudocode for the complete CoSaMP algorithm is given as algorithm 3 below, where we denote the hard thresholding operator, i.e. the operator setting all but the s largest components in magnitude of u to zero, by $H_s(u)$. In our numerical experiments, we used the CoSaMP code from [3] with an additional upper bound of 500 on the total number of iterations.

Algorithm 3 CoSaMP

Parameters: A, f, sparsity level s,
Initialization: $r_0 = f$
while Stopping Criterion **do**

1. Compute $\Omega_{k+1} = supp(H_{2s}(A^T r_k))$

2. Compute $I_{k+1} = \Omega_{k+1} \cup supp(u_k)$

3. Determine $\tilde{u}_{k+1} = \arg\min_u \left\{\|AP_{I_{k+1}} u - f\|^2\right\}$
and set $(\tilde{u}_{k+1})_i = 0$ for $i \notin I_{k+1}$

4. Prune $u_{k+1} = H_s(\tilde{u}_{k+1})$

5. Update $r_{k+1} = f - Au_{k+1}$

end while
return u_k

2.1.4. *Hard thresholding pursuit.* Recently, the HTP algorithm was proposed by Simon Foucart in [16]. HTP can be seen as a combination between IHT and CoSaMP: For a desired sparsity level s, HTP performs a gradient descent step on the objective function ($z^k = u^k + \mu A^T(f - Au^k)$), determines the index set I as the indices corresponding to the s entries of largest magnitude in z^k and obtains the next iterate u^{k+1} by solving a least squares problem on I. HTP is given as algorithm 4 below. Note that convergence of HTP can only be guaranteed for a small enough time step, more precisely, for $\nu\|A\|_{2\to 2}^2 < 1$, where $\|A\|_{2\to 2}$ denotes the induced 2-norm of A.

Algorithm 4 Hard thresholding pursuit

Parameters: A, f, sparsity level s, step size ν,
Initialization: $u^0 = 0$ $r^0 = f$, $z^0 = 0$
for $k = 0$ to $K - 1$ **do**

1. Gradient descend on z: $z^{k+1} = u^k + \nu A^T r^k$

2. Update $I^{k+1} = supp(H_s(z^{k+1}))$.

3. Compute $u^{k+1} = \arg\min_u \left\{\|AP_{I^{k+1}}u - f\|^2\right\}$ and set $(u^{k+1})_i = 0$ for $i \notin I^{k+1}$

4. Update $r_{k+1} = f - Au^{k+1}$

end for
return u^K

The idea behind HTP was generalized in [19] by proposing a new family of sparse greedy reconstruction algorithms (the OMPR family), one member of which is HTP. While theoretical guarantees based on the restricted isometry property (RIP) were optimal for changing just a single index in I^k (a particular algorithm of the OMPR algorithm family), numerical experiments did not show any degradation in recovery ability when using HTP [19], such that we will focus on a comparison to the HTP method in this paper.

Similar to CoSaMP, the major difference between HTP and OMP is that HTP fixes the size of the index set I by an input parameter s and immediately considers larger supports. While this make the solution of the least squares problems more expensive, one can hope to significantly reduce the number of iterations needed for convergence. The clear drawback of fixing the sparsity level s is that s is unknown in practical applications. The OMP approach of increasing the index set I until a desired accuracy $\|Au^k - f\|^2$ is reached is much more intuitive and it might take several runs of HTP to find a suitable sparsity level s. Note that even knowing the sparsity level of the true solution does not mean that HTP actually converges to a u with $Au = f$.

2.2. **The Convex Relaxation: ℓ^1 Minimization.** While the previous subsection summarized some greedy techniques for sparse recovery, let us now focus on sparse recovery by minimizing the convex function that approximates the ℓ^0 norm most closely, i.e., solving the ℓ^1 minimization problem (1.2). It is interesting to see that the adaptive inverse scale space (aISS) method [6] allows to write ℓ^1 minimization

as an iterative method that keeps track of a support and solves a low dimensional least-squares type problem at each iteration (similar to OMP, WOMP, CoSaMP, or HTP). We will summarize the aISS approach in the next subsection.

2.2.1. *The adaptive inverse scale space method for ℓ^1 minimization.* Consider the general optimization problem

$$\min_u J(u) \qquad \text{such that } Au = f \tag{2.2}$$

with $A \in \mathbb{R}^{n\times m}$, $f \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ and a convex functional J. It is well know that the so called *Bregman iteration* (BI) provides an efficient iterative method to determine the above minimizer [25]. Particularly, BI was shown to be equivalent to the *augmented Lagrangian* (AL) method for the above problem (cf. [35, 15]). BI and AL find the solution to (2.2) by constructing the sequence

$$u^{k+1} = \arg\min_u J(u) + \frac{\lambda}{2}\|Au - f\|^2 - \langle p^k, u\rangle, \tag{2.3}$$

$$p^{k+1} = p^k + \lambda A^T(f - Au^{k+1}), \tag{2.4}$$

which involves an unconstrained optimization problem and an explicit update of the dual variable p^{k+1}, where the optimality condition of (2.3) combined with Equation (2.4) yields $p^{k+1} \in \partial J(u^{k+1})$. Note that the second equation can be written as

$$\frac{p^{k+1} - p^k}{\lambda} = A^T(f - Au^{k+1}). \tag{2.5}$$

Interpreting the parameter λ as a discrete time step, one finds BI to be backward time stepping on the evolution equation

$$\partial_t p(t) = A^T(f - Au(t)) \qquad \text{such that } p(t) \in \partial J(u(t)). \tag{2.6}$$

The above differential inclusion is called *inverse scale space flow* and was analyzed in [5].

The main finding of [6] was that in the case of $J(u) = \|u\|_1$, Equation (2.6) can be solved exactly without any time discretization due to its discrete nature. While the subgradient $p(t)$ evolves continuously (piecewise linear), the solution $u(t)$ remains piecewise constant. This allowed the authors of [6] to determine the subgradient $p(t^k)$ at some time t^k at which the solution is changing, before actually knowing the solution $u(t^k)$. Due to the characterization of the ℓ^1 subdifferential

$$p \in \partial\|u\|_1 \Leftrightarrow \begin{cases} p_i = \operatorname{sign}(u_i), & \text{if } u_i \neq 0, \\ |p_i| \leq 1, & \text{else,} \end{cases} \tag{2.7}$$

this restricts the support of $u(t^k)$ to the set $I = \{i \mid |p_i(t^k)| = 1\}$ and leads to a sequence of very low dimensional optimization problems. This approach was called aISS method and is given as algorithm 5 below. It has been shown that the aISS algorithm converges in finitely many iteration to an ℓ^1 minimizing solution. More specifically, one obtains $\|Au(t) - f\|^2 \in O(1/t)$ as well as a convergence rate of $u(t)$ to an ℓ^1-minimizing solution in the Bregman distance. We refer to [6] for a detailed convergence analysis of the aISS method. Also note that the theory for solving the inverse scale space flow equation exactly without any discretization was extended in [23] to arbitrary regularizations $J(u)$ that are polyhedral.

Note that a major difference between the aISS method and the BI is that the optimization problem in the primal variable is low dimensional in the aISS algorithm. In BI, one first computes the full primal variable u^{k+1} and then updates

Algorithm 5 aISS Method

Parameters: $A,\ f,$ threshold ≥ 0
Initialization: $r_0 = f$, $t_1 = 1/\left\|A^T r_0\right\|_\infty$, $p(t_1) = t_1\ A^T r_0$
while $\|r_k\| >$ threshold **do**

1. Compute $I_k = \{i \mid |p_i(t_k)| = 1\}$
2. Compute $u(t_k) = \arg\min_u \left\{\|AP_{I_k}u - f\|^2\right\}$ subject to $u(t_k)p(t_k) \geq 0$
3. Compute the residual $r_k = f - Au(t^k)$.
4. Obtain t_{k+1} as

$$t_{k+1} = \min\{t \mid t > t_k, \exists j : |p_j(t)| = 1, u_j(t_k) = 0, p_j(t) \neq p_j(t_k)\}, \tag{2.8}$$

where

$$p_j(t) = p_j(t_k) + (t - t_k)(A^T r_k)_j \tag{2.9}$$

5. Update the dual variable $p(t)$ via (2.9) with $t = t_{k+1}$

end while
return $u(t_k)$

the dual variable p^{k+1} accordingly. In the aISS method, one knows the subgradient $p(t^k)$ before knowing the corresponding solution, such that one can use the characterization (2.7) to significantly reduce the size of the optimization problem.

On the other hand, the parameter λ in BI can be chosen arbitrarily large, while the corresponding quantity in the aISS flow - the next time step $t^{k+1} - t^k$ - is determined by the previous subgradient as well as the correlation between the current residual and the columns of A by considering $A^T r_k$ in (2.9). While the support in BI changes almost arbitrarily, the support in the aISS method is often increased by a single index per iteration. In this sense, and also due to the consideration of the correlation between the current residual and the columns of A, the aISS method has quite some similarities with the greedy sparse reconstruction method OMP.

In the next section we will first point out similarities and differences between OMP, WOMP, HTP, CoSaMP and the aISS method, before adapting some ideas of OMP and WOMP to the aISS flow to obtain a new sparse reconstruction method.

3. Approximating the Inverse Scale Space Flow

3.1. **Similarities and Differences between OMP, HTP, CoSaMP and the aISS method.** As an overview, we have summarized the main differences between the OMP, WOMP, HTP, CoSaMP and the aISS method in Table 1.

All five, OMP, WOMP, CoSaMP, HTP, and the aISS method, modify the index set I based on the correlation of A to the current residual, which also has the interpretation of the direction of steepest descend of the objective functional. While the size of the index set varies in OMP, WOMP and aISS, it stays fixed for HTP and CoSaMP. It is interesting to see that if $A^T f$ has a unique maximum, the first step of OMP and the aISS method is exactly the same. Both methods start by computing $A^T f$ and determine the first approximation u_1 by the best approximation only using the maximal component of $A^T f$. After this first step, OMP and aISS differ:

While OMP and WOMP only take the correlation to the current residual into account, the aISS method is based on the history of all previous correlations, i.e. $p^{k+1} = p^k + (t^{k+1} - t^k)A^T r^k$ for aISS vs. just $A^T r^k$ for OMP.

The index set in HTP is determined by the largest values in the quantity resulting from moving from the current iterate into the direction of steepest descend. While the aISS method also moves in the direction of steepest descend, it adapts the time step, such that the quantity $p(t^k)$ always maintains the interpretation of a subgradient of $u(t^{k+1})$. Note that having a u^k versus a p^k in the formula for determining the index set makes a significant difference and, for instance, occurred in the ℓ^1 minimization techniques iterative soft thresholding (cf. [1] and references therein) versus linearized Bregman iteration [8].

The CoSaMP method takes yet a different approach to sparse recovery. In a first step it relaxes the constraint that the sparsity of the solution has to be less or equal to s and takes up to $3s$ indices as a support into account. After finding the optimal solution on this larger/relaxed index set the hard thresholding operator is applied to reduce the solutions support size back to s.

A difference between the aISS method and all greedy approaches is that the low dimensional optimization problem is a least squares problem for OMP, WOMP, CoSaMP and HTP, while it is a sign-constrained least squares problem for the aISS method. Although this step can make the aISS algorithm significantly more expensive, it allows indices to leave the support (opposed to OMP, WOMP), and seems to make the difference between greedy approaches and ℓ^1 minimization. (For details regarding the reason why the sign constraint allows indices to leave the support we refer to [6]). By proposing a new greedy method whose only difference to the aISS method is to replace the sign-constrained least squares problem by a simple least squares problem, we will be able to investigate in our numerical experiments (Section 4) how much of a difference the type of least squares problem makes in practice.

Finally, let us mention that the computational costs per iteration are similar in the sense that each algorithm has to multiply by A^T and solve a least squares type of problem. However, the actual costs and runtimes can be quite different as we will confirm with our numerical experiments in Section 4. As mentioned already, the aISS approach can be more computationally expensive since it requires the solution to obey additional sign constraints. Another difference is the size of the least squares problems. For instance, the first iterations of OMP are very cheap since the support set contains only very few indices. In comparison, HTP immediately solves problems of size s and CoSaMP even problems of size $3s$. However, one can hope to need significantly fewer iterations with HTP or CoSaMP. Since OMP needs at least s iterations for s sparse solutions the advantage of having to take fewer iterations with HTP and CoSaMP (also in comparison to aISS) will likely increase with increasing s.

	OMP	**WOMP**	**CoSaMP**	**HTP**	**aISS flow**
Change of the index set is based on	$A^T r^k$	$A^T r^k$	$A^T r^k$	$u^k + \nu A^T r^k$ for fixed ν	$p^k + \Delta t^{k+1} A^T r^k$ for variable time steps $\Delta t^{k+1} = t^{k+1} - t^k$
Size of the index set	increases by one at each iteration	increases by more than one, based on $\lvert(A^T r_k)_i\rvert \geq \rho \lVert A^T r_k \rVert_\infty$	is fixed at a given sparsity level s pruning a solution of support set $3s$ down to s each step	is fixed at a given level s, indices are replaced	most often increases by one but can change arbitrarily
Update for u^k	via a simple least squares problem	via a simple least squares problem	via a simple least squares problem followed by hard thresholding	via a simple least squares problem	via a sign-constrained least squares problem based on the signs of p^{k+1}
Main computational expenses per iteration	Multiplication with A^T, solution of a least squares problem on an index set of varying size	Multiplication with A^T, solution of a least squares problem on an index set of varying size	Multiplication with A^T, solution of a least squares problem on an index set of size $3s$	Multiplication with A^T, solution of a least squares problem on an index set of size s	Multiplication with A^T, solution of sign-constrained least squares problem on an index set of varying size

TABLE 1. Overview over the similarities and differences between OMP, WOMP, CoSaMP, HTP and the aISS method.

3.2. **The greedy inverse scale space (GISS) method.** Considering the similarities between OMP, WOMP, CoSaMP, HTP and the aISS method one could interpret the greedy methods as (faster) approximations of the inverse scale space flow. Thus, considering that ℓ^1 minimization often yields very good/better recovery results than greedy methods, it seems natural to develop a greedy method that approximates the aISS flow even more closely than OMP.

While the fixed index set from HTP seems to be difficult to adapt to the aISS framework, the ideas of OMP and even of WOMP are straight forward to include. To obtain a greedy method like OMP which approximates the ℓ^1 solution more closely, we propose to replace the constrained minimization of the aISS flow by an unconstrained one. We will refer to this method as the *greedy inverse scale space (GISS) method*. In terms of table 1, we will see that the only difference between GISS and OMP will be what the change of index set is based on, $A^T r^k$ for OMP and $p^k + (t^{k+1} - t^k)A^T r^k$ for GISS. Our numerical experiments in Section 4 show that GISS indeed inherits some desirable properties from the aISS method, approximating ℓ^1 minimization much more closely and yielding better recovery results. The GISS algorithm (aISS method after replacing the constrained minimization by an unconstrained minimization) is given as algorithm 6 below.

Algorithm 6 GISS method

Parameters: $A,\ f,$ threshold ≥ 0
Initialization: $r_0 = f$, $t_1 = 1/\left\|A^T r_0\right\|_\infty$, $p(t_1) = t_1\ A^T r_0$
while $\|r_k\| >$ threshold **do**

1. Compute $I_k = \{i \mid |p_i(t_k)| = 1\}$
2. Compute $u(t_k) = \arg\min_u \left\{\|AP_{I_k}u - f\|^2\right\}$
3. Compute the residual $r_k = f - Au(t^k)$.
4. Obtain t_{k+1} as

$$t_{k+1} = \min\{t \mid t > t_k, \exists j : |p_j(t)| = 1, u_j(t_k) = 0, p_j(t) \neq p_j(t_k)\}, \tag{3.1}$$

where

$$p_j(t) = p_j(t_k) + (t - t_k)(A^T r_k)_j \tag{3.2}$$

5. Update the dual variable $p(t)$ via (3.2) with $t = t_{k+1}$

end while
return $u(t_k)$

First of all note, that the above algorithm converges to a solution of $Au = f$: At each iteration, the new time step t^k is chose in a way that an index i enters the index set I_k. Once a certain index j is in I_k the optimality condition to step 2. yields $\left(A^T(f - Au^k)\right)_j = 0$, such that it never leaves the index set. If no more t_{k+1} exists, then we have to have $A^T r_k = 0$, which automatically yields $u(t_k) \in \arg\min_u \|Au - f\|^2$ and the algorithm has converged. Naturally, this means the GISS algorithm converges in a finite number of iterations, and, more precisely, the number of iterations never exceeds the rank of A, i.e. m for $A \in \mathbb{R}^{m\times n}$ in the typical compressed sensing case.

Analog to [6] one can show the strict decrease of $\|Au - f\|^2$ at each step k as well as the exact recovery of any one-sparse solution in a single step given that A has normalized columns. It is interesting to see that even the result about recovering solutions that meet an ERC exactly can be shown similar to Proposition 4 in [6]. Before we actually state and prove this result, let us consider additionally incorporating the idea of WOMP into our framework. Our idea is to take a larger step into the current direction of steepest descend and project the resulting $p(t^k+1)$ back onto the unit ℓ^∞ ball. This slightly modified version of the GISS method is given as algorithm 7 below. We will denote the modification by GISS$^\rho$ where ρ is the factor by which we enlarge the usual time step taken by GISS. Note that $\rho \geq 1$ and that GISS^1 is the same as GISS.

Algorithm 7 GISS$^\rho$ method

Parameters: $A,\ f,\ \rho \geq 1,\ \text{threshold} \geq 0$
Initialization: $r_0 = f,\ t_1 = 1/\left\|A^T r_0\right\|_\infty,\ p(t_1) = t_1\ A^T r_0$
while $\|r_k\| >$ threshold **do**

1. Compute $I_k = \{i \mid |\tilde{p}_i(t_k)| = 1\}$
2. Compute $u(t_k) = \arg\min_u \left\{\|AP_{I_k} u - f\|^2\right\}$
3. Compute the residual $r_k = f - Au(t^k)$.
4. Obtain t_{k+1} as

$$t_{k+1} = \rho \min\{t \mid t > t_k, \exists j : |p_j(t)| = 1, u_j(t_k) = 0, p_j(t) \neq p_j(t_k)\}, \tag{3.3}$$

where

$$p_j(t) = p_j(t_k) + (t - t_k)(A^T r_k)_j \tag{3.4}$$

5. Update the dual variable $p(t)$ via (3.4) with $t = t_{k+1}$ and set $\tilde{p}(t^{k+1}) = \text{sign}(p(t^{k+1})) \min(|p(t^{k+1})|, 1)$

end while
return $u(t_k)$

3.3. **Analysis of the GISS$^\rho$ Method.** The following exact recovery result (similar to OMP/WOMP) is shown for GISS$^\rho$ which - due to GISS^1 being the same as GISS - holds for all versions of our proposed algorithm:

Proposition 3.1. *Let u_{opt} with $Au_{opt} = f$ meet an ERC of order $\frac{1}{\rho}$ and let I be the support of u_{opt}. Then the GISS$^\rho$ algorithm recovers u_{opt} in at most $|I|$ steps, since $|p_j(t)| < 1$ for all t and all $j \notin I$.*

Proof. We prove the above inductively. At $t = 0$, we have $p(t) = 0$ and $u(t) = 0$ such that the support of $u(t)$ clearly is a subset of I. Now let the support of $u(t^k)$ be a subset of I, particularly, let $|p_i(t^k)| < 1$ for all $i \notin I$. We will show that $|p_i(t^{k+1})| < 1$ for all $i \notin I$. Based on the update formula for p we have

$$p(t^{j+1}) = p(t^j) + (t^{j+1} - t^j) A^T A(u_{\text{opt}} - u(t^j)) \tag{3.5}$$

Since $u(t^j)$ is supported on a subset of I for $j \leq k$ we know that

$$u_{\text{opt}} - u(t^j) = P_I(u_{\text{opt}} - u(t^j)), \tag{3.6}$$

where P_I denotes the projection onto the index set I. Projecting equation (3.5) onto the index set I as well as using (3.6) yields

$$P_I(p(t^{j+1}) - p(t^j)) = (t^{j+1} - t^j)P_I A^T A P_I(u_{\text{opt}} - u(t^j)) \tag{3.7}$$

$$\Rightarrow (t^{j+1} - t^j)P_I(u_{\text{opt}} - u(t^j)) = (P_I A^T A P_I)^{-1}\left(P_I(p(t^{j+1}) - p(t^j))\right). \tag{3.8}$$

Inserting the above back into (3.5) leads to

$$p(t^{j+1}) - p(t^j) = A^T \underbrace{A P_I (P_I A^T A P_I)^{-1}}_{=((AP_I)^\dagger)^T} \left(P_I(p(t^{j+1}) - p(t^j))\right), \tag{3.9}$$

$$= A^T((AP_I)^\dagger)^T\left(P_I(p(t^{j+1}) - p(t^j))\right). \tag{3.10}$$

Now, we sum up all the above equations for $j = 0$ up to $j = k$ to obtain

$$p(t^{k+1}) = A^T((AP_I)^\dagger)^T P_I p(t^{k+1}). \tag{3.11}$$

Now the GISS$^\rho$ algorithm is designed such that $\|p(t)\|_\infty \leq \rho$. Let us consider (3.11) at an index $i \notin I$ and denote the i-th column of A by a_i. Then

$$|p_i(t^{k+1})| = a_i^T((AP_I)^\dagger)^T P_I p(t^{k+1}), \tag{3.12}$$

$$\leq \|a_i^T((AP_I)^\dagger)^T\|_1 \underbrace{\|P_I p(t^{k+1})\|_\infty}_{\leq\rho}, \tag{3.13}$$

$$\leq \rho\|(AP_I)^\dagger a_i\|_1, \tag{3.14}$$

$$\overset{ERC}{<} 1. \tag{3.15}$$

Thus, indices $i \notin I$ do not enter the support of $u(t^k)$ for any k. □

To avoid confusion regarding the meaning of ERC, we'd like to point out that although [32] proves a theorem saying that ERC is met whenever the coherence of the matrix A is small enough, ERC itself does not imply anything about the coherence of the dictionary. It is a rather general criterion which, however, is very difficult to verify since it depends on the support of the sparsest solution we are looking for. Theorem 3.10 in [32] shows that whenever an ERC is not met there exist signals which OMP fails to recover. In this sense the ERC is the best criterion that guarantees the exact recovery with OMP. Our previous theorem therefore shows that the GISS algorithm has at least the same exact recovery guarantees as OMP. Looking at the proof of Theorem 3.10 in [32] we can see that the failure to meet an ERC leads to OMP selecting a non-optimal atom in the first step. Since the first steps of OMP and the GISS method coincide, the GISS algorithm will also pick a non-optimal atom, such that OMP and GISS have the same exact recovery guarantees.

Note that exact recovery guarantees are always based on worst case scenarios and two methods having the same theoretical guarantees does not necessarily mean that they show the same recovery properties in practice as we will see in the numerical experiments section.

We introduced our algorithm as a greedy approximation to the aISS method, which means that the solution of the GISS$^\rho$ algorithm should ideally be somewhat close to being an ℓ^1 minimizer. The following results allow an a-posteriori estimate

of how close the GISS[1] solution is to be ℓ^1 minimizing. Later we will generalize this result and show that the GISS$^\rho$ solution is close to a weighted ℓ^1 minimizing solution with weights in $[\frac{1}{\rho},\ 1]$.

Theorem 3.2. *Let $\hat{u}$ be an ℓ^1 minimizing solution of $Au = f$, and let $u(t^K)$ be the solution of algorithm 6. We denote by*

$$M = \{i \mid u_i(t^K) \neq 0,\ sign(u_i(t^K)) = -sign(p_i(t^K))\}$$

the set of indices i where the $u_i(t^K)$ is nonzero but has a different sign than $p_i(t^K)$. Then the estimate

$$\|\hat{u}\|_1 \geq \|u(t^K)\|_1 - 2\sum_{i\in M}\left(|u_i(t^K)| - sign(u_i(t^K))\hat{u}_i\right) \tag{3.16}$$

holds.

Proof. Based on the structure of the GISS algorithm we know that $|p_i(t^K)| \leq 1$ for indices i which are not in the support I of $u(t^K)$. Now let us define an element $\tilde{p}$ by

$$\tilde{p}_i = \begin{cases} p_i(t^K) & \text{if } i \notin I, \\ \operatorname{sign}(u_i(t^K)) & \text{else.} \end{cases} \tag{3.17}$$

Obviously, $\tilde{p} \in \partial\|u(t^K)\|_1$ such that we can conclude that

$$0 \leq \|\hat{u}\|_1 - \|u(t^K)\|_1 - \langle\tilde{p}, \hat{u} - u(t^K)\rangle. \tag{3.18}$$

Now the GISS[1] algorithm produces a $p(t^K) \in \operatorname{range}(A^T)$, which means that

$$\langle p(t^K), \hat{u} - u(t^K)\rangle = \langle q, \underbrace{A\hat{u}}_{=f} - \underbrace{Au(t^K)}_{=f}\rangle = 0. \tag{3.19}$$

Hence we can add zero in Equation (3.18) to obtain

$$0 \leq \|\hat{u}\|_1 - \|u(t^K)\|_1 - \langle\tilde{p} - p(t^K), \hat{u} - u(t^K)\rangle. \tag{3.20}$$

Let us look at the term $\langle\tilde{p} - p(t^K), \hat{u} - u(t^K)\rangle$. By definition of $\tilde{p}$, this term vanishes outside of the support I. On I we can divide the resulting sum in components where $p(t^K)$ has the 'wrong' sign, i.e., $i \in M$, and components where $p(t^K)$ has the 'right' sign, i.e., $i \in M^c$.

$$\begin{aligned}
\langle\tilde{p} - p(t^K), \hat{u} - u(t^K)\rangle &= \sum_{i\in I}(\tilde{p} - p(t^K))_i(\hat{u} - u(t^K))_i, \\
&= \sum_{i\in M}(\tilde{p} - p(t^K))_i(\hat{u} - u(t^K))_i \\
&\quad + \sum_{i\in(I\cap M^c)}(\tilde{p} - p(t^K))_i(\hat{u} - u(t^K))_i, \\
&= \sum_{i\in M}\left(\operatorname{sign}\big(u_i(t^K)\big) + \operatorname{sign}\big(u_i(t^K)\big)\right)(\hat{u} - u(t^K))_i \\
&\quad + \sum_{i\in(I\cap M^c)}\left(\operatorname{sign}\big(u_i(t^K)\big) - \operatorname{sign}\big(u_i(t^K)\big)\right)(\hat{u} - u(t^K))_i, \\
&= \sum_{i\in M} 2\left(\operatorname{sign}(u_i(t^K))\hat{u}_i - |u_i(t^K)|\right). \qquad (3.21)
\end{aligned}$$

Inserting this part into (3.20) we obtain

$$\|\hat{u}\|_1 \geq \|u(t^K)\|_1 + 2\sum_{i\in M}\left(\text{sign}(u_i(t^K))\hat{u}_i - |u_i(t^K)|\right). \tag{3.22}$$

□

Note that Theorem 3.2 assumed $\hat{u}$ be an ℓ^1 minimizing solution of $Au = f$ such that the inequality $\|\hat{u}\|_1 \leq \|u(t^K)\|_1$ is trivial.

The above theorem allows an easy conclusion for the GISS algorithm

Conclusion 3.3. *If $\{i \mid u(t^K)_i \neq 0,\ sign(u_i(t^K)) = -p_i(t^K)\} = \emptyset$, then the GISS algorithm has determined an ℓ^1 minimizing solution.*

It is remarkable that we obtain an extremely simple way to check if our greedy method converged to the ℓ^1 minimizer.

The above criterion is interesting for three reasons: Firstly, it allows some theoretical analysis, namely stating that if criteria like the Null Space Property or the RIP (cf. [29]) (or any other condition that ensures ℓ^1 minimization to recover the sparsest solution) are met, the additional property of $M = \emptyset$ is an exact recovery criterion for the GISS algorithm. Secondly, it tells us if it might be worth it (re-)running an ℓ^1 minimization algorithm on the problem. Thirdly, popular state of the art ℓ^1 minimization algorithms such as the split Bregman (or Augmented Lagrangian) method, the linearized Bregman method, or the aISS method, are typically initialized with a starting subgradient $p = 0$, but still converge to an ℓ^1 minimizing solution if one starts with an arbitrary subgradient $p \in \text{range}(A^T)$. Since the latter is guaranteed by the GISS algorithm, one can use the GISS result as a possible warm start for an ℓ^1 minimization algorithm. The latter of course only makes sense if there are only a few elements in M such that one can hope for $p(t^K)$ to be close to the true ℓ^1 subgradient.

In case of the GISS$^\rho$ method with $\rho > 1$ the situation is more complicated, since the values of $|p(t^K)|$ on the support are somewhere in between 1 and ρ, such that even in the case where the signs of $p(t^K)$ are correct, we are not exactly determining an ℓ^1 minimizer anymore. However, interestingly, we determine a solution which is close to a weighted ℓ^1 norm, where the weights vary in between $\frac{1}{\rho}$ and 1.

Theorem 3.4. *Let $u(t^K)$ be the solution of algorithm 7 and let I denote the support of $u(t^K)$. Let $\hat{u}$ be the solution to the weighted ℓ^1 problem*

$$\min_u J(u) \qquad \text{such that } Au = f, \tag{3.23}$$

with the weighted ℓ^1 norm $J(u) = \sum \frac{1}{\omega_i}|u_i|$ and $\omega_i = |p_i(t^K)|$ if $i \in I$ and $\omega_i = 1$ if $i \notin I$. We denote by $M = \{i \in I \mid sign(u(t^K)_i) = -sign(p_i(t^K))\}$ the set of indices i where the $u(t^K)_i$ is nonzero but has a different sign than $p_i(t^K)$. Then the estimate

$$J(\hat{u}) \geq J(u(t^K)) - 2\sum_{i\in M}|p_i(t^k)|(|u_i(t^K)| - sign(u_i(t^K))\hat{u}_i) \tag{3.24}$$

holds.

Proof. The prove is very similar to the prove of Theorem 3.2 but will be given here for the sake of completeness. The main change is that the subdifferential of the

weighted ℓ^1 norm J can now be characterized as

$$\tilde{p} \in \partial J(u) \Leftrightarrow \begin{cases} \tilde{p}_i = \omega_i sign(u_i) & \text{if } u_i \neq 0 \\ |\tilde{p}_i| \leq \omega_i & \text{else} \end{cases} \tag{3.25}$$

Again the GISS$^\rho$ algorithm yields $|p_i(t^K)| \leq 1$ for indices $i \notin I$ and we can define an element $\tilde{p}$ by

$$\tilde{p}_i = \begin{cases} p_i(t^K) & \text{if } i \notin I, \\ \text{sign}(u_i(t^K))|p_i(t^K)| & \text{else.} \end{cases} \tag{3.26}$$

It is easy to see from Equation (3.25) that $\tilde{p} \in \partial J(u(t^K))$, and since $p(t^K) \in \text{range}(A^T)$, we find

$$0 \leq J(\hat{u}) - J(u(t^K)) - \langle \tilde{p} - p(t^k), \hat{u} - u(t^K) \rangle. \tag{3.27}$$

Again, we can examine the term $\langle \tilde{p} - p(t^k), \hat{u} - u(t^K) \rangle$ by dividing it into the sum over the sets M and $I \cap M^c$. Similar to the proof of Theorem 3.2 the sum over $I \cap M^c$ vanishes and we obtain

$$\langle \tilde{p} - p(t^k), \hat{u} - u(t^K) \rangle = 2 \sum_{i \in M} |p_i(t^k)|(\text{sign}(u_i(t^K))\hat{u}_i - |u_i^K|). \tag{3.28}$$

Inserting (3.28) into (3.27) yields the assertion. □

Again, the correctness of the signs allows to state the convergence to a weighted ℓ^1 minimizing solution.

Conclusion 3.5. *If $\{i \mid u(t^K)_i \neq 0,\ sign(u(t^K)_i) = -sign(p_i)(t^K)\} = \emptyset$, then the GISS$^\rho$ algorithm has determined a $J(\cdot)$ minimizing solution, with J as defined in Theorem 3.4.*

Since the weights $\frac{1}{\omega_i}$ in the definition of J lie in $[\frac{1}{\rho}, 1]$, it is clear that the closer ρ is to one, the closer GISS$^\rho$ will be to the GISS^1 and also to the ℓ^1 minimizing solution. Also, notice that a weighted ℓ^1 norm is the same as minimizing an unweighted ℓ^1 norm with a sensing matrix A in which not all columns have the same norm.

It is interesting to see that if the condition of Conclusion 3.5 is met, there even is a chance of the solution $u(t^K)$ of the GISS$^\rho$ algorithm not only being a weighted ℓ^1 minimizing solution, but also being an unweighted ℓ^1 minimizing solution.

Proposition 3.6. *In the notation of Theorem 3.4, define*

$$\epsilon_i = \begin{cases} 0 & \text{if } i \notin I \\ (\omega_i - 1)sign(u(t^K)) & i \in I \end{cases} \tag{3.29}$$

Let the condition of Conclusion 3.5 be met. If, additionally, there exists a v with $v_i = 0$ for $i \in I$ and $|(p(t_K) - v)_i| \leq 1$ for $i \notin I$, such that $(\epsilon + v) \in range(A^T)$, then $u(t^K)$ is an ℓ^1 minimizing solution.

Proof. Note that based on the assumtions of the above proposition, we can write

$$p(t^K) = p_1 + \epsilon + v \tag{3.30}$$

for an element p_1 which meets $p_1 \in \partial \|u(t^k)\|_1$: For $i \in I$, $v_i = 0$ and hence the above simplifies to $(p(t^K))_i = (p_1)_i + \epsilon_i$, which holds based on the definition of ϵ. For $i \notin I$, $\epsilon_i = 0$ and hence the above simplifies to $(p(t^K))_i - v_i = (p_1)_i$. Since $|(p(t_K) - v)_i| \leq 1$ we can ensure that $|(p_1)_i| \leq 1$ for $i \notin I$.

Since $p(t^K) \in \text{range}(A^T)$ holds as the optimality condition of $u(t^K)$ being a $J(\cdot)$ minimizing solution, we can conclude

$$p_1 = p(t^K) - (\epsilon + v) \in \text{range}(A^T) \tag{3.31}$$

which, together with $p_1 \in \partial \|u(t^k)\|_1$, proves that $u(t^K)$ is also ℓ^1 minimizing.

□

Since the vector ϵ is known after the convergence of GISS$^\rho$, a weaker version of the above proposition is very easy to check:

Conclusion 3.7. *If additionally to the condition of Conclusion 3.5, the vector ϵ defined by* (3.29) *meets $\epsilon \in$ range(A^T), then the GISS$^\rho$ algorithm has determined an ℓ^1 minimizing solution to $Au = f$.*

Despite the greedy steps of leaving out the constrained minimization as well as stretching the time steps by ρ, we obtain an algorithm for which a possible verification of having converged to an ℓ^1 minimizing solution is very simple.

4. Numerical results

Let us look at the numerical performances of each of the methods. Note that the main contribution of this paper is the introduction of a new greedy sparse recovery method which approximates ℓ^1 minimization closely and allows to a-posteriori determine if the solution also is an ℓ^1 minimizing solution. The following numerical experiment serves as an illustration of how GISS$^\rho$ behaves, particularly in comparison to the related methods aISS, OMP and WOMP. We additionally include HTP and CoSaMP in our comparison to have results of two state of the art greedy methods which immediately have a larger support. It is encouraging to see that GISS$^\rho$ shows strong performances in comparison to the other greedy methods.

The numerical results section is organized as follows: First we describe one experimental setup with a particular type of sensing matrix and a particular way of generating the data and analyze the results for this case in detail. We discuss different ways of evaluating the results and comment on the differences of the methods (such as HTP fixing the support size and trying to minimize the error opposed to OMP increasing the support until the desired accuracy is reached). In a second step, we vary the experimental setup, i.e. use different sensing matrices and different ways to generate the data, to investigate to what extend the conclusions from our first experiment seem to hold in general.

For our first experiments, we fix the number of rows of our sensing matrix $A \in \mathbb{R}^{m \times n}$ to be $m = 200$. We vary the number of columns n as well as the sparsity s of what we call *source element* $\tilde{u}$, which denotes the element we create our data with, i.e., $f = A\tilde{u}$. A is generated as a random matrix with values drawn from a Gaussian distribution. Afterwards the columns of A are normalized to have an ℓ^2 norm of 1. The source element $\tilde{u}$ is generated by randomly selecting indices for the nonzero entries, which are then randomly drawn from $\{+1, -1\}$. Note that this type of experimental setup has been used before, for instance in [19]. For each setting, i.e., for each combination of (n, s), we run our experiment 100 times to be more independent of particular realizations of the random matrix and data generation.

We investigate the behavior of the aISS method, OMP, WOMP with a parameter $\alpha = 0.8$, CoSaMP with the (true) sparsity of $\tilde{u}$ as the number of nonzero coefficients s, HTP with $\nu = 1$ and also with the sparsity of $\tilde{u}$ as the number of nonzero

coefficients, GISS[1] and GISS[1.2]. The comparison is somehow unfair since CoSaMP and HTP have the advantage of being given the true sparsity level we wish to reconstruct. For the solution u of each algorithm for each choice of (n, s) we record

(1) the average runtime,
(2) the average relative error to the source element $\tilde{u}$, $\frac{\|\tilde{u}-u\|_2^2}{\|\tilde{u}\|_2^2}$,
(3) the average sparsity $|u|_0$,
(4) the average number of iterations the algorithm needed.

To illustrate the algorithms behavior we can plot the results in a diagram with axes for varying n and s (resulting in a phase diagram). The average sparsity $|u|_0$ of the methods' solution is shown in Figure 1.

The HTP result reflects the sparsity of the source element $\hat{u}$ exactly since the true sparsity level was an input parameter. Since this is also the case for the CoSaMP algorithm, we omitted the CoSaMP plot in Figure 1. Note that although we cannot guarantee that the source element $\hat{u}$ always was the sparsest solution, no algorithm ever found a solution with lower ℓ^0 norm. Comparing the algorithms, we can see that the ℓ^1 minimizing solution determined by the aISS algorithm on average gave the sparsest recovery results. While GISS and GISS[1.2] do not perform quite as well as the aISS method, they clearly outperform OMP and WOMP. Since a precise comparison might be difficult in Figure 1, let us additionally show the isocontours, at which the relative error in sparsity exceeds 10%, i.e., the isocontours of $\frac{|u|_0-|\hat{u}|_0}{|\hat{u}|_0} > 0.1$. Plotting the isocontours of the aISS method, OMP, WOMP, GISS, and GISS[1.2] in one diagramm allows an immediate comparison up to which setting each method reconstructed sufficiently sparse solutions. The result is shown in Figure 2.

It now becomes obvious that the aISS method recovers the sparsest solutions, followed by GISS and GISS[1.2]. OMP and WOMP gave clearly weaker sparse recovery results. Among GISS and GISS[1.2] we can see that the less greedy variant was slightly more successful, although the difference is remarkably small. It is very surprising that in our experiments WOMP actually succeeded more often in finding sparse solutions than OMP.

Besides the sparsity, one could also consider looking at the relative error to the source element, $\frac{\|u-\hat{u}\|^2}{\|\hat{u}\|^2}$. Different from the ℓ^0 comparison above, the relative error will show if in the cases where the algorithms result is not sparse, certain algorithms succeeded in reconstructing some peaks of $\hat{u}$ exactly. Figure 3 shows the relative error in the same type of plot as above - this time including the CoSaMP result as well as a contour plot, showing the isocontours at which $\frac{\|u-\hat{u}\|^2}{\|\hat{u}\|^2} > 0.1$ for each method. Since this relative error can exceed values of 1 we pruned all values larger than 1 to have all images on the same scale.

Clearly, the ℓ^1 minimization was the best method in terms of the relative error, too. We can see that the aISS method has very low relative errors even for settings where the sparsity of the solution indicated that the method was unable to reconstruct $\hat{u}$. The greedy methods generally have a sharper transition between the cases where the method succeeds and where it fails. We can see that GISS inherits some of ℓ^1-like behavior of a smooth relative error transition whereas the GISS[1.2] version shows an almost a binary success-fail diagram. In the contour plot we can see that the aISS approach has the highest success rate, very closely followed by GISS. CoSaMP and GISS[1.2] show comparable reconstruction performances and

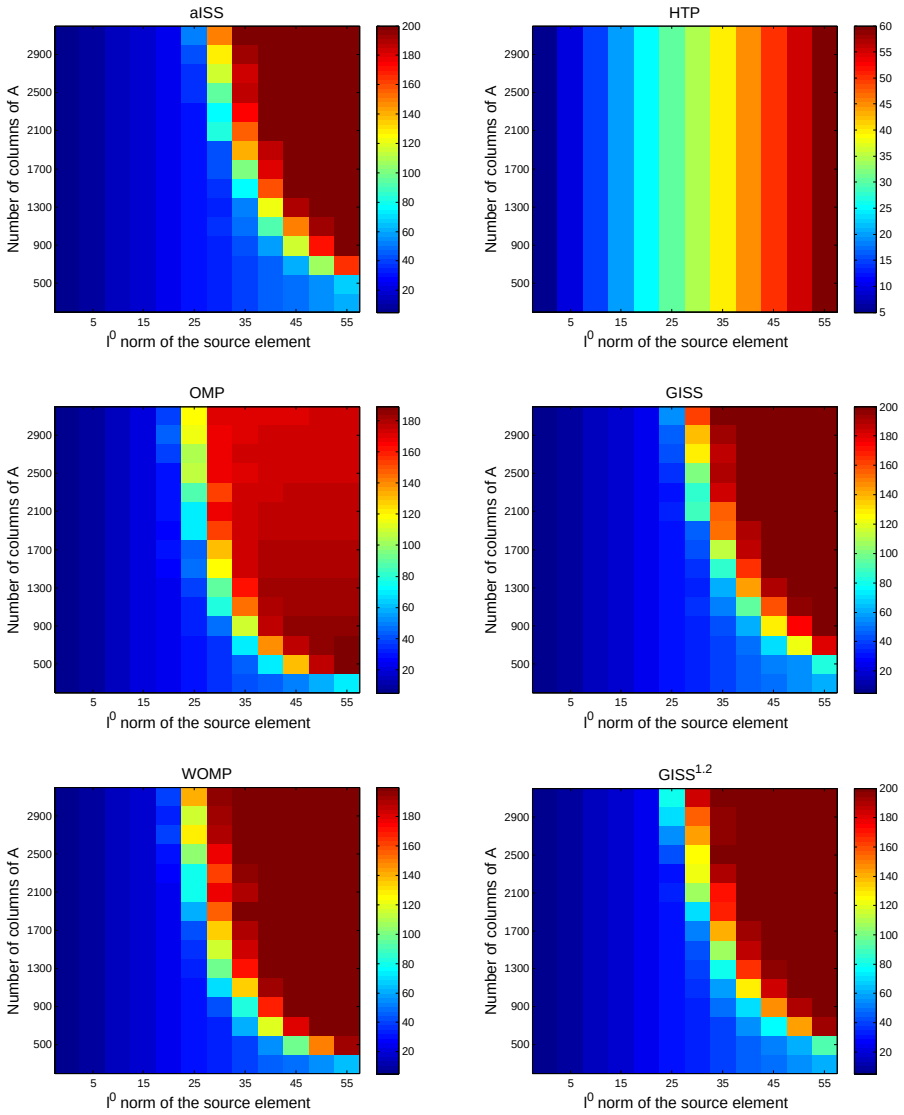

FIGURE 1. Comparison of the average ℓ^0 norm of the solutions of different sparse reconstruction algorithms. The sparsity of the source element the data was created with as well as the number of columns of the sensing matrix A are varied (x- and y-axis). The sensing matrix A is a normalized random matrix. Each setting was run 100 times and the average values are shown.

seem to do better than HTP. The OMP and WOMP methods behave very similarly and seem to give clearly weaker reconstruction results.

Generally, GISS indeed seems to inherit some desirable properties from the ℓ^1 minimizing aISS flow. Furthermore it is encouraging to see that GISS outperforms

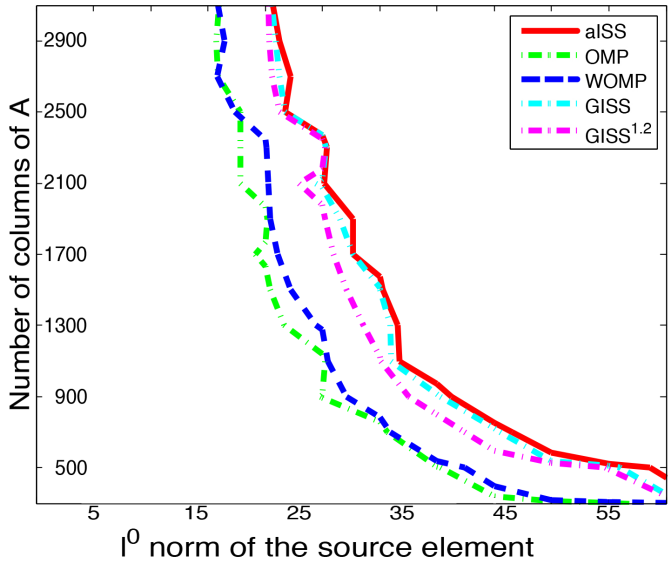

FIGURE 2. Isocontours at which the relative error in sparsity, $\frac{|u|_0-|\hat{u}|_0}{|\hat{u}|_0}$ exceeds 0.1.

HTP and CoSaMP and GISS$^{1.2}$ is comparable to CoSaMP and better than HTP. Although this is a particular test case (of a source element with values drawn from $\{-1,+1\}$) it is remarkable that GISS and GISS$^{1.2}$ reached these performance results despite HTP and CoSaMP being given the correct sparsity level as an input parameter. Furthermore, CoSaMP and HTP do not converge to a sparse solution of $Au = f$. Since their support size is fixed, they try to minimize the objective function $\|Au - f\|$ for the given size of the support, while all five other methods keep iterating until $Au = f$ is solved exactly. To illustrate this problem, Figure 4 shows the errors $\|Au - f\|^2$ for the HTP and CoSaMP methods. Note that all other methods always had a quadratic error of less than 10^{-9}. It is interesting to see in Figure 4 that for HTP the error is particularly high in the transition of the settings where $\hat{u}$ can or cannot be recovered exactly while for CoSaMP the error seems to increase with the ill-posedness. Generally, HTP resulted in lower errors than CoSaMP.

After having analyzed the quality of the algorithms in terms of their ability to reconstruct sparse solutions, let us look at the computational expenses of the algorithms. For all greedy algorithms the most expensive steps are the multiplication with A^T as well as a solution of a least-squares problem. For aISS the least-squares problem has to be solved subject to additional sign constraints. The size of the optimization problems is first very low dimensional and then increasing for aISS, OMP, WOMP, GISS and GISS$^{1.2}$, while it is fixed at $3s$ for CoSaMP and at s for HTP. Let us look at the number of iterations each algorithm needed to converge: Figure 5 shows the average number of iterations in each setting.

We can see that the price the aISS algorithm pays for obtaining the sparsest reconstruction is a significantly higher number of iterations. Note that the aISS method needed up to 350 iterations, while the number of iterations OMP or GISS is

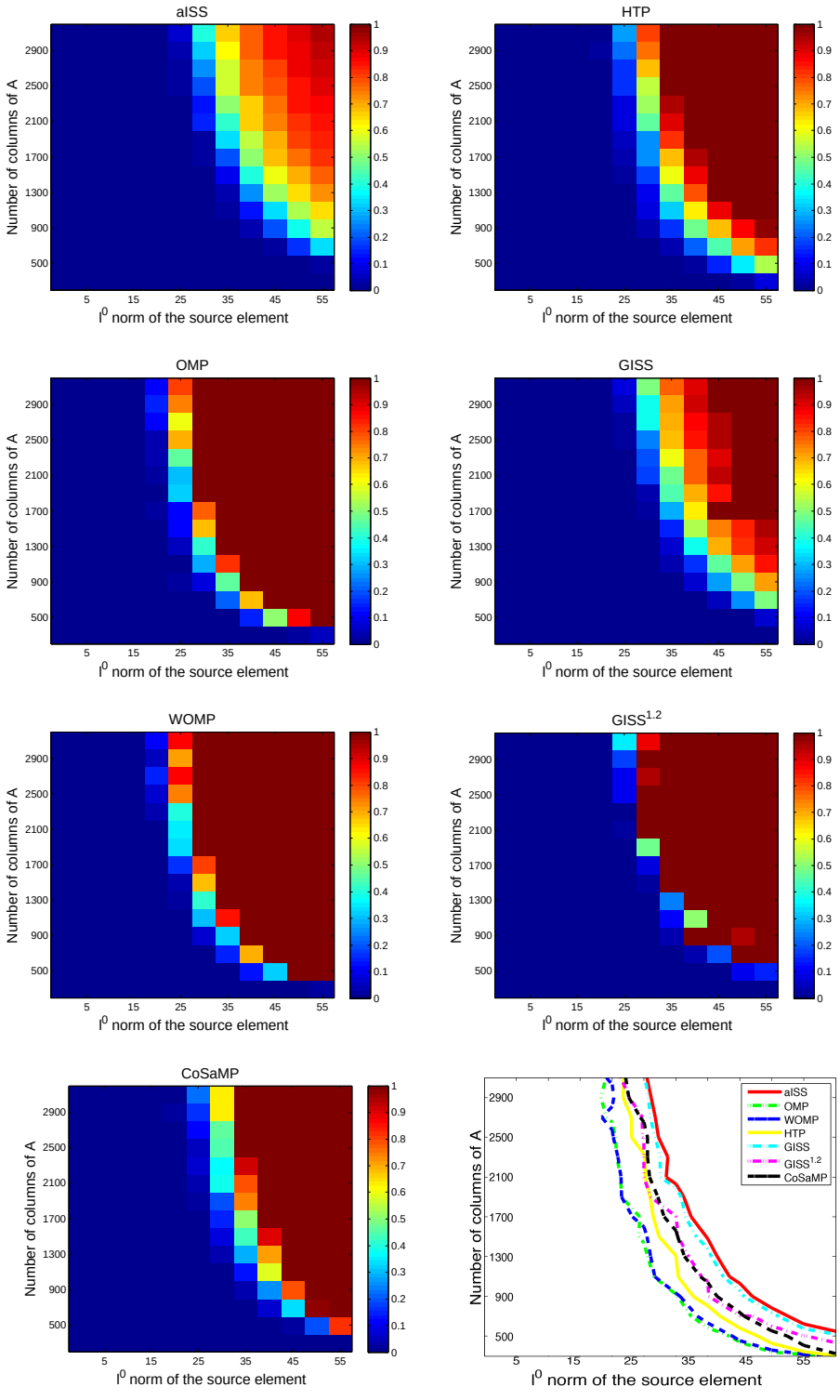

FIGURE 3. Comparison of the average relative error of the solution of each algorithm to $\hat{u}$. The sparsity of the source element $\hat{u}$ the data was created with as well as the number of columns of the sensing matrix A are varied (x- and y-axis). The sensing matrix A is a normalized random matrix. Each setting was run 100 times and the average values are shown.

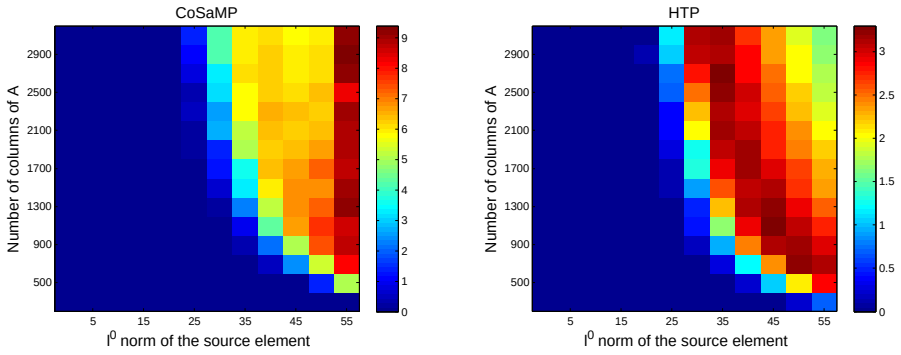

FIGURE 4. Error $\|Au - f\|^2$ for CoSaMP and HTP.

bounded by the rank of the sensing matrix (i.e. by 200 in our case). Furthermore, the aISS iterations can be more expensive since a low dimensional sign constrained least squares problem is solved. The number of iterations OMP and GISS need is comparable and, since one index is allowed to enter the support at each iteration, looks similar to the number of nonzero components shown in Figure 1. The number of iterations needed by WOMP is significantly lower with at most 28 iterations. Even lower is the number of iterations of the GISS$^{1.2}$ algorithm, which needed at most 21 iterations. Note that while the most underdetermined cases were most expensive for the aISS method, they become rather cheap (in terms of iterations) for WOMP.

The number of iterations HTP needed is significantly different from all other methods. While the HTP-iteration image has a peak at 100 iterations, it most often only shows 10-30 iterations. Interestingly, the most expensive cases for HTP are reached for a moderate sparsity level $\hat{u}$. As pointed out in [16], the HTP algorithm does not have to converge but can show periodic behavior, which is why we stop HTP after at most 500 iterations. Note that a reduction of the step size in HTP can guarantee convergence. However, we found in our numerical experiments that larger step sizes work better, despite the risk of eventually periodic behavior. We expect that HTP might have been caught in such a periodic behavior several times for the settings where the corresponding iteration-image has its peaks.

For the CoSaMP algorithm we also used a maximum number of 500 iterations which however is reached for almost all difficult cases (of very underdetermined systems and not very sparse source elements). The number of iterations CoSaMP took in our test cases almost seems to reflect the behavior of successful or unsuccessful reconstruction.

Although the comparison in terms of the number of iterations is interesting, it does not exactly reflect the computational expenses of each method, since, for example, GISS and GISS$^{1.2}$ have to find the next time step while OMP and WOMP just rely on the correlation to the residual. We found the runtime images, when displayed in the same fashion as the number of iterations above, to look rather noisy, which is why we focus on comparing the average run times over all settings. The results are shown in a bar chart in Figure 6.

Interestingly, CoSaMP was the most expensive method on average (0.87 seconds) which makes sense considering that it often took 500 iterations and always solves

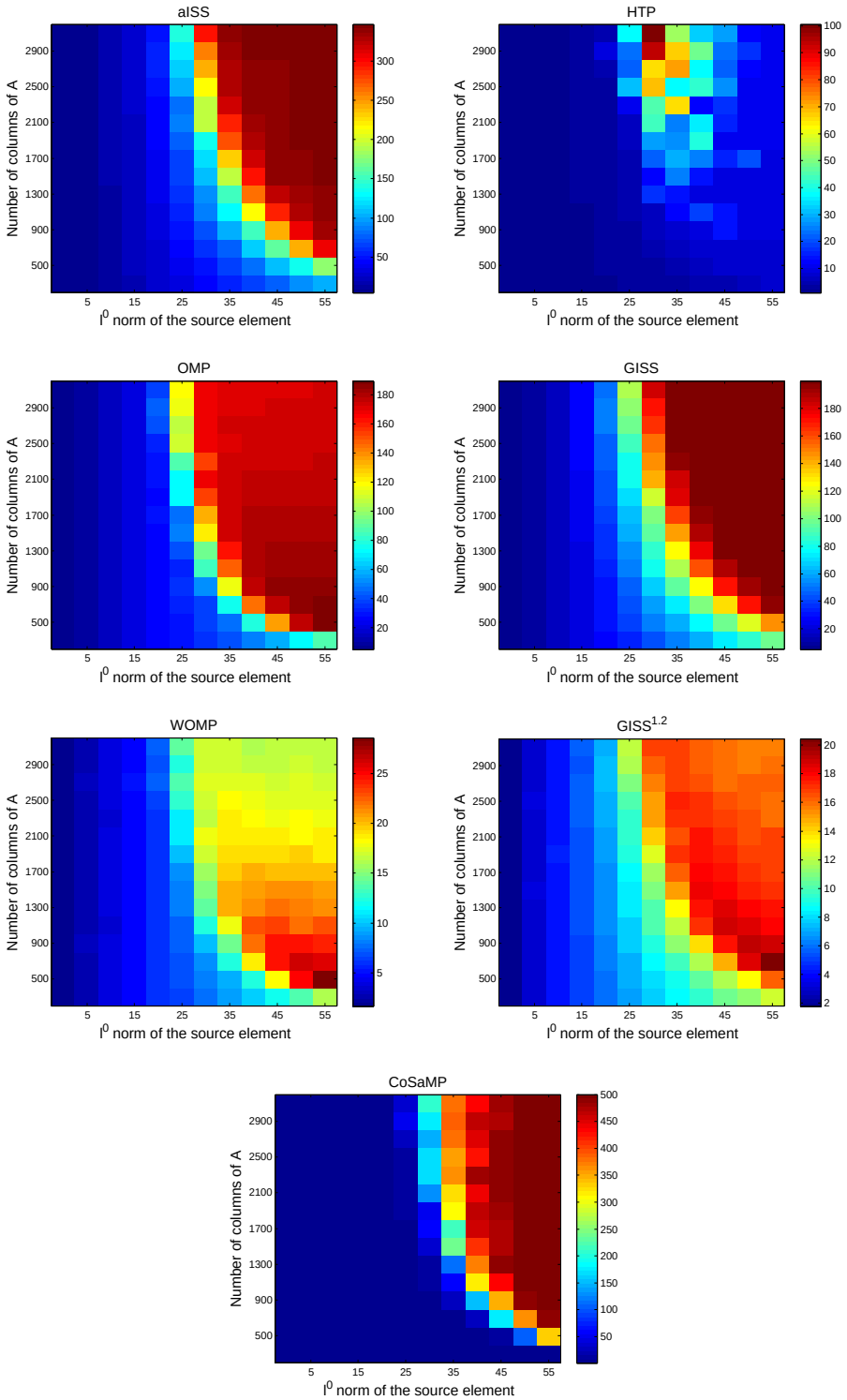

FIGURE 5. Comparison of the number of iterations each algorithm needed for convergence. The sparsity of the source element $\hat{u}$ the data was created with as well as the number of columns of the sensing matrix A are varied (x- and y-axis). The sensing matrix A is a normalized random matrix. Each setting was run 100 times and the average values are shown.

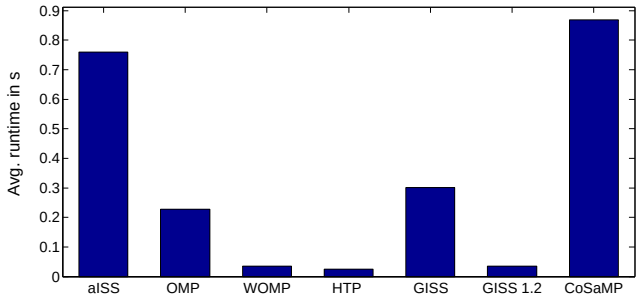

FIGURE 6. Overall average runtimes of the algorithms.

the rather large $3s$ dimensional optimization problems. The aISS method was slightly faster with about 0.76 seconds. Although the aISS algorithm needed fewer iterations, some of the optimization problems were more computationally expensive due to the sign constraint. Omitting the sign constraints (i.e. going from the aISS to the GISS) we see that we get a speed up of more than a factor of two with an average runtime of about 0.30 seconds. This, however, is an average over all test cases: The speed up is smaller for sparse source elements $\tilde{u}$, while it is much higher for the complicated cases (particularly in those, where all methods fail).

We can see that OMP (0.23 seconds) has a small speed advantage over GISS, mostly since GISS requires the computation of a next timestep t^k while OMP just needs to compute $A^T r^k$. A huge speedup is reached with the WOMP, HTP, and GISS$^{1.2}$ methods, which for instance beat aISS by more than a factor of twenty. WOMP and GISS$^{1.2}$ (both 0.034 seconds) have comparable speed while HTP (0.025 seconds) is even slightly faster. However, considering that HTP had the true sparsity as an input, does not always converge to a solution to $Au = f$ and gave worse recovery rates in our test case, we can conclude that GISS$^{1.2}$ seems to be a very good choice combining accuracy and speed.

Of course the above conclusion is only based on the case of a matrix A with entries drawn from a Gaussian distribution, normalized matrix columns and a source element with entries draw from $\{-1, +1\}$. As a next step we conducted four more experiments with different ways of generating the sensing matrix A and different ways of creating the source elements. The types of sensing matrices used for the following tests are random orthonormal matrices and partial discrete cosine matrices (motivated e.g. by [34]), as well as standard matrices with values draw from a Gaussian distribution (in our case with a non-zero mean) and sparse measurement matrices (as for instance motivated by [27]). For the sake of brevity we only show the contours at which the relative error of each method exceeds 10%, i.e. the contour at which $\frac{\|u-\hat{u}\|^2}{\|\hat{u}\|^2} > 0.1$ (similar to the lower right plot in Figure 3). The setups for the four experiments were the following:

- We choose A to be the matrix corresponding to a discrete Cosine transform and select the nonzero entries of the source element $\hat{u}$ by drawing from a uniform sampling of $[-0.5, 0.5]$. The contours for the 10% relative error threshold are shown in Figure 7.

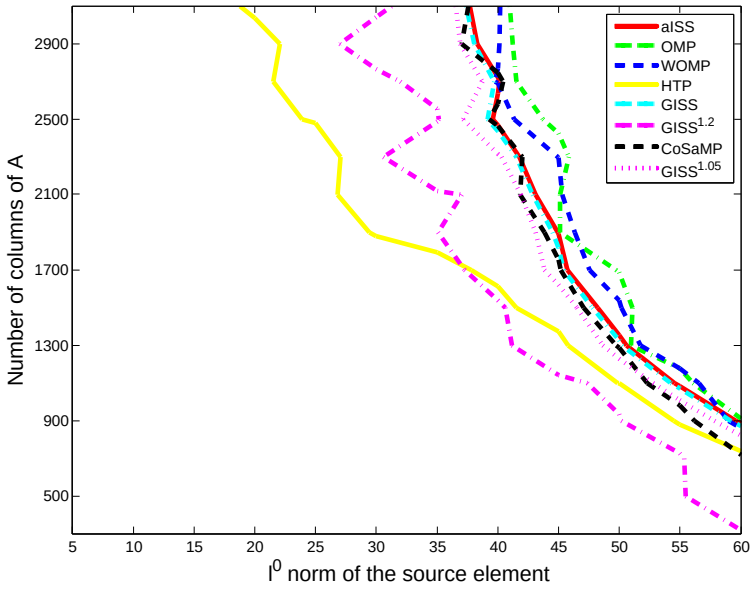

FIGURE 7. Contours of all methods at which the relative error to the true solution exceeds 10%. The sensing matrix A was chosen to be a discrete Fourier matrix and the nonzero entries of the source element $\hat{u}$ are drawn from a uniform sampling of $[-0.5, 0.5]$.

As we can see the results are very different from the results observed in our previous experimental setup. Now OMP and WOMP yield the best reconstruction quality based on the 10% relative error threshold. The ℓ^1 minimization results remain good and are still very well approximated by the GISS algorithm. CoSaMP yields similar reconstruction quality and is closely followed by the GISS$^{1.05}$ relaxation. It is interesting to see that the reconstruction accuracy suffers much more drastically when choosing an even larger ρ for the GISS$^{\rho}$ method: The GISS$^{1.2}$ and HTP algorithms show the worst reconstruction quality in this test.

- For the next experiment the sensing matrix A is generated from a Gaussian distribution with mean 0.5 and variance 1, and the nonzero entries of the source element $\hat{u}$ are again drawn from a uniform sampling of $[-0.5, 0.5]$. Figure 8 illustrates the results of this test case.

  We can see that this change of the experimental setup yet changes the results again. While the HTP algorithm changes from being the worst method to being the best method, the CoSaMP algorithm, previously yielding good results, now turns out to show the worst reconstruction accuracy. Constantly good reconstruction quality is achieved by the ℓ^1 minimization approach, which, once more, is well approximated by the GISS algorithm. OMP yields results with an accuracy similar to ℓ^1 minimization. We can see that choosing more greedy methods, i.e. relaxing OMP to WOMP or

GISS to GISS$^\rho$ with $\rho > 1$, leads to a significant increase in reconstruction error.

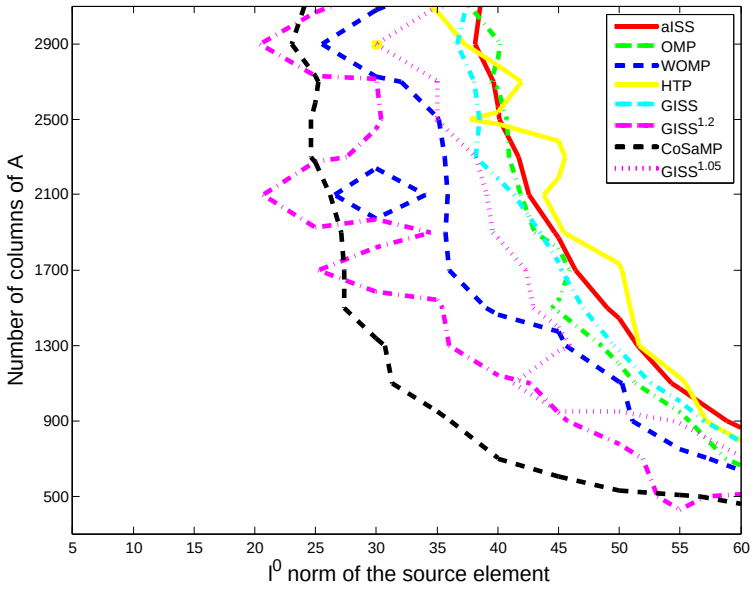

FIGURE 8. Contours of all methods at which the relative error to the true solution exceeds 10%. The entries in the sensing matrix A are generated from a Gaussian distribution with mean 0.5 and the nonzero entries of the source element $\hat{u}$ are drawn from a uniform sampling of $[-0.5,\ 0.5]$.

- In our third experiment, we generate A as a random matrix with orthonormal rows (more precisely as the matrix with orthonormal rows arising from a QR-decomposition of a matrix with values drawn from a standard Gaussian distribution). The nonzero entries of the source element $\hat{u}$ are drawn from a uniform sampling of $[-1,\ 1]$. The results in Figure 9 indicate yet a very different behavior of algorithms. From matrices with more than 700 columns the CoSaMP algorithm is by far the best choice, significantly outperforming even the ℓ^1 minimization approach. The latter shows the second best reconstruction quality with the GISS algorithm replicating the ℓ^1 minimization results almost exactly. Opposed to the previous two cases one loses only very little when using GISS$^\rho$ with $\rho = 1.05$ and even with $\rho = 1.2$. OMP and WOMP show much less favorable behavior with the interesting observation that WOMP succeeds more often than OMP in this test. Completely contrary to the previous test case, HTP yields by far the worst reconstruction results and fails in all but the simplest cases.
- In our last experiment we generated A to be sparse itself with only 10% of the entries in A being nonzero and drawn from a Gaussian distribution, and the nonzero entries of the source element $\hat{u}$ to also be drawn from a Gaussian distribution. Figure 10 shows the corresponding contour plots. In this last

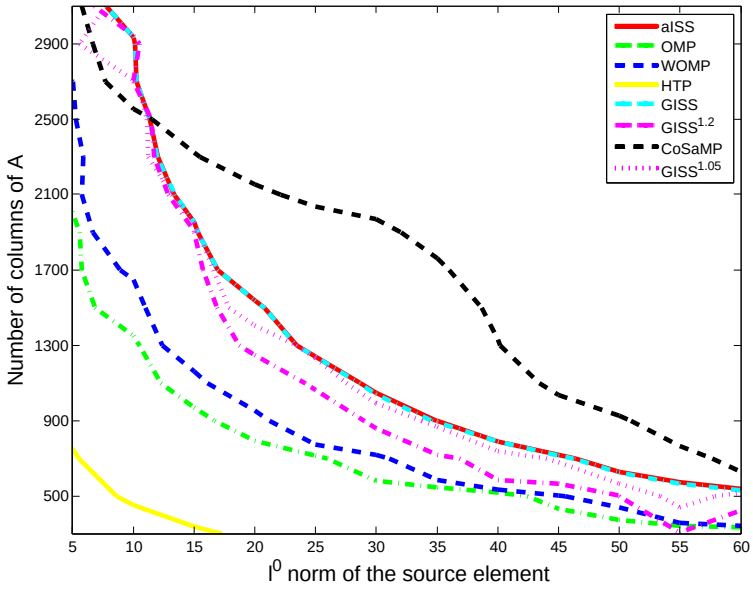

FIGURE 9. Contours of all methods at which the relative error to the true solution exceeds 10%. The matrix A has orthonormal rows and the nonzero entries of the source element $\hat{u}$ are drawn from a uniform sampling of $[-1,\ 1]$.

experiment we can see the typical behavior of ℓ^1 minimization being among the best methods, closely approximated by the GISS algorithm. CoSaMP is very close to and in some cases even beating ℓ^1 minimization. Being more greedy in the GISS algorithm in this case again costs quite a lot of accuracy. OMP yields clearly worse results than CoSaMP, ℓ^1 minimization and the GISS method, and for in this experiment we see the expected behavior of WOMP succeeding less often. HTP did not meet the threshold of an average relative error under 10% for any of the test cases and hence has to be considered as the worst method for this kind of experiment, too.

In summary we can say that the type of sensing matrix and the type of nonzero entries of the true sparse solution heavily influences the performance of the algorithm. Particularly, HTP and CoSaMP can be among the best methods as well as the worst methods depending on the particular setup. In all test and in comparison to all methods, ℓ^1 minimization stably gave very good results and was always well approximated by the GISS algorithm, such that we can say that we proposed a method suitable for stable fast sparse reconstruction. It often compared favorable to OMP in our experiments. Being more greedy and using GISS$^\rho$ with $\rho > 1$ leads to a significant increase in algorithm speed but can - depending on the experimental setup - lead to significantly worse recovery properties such that the parameter ρ has to be chosen carefully to balance the trade-off between accuracy and speed.

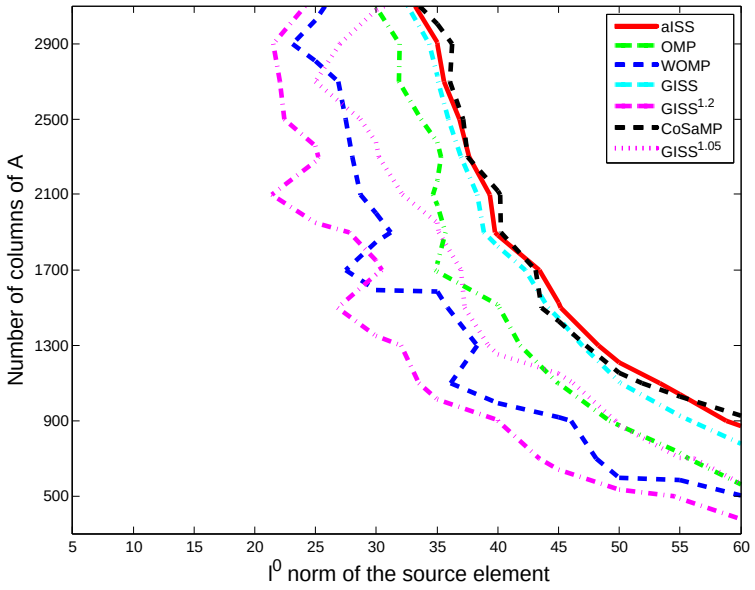

FIGURE 10. Contours of all methods at which the relative error to the true solution exceeds 10%. The matrix A is sparse with only 10% of its entries being nonzero and drawn from a Gaussian distribution. The nonzero entries of the source element $\hat{u}$ were also be drawn from a Gaussian distribution. The contour of HTP is not shown in this plot since it did not reach the desired accuracy for any of the test cases.

5. Conclusions

We proposed a new method, GISS$^\rho$, which is a greedy sparse recovery method that approximates ℓ^1 minimization more closely than OMP and WOMP. We analyzed the algorithms convergence, showed that it yields exact reconstruction for a given ERC condition, and derived very simple criteria to check if the GISS$^\rho$ solution is an ℓ^1 minimizing solution. Our numerical experiments indicate that GISS$^\rho$ indeed inherits some desirable properties from the aISS flow and seems to outperform OMP in terms of the sparsity of the recovered solutions for $\rho = 1$. Choosing a $\rho > 1$ can results in a significant speed up, leading to a method more than 20 times faster than the aISS algorithm, but has to be chose carefully to balance speed and accuracy. We compared our method to OMP, WOMP, HTP, and CoSaMP in five different numerical experiments and found that GISS always approximates the results of ℓ^1 minimization closely, which we found to be very good and - equally importantly - stable in comparison to the results of other greedy algorithms. Generally, we expect GISS and GISS$^\rho$ to work particularly well in those instances where ℓ^1 minimization works well.

In future research we will consider very high dimensional problems motivated by sparsity related problems in image processing and compare the GISS$^\rho$ results

with classical ℓ^1 minimization. Furthermore, we will try to extend this inverse scale space related recovery idea to other types of problems. The key to the success of the GISS method was to obtain a dual variable p which 'almost' is an ℓ^1 subgradient to the recovered solution. It will be worth investigating if this idea can be adapted to other types of greedy recovery methods, maybe even for greedy approximations of minimization problems involving regularizers different from the ℓ^1 norm.

References

1. I. Bayram and I.W. Selesnick, *A subband adaptive iterative shrinkage/thresholding algorithm*, IEEE Trans. on Signal Processing **58** (2010), no. 3, 1131 –1143.
2. A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci **2** (2009), 183–202.
3. S. Becker, CoSaMP and OMP for sparse recovery. Matlab Central File Exchange, 08/01/11 (Updated 04/20/12).
4. T. Blumensath and M.E. Davies, *Iterative hard thresholding for compressed sensing*, Appl. Comp. Harm. Anal **27** (2009), no. 3, 265–274.
5. M. Burger, G. Gilboa, S. Osher, and J. Xu, *Nonlinear inverse scale space methods*, Comm. Math. Sci. **4** (2006), 179–212.
6. M. Burger, M. Moeller, M. Benning, and S. Osher, *An adaptive inverse scale space method for compressed sensing*, Math. Comp. **82** (2013), 269–299.
7. J. Cai, S. Osher, and Z. Shen, *Convergence of the linearized Bregman iteration for ℓ_1-norm minimization*, Math. Comp. **78** (2009), 2127–2136.
8. ———, *Linearized Bregman iterations for compressed sensing*, Math. Comp. **78** (2009), 1515–1536.
9. E.J. Candes and T. Tao, *Near-optimal signal recovery from random projections: universal encoding strategies*, IEEE Trans. Inform. Theory **52** (2004), 5406–5425.
10. ———, *Decoding by linear programming*, IEEE Trans. on Information Theory **51** (2005), no. 12, 4203–4215.
11. W. Dai and O. Milenkovic, *Subspace pursuit for compressive sensing signal reconstruction*, IEEE Trans. on Information Theory **55** (2009), no. 5, 2230 –2249.
12. D.L. Donoho, *Compressed sensing*, IEEE Trans. Inform. Theory **52** (2006), 1289–1306.
13. D.L. Donoho and M. Elad, *Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ^1 minimization*, Proc. Natl. Acad. Sci. USA **100** (2003), 2197–2202.
14. D.L. Donoho, A. Maleki, and A. Montanari, *Message passing algorithms for compressed sensing*, Proceedings of the national academy of sciences **106** (2009), no. 45, 18914–18919.
15. E. Esser, *Applications of Lagrangian-based alternating direction methods and connections to split Bregman*, Tech. report, 2009, UCLA CAM Report [09-31].
16. S. Foucart, *Hard thresholding pursuit: an algorithm for compressive sensing*, SIAM J. on Numerical Analysis **49** (2011), no. 6, 2543–2563.
17. ———, *Stability and robustness of weak orthogonal matching pursuits*, Springer Proceedings in Mathematics and Statistics, vol. 25, 2013, pp. 395–405.
18. E. T. Hale, W. Yin, and Y. Zhang, *A fixed-point continuation method for ℓ_1-regularized minimization with applications to compressed sensing*, SIAM J. Optim. **19** (2008), no. 3, 1107–1130.
19. P. Jain, A. Tewari, and I.S. Dhillon, *Orthogonal matching pursuit with replacement*, Tech. Report arXiv:1106.2774, 2011.
20. A. Maleki, *Coherence analysis of iterative thresholding algorithms*, Proceedings of the 47th annual Allerton conference on Communication, control, and computing, IEEE Press, 2009, pp. 236–243.
21. A. Maleki and D.L. Donoho, *Optimally tuned iterative reconstruction algorithms for compressed sensing*, IEEE J. of Selected Topics in Sig. Processing **4** (2010), no. 2, 330–341.
22. S.G. Mallat and Z. Zhang, *Matching pursuits with time-frequency dictionaries*, IEEE Trans. on Sig. Processing **12** (1993), 3397–3415.
23. M. Moeller and M. Burger, *Multiscale methods for polyhedral regularizations*, SIAM J. on Optimization **23**, no. 3, 1424–1456.

24. D. Needell and J. A. Tropp, *CoSaMP: Iterative signal recovery from incomplete and inaccurate samples*, Applied and Computational Harmonic Analysis **26** (2009), 301–321.
25. S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, *An iterative regularization method for total variation-based image restoration*, SIAM Multiscale Model. Simul. **4** (2005), 460–489.
26. S. Osher, Y. Mao, B. Dong, and W. Yin, *Fast linearized Bregman iteration for compressive sensing and sparse denoising*, Commun. Math. Sci. **8** (2010), 93–111.
27. F. Parvaresh, H. Vikalo, S. Misra, and B. Hassibi, *Recovering sparse signals using sparse measurement matrices in compressed DNA microarrays*, **2** (2008), 275 – 285.
28. Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, *Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition*, Proceedings of the 27 th Annual Asilomar Conference on Signals, Systems, and Computers, 1993, pp. 40–44.
29. H. Rauhut, *Compressive sensing and structured random matrices*, Theoretical Foundations and Numerical Methods for Sparse Recovery (M. Fornasier, ed.), Radon Series Comp. Appl. Math., vol. 9, deGruyter, 2010, pp. 1–92.
30. J.A. Tropp, *Greed is good: Algorithmic results for sparse approximation*, IEEE Trans. Inform. Theory **50** (2004), 2231–2242.
31. J.A. Tropp and A.C. Gilbert, *Signal recovery from random measurements via orthogonal matching pursuit*, IEEE Trans. Inf. Theory **53** (2007), 4655–4666.
32. Joel A. Tropp, *Algorithms for simultaneous sparse approximation: part II: Convex relaxation*, Signal Process. **86** (2006), no. 3, 589–602.
33. Y. Yang, M. Moeller, and S. Osher, *A dual split Bregman method for fast ℓ^1 minimization*, Math. Comp. **82** (2013), 2061–2085.
34. Z. Yang, C. Zhang, J. Deng, and W. Lu, *Orthonormal expansion ℓ^1-minimization algorithms for compressed sensing*, IEEE Trans. on Sig. Processing **59** (2011), no. 12, 6285–6290.
35. W. Yin, S. Osher, D. Goldfarb, and J. Darbon, *Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing*, SIAM J. Imaging Sci. **1** (2008), 143–168.
36. X. Zhang, M. Burger, and S. Osher, *A unified primal-dual algorithm framework based on Bregman iteration*, J. of Scientific Computing **46** (2011), no. 1.

Institute for Computational and Applied Mathematics, University of Münster, Germany

Current address: Department of Mathematics, Technische Universität München, Boltzmannstrasse 3, 85748 Garching, Germany

E-mail address: m.moeller@gmx.net

Department of Mathematics, MOE-LSC and Institute of natural sciences, Shanghai Jiao Tong University, No. 800, Dongchuan Road, Shanghai 200240, P. R. China

E-mail address: xqzhang@sjtu.edu.cn