Information and Inference: A Journal of the IMA (2015) Page 1 of 40
doi:10.1093/imaiai/drn000

CGIHT: Conjugate Gradient Iterative Hard Thresholding for Compressed Sensing and Matrix Completion

JEFFREY D. BLANCHARD*,
Grinnell College, Department of Mathematics and Statistics, Grinnell, IA 50112
*Corresponding author: jeff@math.grinnell.edu
JARED TANNER *and* KE WEI
University of Oxford, Mathematics Institute, Andrew Wiles Building, Radcliffe Observatory Quarter, Woodstock Road, Oxford OX2 6GG
tanner@maths.ox.ac.uk and wei@maths.ox.ac.uk

To Eitan Tadmor in honour of his 60th birthday with thanks for his mentorship and support.

[Received on 2 October 2015]

We introduce the Conjugate Gradient Iterative Hard Thresholding (CGIHT) family of algorithms for the efficient solution of constrained underdetermined linear systems of equations arising in compressed sensing, row sparse approximation, and matrix completion. CGIHT is designed to balance the low per iteration complexity of simple hard thresholding algorithms with the fast asymptotic convergence rate of employing the conjugate gradient method. We establish provable recovery guarantees and stability to noise for variants of CGIHT with sufficient conditions in terms of the restricted isometry constants of the sensing operators. Extensive empirical performance comparisons establish significant computational advantages for CGIHT both in terms of the size of problems which can be accurately approximated and in terms of overall computation time.

Keywords:

Compressed Sensing, Matrix Completion, Row Sparse Approximation, Multiple Measurement Vectors, Hard Thresholding Algorithm, Conjugate Gradient Iterative Hard Thresholding, Restricted Isometry Constants
2010 Math Subject Classification: 41A99, 49N45, 62F30, 65F10, 65F22, 68P30, 90C26, 94A20

1. Introduction

Methods for more efficient data acquisition have received renewed interest in the information community. This greater efficiency is typically achieved by using a low dimensional model of high dimensional data and exploiting the simplicity of the underlying low dimensional model. The prevalence of low dimensional approximations is fundamental to modern compression algorithms and is a cornerstone to efficient analysis and sharing of big data. Two notable examples of techniques which allow for more efficient data acquisition through the existence of such low dimensional models are: compressed sensing where data known to be compressible in a given basis can be measured at a rate proportional to the desired compression rate rather than the high dimension containing the data [20, 32], and low rank matrix completion [19, 74] where suitable matrices known to be approximately low rank can be determined from a number of its entries which is proportional to the number of degrees of freedom of the low rank approximation rather than the full number of entries of the matrix. Particularly remarkable to these techniques is that the information can be acquired from linear measurements which do not

© The author 2015. Published by Oxford University Press on behalf of the Institute of Mathematics and its Applications. All rights reserved.

resort to learning from prior measurements. Compressed sensing and low rank matrix completion can be cast most directly as non-convex optimization problems which seek to find a simple solution to an underdetermined system of linear equations.

In its simplest form, compressed sensing [21, 22, 32] considers the recovery of a *sparse* vector with few nonzeros from a number of linear measurements that is less than the length of the vector. Let x be a vector with at most k nonzero entries, denoted $\|x\|_0 \leqslant k$ where $\|x\|_0$ counts the number of nonzero entries in x. Let the sensing operator $A \in \mathbb{R}^{m\times n}$ be a matrix from which we obtain the $m < n$ measurements $y = Ax$. Then the compressed sensing problem attempts to recover the vector with no more than k nonzeros that fits the measurements as well as possible:

$$\min_{z\in\mathbb{R}^n} \|y - Az\|_2 \quad \text{subject to} \quad \|z\|_0 \leqslant k. \tag{1.1}$$

A variant of compressed sensing considers using a single measurement matrix A to measure multiple sparse vectors whose locations of nonzero values primarily coincide and is referred to as compressed sensing for multiple measurement vectors or joint sparsity [61, 63, 78, 80, 81]. Let the r measured vectors be organized as the columns of a *row sparse* matrix $X \in \mathbb{R}^{n\times r}$ with at most k rows containing nonzero entries, i.e. $\|X\|_{R0} \leqslant k$ where $\|X\|_{R0}$ counts the number of rows in X with at least one nonzero entry. As in the compressed sensing problem, let the sensing operator $A \in \mathbb{R}^{m\times n}$ be a matrix from which the measurements $Y = AX$ are obtained. The compressed sensing problem for row-sparse matrices attempts to recover the matrix with no more than k nonzero rows that fits the measurements as well as possible:

$$\min_{Z\in\mathbb{R}^{n\times r}} \|Y - AZ\|_2 \quad \text{subject to} \quad \|Z\|_{R0} \leqslant k. \tag{1.2}$$

Approximation of row-sparse matrices can be viewed as an intermediate case between traditional compressed sensing of a single vector, recovering (1.1) when $r = 1$, and matrix completion. In row-sparse matrix approximation the matrix X is assumed to have an r–dimensional image space with at most k nonzeros, whereas in matrix completion the image space of X is assumed to be r dimensional but in an unknown subspace which is determined from the leading r left singular vectors of X.

In matrix completion, rather than the sparsity assumption of compressed sensing, the observed matrix is assumed to be low rank. Let $X \in \mathbb{R}^{m\times n}$ and assume $\operatorname{rank}(X) = r < \min(m,n)$. Let the sensing operator $\mathscr{A}(\cdot) : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ be a linear map consisting of p matrices $A_i \in \mathbb{R}^{m\times n}$ for $i = 1,2,\ldots,p$. We take $p < mn$ linear measurements of the low rank matrix X via $y = \mathscr{A}(X)$ where $y_i = \operatorname{trace}(A_i^* X) = \langle A_i, X\rangle_F$. The matrix completion problem attempts to recover the matrix of rank no more than r that fits the measurements as well as possible:

$$\min_{Z\in\mathbb{R}^{m\times n}} \|y - \mathscr{A}(Z)\|_2 \ \text{ subject to } \ \operatorname{rank}(Z) \leqslant r. \tag{1.3}$$

When the set of matrices $\{A_i : 1 \leqslant i \leqslant p\}$ are each composed of a single nonzero entry of value 1 in position s_i, t_i, the linear measurement from the Frobenius inner product simply returns the entry $\langle A_i, X\rangle_F = X_{s_i,t_i}$. This is referred to as *entry sensing*. On the other hand, *dense sensing* occurs when the set $\{A_i : 1 \leqslant i \leqslant p\}$ contains dense matrices and the full Frobenius inner product, $\langle A_i, X\rangle_F := \sum_{j=1}^m \sum_{\ell=1}^n A_i(j,\ell)X(j,\ell)$, is required for each measurement.

Though each of questions (1.1)-(1.3) are known to be NP hard for general sensing operators [68], a considerable theory has been developed establishing the existence of both sensing operators and recovery algorithms which produce arbitrarily accurate approximations to these questions. These advances are particularly striking in terms of the favorable features of the sensing operators and the recovery

algorithms. Suitable sensing operators are ubiquitous and include matrices with highly advantageous properties such as being sparse or having fast and structured matrix products [2, 7, 23, 73, 74]. Furthermore, many algorithms with polynomial computational complexity have theoretical guarantees of successfully recovering the measured data even when the number of measurements is no more than a constant multiple of the degrees of freedom in the object class. See Sec. 2.3 for a partial review with surveys and in depth discussions of these advances and their applications given in [17, 43, 48, 82].

The rapid development of algorithms for the computationally efficient solution to (1.1)-(1.3) has produced numerous effective algorithms with varying sophistication and performance characteristics. One of the most widely studied class of algorithms is to reformulate the nonconvex problems (1.1)-(1.3) by their nearest convex relaxation, and then to use algorithms for the associated standard class of convex optimization problems: linear, quadratic, and semidefinite programming. This approach has been shown to be highly efficient and amenable to a detailed analysis allowing the precise determination of when these classes of algorithms do, or do not, recover the solution to (1.1)-(1.3), [1, 31, 41, 42, 85]. In this manuscript we consider an alternative class of algorithms that attempt to directly solve the original nonconvex questions (1.1)-(1.3) through alternating projection [62] where an approximate solution is iteratively updated by decreasing the quadratic objective along a descent direction, while allowing the update to violate the sparsity or low rank constraint, followed by projection onto the objective constraint of sparse or low rank matrices. The projection step is known as hard thresholding, and algorithms with this alternating projection construction are referred to as *hard thresholding algorithms*. Specifically we present new algorithms that balance the benefits of low per iteration complexity with fast asymptotic convergence rates. The newly proposed algorithms are proven to have uniform recovery guarantees analogous to other hard thresholding algorithms, and to nearly uniformly have superior average case runtime complexity.

2. Main Contributions: CGIHT

We present new algorithms, for the solution to (1.1)-(1.3), which we refer to collectively as Conjugate Gradient Iterative Hard Thresholding (CGIHT). CGIHT is designed to balance the advantage of the low per iteration complexity of methods based on steepest descent with the fast asymptotic convergence rate of methods employing the conjugate gradient method to minimize the quadratic objective in (1.1)-(1.3) when the support set or subspace is held fixed. CGIHT is observed to be able to solve (1.1)-(1.3) to any fixed accuracy in less time than existing algorithms. Moreover, despite the added complexity of CGIHT making explicit use of past update directions, there are variants of CGIHT with provable recovery guarantees analogous to those of other hard thresholding algorithms. The variants of CGIHT for compressed sensing (1.1)-(1.2) and matrix completion (1.3) are stated in Sec. 2.1 and Sec. 2.2 respectively. CGIHT is put in context with other hard thresholding algorithms in Sec. 2.3. An in-depth empirical investigation of CGIHT follows in Sec. 3. The manuscript concludes with the proof of the recovery theorems for CGIHT in Sec. 4.

2.1 *CGIHT for compressed sensing* (1.1)-(1.2)

We begin our presentation of *CGIHT for compressed sensing* with its simplest variant, Alg. 1. In each iteration of CGIHT the current estimate x_{l-1} is updated along the direction p_{l-1}, using a stepsize α_{l-1}, followed by hard thresholding onto the space of vectors with at most k nonzeros. Computationally the hard thresholding operator is composed of two parts. First the principal support of the approximation is identified, then the object is projected onto the principal support. The principal support identification

of cardinality k, denoted here in pseudo-code by $\texttt{PrincipalSupport}_k(\cdot)$, is arrived at by sorting (or computing order statistics) of the ℓ^2 norm of the rows. Projecting onto the principal support set T, denoted here in pseudo-code by $\text{Proj}_T(\cdot)$, follows by setting to zero all entries not on the (row) support set T. The search direction p_{l-1} is selected to be exactly the residual r_0 in iteration $l = 1$, and is otherwise selected to be the residual r_{l-1} projected to be *conjugate orthogonal* to the past search direction p_{l-2} when restricted to the current estimate of the support set T_{l-1}, i.e. $\langle A\text{Proj}_{T_{l-1}}(p_{l-1}), A\text{Proj}_{T_{l-1}}(p_{l-2})\rangle = 0$. This procedure is analogous to the conjugate gradient method. For a square system with $k = m = n$, Alg. 1 will exactly execute the conjugate gradient method [54]. For the compressed sensing regime with $k < m < n$, Alg. 1 lacks some of the important properties of the standard conjugate gradient method. In particular, the search directions do not in general remain orthogonal, except in the simplest case where the support set never changes [72, 87]. Lacking the orthogonalization property over all past support sets precludes the use of the most efficient formulae for computing conjugate gradient stepsizes α_{l-1} and orthogonalization weights β_{l-1}, resulting in one additional matrix vector product per iteration. Additionally, this simplest variant of CGIHT lacks a proof of convergence to the measured k sparse vector. Despite these shortcomings, Alg. 1 is often observed to have superior performance to other variants of CGIHT in terms of both the problem size (k,m,n) it is able to recover and being able to recover the measured vector in less time than other variants of CGIHT.

Algorithm 1 CGIHT for compressed sensing

Initialization: Set $T_{-1} = \{\}$, $p_{-1} = 0$, $w_{-1} = A^*y$, $T_0 = \texttt{PrincipalSupport}_k(|w_{-1}|)$, $x_0 = \text{Proj}_{T_0}(w_{-1})$, and $l = 1$.

Iteration: During iteration l, **do**

1: $r_{l-1} = A^*(y - Ax_{l-1})$ (compute the residual)

2: if $l = 1$,

$\beta_{l-1} = 0$ (set orthogonalization weight)

else

$\beta_{l-1} = -\dfrac{\left\langle A\text{Proj}_{T_{l-1}}(r_{l-1}), A\text{Proj}_{T_{l-1}}(p_{l-2})\right\rangle}{\left\langle A\text{Proj}_{T_{l-1}}(p_{l-2}), A\text{Proj}_{T_{l-1}}(p_{l-2})\right\rangle}$ (compute orthogonalization weight)

3: $p_{l-1} = r_{l-1} + \beta_{l-1}p_{l-2}$ (define the search direction)

4: $\alpha_{l-1} = \dfrac{\left\langle \text{Proj}_{T_{l-1}}(r_{l-1}), \text{Proj}_{T_{l-1}}(p_{l-1})\right\rangle}{\left\langle A\text{Proj}_{T_{l-1}}(p_{l-1}), A\text{Proj}_{T_{l-1}}(p_{l-1})\right\rangle}$ (optimal stepsize if $T_{l-1} = T_*$)

5: $w_{l-1} = x_{l-1} + \alpha_{l-1}p_{l-1}$ (update along search direction)

6: $T_l = \texttt{PrincipalSupport}_k(|w_{l-1}|)$ (support set of k largest entries)

7: $x_l = \text{Proj}_{T_l}(w_{l-1})$ (restriction to support set T_l)

To recover the conjugate gradient property that past search directions maintain orthogonality over a sequence of iterates acting on the same support set requires restarting the conjugate gradient method when the support T_{l-1} changes. The remaining two variants of CGIHT for compressed sensing are designed with such restarting conditions. Moreover, we are able to provide optimal order recovery proofs, similar to other hard thresholding algorithms for (1.1), in terms of the restricted isometry constants (RICs) of the the measurement matrix; see Def. 2.1.

DEFINITION 2.1 (Asymmetric restricted isometry constants for sparse vectors) For an $m \times n$ matrix A, the *asymmetric restricted isometry constants* $L(k,m,n)$ and $U(k,m,n)$ for k sparse vectors are defined

as:

$$L_k = L(k,m,n) := \min_{c \geqslant 0} c \text{ subject to } (1-c) \leqslant \|Ax\|_2^2/\|x\|_2^2, \text{ for all } \|x\|_0 \leqslant k \tag{2.1}$$

$$U_k = U(k,m,n) := \min_{c \geqslant 0} c \text{ subject to } (1+c) \geqslant \|Ax\|_2^2/\|x\|_2^2, \text{ for all } \|x\|_0 \leqslant k. \tag{2.2}$$

The first of these variants of CGIHT with provably optimal order recovery guarantees is referred to as *CGIHT restarted*, Alg. 2. The first iteration, $l = 1$, of Alg. 1 and Alg. 2 are identical. Subsequent iterations differ in their choice of search directions. Alg. 2 uses the residual r_{l-1} as its search direction in any iteration where the support set differs from that of the prior iteration, $T_{l-1} \neq T_{l-2}$. However, in iterations where the support set T_{l-1} is the same as at the prior iteration, the search direction p_{l-1} is selected to be the residual r_{l-1} projected to be conjugate orthogonal to the past search direction p_{l-2} when restricted to the support set T_{l-1}. Starting each instance of the orthogonalization with the first search direction having been the residual recovers the orthogonalization of all search directions over sequences of iterations where the support set remains unchanged; that is, $\left\langle A\text{Proj}_{T_{l-1}}(p_{l-1}), A\text{Proj}_{T_{l-1}}(p_{l-j}) \right\rangle = 0$ for j from 2 increasing until the first value of j such that $T_{l-j} \neq T_{l-1}$. Recovering this orthogonalization property allows the use of efficient formulae for computing the stepsize α_{l-1} and orthogonalization weight β_{l-1} which reduces the per iteration complexity by one matrix vector product.

Algorithm 2 CGIHT restarted for compressed sensing

Initialization: Set $T_{-1} = \{\}$, $p_{-1} = 0$, $w_{-1} = A^*y$, $T_0 = \texttt{PrincipalSupport}_k(|w_{-1}|)$, $x_0 = \text{Proj}_{T_0}(w_{-1})$, and $l = 1$.

Iteration: During iteration l, **do**

1: $r_{l-1} = A^*(y - Ax_{l-1})$ (compute the residual)

2: if $T_{l-1} \neq T_{l-2}$

$\beta_{l-1} = 0$ (set orthogonalization weight)

else

$\beta_{l-1} = \frac{\|\text{Proj}_{T_{l-1}}(r_{l-1})\|^2}{\|\text{Proj}_{T_{l-1}}(r_{l-2})\|^2}$ (compute orthogonalization weight)

3: $p_{l-1} = r_{l-1} + \beta_{l-1} p_{l-2}$ (define the search direction)

4: $\alpha_{l-1} = \frac{\|\text{Proj}_{T_{l-1}}(r_{l-1})\|^2}{\|A\text{Proj}_{T_{l-1}}(p_{l-1})\|^2}$ (optimal stepsize if $T_{l-1} = T_*$)

5: $w_{l-1} = x_{l-1} + \alpha_{l-1} p_{l-1}$ (update along search direction)

6: $T_l = \texttt{PrincipalSupport}_k(|w_{l-1}|)$ (support set of k largest entries)

7: $x_l = \text{Proj}_{T_l}(w_{l-1})$ (restriction to support set T_l)

The recovery guarantee for Alg. 2, Thm. 2.2, considers the typical signal model $y = Ax + e$ where x has at most k nonzeros. In this model x can be viewed as the k sparse vector which minimizes $\|y - Ax\|_2$ and e as the discrepancy between y and Ax. Alternatively x can be viewed as a k sparse vector measured by A and that y is contaminated by additive noise; though, in this perspective, Thm. 2.2 does not consider any particular model for e.

THEOREM 2.2 (Recovery guarantee for CGIHT restarted for compressed sensing, Alg. 2.) Let A be an $m \times n$ matrix with $m < n$, and $y = Ax + e$ for any x with at most k nonzeros. Define the following

constants in terms of the RICs of A:

$$\tau_1 = 2\frac{U_{3k}+L_{3k}}{1-L_k} + \frac{(1+U_k)(U_k+L_k)}{(1-L_k)^2}, \qquad \tau_2 = \frac{2(1-L_{3k})(1+U_k)(U_k+L_k)}{(1-L_k)^3},$$
$$\mu = \frac{1}{2}\left(\tau_1 + \sqrt{\tau_1^2+4\tau_2}\right), \qquad \xi = 2\frac{(1+U_{2k})^{1/2}}{1-L_k}. \tag{2.3}$$

If the RICs of A satisfy

$$\frac{(L_{3k}+U_{3k})(5-2L_k+3U_k)}{(1-L_k)^2} < 1, \tag{2.4}$$

then $\mu < 1$ and the iterates of Alg. 2 satisfy

$$\|x_l - x\|_2 \leqslant \mu^l \|x_0 - x\| + \frac{\xi}{1-\mu}\|e\|. \tag{2.5}$$

The proof of Thm. 2.2 is given in Sec. 4.1. Thm. 2.2 implies that if $e = 0$, the correct support set of x will be identified after logarithmically many iterations [8]. Moreover, once the correct support set is identified, CGIHT is simply the conjugate gradient method applied to the overdetermined system of equations using the submatrix $A_{\mathrm{supp}(x)}$ containing only the columns indexed by the set $\mathrm{supp}(x)$; thus the asymptotic convergence rate is linear with a rate given by $\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$ where $\kappa = \mathrm{cond}(A^*_{\mathrm{supp}(x)}A_{\mathrm{supp}(x)})$. Note that this asymptotic convergence rate is much smaller than the rate μ given in Thm. 2.2; the rate μ defined in Thm. 2.2 indicates instead the minimum rate of ℓ^2 error contraction per iteration, including iterations where the support set is incorrect. Algorithm 2 can also be used to obtain the approximate solution of (1.2) by replacing the vector x with the row-sparse matrix X, $\texttt{PrincipalSupport}_k(\cdot)$ determining k rows of largest ℓ_2 norm, $\mathrm{Proj}_T(\cdot)$ setting to zero all but the entries in rows indexed by T, and all norms being the Frobenius norm. Theorem 2.2 similarly holds for the solution of (1.2), although this worst-case uniform guarantee fails to illustrate the improved performance exhibited for X with rank greater than 1.

As an alternative to the support set based restarting condition of Alg. 2, the conjugate gradient method can be restarted based on a measure of the relative residual for a current support set. *CGIHT projected*, Alg. 3, corresponds to nonlinear restarted conjugate gradient where restarting occurs when $\left\| r_{l-1} - \mathrm{Proj}_{T_{l-1}}(p_{l-1})\right\| / \|\mathrm{Proj}_{T_{l-1}}(r_{l-1})\|$ is sufficiently large. This restarting condition corresponds to the fraction of the current residual aligned with the current search direction, relative to the magnitude of the residual on the current support set. Unlike Alg. 2 which has no tuning parameters, CGIHT projected has a restarting parameter θ controlling the computational effort used to decrease the residual on a given support set.

THEOREM 2.3 (Recovery guarantee for CGIHT projected for compressed sensing, Alg. 3.) Let A be an $m \times n$ matrix with $m < n$, and $y = Ax + e$ for any x with at most k nonzeros. Define the following constants in terms of the RICs of A and the restarting parameter c:

$$\mu = 2(1+c)\frac{U_{3k}+L_{3k}}{1-L_k}, \quad \theta_0 = c\left(\frac{U_{3k}+L_{3k}}{1+U_{2k}}\right), \quad \text{and} \quad \xi = 2(1+\theta_0)\frac{(1+U_{2k})^{1/2}}{1-L_k}.$$

Then provided $\mu < 1$, the iterates of Alg. 3 with restarting condition $\theta < \theta_0$ satisfy

$$\|x_l - x\| \leqslant \mu^l \|x_0 - x\| + \frac{\xi}{1-\mu}\|e\|. \tag{2.6}$$

Algorithm 3 CGIHT projected for compressed sensing

Initialization: Set $T_{-1} = \{\}$, $w_{-1} = A^*y$, $T_0 = \texttt{PrincipalSupport}_k(|w_{-1}|)$, $x_0 = \text{Proj}_{T_0}(w_{-1})$, $r_0 = A^*(y - Ax_0)$, $p_0 = r_0$, Restart_flag $= 1$, set restart parameter θ, and $l = 1$.

Iteration: During iteration l, **do**

1: if $\frac{\left\|r_{l-1} - \text{Proj}_{T_{l-1}}(p_{l-1})\right\|}{\|\text{Proj}_{T_{l-1}}(r_{l-1})\|} > \theta$

$\quad$ Restart_flag $= 1$ (set restart flag)

$\quad$ $\alpha_{l-1} = \frac{\left\|\text{Proj}_{T_{l-1}}(r_{l-1})\right\|^2}{\left\|A\text{Proj}_{T_{l-1}}(r_{l-1})\right\|^2}$ (optimal stepsize if $T_{l-1} = T_*$)

$\quad$ $w_{l-1} = x_{l-1} + \alpha_{l-1} r_{l-1}$ (update along unprojected search direction)

else

$\quad$ Restart_flag $= 0$ (set restart flag)

$\quad$ $\alpha_{l-1} = \frac{\left\|\text{Proj}_{T_{l-1}}(r_{l-1})\right\|^2}{\left\|A\text{Proj}_{T_{l-1}}(p_{l-1})\right\|^2}$ (optimal stepsize if $T_{l-1} = T_*$)

$\quad$ $w_{l-1} = x_{l-1} + \alpha_{l-1}\text{Proj}_{T_{l-1}}(p_{l-1})$ (update along projected search direction)

2: $T_l = \texttt{PrincipalSupport}_k(w_{l-1})$ (support set of k largest entries)

3: $x_l = \text{Proj}_{T_l}(w_{l-1})$ (restriction to support set T_l)

4: $r_l = A^*(y - Ax_l)$ (compute the residual)

5: if Restart_flag $= 1$,

$\quad$ $p_l = r_l$ (define the unprojected search direction)

else

$\quad$ $\beta_l = \frac{\left\|\text{Proj}_{T_l}(r_l)\right\|^2}{\left\|\text{Proj}_{T_l}(r_{l-1})\right\|^2}$ (compute orthogonalization weight)

$\quad$ $p_l = r_l + \beta_l \text{Proj}_{T_l}(p_{l-1})$ (define the projected search direction)

The proof of Thm. 2.3 follows the proof of its matrix completion variant which is given in Sec. 4.2. Again, Alg. 3 has a straightforward extension to the row-sparse approximation problem (1.2) with a uniform guarantee given by Thm. 2.3.

2.2 *CGIHT for matrix completion* (1.3)

We begin our discussion of *CGIHT for matrix completion* with its simplest variant, Alg. 4, which mirrors Alg. 1. In each iteration of CGIHT the current estimate X_{l-1} is updated along the direction P_{l-1}, using a stepsize α_{l-1}, followed by hard thresholding to the matrix of rank at most r that is nearest in the Frobenius norm. Computationally the hard thresholding operator is composed of two parts. First the principal subspace is identified, then the object is projected onto the principal subspace. The principal subspace identification of rank r can be identified in terms of either its column or row space. Here we make use of the column space, using the leading r left singular vectors, denoted here in pseudo-code by $\texttt{PrincipalLeftSingularVectors}_r(\cdot)$; use of the right singular space in algorithms to solve (1.3) is discussed in [58, 77]. Projecting onto the principal subspace U, denoted here in pseudo-code by $\text{Proj}_U(\cdot)$, follows by multiplying from the left by the projection matrix UU^*, or by forming the rank r approximation by computing its singular valued decomposition (SVD), $U\Sigma V$, setting to zero all but its leading r singular values in Σ, and reforming the product of the three SVD matrices [55]. As in Alg. 1,

the search direction P_{l-1} is selected to be exactly the residual R_0 in iteration $l = 1$, and is otherwise selected to be the residual R_{l-1} projected to be conjugate orthogonal to the past search direction when restricted to the current estimate of the subspace U_{l-1}. As with Alg. 1, Alg. 4 lacks the conjugate gradient property of past search directions remaining mutually orthogonal. In fact, as the subspaces are continuously varying, there will typically be no two past subspaces U_l and U_{l-1} which will be identical, and consequently only the past two search directions will remain orthogonal. Despite lacking orthogonalization over longer sequences of iterates, Alg. 4 nearly always has superior performance to the other variant of CGIHT for matrix completion in terms of both the problem size (r,m,n,p) it is able to recover and being able to recover the measured matrix in less time than the restarted variant of CGIHT.

Algorithm 4 CGIHT for matrix completion

Initialization: Set $P_{-1} = 0$, $W_{-1} = \mathscr{A}^*(y)$,
$U_0 = \texttt{PrincipalLeftSingularVectors}_r(W_{-1})$, $X_0 = \text{Proj}_{U_0}(W_{-1})$, and $l = 1$.
Iteration: During iteration l, **do**

1: $R_{l-1} = \mathscr{A}^*\left(y - \mathscr{A}(X_{l-1})\right)$ (compute the residual)
2: if $l = 1$,
$\beta_{l-1} = 0$, (set orthogonalization weight)
else
$\beta_{l-1} = -\dfrac{\left\langle \mathscr{A}\left(\text{Proj}_{U_{l-1}}(R_{l-1})\right), \mathscr{A}\left(\text{Proj}_{U_{l-1}}(P_{l-2})\right)\right\rangle}{\left\langle \mathscr{A}\left(\text{Proj}_{U_{l-1}}(P_{l-2})\right), \mathscr{A}\left(\text{Proj}_{U_{l-1}}(P_{l-2})\right)\right\rangle}$ (compute orthogonalization weight)
3: $P_{l-1} = R_{l-1} + \beta_{l-1} P_{l-2}$ (define the search direction)
4: $\alpha_{l-1} = \dfrac{\left\langle \text{Proj}_{U_{l-1}}(R_{l-1}), \text{Proj}_{U_{l-1}}(P_{l-1})\right\rangle}{\left\langle \mathscr{A}\left(\text{Proj}_{U_{l-1}}(P_{l-1})\right), \mathscr{A}\left(\text{Proj}_{U_{l-1}}(P_{l-1})\right)\right\rangle}$ (optimal stepsize if $U_{l-1} = U_*$)
5: $W_{l-1} = X_{l-1} + \alpha_{l-1} P_{l-1}$ (steepest descent or conjugate gradient step)
6: $U_l = \texttt{PrincipalLeftSingularVectors}_r(W_{l-1})$ (first r left singular vectors)
7: $X_l = \text{Proj}_{U_l}(W_{l-1})$ (restriction to r-dimensional column space U_l)

As alluded to earlier, matrix completion (1.3) differs from (1.1)-(1.2) fundamentally in that the subspace of low rank matrices is a continuously varying manifold whereas the subspace of (row) sparsity is finite dimensional being composed of $\binom{n}{k}$ linear subspaces corresponding to the possible support sets. As a consequence, the subspaces U_{l-1} and U_{l-2} will typically never be exactly the same and there is no exact analog of Alg. 2 for matrix completion. However, the relative residual restarting condition of CGIHT projected, Alg. 3, extends to matrix completion in Alg. 5, which we refer to as *CGIHT projected for matrix completion*. Similar to Alg. 3, CGIHT projected for matrix completion corresponds to nonlinear restarted conjugate gradient where restarting occurs when $\left\| R_{l-1} - \text{Proj}_{U_{l-1}}(P_{l-1})\right\| / \|\text{Proj}_{U_{l-1}}(R_{l-1})\|$ is sufficiently large; see Alg. 5. This restarting condition corresponds to the component of the search direction P_{l-1} contained in the image space of X_{l-1} minus R_{l-1} being small in the Frobenius norm when compared with the size of the residual contained in the image space of X_{l-1}. Unlike CGIHT for matrix completion, Alg. 4, which has no tuning parameters, CGIHT projected has a restarting parameter θ controlling the computational effort used to decrease the residual on a given subspace. As stated in Thm. 2.5, CGIHT projected has a uniform recovery guarantee for low rank matrix approximation (1.3) provided the sensing operator $\mathscr{A}$ has sufficiently small RICs for low rank matrices; see Def. 2.4.

DEFINITION 2.4 For a linear map $\mathscr{A}(\cdot) : \mathbb{R}^{m\times n} \to \mathbb{R}^p$, the *restricted isometry constants* $L(r,m,n,p)$ and

Algorithm 5 CGIHT projected for matrix completion

Initialization: Set $W_{-1} = \mathscr{A}^*(y)$, $U_0 = \texttt{PrincipalLeftSingularVectors}_r(W_{-1})$, $X_0 = \text{Proj}_{U_0}(W_{-1})$, $R_0 = \mathscr{A}^*(y - \mathscr{A}(X_0))$, $P_0 = R_0$, Restart_flag $= 1$, set restart parameter θ, and $l = 1$.

Iteration: During iteration l, **do**

1: if $\frac{\left\|R_{l-1} - \text{Proj}_{U_{l-1}}(P_{l-1})\right\|}{\|\text{Proj}_{U_{l-1}}(R_{l-1})\|} > \theta$

$\quad$ Restart_flag $= 1$ (set restart flag)

$\quad$ $\alpha_{l-1} = \frac{\left\|\text{Proj}_{U_{l-1}}(R_{l-1})\right\|^2}{\left\|\mathscr{A}\left(\text{Proj}_{U_{l-1}}(R_{l-1})\right)\right\|^2}$ (optimal stepsize if $U_{l-1} = U_*$)

$\quad$ $W_{l-1} = X_{l-1} + \alpha_{l-1} R_{l-1}$ (update along unprojected search direction)

else

$\quad$ Restart_flag $= 0$ (set restart flag)

$\quad$ $\alpha_{l-1} = \frac{\left\|\text{Proj}_{U_{l-1}}(R_{l-1})\right\|^2}{\left\|\mathscr{A}\left(\text{Proj}_{U_{l-1}}(P_{l-1})\right)\right\|^2}$ (optimal stepsize if $U_{l-1} = U_*$)

$\quad$ $W_{l-1} = X_{l-1} + \alpha_{l-1}\text{Proj}_{U_{l-1}}(P_{l-1})$ (update along projected search direction)

2: $U_l = \texttt{PrincipalLeftSingularVectors}_r(W_{l-1})$ (first r left singular vectors)

3: $X_l = \text{Proj}_{U_l}(W_{l-1})$ (restriction to r-dimensional column space U_l)

4: $R_l = \mathscr{A}^*(y - \mathscr{A}(X_l))$ (compute the residual)

5: if Restart_flag $= 1$,

$\quad$ $P_l = R_l$ (define the unprojected search direction)

else

$\quad$ $\beta_l = \frac{\left\|\text{Proj}_{U_l}(R_l)\right\|^2}{\left\|\text{Proj}_{U_l}(R_{l-1})\right\|^2}$ (compute orthogonalization weight)

$\quad$ $P_l = R_l + \beta_l \text{Proj}_{U_l}(P_{l-1})$ (define the projected search direction)

$U(r,m,n,p)$ for rank r matrices are defined as:

$$L_r = L(r,m,n,p) := \min_{c \geqslant 0} c \text{ subject to } (1-c) \leqslant \|\mathscr{A}(X)\|_2^2 / \|X\|_F^2, \text{ for all } \text{rank}(X) \leqslant r \tag{2.7}$$

$$U_r = U(r,m,n,p) := \min_{c \geqslant 0} c \text{ subject to } (1+c) \geqslant \|\mathscr{A}(X)\|_2^2 / \|X\|_F^2, \text{ for all } \text{rank}(X) \leqslant r. \tag{2.8}$$

As in Thm. 2.2, Thm. 2.5 considers the model $y = \mathscr{A}(X) + e$ where $\text{rank}(X) \leqslant r$. In this model X is typically viewed as the rank r matrix which minimizes $\|y - \mathscr{A}(X)\|_2$ and e as the measurement error associated with restricting X to be at most rank r. Alternatively X can be viewed as a rank r matrix measured by $\mathscr{A}$ and that y is contaminated by additive noise; though, in this perspective, Thm. 2.5 does not consider any particular model for e.

THEOREM 2.5 (Recovery guarantee for CGIHT projected for matrix completion, Alg. 5.) Let $\mathscr{A}$ be a linear map from $\mathbb{R}^{m\times n}$ to $\mathbb{R}^p$ with $p < mn$, and $y = \mathscr{A}(X) + e$ for any X with $\text{rank}(X) \leqslant r$. Define the following constants in terms of the RICs of $\mathscr{A}$ and the restarting parameter c:

$$\mu = 2(1+c)\frac{U_{3r} + L_{3r}}{1 - L_r}, \quad \theta_0 = c\left(\frac{U_{3r} + L_{3r}}{1 + U_{2r}}\right), \quad \text{and} \quad \xi = 2(1+\theta_0)\frac{(1+U_{2r})^{1/2}}{1 - L_r}.$$

Then provided $\mu < 1$, the iterates of Alg. 5 with restarting condition $\theta < \theta_0$ satisfy

$$\|X_l - X\| \leqslant \mu^l \|X_0 - X\| + \frac{\xi}{1-\mu}\|e\|. \tag{2.9}$$

Theorem 2.5 implies that if $e = 0$, CGIHT projected for matrix completion will recover the measured rank r matrix to arbitrary accuracy. As the manifold of rank r matrices is continuously varying, Alg. 5 will never exactly identify the correct image space, and does not exactly correspond to a traditional numerical linear algebra routine for computing a low rank matrix approximation. However, in iterations where the Restart_flag $= 0$, Alg. 5 does correspond to solving for the low rank approximation with a specified image space corresponding to that of U_{l-1}.

2.3 *Discussion and prior art*

The simplest example of iterative hard thresholding algorithms for compressed sensing (1.1) is referred to simply as Iterative Hard Thresholding (IHT) [15, 49], and corresponds to Alg. 6, with the stepsize ω held constant; typically set to be 1. IHT extends naturally to row sparsity (1.2) [9] and matrix completion (1.3) where it is referred to as either IHT [59, 64] or Singular Value Projection [56]. For each of (1.1)-(1.3), the performance of IHT [37] depends heavily on the selection of the stepsize ω_l. If the stepsize is selected to be too large the method can diverge, whereas overly small values of ω_l can result in slow convergence or convergence to a local minima rather than the desired global minima [24]. Normalized IHT (NIHT) [9, 16, 77] provides a formula for adaptively computing the stepsize; see Alg. 6 for NIHT for compressed sensing (1.1) where the stepsize ω_l is selected to be optimal if $\text{supp}(x_{l-1}) = \text{supp}(x)$.

Algorithm 6 NIHT (Normalized Iterative Hard Thresholding [16])

Initialization: Set $w_{-1} = A^*y$, $T_0 = \texttt{PrincipalSupport}_k(|w_{-1}|)$, $x_0 = \text{Proj}_{T_0}(w_{-1})$, and $l = 1$.
Iteration: During iteration l, **do**

1: $r_{l-1} = A^*(y - Ax_{l-1})$ (update the residual)
2: $\omega_l = \frac{\|\text{Proj}_{T_{l-1}}(r_{l-1})\|^2}{\|A\text{Proj}_{T_{l-1}}(r_{l-1})\|^2}$ (optimal stepsize in the k-subspace T_{l-1})
3: $w_{l-1} = x_{l-1} + \omega_l r_{l-1}$ (steepest descent step)
4: $T_l = \texttt{PrincipalSupport}_k(x_l)$ (proxy to the support set)
5: $x_l = \text{Proj}_{T_l}(w_{l-1})$ (restriction to support set T_l)

Despite the simplicity of IHT and NIHT, they have numerous near optimal properties. (N)IHT has been proven, for suitable linear sensing operators, to be able to recover the solution to (1.1)-(1.2) from a number of measurements that is proportional to the (row) sparsity of the measured data [9, 16], and to recover the solution to (1.3) from within a logarithmic factor of the number of degrees of freedom of rank r matrices [77]. Moreover, for numerous measurement operators, NIHT is observed to be able to solve (1.1)-(1.3) for the same problem sizes as more sophisticated algorithms, and is often able solve (1.1)-(1.3) to moderate accuracy in less computational time than can more sophisticated variants. However, NIHT suffers from the slow asymptotic convergence rate of the steepest descent method if the sensing operator is ill conditioned when restricted to the subspace of the measured data. For example, Alg. 6 for (1.1) has an asymptotic convergence rate per iteration of $\frac{\kappa-1}{\kappa+1}$ where $\kappa = \text{cond}(A^*_{\text{supp}(x)}A_{\text{supp}(x)})$; this is in contrast to the asymptotic convergence rate of $\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$ for each variant of CGIHT for compressed sensing, Algs. 1–3.

Many of the more sophisticated hard thresholding algorithms have been designed in part to overcome this slow asymptotic convergence rate of NIHT. These more advanced algorithms typically achieve this with the inclusion of the conjugate gradient method [51, 54] to solve for the local minima of the objective in (1.1)–(1.3) while the sparse or low rank subspace is held fixed. A highly incomplete list of examples of such methods are: [28, 45, 69] for (1.1), which have been extended to (1.2) in [9] and examples for (1.3) include [29, 59, 60]. Though such methods gain the fast convergence of the conjugate gradient method, they do so at the cost of higher per iteration complexity. It is observed in [12] that when (1.1) is solved to moderate accuracy, $\|y-A\hat{x}\|_2/\|y\|_2 \approx 10^{-4}$, the disadvantage of the higher per iteration complexity causes early iterations, where the support set is typically changing, to result in an overall average computational time that is often as long or *longer* than that of NIHT. When approximate solutions are sought with $\|y-A\hat{x}\|_2/\|y\|_2 \gg 10^{-4}$, NIHT requires even less computational time as compared with more sophisticated algorithms due to the support set identification portion being the dominant task. On the other hand, when approximate solutions are sought with $\|y-A\hat{x}\|_2/\|y\|_2 \ll 10^{-4}$ the more sophisticated algorithms can be substantially faster due to the asymptotic convergence rate more directly impacting the computational time.

2.3.1 *Accelerated iterative hard thresholding algorithms.* Hard Thresholding Pursuit (HTP), Alg. 7 (originally presented as *Iterative thresholding with inversion (ITI)* [66]), and SVP-Newton [56] are the simplest accelerated hard thresholding algorithms for (1.1)–(1.3). HTP corresponds to NIHT with an added pseudo-inverse (Step 4) to compute the optimal values of the k nonzeros given the current estimate of the support set T_l. Typically the pseudo-inverse is computed using the conjugate gradient (CG) method for the system $\text{Proj}_{T_l}\left(A^*A\text{Proj}_{T_l}(x)\right) = \text{Proj}_{T_l}(A^*y)$. Similarly, SVP-Newton corresponds to NIHT for matrix completion with an added step after the principal subspace U_l is identified, where the next estimate is given by $X_l = \arg\min_Z \|y-\mathscr{A}\left(\text{Proj}_{U_l}(Z)\right)\|_2$. An important aspect of the computational effectiveness of HTP and SVP-Newton is to determine how many iterations of CG should be implemented per iteration of HTP and SVP-Newton to approximately solve the least squares sub-problem. CGIHT restarted and CGIHT projected for compressed sensing can be viewed as the internal CG for the least squares sub-problem in HTP terminating upon either the support set changing or the relative fraction $\left\|r_{l-1}-\text{Proj}_{T_{l-1}}(p_{l-1})\right\| / \|\text{Proj}_{T_{l-1}}(r_{l-1})\|$ being sufficiently large. CGIHT projected for matrix completion can be viewed as the internal CG for the least squares sub-problem in SVP-Newton terminating for sufficiently large relative fractions $\left\|R_{l-1}-\text{Proj}_{U_{l-1}}(P_{l-1})\right\| / \|\text{Proj}_{U_{l-1}}(R_{l-1})\|$. These restarting conditions control the computational effort of solving the least squares sub-problem of HTP or SVD-Newton for a current estimate of the support set or subspace before moving to a new support set or subspace. Properly controlling the computational cost of solving the sub-problem is essential to obtain an overall faster recovery time than NIHT [12]. Moreover, excessively solving the sub-problem of minimizing the objective in (1.1)–(1.3) while restricted to a fixed (row) support set or subspace can result in difficulty moving to a different support set or subspace; this can ultimately cause these algorithms to fail to recover sparse vectors of cardinality k which can be recovered by CGIHT for compressed sensing. Likewise, these algorithms can fail to recover low rank matrices of rank r which can be recovered by CGIHT for matrix completion. In addition to the provable recovery guarantees of the restarted and projected variants of CGIHT in Thm. 2.2–2.5, the main innovation of CGIHT is the *nearly uniformly superior empirical performance of CGIHT as compared to NIHT and HTP both in terms of values of* (k,m,n) *and* (r,m,n,p) *recoverable as well as the runtime needed to recover the solution.*

As previously mentioned, there are numerous other accelerated hard thresholding algorithms for

Algorithm 7 HTP (Hard Thresholding Pursuit [45])

Initialization: Set $w_{-1} = A^*y$, $T_0 = \texttt{PrincipalSupport}_k(|w_{-1}|)$, $x_0 = \text{Proj}_{T_0}(w_{-1})$, and $l = 1$.
Iteration: During iteration l, **do**

1: $r_{l-1} = A^*(y - Ax_{l-1})$ (update the residual)
2: $\omega_l = \frac{\|\text{Proj}_{T_{l-1}}(r_{l-1})\|^2}{\|A\text{Proj}_{T_{l-1}}(r_{l-1})\|^2}$ (optimal stepsize in the k-subspace T_{l-1})
3: $x_l = x_{l-1} + \omega_l r_{l-1}$ (steepest descent step)
4: $T_l = \texttt{PrincipalSupport}_k(x_l)$ (proxy to the support set)
5: $x_l = \text{Proj}_{T_l}(A^\dagger y)$ (projection onto the k-subspace T_l)

the solution of (1.1)–(1.3). In [14], Blumensath introduced the Accelerated Iterative Hard Thresholding (AIHT) framework for establishing recovery guarantees for algorithms which utilize a method for reducing the residual norm of an approximation when compared to the NIHT approximation. In that work, an algorithm is considered an Accelerated IHT algorithm if at each iteration the output x_l is a k-sparse vector satisfying $\|y - Ax_l\|_2 \leqslant \|y - A\bar{x}_l\|_2$ where $\bar{x}_l$ is obtained from NIHT. In the AIHT framework, one might consider using a subspace restricted conjugate gradient method to obtain the minimum norm solution; this is precisely the method used in HTP, and could be extended for (1.3) to SVP-Newton. The AIHT framework is designed based on the further reduction of the norm of the residual $\|y - Ax_l\|$. The CGIHT family of algorithms does not fit within this framework. Rather than performing iterations of CG on a subspace in order reduce the residual, as is done in HTP and SVP-Newton, CGIHT is designed to always perform CG on a subspace except when a measure of subspace confidence is violated. It is this perspective that subspace confidence dictates the restarting decisions which separates CGIHT from the AIHT framework including HTP and SVP-Newton. Moreover, in iterations where the true subspace has not been identified, there is, *a priori*, no obvious guarantee a CGIHT step will reduce the residual norm. The stepsize used in CGIHT is only the optimal stepsize to reduce the residual if the current subspace (T_l or U_l) contains the true supporting subspace of the optimal solution to the linear system of equations[1]. Consequently, the CGIHT family of algorithms requires a direct proof of contraction[2] as in Thm. 2.2–2.5.

We briefly discuss some additional notable examples of hard thresholding algorithms, though in less detail due to their being less closely related to CGIHT than are NIHT and HTP (SVP-Newton). For compressed sensing, (1.1) and (1.2), examples of other hard thresholding algorithms include Compressive Sampling Matching Pursuit (CoSaMP) [69], Subspace Pursuit [28] (SP), and the Algebraic Pursuit (ALPS) family of algorithms [25]. These, and other related algorithms, differ from NIHT and HTP by having intermediate steps that further update the values on support sets of cardinality greater than k. Analogous algorithms also exist for matrix completion, (1.3), including the Matrix ALPS family of algorithms [59] and Atomic Decomposition for Minimum Rank Approximation (ADMiRA) [60] which is an extension of CoSaMP to (1.3). There are numerous other hard thresholding algorithms, particularly for matrix completion where the continuous manifold of low rank matrices gives scope for greater diversity of algorithm constructions, see for example [29, 53, 57, 67]. For each of these algorithms there are qualitatively similar recovery guarantees provided the measurement operator has sufficiently small RICs. Of greater practical importance is the region of the problem sizes recoverable for each of

[1]The stepsizes used in NIHT and HTP are also only optimal in terms of residual norm reduction when the current subspace contains the true subspace.

[2]Once the contraction is established, the reduction in residual required by the AIHT framework is simultaneously established.

(1.1)–(1.3) and the relative computational time required for each algorithm. An exhaustive comparison of algorithms in the literature is beyond the scope of this manuscript. Instead we focus our empirical comparisons in Sec. 3 on variants of two representative algorithms which have advantageous properties: CSMPSP (a variant of CoSaMP and SP, see [12]) for (1.1)–(1.2) and FIHT (a variant of 1-ALPS(2)) for (1.1)–(1.3). FIHT, Alg. 8, is a variant of 1-ALPS(2) which is the version of ALPS observed to have the greatest overall computational efficacy. FIHT differs from 1-ALPS(2) in its fourth step, where 1-ALPS(2) uses a support set of size of at least $2k$ by including k indices from the residual in its third step. Empirical testing [83] showed FIHT to be uniformly superior to 1-ALPS(2) for (1.1) and (1.3) in terms of both the problem sizes recoverable and the runtime needed to recover the solution. Central to the fast asymptotic convergence rate of FIHT is the optimal selection of τ listed in the first step of FIHT. For τ fixed, the traditional 1-ALPS(2) has been proven to have RIC recovery guarantees analogous to other hard thresholding algorithms; however, no such proof is currently available when the variable τ from the first step of Alg. 8 is used. In this sense, FIHT can be most directly compared with the variants of CGIHT that do not have restarting and similarly lack a RIC analysis of uniform recovery.

Algorithm 8 FIHT: a variant of 1-ALPS(2) [25]

Initialization: Set $x_{-1} = 0$, $w_{-1} = A^*y$, $T_0 = \texttt{PrincipalSupport}_k(|w_{-1}|)$, $x_0 = \text{Proj}_{T_0}(w_{-1})$, and $l = 1$.

Iteration: During iteration l, **do**

1: if $l = 1$,
$\tau_{l-1} = 0$ (set momentum stepsize)
else
$\tau_{l-1} = \frac{\langle y - Ax_{l-1}, A(x_{l-1} - x_{l-2})\rangle}{\|A(x_{l-1} - x_{l-2})\|^2}$ (calculate momentum stepsize)

2: $w_{l-1} = x_{l-1} + \tau_{l-1}(x_{l-1} - x_{l-2})$ (new extrapolated point)

3: $r^w_{l-1} = A^*(y - Aw_{l-1})$ (residual of extrapolated point)

4: $T^w_{l-1} = \texttt{PrincipalSupport}_k(w_{l-1})$ (support set of extrapolated point)

5: $\tilde{\alpha}_l = \frac{\|\text{Proj}_{T^w_{l-1}}(r^w_{l-1})\|^2}{\|A\text{Proj}_{T^w_{l-1}}(r^w_{l-1})\|^2}$ (optimal stepsize restricted to support of w_{l-1})

6: $x_l = w_{l-1} + \tilde{\alpha}_l r^w_{l-1}$ (steepest descent step)

7: $T_l = \texttt{PrincipalSupport}_k(x_l)$ (proxy to the support set)

8: $x_l = \texttt{Threshold}(x_l, T_l)$ (restriction to proxy support set T_l)

9: $r_l = A^*(y - Ax_l)$ (residual of x_l)

10: $\alpha_l = \frac{\|\text{Proj}_{T_l}(r_l)\|^2}{\|A\text{Proj}_{T_l}(r_l)\|^2}$ (optimal stepsize restricted to T_l)

11: $x_l = x_l + \alpha_l \text{Proj}_{T_l}(r_l)$ (one more gradient descent in the restricted search direction)

Sec. 3.1 shows CGIHT and FIHT are typically the most efficient methods for the solution of (1.1) in terms of (k,m,n) recoverable and the computational time needed. FIHT (and the ALPS family of methods) are based on the optimal first order method for convex problems [70] which is known to lose its accelerated convergence rate in the presence of noise [30]; for detailed comparisons in the presence of noise see [13]. Sec. 3.2 shows CGIHT to be the most efficient algorithm for the solution of (1.2), with notably higher recovery regions than CSMPSP, NIHT and FIHT. Sec. 3.3 shows CGIHT has a recovery region comparable to that of NIHT, but with a notably faster asymptotic convergence rate, similar to FIHT though with a modestly higher phase transition and greater robustness to noise [83]. In totality the empirical results presented in Sec. 3 show CGIHT to generally be the preferred algorithm

for compressed sensing and matrix completion, (1.1)–(1.3).

2.3.2 *Beyond iterative hard thresholding algorithms for* (1.1)–(1.3). There are equally many algorithms for the solution of (1.1)–(1.3) which are not iterative hard thresholding algorithms. Though an exhaustive review is beyond the scope of this manuscript, we briefly review the most notable example. The most widely studied alternative to directly solving (1.1)–(1.3) is replacement of the non-convex constraint with a convex relaxation. That is, the (row) sparsity and rank constraints are replaced by the ℓ_1 and Schatten-1 norms respectively and the objective is reformulated as a variant of

$$\min_{z\in\mathbb{R}^n} \|y-Az\|_2 + \lambda\|z\|_1, \tag{2.10}$$

$$\min_{Z\in\mathbb{R}^{n\times r}} \|Y-AZ\|_2 + \lambda\|Z\|_{1,2}, \tag{2.11}$$

or

$$\min_{Z\in\mathbb{R}^{m\times n}} \|y-\mathscr{A}(Z)\|_2 + \lambda\|Z\|_* \tag{2.12}$$

where $\|Z\|_{1,2}$ is the sum of the ℓ_2 norm of the rows of Z and $\|Z\|_*$ is the sum of the singular values of Z. Under suitable conditions on the sensing operator, A or $\mathscr{A}$, it has been proven that the solution to these convex relaxations is *exactly* the solution to the associated nonconvex question (1.1)–(1.3); see for example [1, 19, 27, 31, 34, 42, 56, 74–76, 80, 85] and references therein. The formulations (2.10)–(2.12) are standard convex optimization questions and can be solved using any of a myriad of algorithms and software packages. In addition, there are many algorithms for (2.10)–(2.12) specifically designed for efficient identification of sparse or low rank solutions; see for example [3–6, 18, 44, 50, 65, 71, 79, 82, 84, 86]. Of these sparsity tailored convex relaxation algorithms, the one most related to CGIHT is CGIST [50] where CG is applied to soft thresholding with restarting conditions based upon the *sign pattern* of past iterates; a fixed sign pattern corresponds to a fixed face of the projected polytope. Though the literature contains several empirical comparisons of the aforementioned algorithms, a detailed average runtime comparison between iterative hard thresholding algorithms and convex relaxation algorithms is particularly challenging due to various implementation heuristics, such as warm starting [44], commonly used in algorithms for the convex relaxations. Moreover, solving (2.10)–(2.12) with a fixed relaxation parameter λ correctly identifies the subspace but results in deflated nonzero values in (2.10)–(2.11) and deflated singular values in (2.12). As a consequence, the solution to (2.10)–(2.12) is typically *debiased* on the fixed solution subspace to obtain the solution to (1.1)–(1.3). An intra-class comparison should include these practical acceleration techniques for both classes of methods as well as debiasing where needed. Though such an intra-class comparison would be highly informative, this manuscript focuses on directly contrasting the CGIHT family of algorithms within the single class of iterative hard thresholding algorithms. Comparisons between the precurser IHT and non-IHT based algorithms are available in [37].

3. Empirical Performance Comparisons

The RIC based recovery guarantees for the solution of (1.1)-(1.3), such as Thm. 2.2–2.5 for CGIHT, are uniform over all measured k (row) sparse or rank r matrices. The uniform nature of these results gives sufficient conditions for recovery, but these conditions are typically highly pessimistic when compared with the ability of an algorithm to solve (1.1)–(1.3) for (row) sparse or low rank matrices selected independently from the linear measurement operator. This observed average case performance is often

far more useful to practitioners since the theoretical guarantees require many more measurements than would typically be useful in applications.

All three problems (1.1)–(1.3) are underdetermined linear systems of equations as the number of linear observations is less than the ambient dimension of the object being measured. If a full set of measurements is obtained, the sensing process is invertible and the exact solution is easily identified. On the other hand, the underlying object lies in a subspace defined by relatively few degrees of freedom. For compressed sensing, the k-sparse signals have exactly k degrees of freedom while in matrix completion the set of rank r matrices lie on a manifold defined by $r(m+n-r)$ degrees of freedom. If the subspace containing the measured object is known *a priori*, the object can be recovered by making appropriate measurements in the known subspace with the number of measurements equal to the number of degrees of freedom. Therefore, these problems inherently define an *undersampling ratio* and an *oversampling ratio*

$$\delta = \frac{\text{number of observations}}{\text{ambient dimension}} \quad \text{and} \quad \rho = \frac{\text{degrees of freedom}}{\text{number of observations}}, \tag{3.1}$$

with the unit square $(\delta,\rho) \in [0,1]^2$ a phase space defining where the number of measurements is sufficient for recovery of (1.1)–(1.3) to be possible, and yet fewer measurements are taken than the ambient dimension.

In this section, we present empirical observations of CGIHT's efficacy as compared with several leading hard thresholding algorithms. First, we present *recovery phase transition curves* for each algorithm which separate the phase space into two regions: success and failure. For problem instances with (δ,ρ) below the recovery phase transition curve, the algorithm is observed to return an approximation matching the solution of (1.1)–(1.3). Alternatively, for problem instances with (δ,ρ) above the recovery phase transition curve, the algorithm is observed to return an approximation that does not appear to be converging to a solution of (1.1)–(1.3). For each algorithm, the region of the phase space below the recovery phase transition curve is referred to as the *recovery region*. Recovery phase transition curves for several algorithms are presented for the compressed sensing problem, the row-sparse approximation problem, and the low rank matrix completion problem in Secs. 3.1, 3.2, and 3.3, respectively.

For a problem instance with sampling ratios (δ,ρ) in the intersection of the recovery region for multiple algorithms, a practitioner must select an algorithm based on some criteria other than ability to obtain the solution of (1.1)-(1.3). In [12], the authors introduced *algorithm selection maps* of the phase space which identify the algorithm from NIHT, HTP, or CSMPSP with the least computational recovery time. In Sec. 3.1.3 we present minimum recovery time algorithm selection maps which also consider CGIHT and FIHT.

3.1 *Empirical Performance Comparisons for Compressed Sensing*

3.1.1 *Experimental Set-up.* The empirical performance comparison of the hard thresholding algorithms for compressed sensing is conducted with the software[3] *GAGA: GPU Accelerated Greedy Algorithms for Compressed Sensing*, Version 1.1.0 [10, 11]. This section compares the performance of CGIHT, CGIHT restarted, CGIHT projected, FIHT, NIHT, HTP, and CSMPSP. CGIHT projected is implemented with the restarting parameter $\theta = 6$ for problem instances with $\delta \leqslant 0.5$ and $\theta = 3$ for $\delta > 0.5$. These values of θ were selected due to their favorable performance in preliminary tests, but have not been extensively tuned. Moreover, the values of θ have not been selected based on the RIC

[3]This software package includes a folder with a Matlab script *do_all.m* which automatically generates the necessary dataset (though with different random seeds) and processes the data to produce all figures contained in Sec. 3.1.

conditions of Thm. 2.2 where a computationally inefficient $\theta \ll 1$ would be suggested for uniform recovery guarantees. The testing was conducted on a Linux machine with Intel Xeon E5-2643 CPUs @ 3.30 GHz, NVIDIA Tesla K10 GPUs, and executed from Matlab R2013a. The algorithms are tested with three random matrix ensembles:

- $\mathscr{N}$: dense Gaussian matrices with entries drawn i.i.d. from $\mathscr{N}(0, m^{-1})$;
- $\mathscr{S}_7$: sparse matrices with 7 nonzero entries per column drawn with equal probability from $\{-1/\sqrt{7}, 1/\sqrt{7}\}$ and locations in each column chosen uniformly;
- *DCT*: m rows chosen uniformly from the $n \times n$ Discrete Cosine Transform matrix.

These three random matrix ensembles are representative of the random matrices frequently encountered in compressed sensing: $\mathscr{N}$ represents dense matrices, $\mathscr{S}_7$ represents sparse matrices, and *DCT* represents subsampling structured matrices with fast matrix-vector products.

The measured vectors x are taken from the random binary vector ensemble, B, which are formed by uniformly selecting k locations for nonzero entries with values $\{-1, 1\}$ chosen with equal probability. A *problem class* (Mat, B) consists of a matrix ensemble, $Mat \in \{\mathscr{N}, \mathscr{S}_7, DCT\}$, and a sparse vector drawn from the binary vector distribution, B. Alternative sparse vector ensembles, such as having nonzero entries drawn from a uniform or normal distribution, have been shown to have larger regions of the (δ, ρ) phase space in which hard thresholding algorithms succeed in finding the solution to the compressed sensing problem (1.1); related phase transitions and additional performance characteristics for alternate vector ensembles are available in [12]. For conciseness we restrict our focus to sparse vectors from B and with measurements y given by Ax; empirical observations of CGIHT in the presence of noise are considered in [13].

When the measured vector is taken from the vector ensemble B, the ℓ_∞ norm accurately determines when the support set of the approximation returned by a hard thresholding algorithm coincides with the support set of the measured vector. Additionally, when the ℓ_∞ norm is small, the algorithms have accurately identified the values of the nonzero entries. Therefore, for the problem classes (Mat, B), we say an algorithm has successfully recovered the measured vector if the output $\hat{x}$ differs by no more than 10^{-3} in any single component, namely $\|\hat{x} - x\|_\infty \leqslant 10^{-3}$ which implies that the correct support has been identified and that

$$\frac{\|\hat{x} - x\|_2}{\|x\|_2} \leqslant 10^{-3}. \tag{3.2}$$

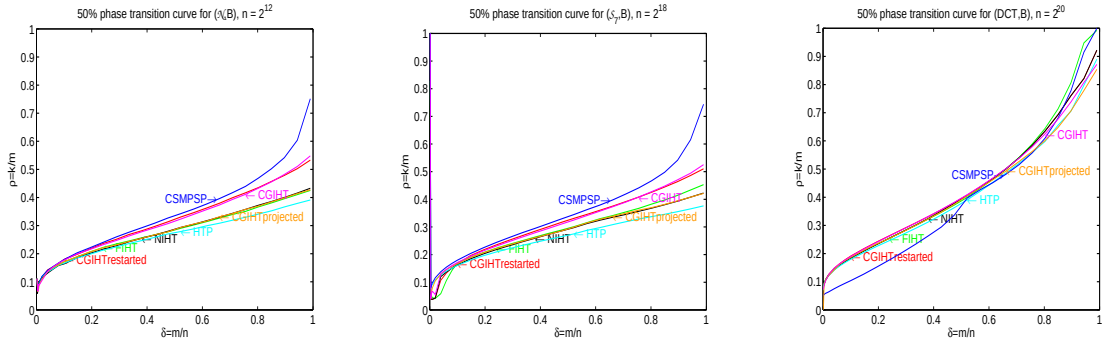
The results of the algorithm tests are presented in the recovery phase transition framework. For the compressed sensing problem, the sparsity of the desired approximation defines the minimum number of measurements required to capture the underlying information. If the support of the vector x is known *a priori*, only $k = \|x\|_0$ measurements are necessary. Therefore, taking m measurements with $k < m < n$ defines the compressed sensing undersampling and oversampling ratios

$$\delta = \frac{m}{n} \quad \text{and} \quad \rho = \frac{k}{m}. \tag{3.3}$$

For testing the compressed sensing problem, the parameter $\delta \in (0, 1)$ takes on thirty values

$$\delta \in \{0.001, 0.002, 0.004, 0.006, 0.008, 0.01, 0.02, 0.04, 0.06, 0.08, 0.1, \ldots, 0.99\} \tag{3.4}$$

with 18 additional uniformly spaced values of δ between 0.1 and 0.99. The parameter $\rho \in (0, 1)$ is sampled in two different ways, one to identify the recovery phase transition for each algorithm and the second for direct performance comparison throughout the recovery regions.



(a) (b) (c)

FIG. 1. 50% recovery probability logistic regression curves for matrix ensembles (a) $\mathcal{N}$ with $n = 2^{12}$, (b) $\mathcal{S}_7$ with $n = 2^{18}$, and (c) *DCT* with $n = 2^{20}$.

3.1.2 *Recovery Phase Transition Curves.* The empirical recovery phase transition curves are logistic regression curves identifying the 50% success rate for the given algorithm and problem class. In order to generate sufficient data for the logistic regression, the testing focuses on the phase transition region where the algorithm transitions from always succeeding to always failing. For a given problem class (Mat, B) and a specific value of n, the 30 values of m are chosen according to $m = \lceil \delta \cdot n \rceil$ with δ taken from (3.4). The phase transition region is found via a binary search which determines an interval $[k_{min}, k_{max}]$ so that the algorithm successfully recovers each of 10 problem instances at $k < k_{min}$ and fails to recover any of 10 problem instances for $k > k_{max}$. The phase transition region $[k_{min}, k_{max}]$ is then extensively tested with 10 problem instances at $\max(50, \sqrt{m})$ distinct, uniformly spaced values of $k \in (k_{min}, k_{max})$. When $k_{max} - k_{min} \leqslant 50$, every value of $k \in [k_{min}, k_{max}]$ is tested 10 times. The results presented in this subsection were obtained in precisely the same manner as those reported in [12] where the interested reader will find further details regarding experimental set-up, stopping criteria, and algorithm implementation.

In [12] it was observed that NIHT, HTP, and CSMPSP have similar recovery phase transition curves for $\delta \lesssim 0.1$ for matrix ensembles $\mathcal{N}$ and $\mathcal{S}_7$ while CSMPSP had an inferior phase transition in this region for the DCT matrix ensemble. For larger values of δ, CSMPSP typically had the highest recovery phase transition curve while NIHT and HTP continued to have similar phase transitions. In Fig. 1 we observe that the recovery phase transition curves for CGIHT and CGIHT restarted are superior to the phase transition curves of NIHT and HTP for essentially all $\delta \in (0,1)$. In particular, the phase transitions curves for CGIHT and CGIHT restarted are closer to that of CSMPSP despite CSMPSP searching over a larger support set in each iteration. Figure 1 also shows that the recovery phase transition for CGIHT projected and FIHT are nearly identical to NIHT and HTP; again this is noteworthy in that FIHT searches over a larger support set when the support set is changing.

3.1.3 *Algorithm Selection Maps.* The recovery phase transition curves of Sec. 3.1.2 define recovery regions where the associated algorithm is typically able to successfully approximate the sparse vector x. Algorithm selection is straightforward when faced with a problem instance in a region of the phase space where only one algorithm is typically successful. However, algorithm selection requires additional information in regions of the phase space where multiple algorithms are typically successful, i.e. in

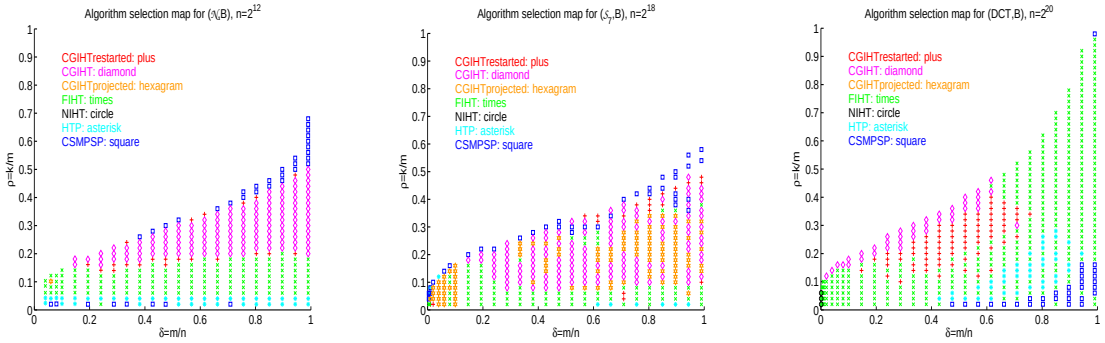


FIG. 2. Algorithm selection maps for matrix ensembles (a) $\mathscr{N}$ with $n = 2^{12}$, (b) $\mathscr{S}_7$ with $n = 2^{18}$, and (c) DCT with $n = 2^{20}$.

the intersection of the recovery regions for different algorithms. In this section, we present algorithm selection maps of the phase space based on least computational time for successful recovery. In order to compare algorithms directly, the phase space is sampled on a mesh consisting of (δ,ρ) with δ from (3.4) and ρ taking the values

$$\rho \in \{\rho_j = 0.02j \mid j = 1,2,\ldots,j^\star\} \tag{3.5}$$

where $\rho_{j^\star}$ is the first value of ρ for which an algorithm fails to recover each of 10 problem instances tested. For the problem instances tested on this mesh, the algorithm with the least computational recovery time is identified on the algorithm selection map in Fig. 2. Algorithm selection maps show consistent general trends across various problem sizes for each problem class (Mat,B) [12]; it is the general trends that are important in the selection maps rather than individual points.

For the problem class $(\mathscr{N},B)$, when $\rho \lesssim 0.2$, the algorithm selection map in Fig. 2(a) recommends the use of FIHT while for $\rho \gtrsim 0.2$ CGIHT reliably recovers the sparse vector in the least time. The algorithm selection map in Fig. 2(b) depicts three clear regions of the phase space for the problem class $(\mathscr{S}_7,B)$. When undersampling by a factor of ten or more so that $\delta < 0.1$, CGIHT projected will return the solution to (1.1) in the least time. For $0.1 \lesssim \delta \lesssim 0.2$, or for $\delta \gtrsim 0.2$ with $\rho \lesssim 0.1$, FIHT will recover the measured vector in the least time. In all other cases, namely $\delta \gtrsim 0.2$ and $\rho \gtrsim 0.1$, CGIHT is recommended. For the problem class (DCT,B), FIHT is recommended through the majority of the recovery region. In the region of the phase space from $0.2 \lesssim \delta \lesssim 0.6$, CGIHT restarted or CGIHT are recommended as ρ approaches the phase transition curve of CGIHT. Note that when the measurements are corrupted by additive noise, namely $y = Ax + e$, FIHT is completely absent from the algorithm selection maps due to the reintroduction of the noise in each iteration at the computation of the extrapolated point (Alg. 8, Steps 1-2); for details see [13].

While the algorithm selection maps identify the algorithm with the least average computation time to recover the vector, Fig. 3 provides a more complete picture for algorithm selection showing the ratio of a given algorithm's recovery time to the least recovery time of the algorithms tested. In particular, Fig. 3 shows that at each point in the (δ,ρ) phase space where recovery is possible, there is a variant of CGIHT that is either the fastest or within a few percent of being the fastest. For example, while the algorithm selection map advocates FIHT for problem class $(\mathscr{N},B)$ and $\rho < 0.2$, Fig. 3 (a),(d), and (g) show that CGIHT, CGIHT restarted, and CGIHT projected rarely require more than 1.25 times as long to find the solution, and never require more than 1.6 times as long as the least computation time among all algorithms. Moreover, for the problem class (DCT,B), while the algorithm selection map Fig. 2(c) seems to advocate for FIHT throughout the majority of the phase space, the ratio maps Fig. 3 (c),(f) and

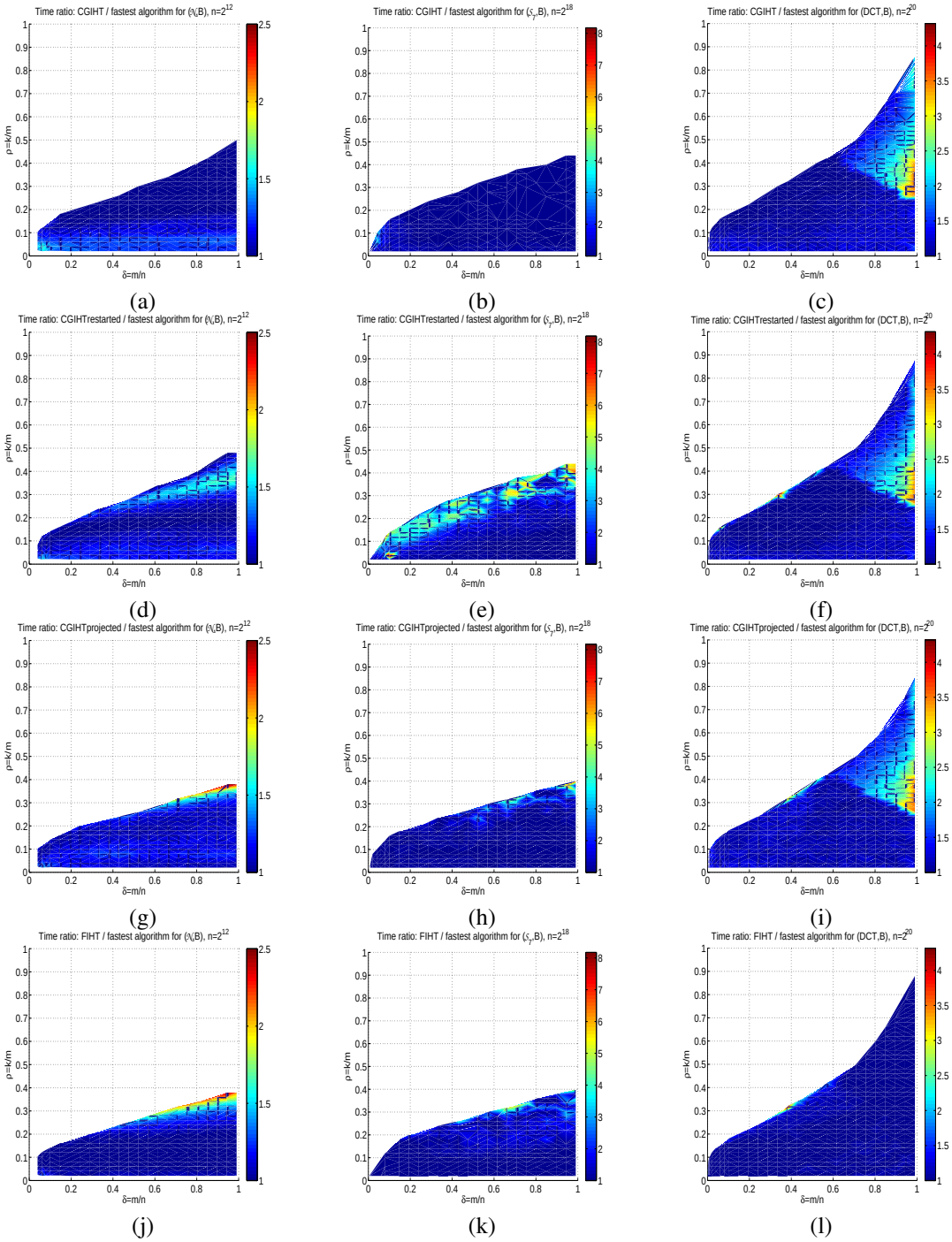
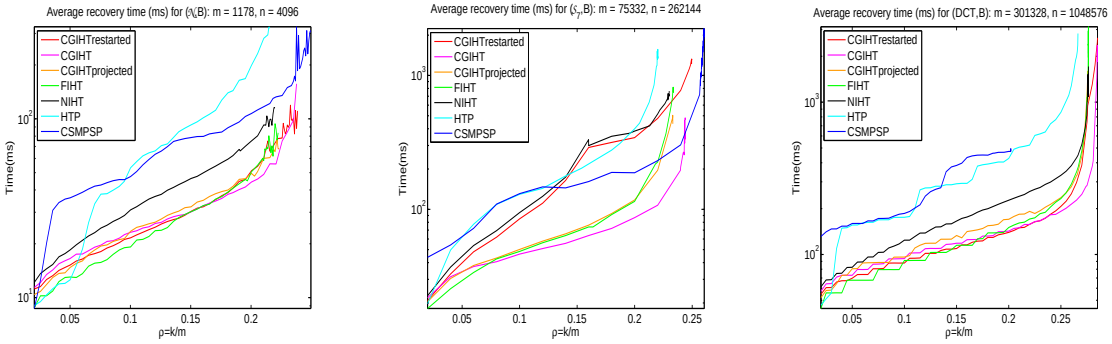


FIG. 3. Average recovery time ratio for CGIHT (a-c), CGIHT restarted (d-f), CGIHT projected (g-i), and FIHT (j-l) compared to the fastest recovery time among all algorithms. Matrix Ensembles: $\mathscr{N}$ with $n = 2^{12}$ (left panels), $\mathscr{S}_7$ with $n = 2^{18}$ (center panels), *DCT* with $n = 2^{20}$ (right panels).

(j) show that for $\delta < 0.7$, CGIHT, CGIHT restarted, and FIHT have essentially the same recovery time. Lastly, the CGIHT variants and FIHT show greater differences in relative recover time for the problem class $(\mathscr{S}_7, B)$, though with a variant of CGIHT nearly always having a smaller recovery time than FIHT. In totality, for the most interesting region of the phase space in compressed sensing, namely $\delta < 1/2$, the average recovery time for CGIHT, CGIHT restarted, CGIHT projected, and FIHT are often within a few percent of one another for each problem class considered.



(a) (b) (c)

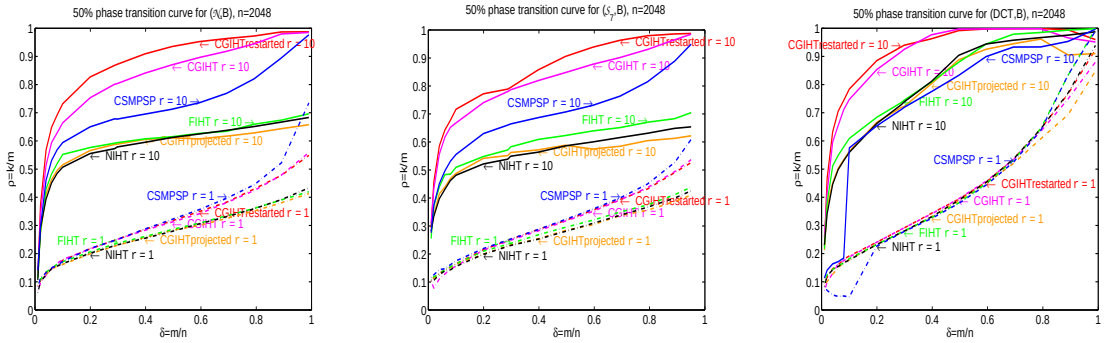
FIG. 4. Average recovery time (ms) dependence on ρ for $\delta \approx 0.287$; (a) $\mathscr{N}$ with $n = 2^{12}$, (b) $\mathscr{S}_7$ with $n = 2^{18}$, (c) *DCT* with $n = 2^{20}$. Vertical scale in log(ms).

3.1.4 *Recovery time dependence on ρ for $\delta \approx 0.3$.* To further investigate the relative computational cost of the algorithms, we provide detailed information for each problem class with a single (m,n) pair. For fixed values of m, n, with a specific undersampling ratio of $\delta \approx 0.287$ (the columns in Fig. 2–3 closest to $\delta = 0.3$), a semi-log plot of the average recovery time is displayed against $\rho = k/m$ up to the maximum phase transition of the algorithms tested at the selected δ. For each problem class and each value of $\rho_j = .02 \cdot j$ with $j \leqslant 15$, 100 problem instances were tested with the average time required for recovery presented. Figure 4 shows two consistent trends across all three problem classes. First, the three variants of CGIHT and FIHT provide significantly improved recovery time when compared to NIHT, HTP, and CSMPSP. Second, among these four accelerated hard thresholding algorithms, CGIHT demonstrates a clear computational advantage as ρ increases toward the recovery phase transition.

RECOMMENDATION Taken together, the algorithm selection maps, average time ratio maps, and average recovery time plot in Figs. 2–4 show that CGIHT, CGIHT restarted, CGIHT projected, and FIHT share a competitive recovery performance which is superior to the other algorithms tested. Due to its consistent performance across problem classes, its computational advantage in the most important region of the phase space with $\delta < 1/2$ and ρ near the phase transition curve, and its observed stability to noise presented in [13], we recommend CGIHT for compressed sensing (Alg. 1) when solving the compressed sensing problem (1.1) with a hard thresholding algorithm.

3.2 *Empirical Performance Comparison for Row-Sparse Matrices*

3.2.1 *Experimental Set-up.* A similar set of empirical results are presented here for the row-sparse approximation problem (1.2). The problem classes are simple extensions of the those from Sec. 3.1. In



(a) (b) (c)

FIG. 5. 50% recovery probability logistic regression curves with $n = 2^{11}$ and $r = 1, 10$ for matrix ensembles (a) $\mathscr{N}$, (b) $\mathscr{S}_7$, and (c) *DCT*.

this section, a problem class (Mat, B) consists of measurement matrix ensemble $Mat \in \{\mathscr{N}, \mathscr{S}_7, DCT\}$ and a matrix drawn from the binary row-sparse matrix ensemble; to form an element of the binary row-sparse matrix ensemble, the row support is selected uniformly at random from the integers $\{1, \ldots, n\}$ and the entries in these rows are populated with values drawn with equality probability from $\{-1, 1\}$. The tests were conducted in Matlab R2013a (with inherent multithreading) on a Linux machine with two Intel Xeon E5620 CPUs @2.40GHZ.

For clarity, we compare the three variants of CGIHT for compressed sensing against FIHT, NIHT, and CSMPSP. In [9], it is shown that HTP and NIHT have nearly identical behavior in the row-sparse setting. The algorithms are extended to the row-sparse approximation setting similar to other hard thresholding algorithms [9, 46, 81]. The algorithms were implemented by extending the Matlab (non-GPU) version of *GAGA* [10]. CGIHT projected is implemented with the restarting parameter $\theta = 3$ for problem instances with $\delta \leqslant 0.5$ and $\theta = 10$ for $\delta > 0.5$; again, these values of θ were selected based on preliminary tests and have not been tuned. For the row-sparse approximation problem, the undersampling and oversampling ratios are again defined by $\delta = m/n$ and $\rho = k/m$, respectively, since the degrees of freedom, number of measurements, and ambient dimension are all scaled by r. In this section, we consider a row-sparse matrix to be successfully recovered when the algorithm returns an approximation $\hat{X}$ which satisfies

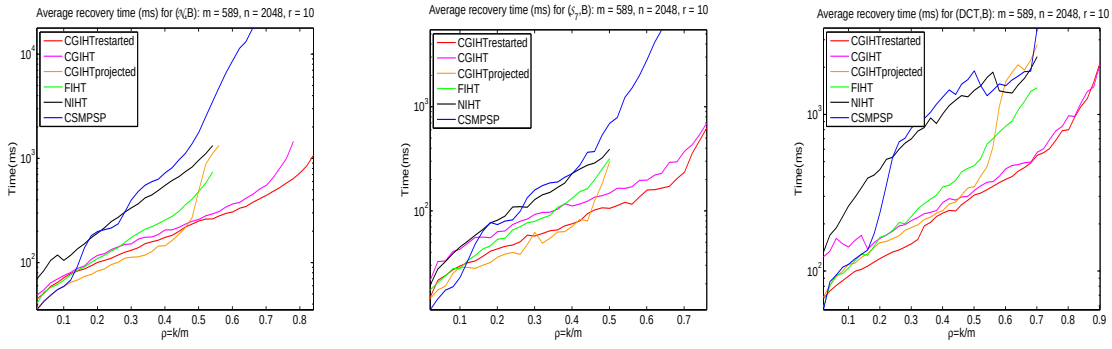
$$\frac{\|\hat{X} - X\|_F}{\|X\|_F} < 10^{-3}. \tag{3.6}$$

3.2.2 *Recovery phase transition curves.* The recovery phase transitions are again defined by the logistic regression curve for the 50% successful recovery rate. Recovery phase transition curves for both a single column ($r = 1$) and ten columns ($r = 10$) appear in Fig. 5. Due to the larger computational burden and the current lack of a parallelized GPU implementation of these algorithms, the algorithms were tested for $n = 2048$ with only fifteen values of δ which are spaced in $[0, 1]$ in a similar fashion to (3.4); in particular, five of the fifteen values of δ lie from 0.01 to 0.1 in order to properly identify the phase transition in the extreme undersampling scenario. The results presented in this section were obtained in the same manner as those reported in [9] where the interested reader will find additional information regarding stopping criteria and experimental set-up.

The significant increase in the area of each algorithm's recovery regions for $r = 10$ compared to

$r = 1$ is consistent with other empirical testing; in particular the empirical recovery phase transition curves for NIHT and CSMPSP reported here are consistent with those reported in [9]. The single vector, $r = 1$, phase transition curves for the variants of CGIHT shown in Figs. 1 and 5 fall between the phase transition curves of NIHT and CSMPSP. On the other hand, for row-sparse matrices with ten independent columns the advantage of using CGIHT is greatly amplified. For row sparse matrices from problem class (Mat, B) with ten columns, the recovery phase transition curves for both CGIHT and CGIHT restarted are substantially higher than that of CSMPSP. Importantly, Fig. 5 shows that for any reasonable undersampling ratio $\delta < 0.75$, CGIHT and CGIHT restarted have substantially larger recovery regions than CSMPSP which had the largest previously reported recovery region for row-sparse approximation [9]. The recovery phase transition curves for CGIHT projected, FIHT, and NIHT are similar across all three problem classes and are not competitive with CGIHT or CGIHT restarted when $r = 10$. For $r = 10$ and for all three problem classes (Mat, B) with $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$, CGIHT restarted has the highest recovery phase transition curve and largest recovery region.

3.2.3 *Recovery time dependence on ρ for $\delta \approx 0.3$.* Similar to Sec. 3.1.4, this subsection investigates the average recovery time of the algorithms for successful identification of the solution to (1.2). For a single value of $\delta \approx 0.287$ with $n = 2048$ and $m = \lceil \delta \cdot n \rceil = 589$, 100 problem instances were tested for each value of $k = \lfloor \rho_j \cdot m \rfloor$ for $\rho_j = \{0.02j : j = 1, \ldots, j^\star\}$ where every algorithm failed to recover any of 100 problem instances for $k = \lfloor \rho_{j^\star} \cdot m \rfloor$. Figure 6 shows that either CGIHT restarted or CGIHT projected is able to recover the row-sparse matrices in the least time for $0.1 < \rho < \rho_{j^\star}$. In combination, Figs. 5–6 show that CGIHT, CGIHT restarted, CGIHT projected, and FIHT offer a significant computational advantage over other leading hard thresholding algorithms for row-sparse approximation. While the phase transition curve of CGIHT projected closely tracks with that of NIHT, CGIHT projected offers a substantial acceleration in recovery time for ρ well within the recovery region. For very small values of ρ, CGIHT projected recovers the measured row-sparse matrix in the least time, similar to its behavior in the single vector $r = 1$ setting. For all values of ρ in Fig. 6, CGIHT restarted has an average recovery time that is less than CGIHT and for ρ approaching the phase transition the average recovery time of CGIHT restarted is considerably better than the average recovery time of CGIHT projected and FIHT.



(a) (b) (c)

FIG. 6. Average recovery time (ms) dependence on ρ for $\delta \approx 0.287$ with $n = 2^{11}$ and $r = 10$; (a) $\mathcal{N}$, (b) $\mathcal{S}_7$, (c) DCT. Vertical scale in $\log(\text{ms})$.

RECOMMENDATION The support set restarting criterion of CGIHT restarted for this discrete problem shows a clear advantage in terms of both recovery phase transition curve and computational time for recovery near the phase transition. Across all problem classes and for all values of ρ in Fig. 6, CGIHT restarted is either superior to or competitive with CGIHT projected in terms of time for recovery; moreover, CGIHT restarted is faster than CGIHT for the row-sparse problem with $r = 10$. Due to its superior phase transition and lower computational time for identifying the solution, we recommend CGIHT restarted for solving the row-sparse approximation problem (1.2) with a hard thresholding algorithm.

3.3 *Empirical Performance Comparison for Matrix Completion*

3.3.1 *Experimental Set-up.* In this section, we present empirical recovery phase transition curves and investigate the rate of recovery for CGIHT and CGIHT projected for matrix completion. The testing was conducted through a massive distribution of problem instances at the IRIDIS High Performance Computing Facility provided by the e-Infrastructure South Centre for Innovation. Recall that the sensing operator in matrix completion is a linear map $\mathscr{A}(\cdot) : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ where $y = \mathscr{A}(X)$ has each measurement defined by the Frobenius inner product of a sensing matrix $A_i \in \mathbb{R}^{m\times n}$ and the matrix X: $y_i = \langle A_i, X\rangle_F$. The algorithms are tested with two representative sensing operator ensembles:

- $\mathscr{G}$: each of the p sensing matrices is a dense Gaussian matrix with entries drawn i.i.d. from $\mathscr{N}(0,(mn)^{-1})$;
- $\mathscr{E}$: each of the p sensing matrices is an entry sensing matrix with a single nonzero value of 1 with the location chosen uniformly at random without replacement.

The sensing operator ensemble $\mathscr{G}$ is representative of dense sensing while operators drawn from the ensemble $\mathscr{E}$ are representative of entry sensing operators. In implementation, the operators drawn from $\mathscr{E}$ are employed by directly acquiring p distinct entries in the sensed matrix rather than through a series of inner products. For matrix completion, the problem class $(Oper,N)$ is comprised of a sensing operator ensemble $Oper \in \{\mathscr{G},\mathscr{E}\}$ and the random low rank matrix ensemble N. To form a rank r matrix in the ensemble N, we compute the product of two random rank r matrices via $X = LR$ where $L \in \mathbb{R}^{m\times r}$ and $R \in \mathbb{R}^{r\times n}$ with entries drawn i.i.d. from the normal distribution $\mathscr{N}(0,1)$. For a given set of parameters (r,m,n,p), a problem instance is formed by drawing a sensing operator $\mathscr{A}$ and low rank matrix X from the problem class $(Oper,N)$.

In the matrix completion setting, the matrices $X \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(X) \leqslant r$ form a manifold of dimension $r(m+n-r)$ in $\mathbb{R}^{mn}$. Therefore, taking p linear measurements with $r(m+n-r) \leqslant p \leqslant mn$ defines matrix completion undersampling and oversampling ratios

$$\delta = \frac{p}{mn} \quad \text{and} \quad \rho = \frac{r(m+n-r)}{p}. \tag{3.7}$$

For conciseness, and consistent with a large portion of the literature, the empirical results presented in this section focus on square matrices with $m = n$; rectangular matrices are observed to have lower phase transitions; see for instance [1, 75, 77].

An empirical average-case performance analysis of NIHT for recovering a low rank matrix from near the minimum number of measurements was first reported by the last two authors in [77]. In that article, the authors compared NIHT with other state of the art methods demonstrating the superiority of NIHT in terms of phase transition for both sensing ensembles $\mathscr{G}$ and $\mathscr{E}$. Due to the already large computational time of more than 5.6 CPU years required to generate the matrix completion data presented here,

we restrict our testing to the matrix completion variants of CGIHT, CGIHT projected, NIHT and FIHT. The matrix completion variant of FIHT is an adaptation of Matrix ALPS II [59]; FIHT is observed to be superior to Matrix ALPS II in terms of both phase transition and computational complexity as the latter requires an additional singular value decomposition in each iteration to include the extra rank r subspace for computing the stepsize [83]. Indirect comparisons can be drawn with other hard thresholding algorithms by contrasting the results presented here and those in [59].

The algorithms were implemented in Matlab R2013a with executables generated using the Matlab Compiler Ver. 4.18.1 and distributed across the cores at the IRIDIS HPC facility. Due to the importance of the convergence rate in recovering low rank matrices with a hard thresholding algorithm, the algorithms are terminated using stopping criteria derived from extensive testing and consistent with the stopping criteria detailed in [12, 77]. In particular, an algorithm terminates if one of the following conditions is satisfied: a maximum of 5000 iterations is met, the relative residual is small $\|y-\mathscr{A}(X_l)\|_2/\|y\|_2 \leqslant 10^{-5}$, or the multiplicative convergence rate is close to 1,

$$\left(\frac{\|y-\mathscr{A}(X_{l+15})\|_2}{\|y-\mathscr{A}(X_l)\|_2}\right)^{\frac{1}{15}} > 0.999. \tag{3.8}$$

Preliminary testing suggested our choice of the restart parameter θ in CGIHT projected, which was set to 5 for sensing ensemble $\mathscr{G}$; and for sensing ensemble $\mathscr{E}$ it was set to 3 if $\delta \leqslant 0.5$, otherwise it was set to 10. In this section, an algorithm is considered to have successfully recovered the matrix X when the algorithm returns a matrix $\hat{X}$ that satisfies

$$\frac{\|\hat{X}-X\|_F}{\|X\|_F} \leqslant 2\times 10^{-3}. \tag{3.9}$$

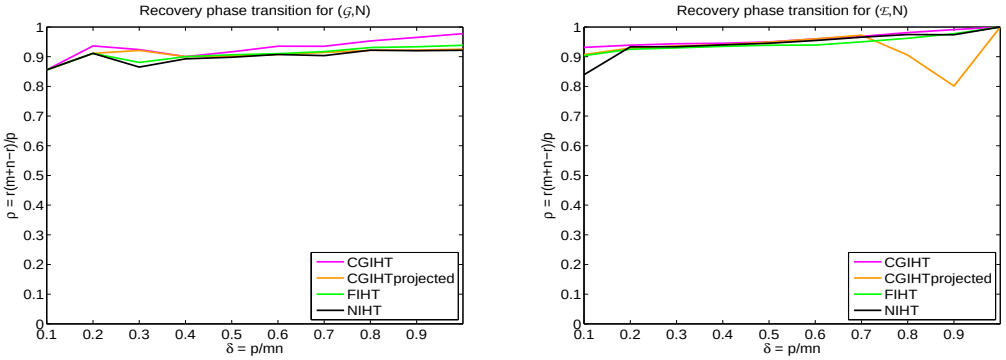


FIG. 7. 50% recovery probability logistic regression curves for matrix completion algorithms: CGIHT, CGIHT projected, FIHT, and NIHT. Horizontal axis δ and vertical axis ρ as defined in (3.7). (a) $\mathscr{G}$ with $m=n=80$, (b) $\mathscr{E}$ with $m=n=800$.

3.3.2 *Recovery Phase Transition Curves.* This section investigates when the matrix completion algorithms can successfully recover a rank r matrix from p linear measurements with p proportional to $r(m+n-r)$. For each problem class $(Oper,N)$ and (m,n) pair, we conduct tests with the undersampling ratio $\delta = p/(mn)$ taking 10 equispaced values from 0.1 to 1.0. The reconstruction algorithms are

substantially faster for sensing ensemble $\mathscr{E}$, as dense sensing operators drawn from $\mathscr{G}$ require p matrix inner products for the application of $\mathscr{A}(\cdot)$, which scales proportionally to n^4 in our testing environment. For this reason the tests for problem class $(\mathscr{G},N)$ are conducted for the small problem size $m = n = 80$, whereas for $(\mathscr{E},N)$ the tests are conducted for $m = n = 800$ which was observed in [77] to resolve the phase transition well. For each triple (m,n,p), we start from a sufficiently small rank r that the algorithm can successfully recover each sensed matrix in all 10 randomly drawn problem instances; we then increase the rank until the algorithm fails to recover the sensed matrix in each of ten random problem instances. Figure 7 displays the empirical phase transitions of the four tested algorithms, again defined by the logistic regression curve for the 50% successful recovery rate.

For the most interesting region of the phase space, namely $\delta < 1/2$, the recovery phase transition curves of CGIHT and CGIHT projected are always greater than 0.8 indicating both algorithms are able to successfully recover the randomly generated rank r matrices with the number of measurements $p = C \cdot r(m+n-r)$ for $C \leqslant 1.25$. For problem class $(\mathscr{G},N)$ with $\delta < 1/2$, the recovery phase transition curves for CGIHT and CGIHT projected are at least as high as the phase transition curve for FIHT, which in turn is either equivalent or superior to that of NIHT. For all $\delta \in (0.1,1)$, none of the other algorithms have a phase transition curve superior to the phase transition curve of CGIHT for matrix completion (Alg. 4). Likewise for $(\mathscr{E},n)$, CGIHT for matrix completion has the highest recovery phase transition curve for all values of $\delta \in (0.1,1)$, although the phase transition curves for all four algorithms are very similar. For $\delta > 0.7$, the observed decrease in the phase transition curve for CGIHT projected is an artifact of the restarting parameter θ, and is likely caused by excessively solving the sub-problem, and in so doing causing the subspace restricted conjugate gradient projections to converge to a non-optimal local minimum; decreasing the restarting parameter θ to 1 for $\delta > 0.7$ increases the phase transition curve to be that of NIHT, but at the cost of increased recovery time.

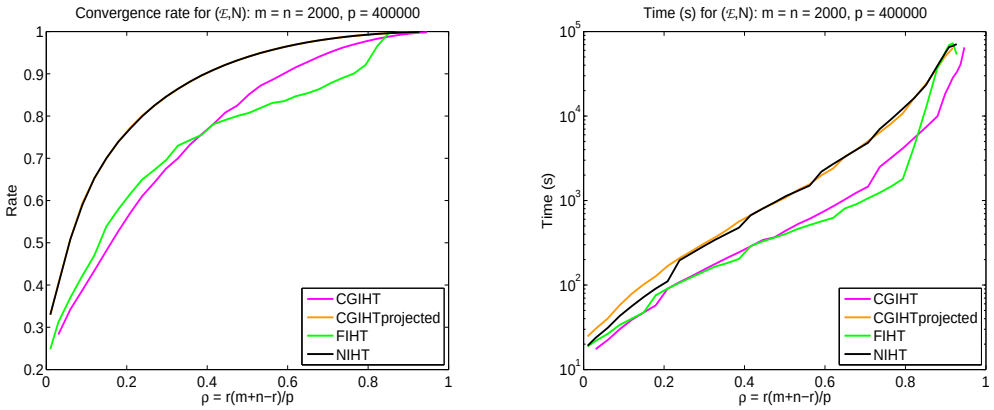


FIG. 8. Average convergence rate and average recovery time (s) of CGIHT, CGIHT projected, FIHT and NIHT for problem class $(\mathscr{E},N)$ with $m = n = 2000$, $p = 0.1 \times mn$ and r ranging from 1 to 96. Horizontal axis ρ defined in (3.7) and convergence rate of left panel defined in (3.8). Vertical scale for (b) is log(s).

3.3.3 *Recovery time dependence on ρ for $\delta = 1/10$.* In this section, we explore the average recovery time and per iteration asymptotic convergence rate for CGIHT, CGIHT projected, FIHT, and NIHT for matrix completion with a fixed value of $\delta = 1/10$. These experiments are performed with $m = n = 2000$, $p = mn/10$, and the rank r sampled from the set $\{r_j = 3j : j = 1,2,\ldots,j^\star\}$ where $r_{j^\star}$ is the smallest

rank for which an algorithm fails to recover a single rank $r_{j^\star}$ matrix in 100 random problem instances. Due to the computational burden of such large-scale testing, the results in this section were exclusively conducted for the problem class $(\mathscr{E},N)$. Figure 8(a) displays the average convergence rates computed according to (3.8) for each algorithm's successful recovery of a rank r matrix. Figure 8(b) provides a semi-log plot of the average computation recovery time.

The empirical results provided in Fig. 8 establish a computational advantage for CGIHT and FIHT when compared to NIHT and CGIHT projected which have indistinguishable convergence rates in Fig. 8(a). For the smallest ranks with $\rho \lesssim 0.5$, the accelerated algorithms CGIHT and FIHT have comparable average convergence rates and average recovery times. For $0.5 \lesssim \rho \lesssim 0.85$, FIHT appears to offer an advantage in terms of both convergence rate and recovery time although theoretical results indicate that FIHT will lose this advantage in the presence of noise. As the rank increases and forces ρ toward the recovery phase transition, CGIHT regains the computational advantage. The inferior rate of recovery for NIHT is expected due to the acceleration of convergence that is the hallmark of the other two algorithms. It should be noted that these hard thresholding algorithms require a computationally expensive partial singular value decomposition in each iteration and algorithms which avoid this burdensome task are likely to have improved average recovery times.

RECOMMENDATION The empirical results presented in this section show that CGIHT and FIHT for matrix completion have similar phase transition curves and rates of recovery which are superior to CGIHT projected and NIHT. We conjecture that the instability to noise observed for FIHT in compressed sensing will carry over to the matrix completion setting as the computation of the momentum step will still reintroduce the noise in each iteration. With its superior recovery phase transition curve for both Gaussian and entry sensing and its advantageous recovery rate near the phase transition curve, we recommend CGIHT for matrix completion (Alg. 4) for solving the low rank matrix completion problem (1.3) with a hard thresholding algorithm.

4. Proof of main results

4.1 *Proof of Theorem 2.2: CGIHT restarted for compressed sensing, Alg. 1*

The proof of Thm. 2.2 is partitioned here into three steps: a technical lemma, bounds on the update and orthogonalization stepsizes, and the analysis of the algorithm. The first two steps are presented as Lemmas 4.1 and 4.2. In CGIHT, each new approximation could possibly depend on all previous iterations. This will ultimately lead to a three term recurrence relation on the approximation error. The following induction argument assumes that we have established a base case prior to calling the lemma as in the proof of Thm. 2.2.

LEMMA 4.1 Suppose $c_0,\eta,\tau_1,\tau_2 \geqslant 0$ and let $\mu = \frac{1}{2}\left(\tau_1 + \sqrt{\tau_1^2+4\tau_2}\right)$. Assume $c_1 \leqslant \mu c_0 + \eta$ and define $c_l = \tau_1 c_{l-1} + \tau_2 c_{l-2} + \eta$ for $l \geqslant 2$. If $\tau_1+\tau_2 < 1$, then $\mu < 1$ and

$$c_l \leqslant \mu^l c_0 + \eta \sum_{i=0}^{l-1} \mu^i. \tag{4.1}$$

Proof. If $\tau_1+\tau_2<1$, then $\mu < \frac{1}{2}(\tau_1+\tau_2+1) < 1$. By assumption, (4.1) is valid for c_1. Assume it is

also valid for c_j with $j \leqslant l-1$. Then, since $\mu = \tau_1 + \frac{\tau_2}{\mu}$,

$$c_l \leqslant \tau_1 \left(\mu^{l-1} c_0 + \eta \sum_{i=0}^{l-2} \mu^i \right) + \frac{\tau_2}{\mu} \mu \left(\mu^{l-2} c_0 + \eta \sum_{i=0}^{l-3} \mu^i \right) + \eta \leqslant \mu^l c_0 + \eta \sum_{i=0}^{l-1} \mu^i.$$

□

Central to the performance of CGIHT is the calculation of stepsizes for the update and orthogonalization of the support set restricted conjugate gradient method. The stepsizes α_l are uniformly bounded near one with the same RIC bounds as the NIHT stepsize. The relative orthogonality is measured by β_l. When the support set has changed, β_l is defined to be zero, otherwise each β_l is uniformly bounded near zero. In the process of establishing these bounds on the stepsizes, we also bound the spectrum of a projection operator which appears regularly in this type of analysis.

LEMMA 4.2 By the definition of RICs of the measurement matrix A, the stepsize is uniformly bounded by

$$\frac{1}{1+U_k} \leqslant \alpha_l \leqslant \frac{1}{1-L_k} \tag{4.2}$$

and the orthogonalization coefficients are uniformly bounded by

$$|\beta_l| \leqslant \frac{(1+U_k)(L_k+U_k)}{(1-L_k)^2}. \tag{4.3}$$

Furthermore, if $Q, S \subset \{1,\ldots,n\}$ are two index sets and $ck = |Q \cup S|$, then for any z

$$\left\| \mathrm{Proj}_Q \left((I - \alpha_l A^* A) \mathrm{Proj}_S(z) \right) \right\| \leqslant \frac{U_{ck} + L_{ck}}{1-L_k} \left\| \mathrm{Proj}_S(z) \right\|. \tag{4.4}$$

Proof of Lemma 4.2. When $T_l \neq T_{l-1}$ the stepsize α_l is the same as proposed by NIHT, and is bounded directly as

$$\frac{1}{1+U_k} \leqslant \alpha_l = \frac{\|\mathrm{Proj}_{T_l}(r_l)\|^2}{\|A\mathrm{Proj}_{T_l}(r_l)\|^2} \leqslant \frac{1}{1-L_k}$$

by inverting the standard RIC bounds of A. If $T_l = T_{l-1}$, we utilize two important inequalities

$$\|A\mathrm{Proj}_{T_{l-1}}(p_{l-1})\| \leqslant \|A\mathrm{Proj}_{T_{l-1}}(r_{l-1})\|, \tag{4.5}$$

$$\|\mathrm{Proj}_{T_{l-1}}(r_{l-1})\| \leqslant \|\mathrm{Proj}_{T_{l-1}}(p_{l-1})\|. \tag{4.6}$$

The first inequality (4.5) follows from noting that $A\mathrm{Proj}_{T_{l-1}}(p_{l-1})$ is orthogonal to $A\mathrm{Proj}_{T_{l-1}}(p_{l-2})$ by construction (orthogonalization after the application of A is referred to as conjugate orthogonal); consequently, multiplying $r_{l-1} = p_{l-1} + \beta_{l-1} p_{l-2}$ by A and noting the orthogonality gives $\|A\mathrm{Proj}_{T_{l-1}}(r_{l-1})\|^2 = \|A\mathrm{Proj}_{T_{l-1}}(p_{l-1})\|^2 + \beta_{l-1}^2 \|A\mathrm{Proj}_{T_{l-1}}(p_{l-2})\|^2 \geqslant \|A\mathrm{Proj}_{T_{l-1}}(p_{l-1})\|^2$. The second inequality (4.6) follows from applying the Cauchy-Schwartz inequality to the conjugate gradient property [51, pg. 34] that $\|\mathrm{Proj}_{T_{l-1}}(r_{l-1})\|^2 = \left\langle \mathrm{Proj}_{T_{l-1}}(r_{l-1}), \mathrm{Proj}_{T_{l-1}}(p_{l-1}) \right\rangle$.

To establish the lower bound, we utilize (4.5) to observe

$$\frac{1}{\alpha_l} = \frac{\|A\mathrm{Proj}_{T_{l-1}}(p_{l-1})\|^2}{\|\mathrm{Proj}_{T_{l-1}}(r_{l-1})\|^2} \leqslant \frac{\|A\mathrm{Proj}_{T_{l-1}}(r_{l-1})\|^2}{\|\mathrm{Proj}_{T_{l-1}}(r_{l-1})\|^2} \leqslant 1 + U_k. \tag{4.7}$$

The upper bound follows from (4.6) since

$$\alpha_l = \frac{\|\text{Proj}_{T_{l-1}}(r_{l-1})\|^2}{\|A\text{Proj}_{T_{l-1}}(p_{l-1})\|^2} \leqslant \frac{\|\text{Proj}_{T_{l-1}}(p_{l-1})\|^2}{\|A\text{Proj}_{T_{l-1}}(p_{l-1})\|^2} \leqslant \frac{1}{1-L_k}. \tag{4.8}$$

Thus from (4.7) and (4.8), when $T_l = T_{l-1}$, α_l is also bounded by

$$\frac{1}{1+U_k} \leqslant \alpha_l \leqslant \frac{1}{1-L_k}. \tag{4.9}$$

For any index sets $Q, S \subset \{1,\dots,n\}$, let $A_{Q\cup S}$ be the submatrix formed by the columns of A indexed by the set $Q\cup S$. Then for any l and any vector z, the $|Q|$ nonzeros in the vector $\text{Proj}_Q\left((I-\alpha_l A^*A)\text{Proj}_S(z)\right)$ are a subset of the $|Q\cup S|$ nonzeros in the vector $\left(I-\alpha_l A^*_{Q\cup S}A_{Q\cup S}\right)\text{Proj}_S(z)$, so that

$$\left\|\text{Proj}_Q\left((I-\alpha_l A^*A)\text{Proj}_S(z)\right)\right\| \leqslant \left\|I-\alpha_l A^*_{Q\cup S}A_{Q\cup S}\right\| \|\text{Proj}_S(z)\| \leqslant \frac{U_{ck}+L_{ck}}{1-L_k}\|\text{Proj}_S(z)\| \tag{4.10}$$

where $ck = |Q\cup S|$. The bound on the operator $I-\alpha_l A^*_{Q\cup S}A_{Q\cup S}$ in terms of the RIC of A follows from Def. 2.1 and (4.9) as in [9, Lem. 5].

The orthogonalization factor β_l is equal to zero when $T_l \neq T_{l-1}$, and can be bounded when $T_l = T_{l-1}$ by the using alternative formula

$$\beta_l = -\frac{\langle \text{Proj}_{T_l}(r_l), \text{Proj}_{T_l}(A^*A\text{Proj}_{T_l}(p_{l-1}))\rangle}{\|A\text{Proj}_{T_l}(p_{l-1})\|^2} \tag{4.11}$$

and expressing $\text{Proj}_{T_l}(r_l)$ in terms of prior search directions. When $T_l = T_{l-1}$

$$\begin{aligned}
\text{Proj}_{T_l}(r_l) &= \text{Proj}_{T_l}(r_{l-1}) - \alpha_{l-1}\text{Proj}_{T_l}(A^*A\text{Proj}_{T_l}(p_{l-1})) \\
&= \text{Proj}_{T_l}(p_{l-1}) - \beta_{l-1}\text{Proj}_{T_l}(p_{l-2}) - \alpha_{l-1}\text{Proj}_{T_l}(A^*A\text{Proj}_{T_l}(p_{l-1})) \\
&= \text{Proj}_{T_l}\left((I-\alpha_{l-1}A^*A)\text{Proj}_{T_l}(p_{l-1})\right) - \beta_{l-1}\text{Proj}_{T_l}(p_{l-2}).
\end{aligned} \tag{4.12}$$

where the first equality follows by substituting $x_l = \text{Proj}_{T_l}(x_{l-1}+\alpha_{l-1}p_{l-1})$ into r_l, the second equality by substituting $r_{l-1} = p_{l-1} - \beta_{l-1}p_{l-2}$, and the final equality by rearrangement. Inserting (4.12) for $\text{Proj}_{T_l}(r_l)$ into (4.11) gives

$$\begin{aligned}
|\beta_l| &= \frac{\left|\langle \text{Proj}_{T_l}(I-\alpha_{l-1}A^*A\text{Proj}_{T_l}(p_{l-1})), \text{Proj}_{T_l}(A^*A\text{Proj}_{T_l}(p_{l-1}))\rangle\right|}{\|A\text{Proj}_{T_l}(p_{l-1})\|^2} \\
&\leqslant \frac{\|\text{Proj}_{T_l}(I-\alpha_{l-1}A^*A\text{Proj}_{T_l}(p_{l-1}))\|\|\text{Proj}_{T_l}(A^*A\text{Proj}_{T_l}(p_{l-1}))\|}{\|A\text{Proj}_{T_l}(p_{l-1})\|^2} \\
&\leqslant \frac{L_k+U_k}{1-L_k}\|\text{Proj}_{T_l}(p_{l-1}))\|\frac{(1+U_k)\|\text{Proj}_{T_l}(p_{l-1})\|}{(1-L_k)\|\text{Proj}_{T_l}(p_{l-1})\|^2} \\
&= \frac{1+U_k}{1-L_k}\cdot\frac{L_k+U_k}{1-L_k}
\end{aligned} \tag{4.13}$$

where the first equality follows by noting that $A\text{Proj}_{T_l} p_{l-1}$ and $A\text{Proj}_{T_l} p_{l-2}$ are orthogonal, the first inequality follows from the Cauchy-Schwarz inequality, and the second inequality from (4.10) with $Q = S = T_l$. □

With Lemmas 4.1 and 4.2, the proof of Thm. 2.2 is similar to the proofs of other hard thresholding algorithms. Following Foucart's general outline [47], the approximation error is bounded by observing that the hard thresholding operator produces the best k-sparse approximation to the current update. Lemma 4.2 offers a simple way to bound the approximation error. The proof is complicated by the fact that the current search direction can depend on all previous search directions. This is handled by establishing a recurrence relation and invoking Lem. 4.1.

Proof of Theorem 2.2. The proof begins following the proof of an analogous theorem for IHT [47]. Note that

$$\|x_l - w_{l-1}\|^2 = \|x_l - x + x - w_{l-1}\|^2 = \|x_l - x\|^2 + \|x - w_{l-1}\|^2 + 2\langle x_l - x, x - w_{l-1}\rangle \leqslant \|x - w_{l-1}\|^2 \tag{4.14}$$

where the inequality follows from x_l being, by definition, the k-sparse vector nearest to w_{l-1}. Canceling $\|x - w_{l-1}\|^2$ in (4.14) and rearranging gives the inequality

$$\|x_l - x\|^2 \leqslant 2\langle x_l - x, w_{l-1} - x\rangle = 2\langle x_l - x, \text{Proj}_{T_l \cup T}(w_{l-1} - x)\rangle,$$

where $\text{Proj}_{T_l \cup T}$ reflects the sparsity of $x_l - x$. Applying the Cauchy-Schwarz inequality and canceling a power of $\|x_l - x\|$ gives

$$\|x_l - x\| \leqslant 2\|\text{Proj}_{T_l \cup T}(w_{l-1} - x)\|. \tag{4.15}$$

In anticipation of substituting $w_{l-1} = x_{l-1} + \alpha_{l-1} p_{l-1}$ into (4.15), we note that p_{l-1} can be expressed as

$$p_{l-1} = r_{l-1} + \sum_{j=0}^{l-2} r_j \left(\Pi_{q=j+1}^{l-1} \beta_q \right). \tag{4.16}$$

Equation (4.15) can then be expressed purely in terms of x_j for $j \leqslant l$ by substituting $w_{l-1} = x_{l-1} + \alpha_{l-1} p_{l-1}$ into (4.15) with p_{l-1} given by (4.16) and replacing all instances of r_j with $A^*A(x - x_j) + A^*e$,

$$\begin{aligned} \|x_l - x\| &\leqslant 2\left\| \text{Proj}_{T_l \cup T} \left(x_{l-1} - x + \alpha_{l-1} r_{l-1} + \alpha_{l-1} \sum_{j=0}^{l-2} r_j \left(\Pi_{q=j+1}^{l-1} \beta_q \right) \right) \right\| \\ &\leqslant 2\left\| \text{Proj}_{T_l \cup T} \left((I - \alpha_{l-1} A^*A)(x_{l-1} - x) \right) \right\| + 2|\alpha_{l-1}| \left\| \text{Proj}_{T_l \cup T}(A^*e) \right\| \\ &\quad + 2|\alpha_{l-1}| \sum_{j=0}^{l-2} \left(\Pi_{q=j+1}^{l-1} \beta_q \right) \left(\left\| \text{Proj}_{T_l \cup T}(A^*A(x - x_j)) \right\| + \left\| \text{Proj}_{T_l \cup T}(A^*e) \right\| \right) \end{aligned} \tag{4.17}$$

We proceed by bounding each of the terms in the final inequality of (4.17). Let $\varepsilon_\alpha = \frac{1}{1 - L_k}$, $\varepsilon_\beta =$

$\frac{(1+U_k)(U_k+L_k)}{(1-L_k)^2}$, and $\varepsilon_\lambda = \frac{U_{3k}+L_{3k}}{1-L_k}$ denote the upper bounds established by Lem. 4.2. The we have

$$\left\|\text{Proj}_{T_l\cup T}\left((I-\alpha_{l-1}A^*A)(x_{l-1}-x)\right)\right\| \leqslant \varepsilon_\lambda \|x_{l-1}-x\|, \tag{4.18}$$

$$|\alpha_{l-1}| \left\|\text{Proj}_{T_l\cup T}(A^*e)\right\| \leqslant \varepsilon_\alpha (1+U_{2k})^{1/2} \|e\|, \tag{4.19}$$

$$|\alpha_{l-1}| \sum_{j=0}^{l-2} \left(\Pi_{q=j+1}^{l-1}\beta_q\right) \left\|\text{Proj}_{T_l\cup T}(A^*A(x-x_j))\right\| \leqslant \varepsilon_\alpha(1+U_{3k}) \sum_{j=0}^{l-2} \varepsilon_\beta^{l-j-1}\|x-x_j\|, \tag{4.20}$$

$$|\alpha_{l-1}| \sum_{j=0}^{l-2} \left(\Pi_{q=j+1}^{l-1}\beta_q\right) \left\|\text{Proj}_{T_l\cup T}(A^*e)\right\| \leqslant \varepsilon_\alpha(1+U_{2k})^{1/2}\|e\| \sum_{j=0}^{l-2} \varepsilon_\beta^{l-j-1}. \tag{4.21}$$

Applying (4.18)–(4.21) to (4.17), gathering like terms, and reindexing the sums,

$$\|x_l-x\| \leqslant 2\varepsilon_\lambda \|x_{l-1}-x\| + 2\varepsilon_\alpha(1+U_{3k}) \sum_{j=1}^{l-1} \varepsilon_\beta^j \|x_{l-j-1}-x\| + 2\varepsilon_\alpha(1+U_{2k})^{1/2}\|e\| \sum_{j=0}^{l-1} \varepsilon_\beta^j. \tag{4.22}$$

A simplified argument focused only the first iterate yields

$$\|x_1-x\| \leqslant 2\varepsilon_\lambda \|x_0-x\| + 2\varepsilon_\alpha(1+U_{2k})^{1/2}\|e\|. \tag{4.23}$$

Seeking a bound on $\|x_l-x\|$ purely in terms of $\|x_0-x\|$, define $c_0 = \|x_0-x\|$, $c_1 = 2\varepsilon_\lambda c_0 + \xi\|e\|$, and recursively define

$$c_l = 2\varepsilon_\lambda c_{l-1} + 2\varepsilon_\alpha(1+U_{3k}) \sum_{j=1}^{l-1} \varepsilon_\beta^j c_{l-j-1} + 2\varepsilon_\alpha(1+U_{2k})^{1/2}\|e\| \sum_{j=0}^{l-1} \varepsilon_\beta^j \qquad \text{for } l \geqslant 2. \tag{4.24}$$

Note that (4.23) shows that $\|x_1-x\| \leqslant c_1$ and (4.22) ensures $\|x_j-x\| \leqslant c_j$ for $j \geqslant 2$. By computing $c_l - \varepsilon_\beta c_{l-1}$ and isolating c_l, (4.24) can be rewritten as a three term recurrence relation

$$c_l = (2\varepsilon_\lambda+\varepsilon_\beta)c_{l-1} + 2(\varepsilon_\alpha(1+U_{3k})-\varepsilon_\lambda)\varepsilon_\beta c_{l-2} + 2\varepsilon_\alpha(1+U_{2k})^{1/2}\|e\|. \tag{4.25}$$

Now define

$$\tau_1 = 2\varepsilon_\lambda+\varepsilon_\beta, \quad \tau_2 = 2\left(\varepsilon_\alpha(1+U_{3k})-\varepsilon_\lambda\right)\varepsilon_\beta, \quad \text{and} \quad \xi = 2\varepsilon_\alpha(1+U_{2k})^{1/2},$$

so that $c_l = \tau_1 c_{l-1} + \tau_2 c_{l-2} + \xi\|e\|$. Let $\mu = \frac{1}{2}\left(\tau_1 + \sqrt{\tau_1^2+4\tau_2}\right)$ and observe that since $2\varepsilon_\lambda < \tau_1 < \mu$, $c_1 \leqslant \mu c_0 + \xi\|e\|$. If $\tau_1+\tau_2 < 1$, then Lem. 4.1 implies $\mu < 1$ and

$$\|x_l-x\| \leqslant c_l \leqslant \mu^l c_0 + \xi\|e\| \sum_{i=0}^{l-1} \mu^i \leqslant \mu^l c_0 + \frac{\xi}{1-\mu}\|e\| = \mu^l\|x_0-x\| + \frac{\xi}{1-\mu}\|e\|. \tag{4.26}$$

Finally, note that μ and ξ are equivalent to the definitions in (2.3) and the sufficient condition $\tau_1+\tau_2 < 1$ is satisfied when

$$\frac{(L_{3k}+U_{3k})(5-2L_k+3U_k)}{(1-L_k)^2} < 1. \tag{4.27}$$

□

A nearly identical proof with row-sparse matrices establishes an identical sufficient condition for row-sparse recovery via CGIHT restarted.

4.2 *Proof of Theorem 2.5: CGIHT projected for matrix completion, Alg. 5*

By establishing the standard orthogonality relationships for the conjugate gradient method when restricted to a fixed column space, the matrix completion analogue of the uniform bounds on the stepsize and orthogonalization coefficient follows the same general proof as that of Lemma 4.2.

LEMMA 4.3 By the definition of RICs of the measurement operator $\mathscr{A}$, the stepsize is uniformly bounded by

$$\frac{1}{1+U_r} \leqslant \alpha_l \leqslant \frac{1}{1-L_r} \tag{4.28}$$

and the orthogonalization coefficients are uniformly bounded by

$$|\beta_l| \leqslant \frac{(1+U_r)(L_r+U_r)}{(1-L_r)^2}. \tag{4.29}$$

Furthermore, if Q,S define column spaces with $cr = \text{rank}(\text{Proj}_{Q\cup S})$, then for any Z

$$\left\|\text{Proj}_Q\left((I-\alpha_l\mathscr{A}^*\mathscr{A})\text{Proj}_S(Z)\right)\right\| \leqslant \frac{U_{cr}+L_{cr}}{1-L_r}\left\|\text{Proj}_S(Z)\right\|. \tag{4.30}$$

With Lemma 4.3, we prove the convergence guarantee for CGIHT projected for matrix completion, Alg. 5. Unlike in the discrete compressed sensing problem, CGIHT projected for matrix completion suffers from additional computational burdens and reduced empirical performance. Although we require the projected version in order to ensure orthogonality and establish a convergence result, the non projected version, CGIHT for matrix completion (Alg. 4), is recommended for implementations.

Proof of Theorem 2.5. The early steps of the proof of Thm. 2.5 follows the proof of Thm. 2.2 through to (4.15), giving the upper bound

$$\|X_l - X\| \leqslant 2\|\text{Proj}_{U_l\cup U}(W_{l-1}-X)\|. \tag{4.31}$$

We seek a bound on $\|\text{Proj}_{U_l\cup U}(W_{l-1}-X)\|$ which initially is split into two cases based on the value of Restart_flag. When Restart_flag $= 1$

$$W_{l-1}-X = X_{l-1}-X+\alpha_{l-1}R_{l-1} = (I-\alpha_{l-1}\mathscr{A}^*\mathscr{A})(X_{l-1}-X)+\alpha_{l-1}\mathscr{A}^*(e).$$

Applying the triangle inequality, Lem. 4.3, and Def. 2.4,

$$\begin{aligned}\|\text{Proj}_{U_l\cup U}(X_{l-1}-X+\alpha_{l-1}R_{l-1})\| &\leqslant \|\text{Proj}_{U_l\cup U}\left((I-\alpha_{l-1}\mathscr{A}^*\mathscr{A})\text{Proj}_{U_{l-1}\cup U}(X_{l-1}-X)\right)\| \\ &\quad + |\alpha_{l-1}|\|\text{Proj}_{U_l\cup U}\mathscr{A}^*(e)\| \\ &\leqslant \frac{U_{3r}+L_{3r}}{1-L_r}\|X_{l-1}-X\| + \frac{(1+U_{2r})^{1/2}}{1-L_r}\|e\|. \end{aligned} \tag{4.32}$$

Therefore, if Restart_flag $= 1$,

$$\|\text{Proj}_{U_l\cup U}(W_{l-1}-X)\| \leqslant \frac{U_{3r}+L_{3r}}{1-L_r}\|X_{l-1}-X\| + \frac{(1+U_{2r})^{1/2}}{1-L_r}\|e\|. \tag{4.33}$$

On the other hand, when Restart_flag $= 0$ (continuing with the same column space)

$$W_{l-1}-X = X_{l-1}-X+\alpha_{l-1}\text{Proj}_{U_{l-1}}P_{l-1} = X_{l-1}-X+\alpha_{l-1}+\alpha_{l-1}R_{l-1}-\alpha_{l-1}\left(R_{l-1}-\text{Proj}_{U_{l-1}}(P_{l-1})\right). \tag{4.34}$$

When Restart_flag $= 0$, the restarting parameter gives the bound,

$$\frac{\left\| R_{l-1} - \text{Proj}_{U_{l-1}}(P_{l-1}) \right\|}{\|\text{Proj}_{U_{l-1}}(R_{l-1})\|} \leqslant \theta < \theta_0 = c\left(\frac{U_{3r}+L_{3r}}{1+U_{2r}}\right). \tag{4.35}$$

Writing $R_{l-1} = \mathscr{A}^*(\mathscr{A}(X - X_{l-1})) + \mathscr{A}^*(e)$, (4.35) leads to

$$\begin{aligned}
\left\| R_{l-1} - \text{Proj}_{U_{l-1}}(P_{l-1}) \right\| &\leqslant \theta \|\text{Proj}_{U_{l-1}}(R_{l-1})\| \\
&\leqslant \theta \|\text{Proj}_{U_{l-1}}(\mathscr{A}^*(\mathscr{A}(X - X_{l-1})) + \mathscr{A}^*(e))\| \\
&\leqslant \theta \left(\|\text{Proj}_{U_{l-1}\cup U}(\mathscr{A}^*(\mathscr{A}(X - X_{l-1})))\| + \|\text{Proj}_{U_{l-1}}(\mathscr{A}^*(e))\| \right) \\
&\leqslant \theta \left((1+U_{2r})\|X_{l-1} - X\| + (1+U_r)^{1/2}\|e\| \right) \\
&\leqslant c(U_{3r}+L_{3r}) \left(\|X_{l-1} - X\| + \frac{(1+U_r)^{1/2}}{1+U_{2r}}\|e\| \right).
\end{aligned} \tag{4.36}$$

So, in this case with Restart_flag $= 0$,

$$\begin{aligned}
\|\text{Proj}_{U_l \cup U}(W_{l-1} - X)\| &\leqslant \|\text{Proj}_{U_l \cup U}(X_{l-1} - X + \alpha_{l-1} R_{l-1})\| + |\alpha_{l-1}| \left\| R_{l-1} - \text{Proj}_{U_{l-1}}(P_{l-1}) \right\| \\
&\leqslant \left(\frac{U_{3r}+L_{3r}}{1-L_r}\|X_{l-1} - X\| + \frac{(1+U_{2r})^{1/2}}{1-L_r}\|e\| \right) \\
&\quad + c\frac{U_{3r}+L_{3r}}{1-L_r} \left(\|X_{l-1} - X\| + \frac{(1+U_r)^{1/2}}{1+U_{2r}}\|e\| \right) \\
&\leqslant (1+c)\frac{U_{3r}+L_{3r}}{1-L_r}\|X_{l-1} - X\| \\
&\quad + \frac{(1+U_{2r})^{1/2}}{1-L_r} \left(1 + c\frac{U_{3r}+L_{3r}}{1+U_{2r}} \right) \|e\|.
\end{aligned} \tag{4.37}$$

Clearly, the bound on $\|\text{Proj}_{U_l \cup U}(W_{l-1} - X)\|$ from (4.33) is smaller than the bound in (4.37); thus, (4.37) applies to both cases. Therefore, with

$$\mu = 2(1+c)\frac{U_{3r}+L_{3r}}{1-L_r} \quad \text{and} \quad \xi = 2\frac{(1+U_{2r})^{1/2}}{1-L_r}(1+\theta_0),$$

equations (4.31), (4.33), and (4.37) combine to show that

$$\|X_l - X\| \leqslant \mu \|X_{l-1} - X\| + \xi \|e\|. \tag{4.38}$$

If $\mu < 1$, a straightforward induction argument with (4.38) provides the desired final bound,

$$\|X_l - X\| \leqslant \mu^l \|X_0 - X\| + \frac{\xi}{1-\mu}\|e\|. \tag{4.39}$$

□

The restarting parameter θ plays an important role in CGIHT projected for matrix completion. This theorem requires $\theta < c\frac{U_{3r}+L_{3r}}{1+U_{2r}}$ and that $\mu = 2(1+c)\frac{U_{3r}+L_{3r}}{1-L_r} < 1$. First, consider the two extreme cases of $c = 0$ or $c = \infty$. When $c = 0$, the restarting condition is met at every iteration and the algorithm is identical to NIHT for matrix completion. Moreover, the sufficient condition $\mu = 2\frac{U_{3r}+L_{3r}}{1-L_r} < 1$ is identical to the sufficient condition for NIHT for matrix completion [77]. At the other extreme, the algorithm will never restart and instead will project the observations y onto the column space spanned by the r principal left singular vectors of $\mathscr{A}^*(y)$. This is the one step hard thresholding algorithm and the theorem could only apply when $\mathscr{A}$ is an isometry, i.e. $U_{3r} = L_{3r} = 0$. For the values of $c \in (0,\infty)$, CGIHT traces between matrix completion versions of NIHT and HTP. As c increases, the sufficient condition $\mu < 1$ is satisfied by a smaller set of linear operators $\mathscr{A}$ while the algorithm is permitted to take more conjugate gradient steps on each subspace. Eventually, the subspace restricted conjugate gradient method will force $\|\text{Proj}_{U_{l-1}} R_{l-1}\|$ to a small enough value that the restarting criterion is satisfied. In other words, the parameter c determines an error tolerance for a projection of y onto each subspace spanned by U_{l-1}.

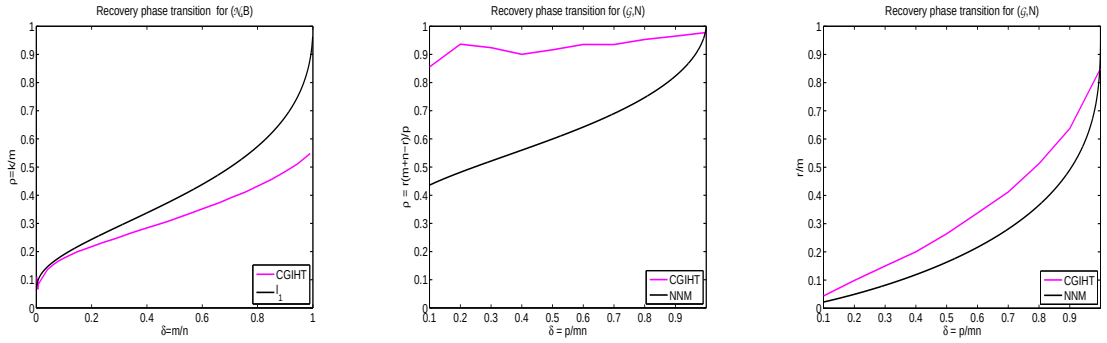
The proof of Thm. 2.3 follows the proof of Thm. 2.5 and is omitted for brevity.

5. Conclusion and future directions

The large scale testing of CGIHT on synthetic problems in Sec. 3 shows it to typically outperform existing hard thresholding algorithms for each of compressed sensing, row sparse approximation, and matrix completion. Though no single variant of CGIHT is universally superior for each of these questions, their differing performance reflects the dominant aspects of the problem. Despite CGIHT restarted for compressed sensing having a per iteration complexity that is modestly lower than that of CGIHT, it is the latter that is typically faster in practice and possesses a larger recovery region. In contrast, CGIHT restarted shows performance superior to that of CGIHT for row sparse approximation where the discrete nature of the problem is further emphasized; again, the superior performance is both in terms of recovery time and a unusually large recovery region, compared to other hard thresholding algorithms, as the number of vectors is increased. For matrix completion it is the non restarted CGIHT that is fastest and with the largest recovery region. CGIHT projected is not observed to be superior for any of the problems tested, but we conjecture that it may well be superior for compressed sensing and row sparse approximation once the problem size is sufficiently large so that the discrete support set restarting condition would be activated for an increasing fraction of the iterates which would degrade the asymptotic convergence rate. However, it is worth noting that this conjecture is not realized for compressed sensing problems tested at ambient dimensions up to a million, nor for the largest matrix completion problems tested here.

The efficacy of CGIHT across these three problems suggests it may well be similarly effective for other constrained underdetermined linear systems of equations. In future work we suggest extending CGIHT to related problems such as separation of sparse outliers from low rank matrices and estimation of sparse inverse covariance matrices. In a different direction, the non restated variant of CGIHT was typically the best performing variant, but unfortunately lacks a recovery guarantee. Developing a recovery guarantee is likely to require an analysis of the fraction of a support set or subspace correctly identified at each iteration. It may also be possible to extend CGIHT to include exploration of larger support sets or subspaces analogous to CoSaMP [69], ADMiRA [60], and the ALPS family of algorithms [25, 58] without the negative side effects.

Though CGIHT has been observed to, overall, be more computationally efficient than other hard thresholding algorithms, a direct comparison with other classes of algorithms is needed in order to



(a) (b) (c)

FIG. 9. 50% recovery probability logistic regression curves: (a) CGIHT for ensemble $(\mathcal{N},B)$ with $n=2^{12}$ (black) and the analytic ℓ^1 regularization phase transition [31], (b) CGIHT for ensemble $(\mathcal{G},N)$ with $n=m=80$ (black) and the analytic Schatten-1 regularization phase transition [1, 38], (c) the same as (b) with a different vertical axis.

determine its overall efficacy. Particularly important classes of algorithms are convex regularizations, see for instance [3–6, 18, 44, 50, 65, 71, 79, 82, 84, 86], and approximate message passing algorithms (AMP), see for instance [26, 36, 38–40, 52] and references therein. As a preliminary comparison of recovery ability, we contrast the empirically observed phase transition of CGIHT with Gaussian sensing and the analytic asymptotic phase transitions of the convex regularizations and AMP algorithms. Fig. 9(a) shows the 50% recovery phase transition of CGIHT for $(\mathcal{N},B)$ with the analytic phase transition [1, 31, 33, 38] associated with solving (2.10) where $\|y-Az\|_2=0$. For the most challenging, binary, vector ensemble, the recovery phase transition of CGIHT is strictly below that of ℓ_1 regularization, although the relative difference between the recovery phase transitions decreases as m/n moves toward zero. In this region of greatest interest for applications, the similar phase transitions suggest the dominant difference between the algorithms will be recovery time rather than recovery ability. Fig. 9(b) shows the 50% recovery phase transition of CGIHT for $(\mathcal{G},N)$ with the analytic phase transition [1, 35] associated with solving (2.12) where $\|y-\mathcal{A}(z)\|_2=0$. In contrast with the compressed sensing setting, the recovery phase transition for CGIHT is uniformly above that of solving (2.12). In particular, this difference is enhanced as p/mn decreases to zero, which is again the region of greatest interest for applications. This indicates that CGIHT, as well as other iterative hard thresholding algorithms, see Fig. 7, are less reliant on the smallness of the rank of the measured matrix. Fig. 9(c) shows the same curves as in Fig. 9(b), though with the vertical axis r/m, to aid in comparison with other manuscripts in the matrix completion literature which use this alternative scaling. Although large-scale empirical comparisons of CGIHT and algorithms designed for solving (2.12) are not currently available, preliminary comparisons with some existing software indicate CGIHT is dramatically more efficient than some specifically designed algorithms for (2.12) such as SVT [18].

Funding

This work was supported by the National Science Foundation [DMS 1112612 to JDB], a Harris Faculty Fellowship [2013-2014 to JDB], an Nvidia Professor Partnership [to JT], and the China Scholarship Council [to KW].

Acknowledgment

The authors acknowledge the use of the IRIDIS High Performance Computing Facility, and associated support services at the University of Southampton, in the completion of this work.

References

[1] AMELUNXEN, D., LOTZ, M., MCCOY, M. B. & TROPP, J. A. (2014) Living on the edge: A geometric theory of phase transitions in convex optimization. *Information and Inference: A Journal of the IMA*, **3**(3), 224–294.

[2] BAH, B. & TANNER, J. (2010) Improved bounds on restricted isometry constants for Gaussian matrices. *SIAM Journal on Matrix Analysis*, **31**(5), 2882–2898.

[3] BECK, A. & TEBOULLE, M. (2009) A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, **2**(1), 183–202.

[4] ——— (2014) A fast dual proximal gradient algorithm for convex minimization and applications. *Oper. Res. Lett.*, **42**(1), 1–6.

[5] BERG, E. V. D. & FRIEDLANDER, M. P. (2008) Probing the Pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, **31**(2), 890–912.

[6] BIOUCAS-DIAS, J. M. & FIGUEIREDO, M. A. T. (2007) A new TwIST: Two-step iterative shrinkage/thresholding algorithms for image restoration. *IEEE Trans. Image Process.*, **16**(12), 2992–3004.

[7] BLANCHARD, J. D., CARTIS, C. & TANNER, J. (2011) Compressed sensing: How sharp is the restricted isometry property?. *SIAM Review*, **53**(1), 105–125.

[8] BLANCHARD, J. D., CARTIS, C., TANNER, J. & THOMPSON, A. (2011) Phase transitions for greedy sparse approximation algorithms. *Appl. Comput. Harmon. Anal.*, **30**(2), 188–203.

[9] BLANCHARD, J. D., CERMAK, M., HANLE, D. & JING, Y. (2014) Greedy algorithms for joint sparse recovery. *IEEE Trans. Sig. Proc.*, **62**(7), 1694–1704.

[10] BLANCHARD, J. D. & TANNER, J. (2013a) GAGA: GPU accelerated greedy algorithms. Version 1.1.0. [Online]. Available: www.gaga4cs.org.

[11] ——— (2013b) GPU accelerated greedy algorithms for compressed sensing. *Math. Prog. Computation*, **5**(3), 267–304.

[12] ——— (2015) Performance comparisons of greedy algorithms in compressed sensing. *Num. Lin. Alg. Appl.*, **22**(2), 254–282.

[13] BLANCHARD, J. D., TANNER, J. & WEI, K. (2015) Conjugate gradient iterative hard thresholding: Observed noise stability for compressed sensing. *IEEE Trans. Signal Processing*, **63**(2), 528–537.

[14] BLUMENSATH, T. (2012) Accelerated iterative hard thresholding. *Signal Processing*, **92**, 752–756.

[15] BLUMENSATH, T. & DAVIES, M. E. (2009) Iterative hard thresholding for compressed sensing. *Appl. Comput. Harmon. Anal.*, **27**(3), 265–274.

[16] ——— (2010) Normalised iterative hard thresholding; guaranteed stability and performance. *IEEE Selected Topics in Signal Processing*, **4**(2), 298–309.

[17] BRUCKSTEIN, A. M., DONOHO, D. L. & ELAD, M. (2009) From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, **51**(1), 34–81.

[18] CAI, J.-F., CANDÈS, E. J. & SHEN, Z. (2010) A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, **20**(4), 1956–1982.

[19] CANDÈS, E. & RECHT, B. (2009) Exact matrix completion via convex optimization. *Foundations of Comp. Math.*, **9**(6), 717–772.

[20] CANDÈS, E. J. (2006) Compressive sampling. in *International Congress of Mathematicians. Vol. III*, pp. 1433–1452. Eur. Math. Soc., Zürich.

[21] CANDÈS, E. J., ROMBERG, J. & TAO, T. (2006) Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inform. Theory*, **52**(2), 489–509.

[22] CANDÈS, E. J. & TAO, T. (2005) Decoding by linear programming. *IEEE Trans. Inform. Theory*, **51**(12), 4203–4215.

[23] ——— (2006) Near-optimal signal recovery from random projections: Universal encoding strategies?. *IEEE Trans. Inform. Theory*, **52**(12), 5406–5425.

[24] CARTIS, C. & THOMPSON, A. (2015) A new and improved quantitative recovery analysis for iterative hard thresholding algorithms in compressed sensing. *IEEE Trans. Inform. Theory*, **61**(4), 1–24.

[25] CEVHER, V. (2011) An ALPS view of sparse recovery. in *Acoustics Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pp. 5808 –5811.

[26] CHANDAR, V., SHAH, D. & WORNELL, G. W. (2010) A simple message-passing algorithm for compressed sensing. in *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, pp. 1968–1972.

[27] CHEN, S. S., DONOHO, D. L. & SAUNDERS, M. A. (2001) Atomic decomposition by basis pursuit. *SIAM Rev.*, **43**(1), 129–159 (electronic), Reprinted from SIAM J. Sci. Comput. **2**0 (1998), no. 1, 33–61.

[28] DAI, W. & MILENKOVIC, O. (2009) Subspace pursuit for compressive sensing signal reconstruction. *IEEE Trans. Inform. Theory*, **55**(5), 2230–2249.

[29] DAI, W., MILENKOVIC, O. & KERMAN, E. (2011) Subspace evolution and transfer (SET) for low-rank matrix completion. *IEEE Transactions on Signal Processing*, **59**(7), 3120–3132.

[30] DEVOLDER, O., GLINEUR, F. & NESTEROV, Y. (2014) First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming Series A*.

[31] DONOHO, D. L. (2004) Neighborly polytopes and sparse solution of underdetermined linear equations. Technical Report, Department of Statistics, Stanford University.

[32] ——— (2006a) Compressed sensing. *IEEE Trans. Inform. Theory*, **52**(4), 1289–1306.

[33] ——— (2006b) High-dimensional centrally symmetric polytopes with neighborliness proportional to dimension. *Discrete Comput. Geom.*, **35**(4), 617–652.

[34] DONOHO, D. L. & GAVISH, M. (2014) Minimax risk of matrix denoising by singular value thresholding. *Ann. Statist.*, **42**(6), 2413–2440.

[35] DONOHO, D. L., GAVISH, M. & MONTANARI, A. (2013) The phase transition of matrix recovery from Gaussian measurements matches the minimax MSE of matrix denoising. *Proc. Natl. Acad. Sci. USA*, **110**(21), 8405–8410.

[36] DONOHO, D. L., JOHNSTONE, I. & MONTANARI, A. (2013) Accurate prediction of phase transitions in compressed sensing via a connection to minimax denoising. *IEEE Trans. Inform. Theory*, **59**(6), 3396–3433.

[37] DONOHO, D. L. & MALEKI, A. (2010) Optimally tuned iterative thresholding algorithms for compressed sensing. *IEEE Selected Topics in Signal Processing*, **4**(2), 330–341.

[38] DONOHO, D. L., MALEKI, A. & MONTANARI, A. (2009) Message-passing algorithms for compressed sensing. *Proc. Natl. Acad. Sci. USA*, **106**(45), 18914–18919.

[39] ——— (2010a) Message passing algorithms for compressed sensing: I. motivation and construction. in *Proc. IEEE Inform. Theory Workshop*.

[40] ——— (2010b) Message Passing Algorithms for Compressed Sensing: II. analysis and validation. in *Proc. IEEE Inform. Theory Workshop*.

[41] DONOHO, D. L. & TANNER, J. (2005) Sparse nonnegative solution of underdetermined linear equations by linear programming. *Proc. Natl. Acad. Sci. USA*, **102**(27), 9446–9451 (electronic).

[42] ——— (2009) Counting faces of randomly projected polytopes when the projection radically lowers dimension. *J. AMS*, **22**(1), 1–53.

[43] ELDAR, Y. C. & KUTYNIOK, G. (2012) *Compressed sensing: Theory and applications*. Cambridge University Press.

[44] FIGUEIREDO, M. A. T., NOWAK, R. D. & WRIGHT, S. J. (2007) Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Selected Topics in Signal Processing*, **1**(4), 586–597.

[45] FOUCART, S. (2011a) Hard thresholding pursuit: An algorithm for compressive sensing. *SIAM Journal on Numerical Analysis*, **49**(6), 2543–2563.

[46] ——— (2011b) Recovering jointly sparse vectors via hard thresholding pursuit. in *Proc. of SAMPTA*. Online.

[47] FOUCART, S. (2012) Sparse recovery algorithms: Sufficient conditions in terms of restricted isometry constants. in *Approximation Theory XIII: San Antonio 2010*, ed. by M. Neamtu, & L. Schumaker, vol. 13 of *Springer Proceedings in Mathematics*, pp. 65–77. Springer New York.

[48] FOUCART, S. & RAUHUT, H. (2013) *A mathematical introduction to compressive sensing*. Birkhauser.

[49] GARG, R. & KHANDEKAR, R. (2009) Gradient descent with sparsification: An iterative algorithm for sparse recovery with restricted isometry property. in *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 337–344, New York, NY, USA. ACM.

[50] GOLDSTEIN, T. & SETZER, S. (2004) High-order methods for basis pursuit. Computational applied mathematics (CAM) Technical Report, Department of Mathematics, University of California at Los Angeles.

[51] GREENBAUM, A. (1997) *Iterative methods for solving linear systems*, vol. 17 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

[52] GUO, C. & DAVIES, M. E. (2015) Near optimal compressed sensing without priors: Parametric SURE approximate message passing. *IEEE Trans. Signal Processing*, **63**(8), 2130–2141.

[53] HALDAR, J. P. & HERNANDO, D. (2009) Rank-constrained solutions to linear matrix equations using PowerFactorization. *IEEE Signal Processing Letters*, **16**(7), 584 –587.

[54] HESTENES, M. R. & STIEFEL, E. (1952) Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, **49**, 409436.

[55] HORN, R. A. & JOHNSON, C. R. (1990) *Matrix analysis*. Cambridge University Press.

[56] JAIN, P., MEKA, R. & DHILLON, I. (2010) Guaranteed rank minimization via singular value projection. *Proc. Neural Information Processing Systems Conf. (NIPS)*, pp. 937–945.

[57] KESHAVAN, R. H., MONTANARI, A. & OH, S. (2010) Matrix completion from a few entries. *IEEE Trans. Inform. Theory*, **56(6)**, 2980–2998.

[58] KYRILLIDIS, A. & CEVHER, V. (2012) Matrix ALPS: Accelerated low rank and sparse matrix reconstruction. in *Statistical Signal Processing Workshop (SSP), 2012 IEEE*, pp. 185 –188.

[59] KYRILLIDIS, A. & CEVHER, V. (2014) Matrix recipes for hard thresholding methods. *Journal of Mathematical Imaging and Vision*, **48**(2), 235–265.

[60] LEE, K. & BRESLER, Y. (2010) ADMiRA: Atomic decomposition for minimum rank approximation. *IEEE Trans. Inform. Theory*, **56**(9), 4402 –4416.

[61] LEVIATAN, D. & TEMLYAKOV, V. N. (2006) Simultaneous approximation by greedy algorithms. *Adv. Comput. Math.*, **25**(1-3), 73–90.

[62] LEWIS, A. S. & MALICK, J. (2008) Alternating projections on manifolds. *Mathematics of Operations Research*, **33**, 216–234.

[63] LUTOBORSKI, A. & TEMLYAKOV, V. N. (2003) Vector greedy algorithms. *J. Complexity*, **19**(4), 458–473.

[64] MA, S. Q., GOLDFARB, D. & CHEN, L. F. (2011a) Convergence of fixed-point continuation algorithms for matrix rank minimization. *Foundations of Computational Mathematics*, **11**(2), 183–210.

[65] ——— (2011b) Fixed point and Bregman iterative methods for matrix rank minimization. *Mathematical Programming Series A.*, **128**(1), 321–353.

[66] MALEKI, A. (2009) Coherence analysis of iterative thresholding algorithms. in *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, pp. 236 –243.

[67] MEYER, G., BONNABEL, S. & SEPULCHRE, R. (2011) Linear regression under fixed-rank constraints: a Riemannian approach. in *Proc. of the 28th International Conference on Machine Learning (ICML2011), Bellevue (USA)*.

[68] NATARAJAN, B. K. (1995) Sparse approximate solutions to linear systems. *SIAM J. Comput.*, **24**(2), 227–234.

[69] NEEDELL, D. & TROPP, J. A. (2009) CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Appl. Comp. Harm. Anal.*, **26**(3), 301–321.

[70] NESTEROV, Y. (2004) *Introductory lectures on convex optimization : A basic course*, Applied optimization. Kluwer Academic Publ., Boston, Dordrecht, London.

[71] ——— (2007) Gradient methods for minimizing composite objective functions. CORE Discussion Paper 2007/76, Center for Operations Research and Econometrics, Université Catholique de Louvain, Belgium.

[72] POWELL, M. J. D. (1976) Some convergence properties of the conjugate gradient method. *Mathematical Programming*, **11**, 42–49.

[73] RAUHUT, H., ROMBERG, J. & TROPP, J. A. (2012) Restricted isometries for partial random circulant matrices. *Appl. Comput. Harmon. Anal.*, **32**(3), 242–254.

[74] RECHT, B., FAZEL, M. & PARRILO, P. A. (2010) Guaranteed minimum rank solutions to linear matrix equations via nuclear norm minimization. *SIAM Review*, **52**(3), 471–501.

[75] RECHT, B., XU, W. & HASSIBI, B. (2011) Null space conditions and thresholds for rank minimization. *Mathematical Programming Series B*, **127**, 175–211.

[76] RUDELSON, M. & VERSHYNIN, R. (2008) On sparse reconstruction from Fourier and Gaussian measurements. *Comm. Pure Appl. Math.*, **61**(8), 1025–1045.

[77] TANNER, J. & WEI, K. (2013) Normalized iterative hard thresholding for matrix completion. *SIAM J. Scientfic Comput.*, **35**(5), S104–S125.

[78] TEMLYAKOV, V. N. (2004) A remark on simultaneous greedy approximation. *East J. Approx.*, **10**(1-2), 17–25.

[79] TOH, K.-C. & YUN, S. (2010) An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pacific Journal of Optimization*, **6**(3), 615–640.

[80] TROPP, J. A. (2006) Algorithms for simultaneous sparse approximation. Part II: Convex relaxation. *Signal Processing*, **86**, 589–602.

[81] TROPP, J. A., GILBERT, A. C. & STRAUSS, M. J. (2006) Algorithms for simultaneous sparse approximation. Part I: Greedy pursuit. *Signal Processing*, **86**, 572–588.

[82] TROPP, J. A. & WRIGHT, S. J. (2010) Computational methods for sparse solution of linear inverse problems. *Proceedings of the IEEE*, **98**(6), 948–958.

[83] WEI, K. (2014) Efficient algorithms for compressed sensing and matrix completion. Doctoral thesis, University of Oxford.

[84] WRIGHT, S. J., NOWAK, R. D. & FIGUEIREDO, M. A. T. (2008) Sparse reconstruction by separable approximation. *Proc. International Conference on Acoustics, Speech, and Signal Processing*.

[85] XU, W. & HASSIBI, B. (2008) Precise stability phase transitions for ℓ_1 minimization: A unified geometric framework. *IEEE Trans. Inform. Theory*, **57**(10), 6894–6919.

[86] YIN, W., OSHER, S., GOLDFARB, D. & DARBON, J. (2008) Bregman iterative algorithms for ℓ^1-minimization with applications to compressed sensing. *SIAM Journal on Imaging Science*, **1**(1), 143–168.

[87] YUAN, Y. X. (1993) Analysis of conjugate gradient method. *Optimization Methods and Software*, **2**, 19–29.