Conjugate Gradient Iterative Hard Thresholding: Observed Noise Stability for Compressed Sensing

Jeffrey D. Blanchard, Jared Tanner, and Ke Wei

***Abstract*—Conjugate Gradient Iterative Hard Thresholding (CGIHT) for compressed sensing combines the low per iteration computational cost of simple line search iterative hard thresholding algorithms with the improved convergence rates of more sophisticated sparse approximation algorithms. This article shows that the average case performance of CGIHT is robust to additive noise well beyond its theoretical guarantees, and in this setting is typically the fastest iterative hard thresholding algorithm for sparse approximation. Moreover, CGIHT is observed to benefit more than other iterative hard thresholding algorithms when jointly considering multiple sparse vectors whose sparsity patterns coincide.**

I. Introduction

With the advent of compressed sensing [1], [2] and related questions, there has been significant development of algorithms designed to recover sparse solutions to underdetermined linear systems of equations. Consider $y = Ax + e$ where $x \in \mathbb{R}^n$ is k-sparse (i.e. the number of nonzeros entries in x is at most k, denoted $\|x\|_0 \leq k$), $A \in \mathbb{R}^{m\times n}$ and $e \in \mathbb{R}^m$ representing model misfit between representing y with k columns of A and/or additive noise. The compressed sensing recovery question is to identify the minimizer

$$\hat{x} = \underset{z\in\mathbb{R}^n}{\arg\min} \|y - Az\|_2 \quad \text{subject to} \quad \|z\|_0 \leq k. \tag{1}$$

Algorithms for the solution of (1) can, suitably extended, also be used to solve the *row-sparse* approximation problem. For $X \in \mathbb{R}^{n\times r}$, X is k-row-sparse when X has at most k rows containing nonzero values, denoted $\|X\|_{R0} \leq k$. This is equivalent to the *multiple measurement vector* or *joint sparsity* settings in compressed sensing where the columns of X are each k-sparse vectors with a common, joint support set. Consider the model $Y = AX + E$, in this case, iterative hard thresholding algorithms attempt to identify the k-row-sparse minimizer

$$\hat{X} = \underset{Z\in\mathbb{R}^{n\times r}}{\arg\min} \|Y - AZ\|_F \quad \text{subject to} \quad \|Z\|_{R0} \leq k. \tag{2}$$

JDB is with the Department of Mathematics and Statistics, Grinnell College, Grinnell, IA 50112. JDB is supported by the National Science Foundation [DMS 11126152] and a 2013-2014 Harris Faculty Fellowship. (email: jeff@math.grinnell.edu)

JT and KW are with the Mathematics Institute, University of Oxford, Oxford, UK. JT is supported by an NVIDIA Professor Partnership. KW is supported by the China Scholarship Council. (email: {tanner,wei}@maths.ox.ac.uk)

Manuscript submitted May 2014.

This paper has supplementary downloadable material available at http://eprints.maths.ox.ac.uk/1806/, provided by the authors.

Question (1) is the special case of (2) with $r = 1$.

Most algorithms designed for the solution of (1)-(2) have recovery guarantees (including stability for the model $y = Ax+e$ with $\|x\|_0 \leq k$) based on the now ubiquitous *restricted isometry constants* (RICs) [3]. The lower and upper RICs of a matrix $A \in \mathbb{R}^{m\times n}$ are defined as the smallest values which satisfy

$$(1 - L_{ck})\|z\|_2^2 \leq \|Az\|_2^2 \leq (1 + U_{ck})\|z\|_2^2 \tag{3}$$

for all ck-sparse vectors $z \in \mathbb{R}^n$. A partial list of leading iterative hard thresholding algorithms which, provided A has sufficiently small RICs, are proven to have uniform recovery guarantees over all k sparse vectors include: Normalised Iterative Hard Thresholding (NIHT) [4], Hard Thresholding Pursuit (HTP) [5], Compressive Sampling Matching Pursuit (CoSaMP) [6], Subspace Pursuit (SP) [7], the ALPS family of algorithms [8], and Conjugate Gradient Iterative Hard Thresholding (CGIHT) [9]. Identical guarantees for the row-sparse approximation problem are presented in [9]–[11]. The authors introducing these algorithms, as well as other numerical investigations [11]–[14], have shown that the RIC-based sufficient conditions establishing *uniform* recovery over all x are dramatically more pessimistic than the observed recovery ability of the algorithms when x is drawn *independently* from A. In this manuscript we perform extensive testing of these algorithms for the model of an exactly (row)-sparse matrix with relatively moderate additive noise, and observe that in this setting CGIHT is typically able to recover an approximate solution to (1)-(2) in less time than the aforementioned algorithms.

II. CGIHT for compressed sensing

The relative empirical performance of NIHT, HTP, and an algorithm denoted CSMPSP (a variant of CoSaMP and SP) were extensively tested in [14] using the software *GAGA: GPU Accelerated Greedy Algorithms for Compressed Sensing* [15], [16]. In particular, [14] presents *algorithm selection maps*, throughout the $(\delta, \rho) = (m/n, k/m)$ phase space, to indicate which algorithm is able to solve (1) to a specified tolerance in the least time. For values of (δ, ρ) when multiple algorithms are typically able to solve (1), it is often the least complex algorithm NIHT which can recover the solution to moderate accuracy in the least time.

The efficiency of NIHT is a result of its lower per iteration complexity coupled with its flexible option to change the subspace in which it is searching. However, the projections inherent in the iterations of HTP and CSMPSP provide a

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)
(set orthogonalization weight)
2: if $l = 1$,
$\beta_{l-1} = 0$
else
$\beta_{l-1} = -\frac{\left\langle A\text{Proj}_{T_{l-1}}(r_{l-1}), A\text{Proj}_{T_{l-1}}(p_{l-2})\right\rangle}{\left\langle A\text{Proj}_{T_{l-1}}(p_{l-2}), A\text{Proj}_{T_{l-1}}(p_{l-2})\right\rangle}$
3: $p_{l-1} = r_{l-1} + \beta_{l-1}p_{l-2}$ (define the search direction)
4: $\alpha_{l-1} = \frac{\left\langle \text{Proj}_{T_{l-1}}(r_{l-1}), \text{Proj}_{T_{l-1}}(p_{l-1})\right\rangle}{\left\langle A\text{Proj}_{T_{l-1}}(p_{l-1}), A\text{Proj}_{T_{l-1}}(p_{l-1})\right\rangle}$ (optimal stepsize if $T_{l-1} = T_*$)
5: $w_{l-1} = x_{l-1} + \alpha_{l-1}p_{l-1}$ (update along search direction)
(support set of k largest entries)
6: $T_l = \texttt{PrincipalSupport}_k(|w_{l-1}|)$ entries)
7: $x_l = \text{Proj}_{T_l}(w_{l-1})$ (restriction to support set T_l)

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)
(set orthogonalization weight)
2: if $T_{l-1} \neq T_{l-2}$
$\beta_{l-1} = 0$
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)
(support set of k largest entries)
6: $T_l = \texttt{PrincipalSupport}_k(|w_{l-1}|)$ entries)
7: $x_l = \text{Proj}_{T_l}(w_{l-1})$ (restriction to support set T_l)

substantially improved convergence rate[1] once the supporting subspace (or support set) is identified. In [9], we present *Conjugate Gradient Iterative Hard Thresholding* (CGIHT, Alg. 1) which combines the two advantages into a single algorithm[2]. By allowing the support set to change after each low complexity iteration, CGIHT is able to rapidly identify the correct support while exploiting the computational advantages of a subspace restricted conjugate gradient method when the current estimate for the support set remains unchanged. This enables CGIHT to quickly update the current estimate of the support set while efficiently capturing the correct values on the intersection of the current support estimate and the true support set. Once the true support is found, CGIHT shares the same convergence rate as HTP and CSMPSP.

In iteration l, NIHT computes a step size that is the optimal steepest descent step if the true support T coincides with the current estimate for the support T_{l-1}. Similarly, CGIHT computes an orthogonalization weight, search direction, and step size that executes a subspace restricted conjugate gradient (CG) method when the true support is contained in the current support estimate and the current support estimate remains unchanged. Since orthogonality is only preserved if the initial step on a given support estimate is the steepest descent step, it is natural to restart the subspace restricted conjugate gradient search when the support changes. Also essential for low per iteration complexity, implementing CGIHT with some form of restarting permits one to employ the lower cost implementation of CG. While the orthogonalization is lost without restarting, the non-restarted version of CGIHT (Alg. 1) frequently identifies the true support set in fewer iterations than the variants of CGIHT which employ restarting. *CGIHT restarted*, Alg. 2, exploits the discrete nature of the support constraint and restarts the conjugate gradient method each time the current support estimate changes. Alternatively, *CGIHT projected*[3] employs a restarting condition based on the fraction of the residual aligned with the search direction relative to the residual projected onto the current support set. In Algs. 1–2, the hard thresholding operation is executed by first determining the support estimate T_l of the k largest magnitudes in the current approximation x_l via the subroutine $\texttt{PrincipalSupport}_k(|w_{l-1}|)$, followed by projecting w_{l-1} onto T_l denoted $\text{Proj}_{T_l}(w_{l-1})$.

The restarting conditions in CGIHT restarted and CGIHT projected ensure conjugate orthogonality of the prior search directions and orthogonality of the residuals on the current estimate of the support. These orthogonality guarantees are amenable to analysis using RICs. Both CGIHT restarted and CGIHT projected are guaranteed to recover an approximate solution of (1) for the model $y = Ax + e$ to within a multiple of $\|e\|_2$, provided the matrix A has sufficiently small RICs[4]; see Thm. 1.

Theorem 1 (CGIHT [9]): Let A be an $m \times n$ matrix with $m < n$, and $y = Ax + e$ for any x with at most k nonzeros. Suppose A has sufficiently small restricted isometry constants as defined in [9]. Then for $alg \in \{$CGIHT restarted, CGIHT projected$\}$, there exists constants $\mu^{alg}, \xi^{alg} > 0$ such that $\mu^{alg} < 1$ and

$$\|x_l - x\|_2 \leq \left(\mu^{alg}\right)^l \|x_0 - x\|_2 + \frac{\xi^{alg}}{1 - \mu^{alg}}\|e\|_2 \qquad (4)$$

where x_l is the l^{th} iterate of alg.

Though CGIHT's sufficient RIC conditions are inferior to known conditions for NIHT and HTP in terms of probability the condition is satisfied by a Gaussian matrix, as stated previously it is well known that such uniform recovery conditions

[1]The projection in HTP and CSMPSP is implemented as a subspace restricted conjugate gradient method.

[2]CGIHT for matrix completion is also presented in [9].

[3]The empirical performance of CGIHT projected is inferior to that of Algs. 1–2 for solving (1). Thus the pseudo code, available in [9], is omitted.

[4]Explicit formulae for bounds on the constants μ^{alg}, ξ^{alg} and proofs of the results are available in [9].

are not indicative of the often more important average case performance of these algorithms when x is independent of A [14], [17].

In this paper, we explore the empirical recovery performance of CGIHT for measurements corrupted by additive noise. Utilizing the GAGA software [16], we identify the empirical recovery phase transitions for the three variants of CGIHT under multiple levels of $\|e\|_2$ (Sec. III-B). Furthermore, we compare the algorithms with NIHT, CSMPSP, HTP, and Fast IHT (FIHT) which is a variant of 1-ALPS(2) [18] adapting the optimal first order gradient method [19] for the solution of (1). In Section III-C we present algorithm selection maps of the phase space which identify the algorithm with the least recovery time. This can be interpreted as an expansion of the work in [14] which includes the three variants of CGIHT and FIHT. Section IV further extends the investigation to the question (2). The three main findings are:

- CGIHT, CGIHT restarted, and CGIHT projected are stable in the presence of additive noise, well beyond the region dictated by Thm. 1;
- typically, CGIHT accurately approximates a sparse vector from noise corrupted measurements in less time than other hard thresholding algorithms;
- for the row-sparse approximation problem (2), the computational performance advantages for the CGIHT and CGIHT restarted are substantially amplified, both in terms of recovery region and relative computational efficiency.

III. Observed Noise Stability for Compressed Sensing

In this section, we infer an observed stability to additive noise for all three variants of CGIHT in terms of problem sizes that are recoverable when solving the compressed sensing problem (1); this stability is on par with the stability of other greedy algorithms such as NIHT, CoSaMP, and HTP. The critical observation is that across three representative matrix ensembles a variant of CGIHT is typically the algorithm which successfully recovers the measured vector in the least computational time. We compare the performance of CGIHT, CGIHT restarted, and CGIHT projected with FIHT, NIHT, HTP, and CSMPSP expanding the results presented in [14] for the last three algorithms. The pseudo code for each of these algorithms is included in [9], [14].

This article focuses on the most demanding sparse vector ensemble for (1), namely vectors with equal magnitude nonzero entries. The algorithms considered here are able to successfully recover vectors with a greater number of nonzero entries from alternate vector ensembles. Moreover, we focus on hard thresholding algorithms and do not explicitly compare soft thresholding algorithms tailored to solve (1). The restricted focus of this study is supported by previous empirical studies [12]–[14] from which indirect comparisons are easily made.

A. Empirical Testing Environment for Compressed Sensing

The testing is conducted with the algorithm implementations and testing suites in the software *GAGA* [16]. 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:

- $\mathcal{N}$: dense Gaussian matrices with entries drawn i.i.d. from $\mathcal{N}(0, m^{-1})$;
- $\mathcal{S}_7$: 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: $\mathcal{N}$ represents dense matrices, $\mathcal{S}_7$ represents sparse matrices, and DCT represents subsampling structured matrices with fast matrix-vector products.

The measured vectors are taken from the random binary vector distribution, B, and are formed by uniformly selecting k locations for nonzero entries with values $\{-1, 1\}$ chosen with equal probability. For noisy measurements, we denote the sparse vector distribution by B_ϵ. In B_ϵ, the vectors are drawn from the sparse vector ensemble B and the measurements are corrupted by a misfit vector e drawn from the sphere of radius $\epsilon\|Ax\|_2$; ϵ is referred to colloquially as the "noise level" for the model $y = Ax + e$. A *problem class* consists of a matrix ensemble and a sparse vector ensemble and is denoted (Mat, vec) with $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$ and $vec \in \{B, B_\epsilon\}$ with $\epsilon \in \{0.1, 0.2\}$. The problem classes B_ϵ with $\epsilon = 0$ and B are equivalent and are used interchangeably.

For each problem class (Mat, B_ϵ), problem instances of size (k, m, n) are generated by selecting $n = 2^{13}$ for $Mat = \mathcal{N}$ and $n = 2^{17}$ for $Mat \in \{\mathcal{S}_7, DCT\}$, defining $m = \lceil \delta \cdot n \rceil$ and $k = \lceil \rho \cdot m \rceil$, and using sampling ratios (δ, ρ) chosen as follows. For testing the compressed sensing problem (1), the parameter $\delta \in (0, 1)$ takes on thirty values

$$\begin{aligned}\delta \in \{&0.001, 0.002, 0.004, 0.006, 0.008, 0.01,\\ &\ldots, 0.02, 0.04, 0.06, 0.08, 0.1, \ldots, 0.99\}\end{aligned} \tag{5}$$

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 standard theoretical stability result for an algorithm approximately solving (1) with the model $y = Ax + e$ where $\|x\|_0 \leq k$ has the form (4) for some constants μ^{alg} and ξ^{alg} depending on the algorithm and RIC constants of A. For the model $y = Ax + e$ considered here, $\|e\|_2 = \epsilon\|Ax\|_2 \leq \epsilon\|A_T\|_2\|x\|_2$ where A_T is the submatrix formed by the columns of A with indices in $T = \text{supp}(x)$. In this empirical study, we consider an approximate solution to be a successful approximation when the relative approximation error satisfies

$$\frac{\|\hat{x} - x\|_2}{\|x\|_2} < .001 + 2\epsilon. \tag{6}$$

The success condition (6) roughly captures the form of the theoretical recovery bound (4).

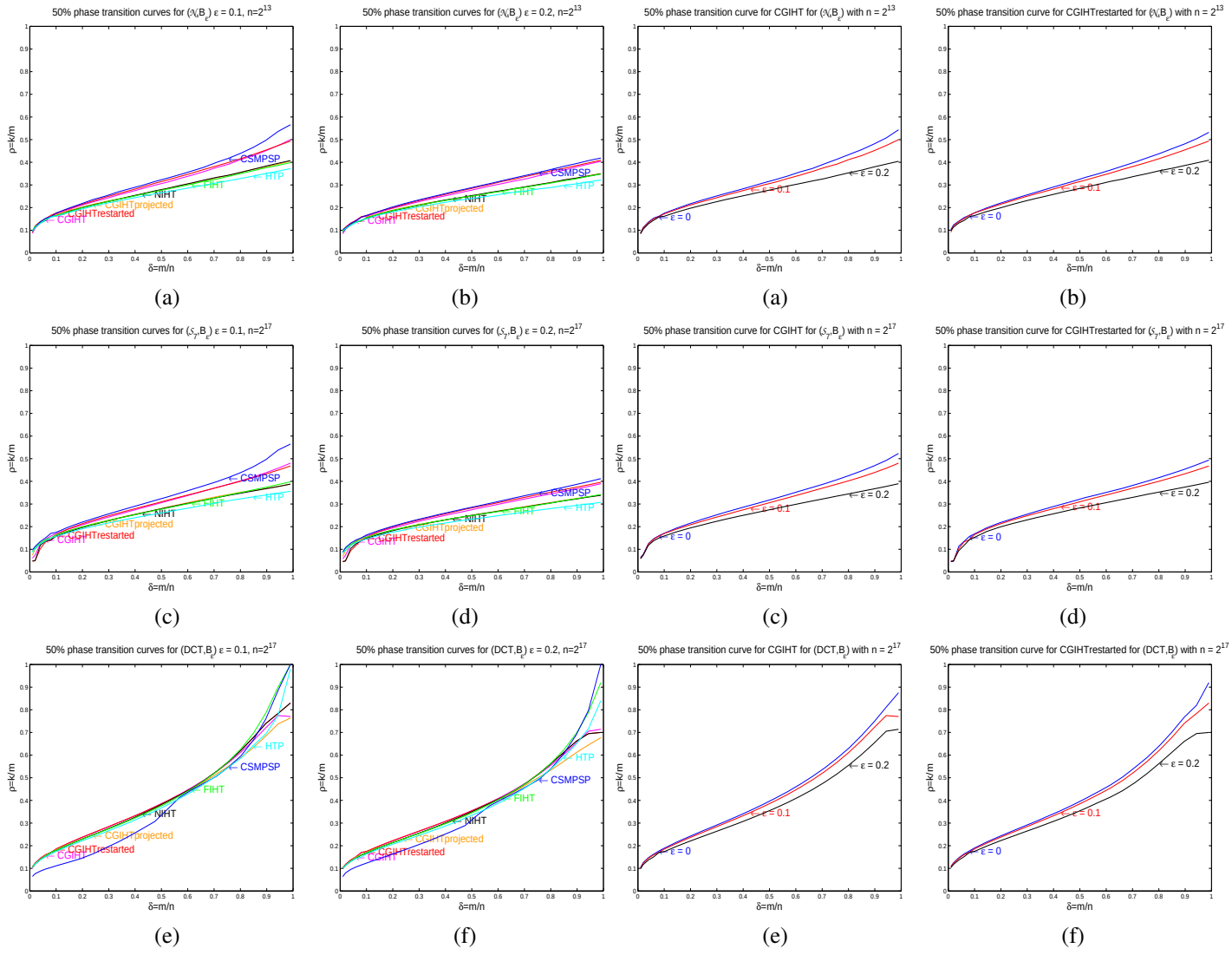

Fig. 1. Compressed sensing (1). 50% recovery probability logistic regression curves for problem classes (Mat, B_ϵ). Left column: $\epsilon = 0.1$. Right column: $\epsilon = 0.2$. (a–b) $Mat = \mathcal{N}$, $n = 2^{13}$; (c–d) $Mat = \mathcal{S}_7$, $n = 2^{17}$; (e–f) $Mat = DCT$, $n = 2^{17}$.

Fig. 2. Compressed sensing (1). 50% recovery probability logistic regression curves for problem classes (Mat, B_ϵ) with $\epsilon = 0, 0.1, 0.2$. Left column: CGIHT. Right column: CGIHT restarted. (a–b) $Mat = \mathcal{N}$, $n = 2^{13}$; (c–d) $Mat = \mathcal{S}_7$, $n = 2^{17}$; (e–f) $Mat = DCT$, $n = 2^{17}$.

As described in [14], the algorithms terminate when any one of several stopping criteria are met: the residual is small, the residual is no longer improving, the algorithm is diverging, or the convergence rate is near 1. Through empirical testing, it was observed that when noise is present in the measurements the iterative hard thresholding algorithms tested frequently cycle between multiple limit points, often with distinct support sets. A simple additional stopping criterion alleviates this problem: if the ℓ_2 norm of the residual is identical (to single precision) to the ℓ_2 norm of one of the previous fifteen residuals, a cycle is declared and the algorithm terminates[5]. For comparison of recovery times, all algorithms utilize identical stopping criteria including the detection for cycling between multiple limit points.

[5]While this will fail to find a cycle of length greater than fifteen, and could theoretically stop prematurely for two different approximations with identical residual norms, this stopping criterion is empirically robust.

B. Recovery Phase Transitions for Compressed Sensing

For each problem class, the empirical recovery phase transitions are identified by first determining a phase transition region $[k_{min}, k_{max}]$ where an algorithm successfully recovers each of 10 random vectors at k_{min} but fails to recover any of 10 random vectors at k_{max}. The phase transition region is then extensively tested and the *recovery phase transition* is the curve defined by a logistic regression for a 50% success rate. The *recovery region* is the region below the recovery phase transition and identifies the region of the phase space where an algorithm is typically able to recover randomly drawn vectors.

In Fig. 1, we compare the empirical recovery phase transitions of the seven algorithms. In the most interesting region of the phase space for compressed sensing, namely $\delta = m/n < 0.5$, the recovery phase transitions for CGIHT and CGIHT restarted are superior to NIHT, HTP, CGIHT projected and FIHT for all three matrix ensembles. For matrix ensembles $\mathcal{N}$ and $\mathcal{S}_7$, CSMPSP boasts the highest (best) recovery phase

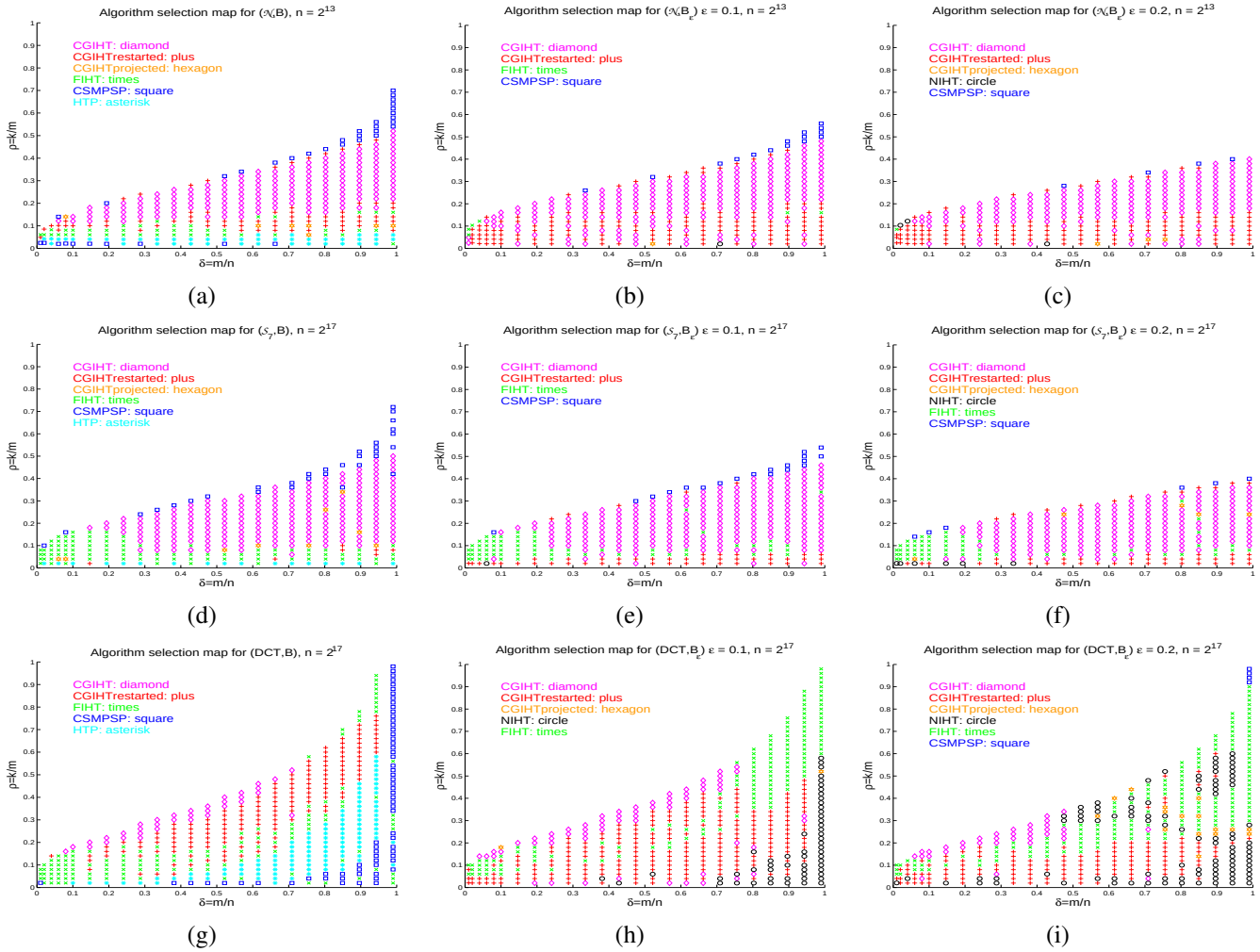

Fig. 3. Compressed sensing (1). Algorithm selection maps for problem classes (Mat, B_ϵ). Left column: $\epsilon = 0$. Center column: $\epsilon = 0.1$. Right column: $\epsilon = 0.2$. (a-c) $Mat = \mathcal{N}$, $n = 2^{13}$; (d-f) $Mat = \mathcal{S}_7$, $n = 2^{17}$; (g-i) $Mat = DCT$, $n = 2^{17}$.

transition for all values of $\delta = m/n$ with the advantage increasing as $\delta \to 1$. As the noise level ϵ increases, the differences between these phase transition curves decrease. In terms of location of the recovery phase transition, we claim the empirical recovery phase transition of CGIHT, CGIHT restarted, and CGIHT projected is competitive with the other algorithms.

In Fig. 2, we observe that the areas below the recovery phase transition curves for CGIHT and CGIHT restarted smoothly decrease as the noise level ϵ increases. Moreover, for the moderate noise level $\epsilon = 0.1$, the recovery phase transition curves for CGIHT closely track the recovery phase transitions for $\epsilon = 0$ with the relative discrepancy decreasing in the most significant region of $\delta \ll 1$. Even for $\epsilon = 0.2$, the recovery region captures a significant portion of the noise free ($\epsilon = 0$) recovery region. Thus, we claim that through the majority of the noise-free recovery region, the relative ℓ_2 norms of the approximations from both the restarted and non-restarted variants of CGIHT scale with the noise level ϵ. This observation holds over three representative matrix ensembles. These empirical observations show that the average case recovery of CGIHT is substantially beyond the region where the uniform recovery condition of Thm. 1 is valid, and show that the recovery regions degrade slowly with additive noise. Similar plots to those in Fig. 2 for the other five algorithms are available in the supplementary material [20], and show a similar smooth decrease in the area of the recovery regions relative to noise level ϵ.

The recovery phase transition curves from Figs. 1 and 2 show the location of the 50% success rate for each of the algorithms. These clearly delineate the recovery region for each algorithm and inform practitioners if an algorithm is likely to succeed in recovering a sparse vector. In the region of the phase space between the algorithm with the highest phase transition and the next highest phase transition curve, one should clearly employ the only successful algorithm. In the region of the phase space below the phase transition curves for multiple algorithms, one must choose which algorithm best serves their application.

C. Algorithm Selection Maps for Compressed Sensing

When selecting from multiple algorithms, a natural choice is to select the algorithm able to reliably solve the question in the least time. In [14], the first two authors introduced

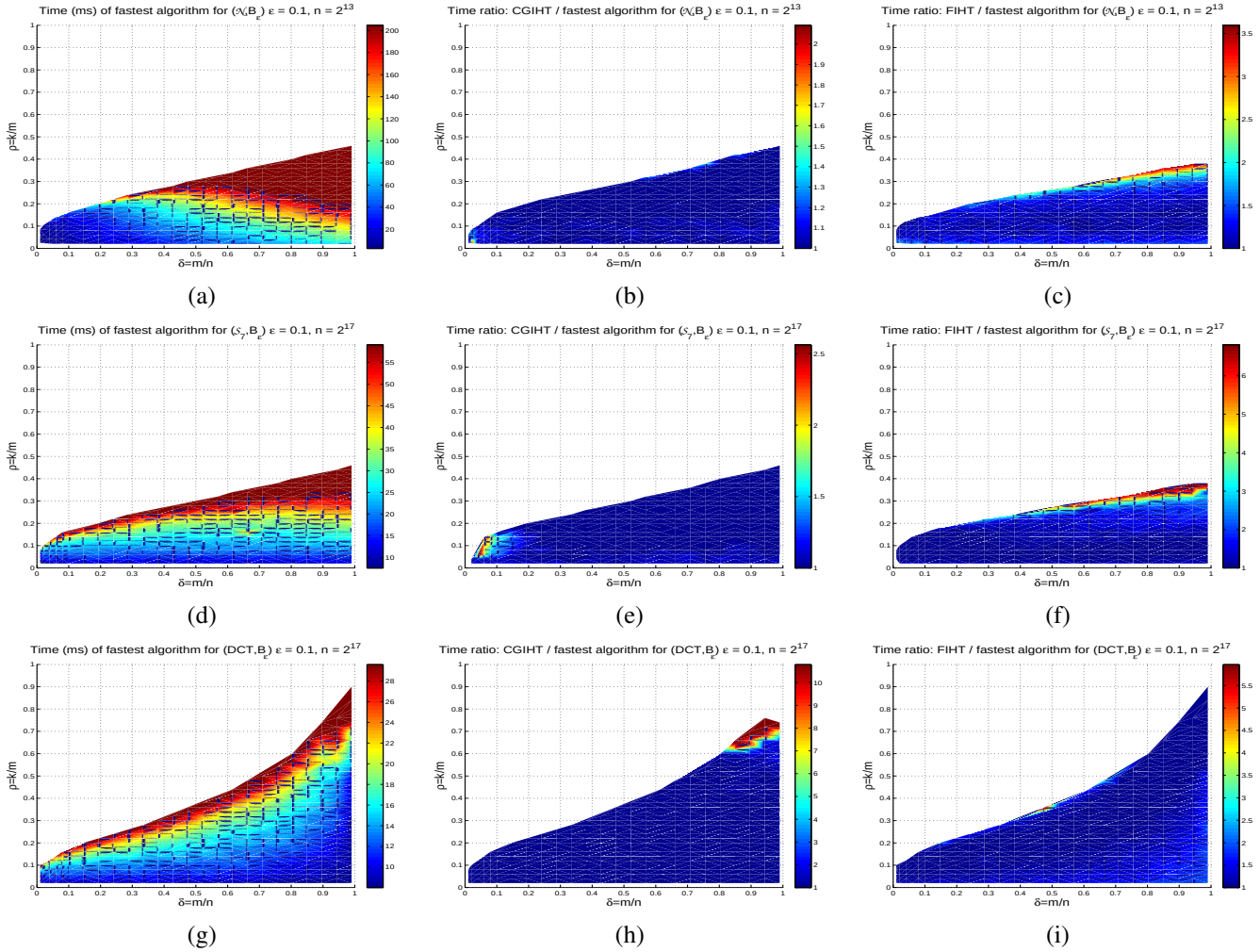

Fig. 4. Compressed sensing (1). Minimum average recovery time and ratios of average recovery time to the minimum for problem classes (Mat, B_ϵ), $\epsilon = 0.1$: (a-c) $Mat = \mathcal{N}$, $n = 2^{13}$; (d-f) $Mat = \mathcal{S}_7$, $n = 2^{17}$; (g-i) $Mat = DCT$, $n = 2^{17}$. Left column: minimum average recovery time over all algorithms. Ratios of average recovery time over minimum average recovery time for CGIHT (Center Column) and FIHT (Right Column).

algorithm selection maps which identify the algorithm with the least computational recovery time. In that empirical study, the dominant algorithms under consideration were NIHT, HTP, CSMPSP. While the noise-free algorithm selection maps included a mix of these algorithms, NIHT was the overwhelming selection in the presence of additive noise with $\epsilon = 0.1$. In this section, we extend the algorithm selection maps to consider the three variants of CGIHT and FIHT. To generate data on a mesh for direct comparison, the parameter $\rho \in (0, 1)$ takes the values

$$\rho \in \{0.02j : j = 1, \ldots, 50\}. \tag{7}$$

For each algorithm and each value of δ from (5), ten problem instances of size (k, m, n) are tested for each value of ρ until the algorithm fails to successfully recover the measured vector for all ten tests.

In Fig. 3, we present algorithm selection maps for all three matrix ensembles and noise levels $\epsilon \in \{0, 0.1, 0.2\}$. In the noise-free setting (Fig. 3(a),(d),(g)) FIHT often recovers the measured vector in the least time when $\rho = k/m \lesssim 0.15$ or $\delta = m/n \lesssim 0.1$. In the region where CSMPSP is the only successful algorithm, it is marked as the recommended algorithm, though in all other regions it typically requires more time to converge. For all three matrix ensembles, a variant of CGIHT is often the algorithm requiring the least time for problem instances (k, m, n) with $(\delta, \rho) = (k/m, m/n)$ just below the phase transition. This is a similar finding to the recovery behavior of NIHT for noiseless measurements presented in [14]. Essentially, for noiseless signals, CGIHT or FIHT replaces NIHT in the algorithm selection maps of all three matrix ensembles. When the measurements are corrupted with the additive misfit vector e, CGIHT and CGIHT restarted mark the overwhelming majority of the phase space for matrix ensembles $\mathcal{N}$ and $\mathcal{S}_7$. For matrix ensemble DCT and $\delta \lesssim 0.5$, CGIHT and CGIHT restarted continue to dominate the selection maps.

The trends established by the algorithm selection maps delineate regions of the phase space where a particular algorithm has the least average time for recovery. When selecting an algorithm, the additional information provided in Fig. 4 is equally important. The left column of Fig. 4 shows the minimum average recovery time from all tested algorithms. Note that the GPU accelerated implementations from *GAGA*

[16] find the solution to (1) for the problem class (Mat, B_ϵ) with $\epsilon = 0.1$ in a small fraction of a second: less than 200 milliseconds (ms) for $Mat = \mathcal{N}$ with $n = 2^{13}$, 60 ms for $Mat = \mathcal{S}_7$ with $n = 2^{17}$, and 30 ms for $Mat = DCT$ with $n = 2^{17}$. Similar timings are observed for $\epsilon \in \{0, 0.2\}$ with plots available in the supplementary material [20].

It is also important to understand the relative performance of each algorithm compared to the best time among all algorithms. The center and right columns of Fig. 4 show the ratio of the average recovery time for CGIHT (center) and FIHT (right) compared to the best time. The two algorithms were selected for Fig. 4 as the variant of CGIHT and the non-CGIHT algorithm with the smallest variation from the least time. Throughout the preponderance of the phase space, CGIHT is within a tiny fraction of the best time, even when not marked as the fastest algorithm on the selection map. Similarly, the time required for FIHT is usually within a small multiple of the minimal time. While CIGHT restarted, CGIHT projected, and NIHT are competitive, HTP and CSMPSP often require more than ten times the minimal computation time. Ratio plots for all algorithms and for $\epsilon \in \{0, 0.2\}$ are available in the supplementary material [20].

In algorithm development for compressed sensing (1), one seeks an algorithm which can rapidly recover a problem of size (k, m, n) with $m \ll n$ and k as close to m as possible; this translates to the phase space as (δ, ρ) pairs with $\delta \ll 1$ and ρ as large as possible. From this point of view, the recovery region near the phase transition is the critical region of the (δ, ρ) phase space. From the data presented in Figs. 1–4 and [20], we observe that for any reasonable undersampling ratio $\delta \lesssim 0.5$, CGIHT is the recommended algorithm for matrix ensembles $\mathcal{N}$ and DCT especially in the presence of noise. For the matrix ensemble $\mathcal{S}_7$ with $0.1 \lesssim \delta \lesssim 1$, CGIHT is also the dominant algorithm. However, for $\delta \lesssim 0.1$ FIHT is the dominant algorithm for the matrix ensemble $\mathcal{S}_7$. While other algorithms are competitive in various regions of the phase space for solving (1), CGIHT demonstrates a consistent computational performance advantage throughout the phase space.

IV. Observed noise stability for row-sparse approximation

In this section we observe an even more substantial computational advantage for CGIHT and CGIHT restarted when solving the row-sparse approximation problem (2). The results presented as Figs. 5–7 are generated in a similar fashion as those reported in Sec. III with natural extensions to row-sparse approximation. The recovery phase transitions are again computed as a logistic regression of the 50% success rate, the algorithm selection maps identify the algorithm with the least computational recovery time, and the timings and ratios are computed in the same fashion as in Fig. 4.

A. Empirical Testing Environment for Row-Sparse Approximation

Currently, there is no available GPU software for testing the row-sparse approximation problem. In this section, the algorithms were implemented by extending the Matlab version of GAGA [15], [16] to the row-sparse approximation problem. The testing was conducted on a Linux machine with two Intel Xeon E5620 CPUs using Matlab R2013a with inherent multi-threading.

The problem classes (Mat, B_ϵ) are also extended to row-sparse approximation. The three matrix enxembles, $\{\mathcal{N}, \mathcal{S}_7, DCT\}$, are identical to those defined in Sec. III-A. The row-sparse matrix ensemble B_ϵ is a natural extension of the sparse vector ensembles defined in Sec. III-A. A matrix $X \in \mathbb{R}^{n\times r}$ from the row-sparse ensemble B has its row support set chosen uniformly from $\{1, \dots, n\}$ with all nonzero values in these rows drawn from $\{-1, 1\}$ with equal probability. The misfit model $E \in \mathbb{R}^{m\times r}$ is formed by selecting each column E_i from the sphere of radius $\epsilon\|\tilde{Y}_i\|_2$ where $\{\tilde{Y}_i : i = 1, \dots, r\}$ are the columns of $\tilde{Y} = AX$. When testing the problem class (Mat, B_ϵ) the algorithms are given a matrix A drawn from the measurement matrix ensemble Mat and the measurements $Y = AX + E$.

For testing the row-sparse approximation problem (2), the parameter $\delta \in (0, 1)$ takes on fifteen values

$$\delta \in \{0.01, 0.02, 0.04, 0.06, 0.08, 0.1, \dots, 0.96\} \tag{8}$$

with 13 additional uniformly spaced values of δ between 0.1 and 0.96. Again, 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. Throughout this section we fix $n = 2^{10}$ and define $m = \lceil \delta \cdot n \rceil$, $k = \lceil \rho \cdot m \rceil$. All results are presented for row-sparse matrices with $r = 10$ columns.

The definition of success is the natural extension to the relative Frobenius norm. An algorithm has successfully recovered the solution to (2) when the relative approximation error satisfies

$$\frac{\|\hat{X} - X\|_F}{\|X\|_F} < .001 + 2\epsilon. \tag{9}$$

The stopping criteria described in Sec. III-A utilize the Frobenius norm where appropriate.

B. Phase Transitions for Row-Sparse Approximation

The data for the phase transitions is generated in the same manner as detailed in Sec. III-B where the phase transition region is identified by determining the interval $[k_{min}, k_{max}]$ and then extensively testing the phase transition region. The increased phase transitions for the row-sparse approximation problem with $r = 10$ columns compared to the phase transitions for $r = 1$ in Sec. III-B is consistent with other other empirical investigations of (2). In particular, the phase transition curves reported in [11] demonstrated that CSMPSP has a considerably larger recovery region than NIHT and HTP for the solving (2) with $r > 1$. In Fig. 5 it is clear that CGIHT and CGIHT restarted have substantially larger recovery regions than CSMPSP, especially for the most interesting region of the phase space with $\delta < 0.5$. At the same time, Fig. 5 shows that the recovery regions for CGIHT projected and FIHT are

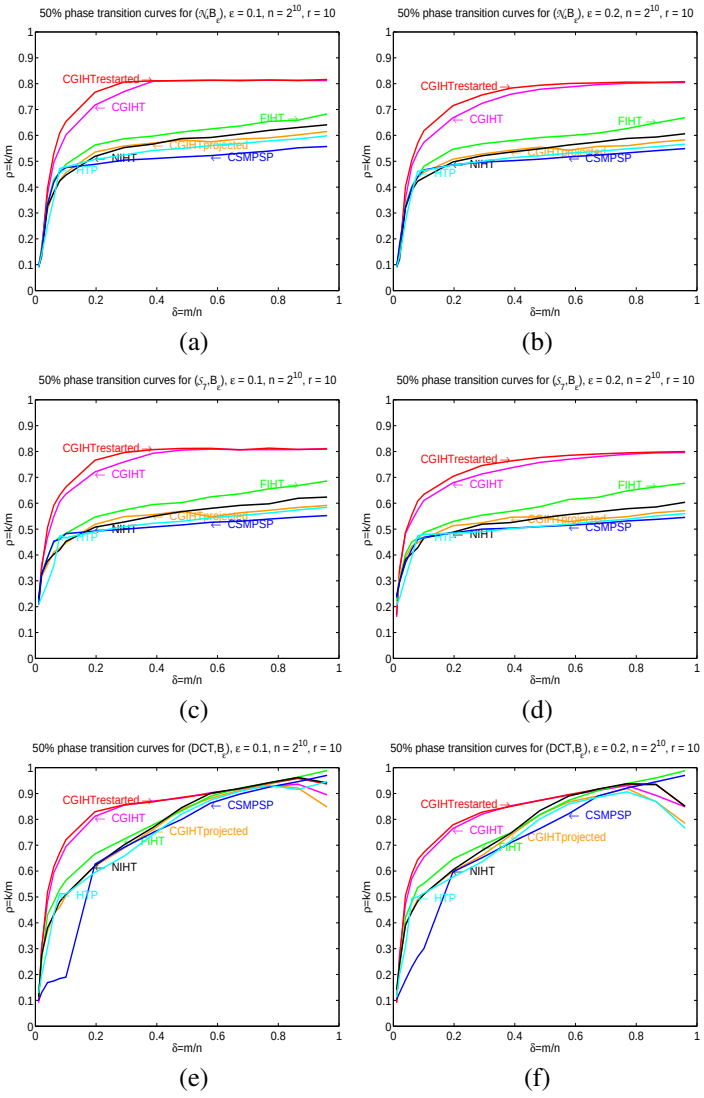

Fig. 5. Row-sparse approximation (2). 50% recovery probability logistic regression curves for problem classes (Mat, B_ϵ) with $n = 2^{10}$ and $r = 10$. Left column: $\epsilon = 0.1$. Right column: $\epsilon = 0.2$. (a–b) $Mat = \mathcal{N}$; (c–d) $Mat = \mathcal{S}_7$; (e–f) $Mat = DCT$.

roughly equivalent to that of NIHT. The significantly improved phase transition for CGIHT and CGIHT restarted suggest they are the preferred iterative hard thresholding algorithms for solving the row-sparse aprroximation problem (2).

C. Algorithm Selection Maps for Row-Sparse Approximation

Similar to Sec. III-C, the problem instances of size (k, m, n) are drawn from the problem class (Mat, B_ϵ) with ρ chosen from a grid. In this section, ρ takes the values

$$\rho \in \{0.04j : j = 1 : 25\} \tag{10}$$

with ten tests performed for each algorithm at each (δ, ρ) until the algorithm fails to recover a single row-sparse matrix in any of the ten tests.

In Fig. 6, CGIHT and CGIHT restarted cover essentially all of the algorithm selection maps for matrix enxembles $\mathcal{N}$ and $\mathcal{S}_7$, particularly near the phase transition. For problem classes (Mat, B_ϵ) with $\epsilon \in \{0.1, 0.2\}$ and $Mat \in \{\mathcal{N}, \mathcal{S}_7\}$, CGIHT restarted is the overwhelming choice for solving (2). However, for the problem classes (DCT, B_ϵ) with $\epsilon \in \{0.1, 0.2\}$, CGIHT has the least computation time for successfully approximating the solution to (2). In Fig. 7, the minimal time for recovery is presented in the left column where we observe that the fastest algorithm solves the problem in less than 170 ms for $Mat = \mathcal{N}$, 100 ms for $Mat = \mathcal{S}_7$, and 260 ms for $Mat = DCT$. The ratio of average recovery time to the minimum average recovery time is again given for the variant of CGIHT and the non-CGIHT algorithm with smallest variation from the minimal time. CGIHT restarted is the fastest of the CGIHT family of algorithms for solving (2) with problem classes (Mat, B_ϵ) for $Mat \in \{\mathcal{N}, \mathcal{S}_7\}$. For $Mat = DCT$, CGIHT is presented in Fig. 7(h). For all three matrix ensembles the non-CGIHT algorithm presented is FIHT whose average recovery time is often twice that of the CGIHT algorithm, especially near the phase transition.

Based on the data presented in Figs. 5–7 and [20], CGIHT restarted is the recommended hard thresholding algorithm for solving the row-sparse approximation problem (2) for problem classes (Mat, B_ϵ) with $Mat \in \{\mathcal{N}, \mathcal{S}_7\}$, $\epsilon \in \{0, 0.1, 0.2\}$, and $r = 10$. CGIHT is the recommended algorithm for the problem class (DCT, B_ϵ) with $\epsilon \in \{0, 0.1, 0.2\}$ and $r = 10$. Similar performance results extend to other values of $r > 1$. The supplementary material [20] contains plots for all algorithms, all problem classes, and all noise levels discussed in this section.

REFERENCES

[1] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inform. Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[2] E. J. Candès, "Compressive sampling," in *Inter. Congress of Mathematicians. Vol. III*. Eur. Math. Soc., Zürich, 2006, pp. 1433–1452.

[3] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Trans. Inform. Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[4] T. Blumensath and M. E. Davies, "Normalised iterative hard thresholding; guaranteed stability and performance," *IEEE J. Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 298–309, 2010.

[5] S. Foucart, "Hard thresholding pursuit: an algorithm for compressive sensing," *SIAM J. Num. Anal.*, vol. 49, no. 6, pp. 2543–2563, 2011.

[6] D. Needell and J. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comput. Harmon. Anal.*, vol. 26, no. 3, pp. 301–321, 2009.

[7] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. Inform. Theory*, vol. 55, no. 5, pp. 2230–2249, 2009.

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

[9] J. D. Blanchard, J. Tanner, and K. Wei, "Conjugate gradient iterative hard thresholding for compressed sensing and matrix completion," p. http://eprints.maths.ox.ac.uk/1833/, 2014.

[10] S. Foucart, "Recovering jointly sparse vectors via hard thresholding pursuit," in *Proc. of SAMPTA*, 2011, Online.

[11] J. D. Blanchard, M. Cermak, D. Hanle, and Y. Jing, "Greedy algorithms for joint sparse recovery," *IEEE Trans. Sig. Proc.*, vol. 62, no. 7, pp. 1694–1704, 2014.

[12] A. Maleki and D. Donoho, "Optimally tuned iterative reconstruction algorithms for compressed sensing," *IEEE J. Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 330–341, 2010.

[13] B. Sturm, "Sparse vector distributions and recovery from compressed sensing," 2011, arXiv:1103.6246v2.

[14] J. D. Blanchard and J. Tanner, "Performance comparisons of greedy algorithms for compressed sensing," *Numerical Linear Algebra with Appl.*, 2014, In press.

[15] ——, "GPU accelerated greedy algorithms for compressed sensing," *Math. Prog. Computation*, vol. 5, no. 3, pp. 267–304, 2013.

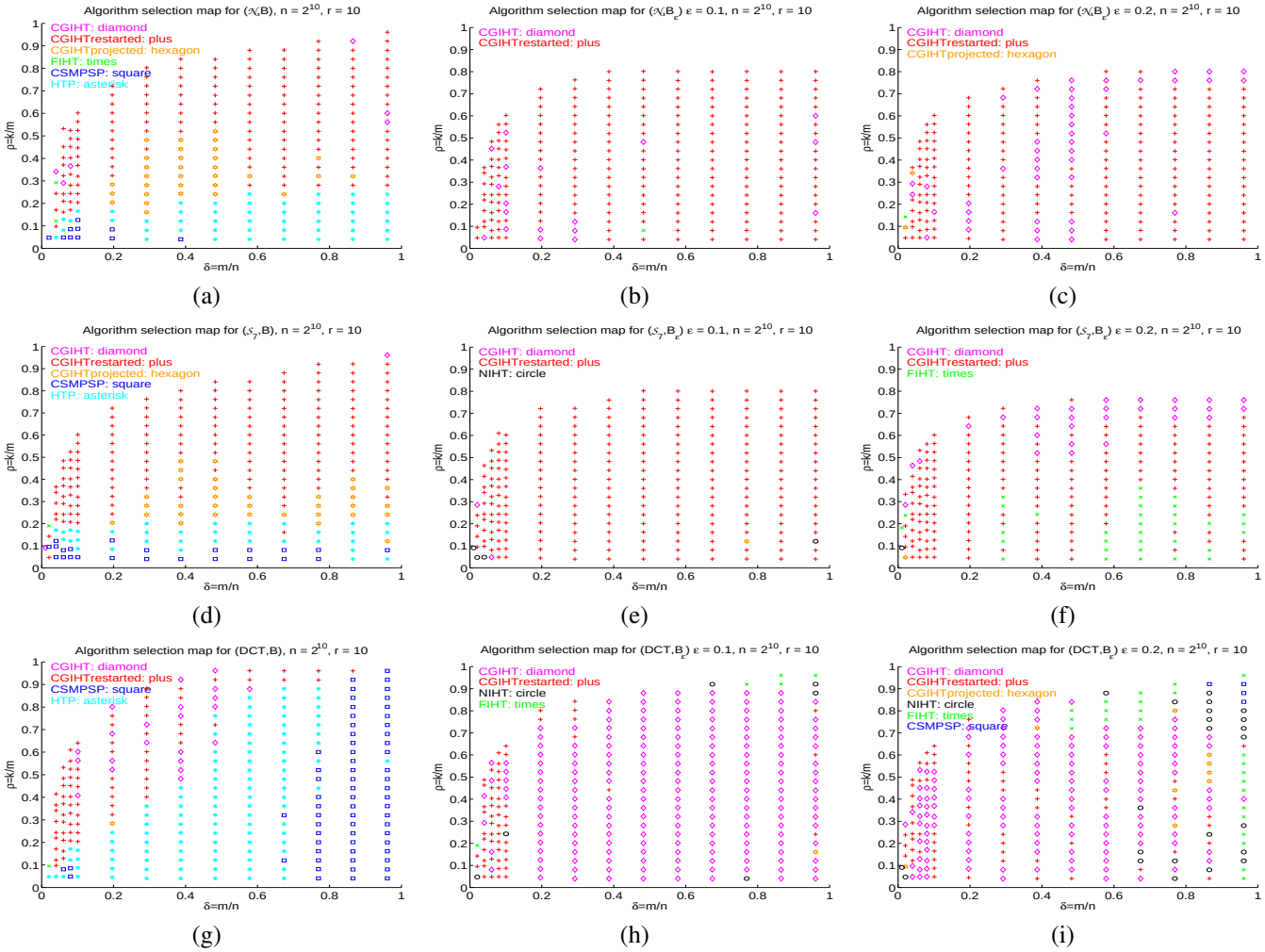

Fig. 6. Row-sparse approximation (2). Algorithm selection maps for problem classes (Mat, B_ϵ) with $n = 2^{10}$ and $r = 10$. Left column: $\epsilon = 0$. Center column: $\epsilon = 0.1$. Right column: $\epsilon = 0.2$. (a-c) $Mat = \mathcal{N}$; (d-f) $Mat = \mathcal{S}_7$; (g-i) $Mat = DCT$.

[16] ——, "GAGA: GPU Accelerated Greedy Algorithms," 2013, version 1.1.0. [Online]. Available: www.gaga4cs.org

[17] J. D. Blanchard, C. Cartis, J. Tanner, and A. Thompson, "Phase transitions for greedy sparse approximation algorithms," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 188–203, 2011.

[18] A. Kyrillidis and V. Cevher, "Recipes on hard thresholding methods," in *4th IEEE Inter. Workshop on CAMSAP*, 2011, pp. 353–356.

[19] Y. Nesterov, "Gradient methods for minimizing composite objective functions," 2007, cORE Discussion Paper 2007/76, Center for Operations Research and Econometrics, Université Catholique de Louvain, Belgium.

[20] J. D. Blanchard, J. Tanner, and K. Wei, "*Supplementary Material for* Conjugate Gradient Iterative Hard Thresholding: Observed noise stability," 2014, [Available] http://eprints.maths.ox.ac.uk/1806/.

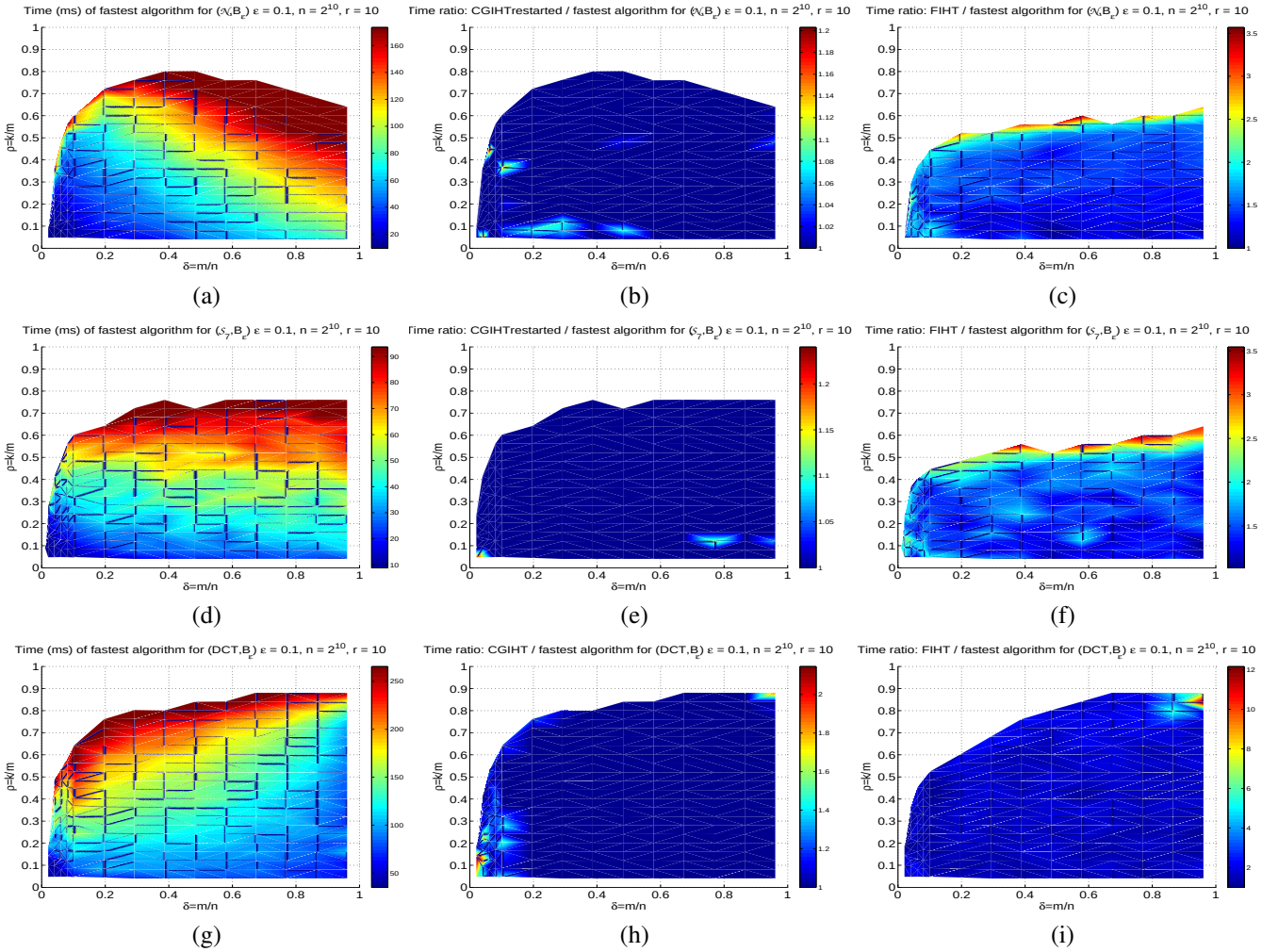

Fig. 7. Row-sparse approximation (2). Minimum average recovery time and ratios of average recovery time to the minimum for problem classes (Mat, B_ϵ), $\epsilon = 0.1$, $n = 2^{10}$, and $r = 10$: (a-c) $Mat = \mathcal{N}$; (d-f) $Mat = \mathcal{S}_7$; (g-i) $Mat = DCT$. Left column: minimum average recovery time over all algorithms. Ratios of average recovery time over minimum average recovery time for CGIHT restarted (b,e), CGIHT (h) and FIHT (c,f,i).