Performance Comparisons of Greedy Algorithms in Compressed Sensing

Jeffrey D. Blanchard[1]*and Jared Tanner[2]

[1] *Department of Mathematics and Statistics, Grinnell College, Grinnell, IA 50112*
[2] *Mathematics Institute, University of Oxford, 24-29 St Giles', Oxford OX1 3LB, UK*

SUMMARY

Compressed sensing has motivated the development of numerous *sparse approximation* algorithms designed to return a solution to an underdetermined system of linear equations where the solution has the fewest number of nonzeros possible, referred to as the sparsest solution. In the compressed sensing setting, *greedy* sparse approximation algorithms have been observed to be both able to recover the sparsest solution for similar problem sizes as other algorithms and to be computationally efficient; however, little theory is known for their average case behavior. We conduct a large scale empirical investigation into the behavior of three of the state of the art greedy algorithms: NIHT, HTP, and CSMPSP. The investigation considers a variety of random classes of linear systems. The regions of the problem size in which each algorithm is able to reliably recover the sparsest solution is accurately determined, and throughout this region additional performance characteristics are presented. Contrasting the recovery regions and average computational time for each algorithm we present *algorithm selection maps* which indicate, for each problem size, which algorithm is able to reliably recover the sparsest vector in the least amount of time. Though no one algorithm is observed to be uniformly superior, NIHT is observed to have an advantageous balance of large recovery region, absolute recovery time, and robustness of these properties to additive noise across a variety of problem classes. A principle difference between NIHT and the more sophisticated HTP and CSMPSP is the balance of asymptotic convergence rate against computational cost prior to potential support set updates. The data suggests NIHT is typically faster than HTP and CSMPSP due to greater flexibility in updating the support which limits unnecessary computation on incorrect support sets. The algorithm selection maps presented here are the first of their kind for compressed sensing. Copyright © J.D. Blanchard and J. Tanner.

Submitted March 2013, Revised January 2014

KEY WORDS: Greedy algorithm, compressed sensing, GPU computing, hard thresholding, sparse approximation

1. INTRODUCTION

Compressed sensing [1, 2] is a technique where the prior knowledge that data is compressible allows for the data to be acquired with fewer measurements than would otherwise be necessary. The resulting reduced number of measurements is proportional to the desired compression rate and can therefore result in dramatic savings in the number of measurements needed. In its simplest form, compressed sensing can be expressed in terms of linear algebra. Let $x \in \mathbb{R}^n$ be a vector with

*Correspondence to: J.D. Blanchard, Department of Mathematics and Statistics, Grinnell College, Grinnell, IA 50112. Email: jeff@math.grinnell.edu

Contract/grant sponsor: Leverhulm Trust and Nvidia Professor Partnership Program [Tanner]

Contract/grant sponsor: National Science Foundation; contract/grant number: DMS 1112612, OISE 0854991 [Blanchard]

k nonzero entries (referred to as k sparse) and let A be an $m \times n$ matrix whose m rows represent inner products used to acquire the measurements $y = Ax \in \mathbb{R}^m$. If one is willing to take a full set of $m = n$ measurements, the vector can be acquired by measuring the entries in x directly with the $n \times n$ identity matrix ($A = I$) so that $y = Ax = Ix = x$. In contrast, if one knew before hand the location of the k nonzero entries one could simply measure those k values similarly. While one often knows *a priori* that a vector has few nonzeros (or few large entries), one rarely knows the location of the nonzeros requiring more than k measurements. Thus, the number of measurements, m, satisfies $k < m < n$.

When the location of the nonzeros is unknown one can formulate the recovery of the sparse vector x from knowledge of A and the measurements $y = Ax$ as the combinatorial optimization problem

$$\min_{z\in\mathbb{R}^n} \|z\|_0 \text{ subject to } y = Az \tag{1}$$

where $\|z\|_0$ denotes the number of nonzeros in the vector z. A naive method for solving (1) is to begin by checking if there is a single column of A that can be used to represent y, failing this to check if there is any combination of two columns of A that can be used to represent y, and increasing the number of columns under consideration until a solution is determined. As A is underdetermined it is always possible to find a solution with m or fewer nonzeros that will satisfy (1). Unfortunately this exhaustive search is computationally implausible for all but the smallest values of k, m, and n.

Much of the development of compressed sensing has focused on the design and analysis of computationally efficient algorithms which can solve (1). An intensively studied technique is to replace (1) by the convex relaxation of the objective

$$\min_{z\in\mathbb{R}^n} \|z\|_1 \text{ subject to } y = Az \tag{2}$$

in which case (2) can be recast as a linear program and solved using well developed, off-the-shelf software [3, 4, 5] or algorithms more recently designed specifically for compressed sensing [6, 7, 8, 9]. The equivalence of when the solution to (2) will be identical to the solution to (1) has been fully characterized by Donoho in [10, 11] with precise sampling theorems which can be stated in terms of an undersampling ratio $\delta = m/n$ and oversampling ratio $\rho = k/m$. In particular for A with entries drawn Gaussian i.i.d., precise values of a function $\rho(\delta = m/n)$ are derived so that for $k < m \cdot \rho(m/n)$ the solution to (2) will coincide with the solution to (1); moreover, when $k > m \cdot \rho(m/n)$ the solution to (2) is typically observed to be different from the solution to (1). The abrupt change in the probability of recovery for an algorithm is referred to as a *phase transition* and the curve $\rho(m/n)$ denoting the transition is referred to as a *phase transition curve*. Other notable examples of precise sampling theorems have been derived for similar formulations such as: robust variants of (1) and (2) by Xu and Hassibi [12], inclusion of non negativity by Donoho and Tanner [13, 14, 15], uniqueness of solutions with bound constraints by Donoho and Tanner [16], recovery of low rank matrices by Recht, Xu and Hassibi [17], block sparse signals by Stojnic [18], and equivalence to the state evolution of message passing algorithms by Donoho, Maleki, and Montanari [19, 20]. Large-scale empirical testing has shown that many of these results hold for many more matrix ensembles than the Gaussian ensemble implied by the theory [21, 22].

A second line of compressed sensing algorithm development has considered algorithms which attempt to directly solve (1) or a noise resistant extension such as

$$\min_{z} \|y - Az\| \text{ subject to } \|z\|_0 \leq k. \tag{3}$$

Examples of such algorithms which have been observed to perform well in testing include Compressive Sampling Matching Pursuit (CoSaMP) [23], Subspace Pursuit (SP) [24], Iterative Hard Thresholding (IHT) [25], Normalized Iterative Hard Thresholding (NIHT) [26], and Hard Thresholding Pursuit (HTP) [27] though this list is far from exhaustive. Each of these algorithms has been analyzed using techniques such as the restricted isometry property [28] and have been proven to recover the solution to (1) for a wide class of sensing matrices A at the optimal order of m proportional to k. However, these sufficient conditions are extremely pessimistic with associated phase transitions well below observed performance [29, 30]. These algorithms employ projections via hard thresholding which sets selected coefficients to zero while leaving the remaining coefficients unchanged. We refer to this class of algorithms broadly as *hard thresholding algorithms.* Unfortunately there are no precise average case characterizations for when (1) is solved by hard thresholding algorithms, with the technical difficulty being that the hard thresholding projections are not Lipschitz continuous [31].

1.1. Relationship to Prior Work

This paper provides an empirical average case analysis of the hard thresholding algorithms via large-scale testing of problems with realistic, application sized problems. The literature already has several examples of empirical testing of algorithms for compressed sensing, notably [32, 33, 22]. In [32] Donoho and Maleki studied optimal tuning parameters for various algorithms, including hard thresholding algorithms. In [33], Sturm studied 15 algorithms with 7 vector distributions focusing entirely on locations of the phase transition for Gaussian matrices. In [22], Monajemi et. al studied the behavior of linear programing (2) recovery probability for numerous deterministic matrices. The information provided in the previous work, such as the tuning parameters for IHT and CSMPSP from [32] have been incorporated into our work. This work is distinct from these predecessors in three main ways.

First, each of those empirical analyses focus on small problem dimensions of $n = 800$, $n = 400$, and $n \approx 1000$, in [32], [33], and [22] respectively. This present empirical analysis is based on problems with ambient dimension frequently set at $n = 2^{18}$ and $n = 2^{20}$. Empirical studies with more realistic, application sized problems offer better insight into the large-scale behavior and should bolster practitioners' confidence in the results. The ability to test large problem sizes is particularly important in evaluating the behavior of algorithms in the extreme undersampling regime of $m \ll n$. When n is small as in previous empirical studies, significant undersampling quickly forces the sparsity level toward zero resulting in poor resolution of k/m. As compressed sensing is primarily concerned with extreme undersampling, this is the most critical region for empirical testing.

Second, none of the previous works consider the very competitive greedy algorithms NIHT or HTP. While [22] is focused exclusively on ℓ_1 minimization, the other two projects examined greedy algorithms, including IHT, CoSaMP, and Subspace Pursuit. Whereas, [32] identified a tuned, fixed step size for IHT, Blumensath and Davies [26] updated IHT to NIHT by incorporating the optimal step size in the current supporting subspace, leading to considerable improvement in both recovery capability and time. Foucart [27] altered IHT adopting the projection ideas from CoSaMP and Subspace Pursuit and combining them with the iterative update step from IHT. These two greedy algorithms are highly competitive with other greedy algorithms and their practical behavior is extensively studied in this work.

Finally, the most significant delineating contribution of this paper is the focus on other behavior characteristics of the algorithms. While [32, 33, 22] are almost entirely focused on the location of the empirical, average case, recovery phase transition, this paper is focused on providing information about how an algorithm performs in relationship to the others, specifically in regions where more than one algorithm is successful. The location of the phase transition is a critical element in this analysis as it identifies the region in which an algorithm reliably recovers the measured vector. However, which algorithm should be selected when more than one algorithm is able to reliably recover the measured vector? This work not only provides empirical phase transitions for large, realistic sized problems, it also provides an algorithm selection map for each problem class, indicating which algorithm is able to reliably recover the measured vector in the least time. Information about the mean and variance for certain performance characteristics is also presented, including time for recovery, ratio of time to fastest algorithm, iterations, convergence rate, and fraction of support set recovered.

The increased scope of testing here is made possible by our development of a software package designed for such large scale, rapid testing using graphics processing units (GPUs). The software, *GPU Accelerated Greedy Algorithms for Compressed Sensing* (GAGA) [34, 35], includes a full testing suite which generates a problem on the GPU, solves the problem with one of five hard thresholding algorithms (also referred to as greedy algorithms), and returns performance characteristics to a text file. The software is also capable of solving a problem directly rather than generating a random problem and is therefore useful for applications; a detailed description of the software including timings and acceleration ratios is presented in [34]. The software is available for download at [35].

1.2. Preliminaries

Our empirical investigation of the hard thresholding algorithms for compressed sensing considers the ability to recover the underlying signal, recovery time, and other performance characteristics of NIHT, HTP, and CSMPSP (a hybrid algorithm akin to Subspace Pursuit and CoSaMP). We denote by $A_S^\dagger$ the pseudo inverse of A_S where A_S is the $m \times |S|$ submatrix consisting of the columns of A indexed by S. In our implementation, the projection $A_S^\dagger y$ is performed via the conjugate gradient method [36] restricted to the subspace defined by S. Thus, if $z = A_S^\dagger y$, then $z = z_S$ is a vector supported on the index set S. The pseudo code presented for the algorithms include the subroutines $\texttt{DetectSupport}(z)$, returning the index set of the k largest magnitude entries in z, and $\texttt{Threshold}(z, S)$, the hard thresholding operator that leaves the values in z_S unchanged and sets all other values of z to zero. Details of the implementations of the algorithms and subroutines in GAGA are described in [34].

This manuscript describes the average case behavior of three hard thresholding algorithms: NIHT, HTP, and CSMPSP whose pseudo code are stated in Alg. 1-3. Each of the three hard thresholding algorithms take as their arguments the measurements, y, the matrix by which the measurements were acquired, A, and the sparsity level k and begins by computing an initial estimate, support set, and residual via Subroutine 1. For conciseness the algorithm pseudo code omits the stopping criteria, which are described in Sec. 2.

```
Subroutine 1 Initialization Procedure
Input: $A$, $y$, $k$
Output: A $k$-sparse approximation $\hat{x}$ of the target signal $x$

Initialization and Initial Support Detection
  $x_0 = A^* y$                                   (initial approximation)
  $T_0 = \texttt{DetectSupport}(x_0)$              (proxy to the support set)
  $x_0 = \texttt{Threshold}(x_0, T_0)$             (restriction to proxy support set $T_0$)
  $r_0 = y - Ax_0$                                 (initialize residual)
```

```
Algorithm 1 NIHT (Normalized Iterative Hard Thresholding [26])
Iteration: During iteration $l$, do
  $\omega_l = \frac{\|(A^* r_{l-1})_{T_{l-1}}\|_2^2}{\|A_{T_{l-1}}(A^* r_{l-1})_{T_{l-1}}\|_2^2}$    (optimal step size in the $k$-subspace $T_{l-1}$)
  $x_l = x_{l-1} + \omega_l A^* r_{l-1}$           (steepest descent step)
  $T_l = \texttt{DetectSupport}(x_l)$              (proxy to the support set)
  $x_l = \texttt{Threshold}(x_l, T_l)$             (restriction to proxy support set $T_l$)
  $r_l = y - Ax_l$                                 (update the residual)
```

1.3. Main Findings

The main findings of our empirical tests are informally summarized in this section, with the remainder of the manuscript presenting a distillation of the data generated. These findings by no means exhaust the information contained in the data, nor do they answer all questions a practitioner might have. More quantitative variants of these findings are presented as formal Claims

Algorithm 2 HTP (Hard Thresholding Pursuit [27])

```
Iteration: During iteration $l$, do
    $\omega_l = \frac{\|(A^* r_{l-1})_{T_{l-1}}\|_2^2}{\|A_{T_{l-1}}(A^* r_{l-1})_{T_{l-1}}\|_2^2}$        (optimal step size in the $k$-subspace $T_{l-1}$)
    $x_l = x_{l-1} + \omega_l A^* r_{l-1}$        (steepest descent step)
    $T_l = \mathtt{DetectSupport}(x_l)$        (proxy to the support set)
    $x_l = A_{T_l}^{\dagger} y$        (projection onto the $k$-subspace $T_l$)
    $r_l = y - Ax_l$        (update the residual)
```

Algorithm 3 CSMPSP (CoSaMP [23], Subspace Pursuit [24])

```
Iteration: During iteration $l$, do
    $S_l = \mathtt{DetectSupport}(A^* r_{l-1})$        ($k$ columns most correlated with residual)
    $\Lambda_l = T_{l-1} \cup S_l$        (form a larger proxy for the support set)
    $x_l = A_{\Lambda_l}^{\dagger} y$        (projection onto the $2k$-subspace $\Lambda_l$)
    $T_l = \mathtt{DetectSupport}(x_l)$        (proxy to the support set)
    $x_l = \mathtt{Threshold}(x_l, T_l)$        (restriction to proxy support set $T_l$)
    $r_l = y - Ax_l$        (update the residual)
```

and Observations in Secs. 3–4. The software used to conduct these tests and process the data are available at [35], written with the aim that interested parties can easily generate different data and easily process it in order to address other questions.

Summary of Main Findings The following seven, informal findings summarize the more detailed Claims and Observations in Sections 3 and 4. While some of the findings extend and reinforce similar statements from previous work, most offer new insight to the behavior of the algorithms across a wide range of problem instances.

- The recovery phase transition curves for NIHT, HTP, and CSMPSP are increasingly similar as $\delta = m/n \to 0$ for A drawn from each of three random matrix ensembles: Gaussian, sparse expanders, and random rows of the discrete cosine transform . As $\delta = m/n \to 1$, CSMPSP typically has the highest recovery phase transition curve. (Sec. 3.1, Claim 1)
- NIHT frequently recovers the measured vector in the least amount of time; though there are regions where CSMPSP is the only successful algorithm. (Sec. 3.2, Claim 2)
- When converging to the measured signal, all three algorithms exhibit a predictable behavior in terms of several performance characteristics such as fraction of the true support set identified, average time for recovery, iterations, and asymptotic convergence rate. (Sec. 3.3, Claim 3)
- All three algorithms are stable to moderate levels of non-adversarial, additive noise; the observed average case performance suggests the relative ℓ_2 error of the recovered vector scales linearly with the noise level. (Sec. 4.1, Claim 4)
- In the presence of moderate levels of noise, NIHT is almost always the fastest algorithm. (Secs. 4.2 and 5, Claims 5 and 8)
- The predictable behavior of the algorithms persists even in the presence of moderate levels of noise. (Sec. 4.3, Claim 6)
- All three algorithms are less successful when attempting to recover vectors with equal magnitude entries. (Sec. 5, Claim 7)

These findings and the more formal Claims 1–8 are informed by the totality of the data presented in Secs. 3–4 and the supplementary material [37].

Discussion of Algorithm Performance A theoretical average case analysis for NIHT, HTP, or CSMPSP is absent from the literature. The following is a plausible discussion of why the algorithms behave in the manner observed in these empirical results. This interpretation is drawn from the authors study, implementation, and extensive testing of the algorithms presented in Secs. 3–4 and the supplementary material [37]. As with any empirical study, the interpretations are unproven and a mathematical anslysis of the observed phenomenon would be beneficial.

The greedy algorithms under consideration attempt to directly reconstruct the measured vector x from the information contained in the measurements y with knowledge of both the measurement process A and the number of nonzeros in x. The algorithms are primarily support set identification algorithms. However, the algorithms also attempt to balance two competing tasks: identification of the support set and the true values of the nonzero entries on the support. The algorithms differ in the degree of confidence given in each iteration to the current estimate of the support set. NIHT gives little confidence to each iteration's estimate of the support set; the algorithm simply takes a single steepest descent step on the current support and then checks if the support set should be updated. HTP and CSMPSP give greater confidence to the current support estimate and theoretically perform a pseudo inverse projection in the associated subspace. A relaxation of this confidence for HTP and CSMPSP can be obtained by implementing fast variants of the algorithms described by the original authors of each algorithm [27, 23, 24]. The implementation studied in this article is somewhere between the theoretical and fast variants of HTP and CSMPSP where the project is replaced by a subspace restricted conjugate gradient search with a maximum of 15 iterations[†]; if the support does not change after 15 iterations this can be viewed as a standard restarting of the conjugate gradient method.

The observed performance of NIHT indicates that its considerable flexibility to change the support set saves computational expenditure while searching for the support. When the correct support is identified, the normalized step size is the optimal steepest descent step on the supporting subspace. This combination of rapid support detection and reasonable convergence rate often makes NIHT the fastest of the algorithms. On the other hand, when the problems permit a rapid identification of the support, the projection based methods, HTP and CSMPSP, are advantageous due to their substantially superior asymptotic convergence rates from the subspace restricted conjugate gradient projection. When noise is present, the support set is more difficult to identify; thus, giving less confidence to the estimated support in each iteration increases the advantage of NIHT over the other algorithms.

Finally, CSMPSP often has the highest phase transition which is likely due to searching for the support set over a larger subspace. While NIHT and HTP always operate on a k-dimensional space, CSMPSP first projects the measurements onto a $2k$-dimensional space and then restricts to a k dimensional space. This projection over a larger space appears to permit the correct identification of the support for larger values of $\rho = k/m$ than the other algorithms.

1.4. Outline

The remainder of the manuscript is organized as follows. Sec. 2 describes the experimental setup including: matrix and sparse vector ensembles tested, stopping criteria, problem sizes and parameters for which tests were conducted, and the computational environments in which the tests were performed. Sec. 3 presents the data for the problem class of sparse vectors with nonzeros of equal magnitude and exact measurements, with recovery phase transitions presented, including: regions of high probability of recovery, average time for recovery, algorithm selection maps, and the variance of these and other performance properties of NIHT, HTP, and CSMPSP. Sec. 4 presents a similar analysis as in Sec. 3, extended to show how the algorithm properties change with the introduction of moderate values of noise. With noise level $\epsilon = 1/10$, Sec. 5 focuses on sparse vectors with nonzeros drawn from ensembles of nonequal magnitude. Sec. 6 outlines future extensions. Only a small portion of the algorithm performance data calculated as described in Sec. 2 is presented in the main body of the manuscript; the remainder of the data is presented in the supplementary document [37] which is outlined in Sec. 2.4.

[†] A maximum of 15 CG iterations per projection was selected to approximately minimize reconstruction time of HTP and CSMPSP while allowing for competitive successful recovery rates.

2. EXPERIMENTAL SETUP

This manuscript describes the average case behavior of three hard thresholding algorithms: NIHT, HTP, and CSMPSP whose pseudo code are stated in Alg. 1-3. This section outlines the tests conducted with the results presented in Sec. 3-4 and the supplementary document [37].

Problem Class For each algorithm we conduct tests for random matrices, drawn from different ensembles, used to measure sparse vectors whose nonzeros are similarly drawn from a variety of distributions. We refer to the combination of a matrix ensemble and sparse vector ensemble as a *problem class* and denote it by (Mat, vec). Each algorithm is evaluated for a specific problem class by repeatedly testing different problem instances at a variety of problem sizes.

Problem Instance Let A and x be drawn from a specific problem class (Mat, vec) at size (k, m, n) where the matrix A is of size $m \times n$ and the vector x has k nonzeros. We refer to a problem instance without noise as recovery of x from the pair (A, y) where $y = Ax$, and a problem instance with noise level ϵ as recovery of a k sparse vector from the pair (A, y) where $y = Ax + e$ with e an additive noise vector drawn uniformly from the sphere of radius $\epsilon\|Ax\|_2$. We denote the problem class with additive noise by (Mat, vec_ϵ). Details of the matrix and sparse vector ensembles investigated in this work conclude this subsection.

Matrix ensemble The measurement matrices are drawn from one of three matrix ensembles denoted by $\mathcal{N}$ (normal/Gaussian), $\mathcal{S}_p$ (sparse), and DCT (discrete cosine transform):

- $\mathcal{N}$: Gaussian matrix normalized to have expected unit Euclidean length columns, i.e. entries drawn i.i.d. from $\mathcal{N}(0, m^{-1})$.
- $\mathcal{S}_p$: Sparse matrix with p nonzeros per column with support set drawn uniformly at random and nonzero values drawn uniformly at random from plus or minus $p^{-1/2}$.
- DCT: Random subsampled rows of the discrete cosine transform matrix.

These three matrix ensembles are selected to represent the cases of fully dense matrices, sparse matrices, and structured matrices with a fast matrix-vector product. In this paper, the results for the sparse matrix ensemble are given for $p = 7$ nonzeros per column. The supplementary document [37, Sec. S6] contains results for other values of p, in particular $p = 4$ and $p = 13$. The results for $\mathcal{S}_4$ illustrate that such sparse matrices have substantially lower recovery phase transitions than $\mathcal{S}_7$ while there is a limited increase in the recovery phase transition curve as p is increased from 7 to 13. The choice of $p = 7$ was made to balance the speed of very sparse matrix-vector multiplications against competitive recovery phase transitions, and was selected from testing with $2 \leq p \leq 16$.

Sparse vector ensembles Each vector has a support set chosen uniformly at random and the nonzero entries drawn i.i.d. from one of the following distributions denoted by B (binary), U (uniform), and N (normal):

- B: Nonzeros drawn from plus and minus one, $\{-1, 1\}$, with equal probability;
- U: Nonzeros drawn uniformly from the unit interval, $\mathcal{U}(0, 1)$;
- N: Nonzeros drawn from the standard normal distribution, $\mathcal{N}(0, 1)$.

The sparse vector ensemble B represents vectors with equal magnitude nonzeros entries as such vectors have been observed in prior studies to be the most challenging for hard thresholding algorithms. The ensemble N represents vectors whose nonzero entries have widely varying magnitudes. Finally, the ensemble U is an intermediate case with positive nonzero entries.

Stopping Criteria Each algorithm is initialized using Subroutine 1 and continues iterating until one of the following stopping criteria are met:

(i) the residual is small: $\|r_l\|_2 < .001 \cdot \frac{m}{n}$;

(ii) the algorithm is diverging: $\|r_l\|_2 > 100 \cdot \|r_0\|_2$.

(iii) the residual has failed to change significantly in 16 iterations:

$$\max_{j=1,\dots,15} \left| \|r_{l-j+1}\|_2 - \|r_{l-j}\|_2 \right| \leq 10^{-6};$$

(iv) the asymptotic convergence rate is close to one:

$$\left(\frac{\|r_{l-15}\|_2^2}{\|r_l\|_2^2} \right)^{\frac{1}{15}} > 0.999;$$

for Alg. 1 with $l > 750$ and for Algs. 2–3 with $l > 125$.

When any one of the stopping criteria *(i)-(iv)* are met at iteration l, the algorithm terminates and returns the k sparse approximation x_l which we denote by $\hat{x}$.

Stopping criterion *(i)* measuring the norm of the residual is dependent on the matrix dimensions in order to account for smaller energy levels in smaller problems. Stopping criterion *(i)* indicates that the algorithm has converged and is typically invoked when an algorithm successfully recovers the measured vector. If the residual norm has grown two orders of magnitude larger than the initial approximation, the algorithm is diverging and criterion *(ii)* terminates the algorithm. Stopping criterion *(ii)* is active in instances where an unusually large step size causes an algorithm to remain on an incorrect support set. Stopping criterion *(iii)* indicates that the algorithm has converged, but to a solution with a large residual. This criterion is effective in halting the algorithms when they have converged to an incorrect solution and when the algorithm has converged to a good approximation of the vector when the measurements are corrupted with noise. The asymptotic convergence rate of the algorithms can be arbitrarily close to 1 for problem instances very near an algorithm's recovery phase transition. Stopping criterion *(iv)* halts an algorithm when the asymptotic convergence rate is near 1 in order to balance overall computation time, especially when an algorithm is failing. Stopping the algorithm based on asymptotic convergence rate only occurs after many iterations[‡] in order to prevent the condition from prematurely stopping an algorithm from a residual increase based on changing the estimate of the support set from one iteration to the next. Stopping criterion *(iv)* essentially serves the role of a maximum number of iterations; it is possible that increasing the number of iterations required to activate criterion *(iv)* could modestly increase the recovery phase transitions. These stopping conditions were determined through extensive testing and provide a clear separation of success and failure for (Mat, B) as described in the supplementary document [37].

There are two distinct analysis frameworks used to interpret the data generated from the empirical testing. In Sec. 3, the analysis focuses on extensive testing of the problem classes (Mat, B) for $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$. As stated in Claim 7 and in previous studies [32, 33], the random vector ensemble $vec = B$ is considered the most challenging in that it has the lowest recovery phase transition for hard thresholding algorithms. In this noise free setting, we perform extensive testing for many values of n in order to provide insight into dependence on the problem size. The results for various values of n are included in the supplementary material [37] while the testing detailed in Sec. 2.1 focuses on the largest dimension tested for each matrix ensemble. Sec. 4 provides an analysis of testing conducted on all problem classes (Mat, vec) and (Mat, vec_ϵ); here a single value of n is tested per problem class and there is a minor relaxation in the definition of successful recovery as described in Sec. 2.2. Details of all tests and the distinctions between the testing conducted in Secs. 3 and 4 are outlined in the following subsections.

[‡] Stopping criterion *(iv)* is only invoked after 750 iterations for Alg. 1 or 125 iterations for Algs. 2–3; since each iteration of Algs. 2–3 can contain up to 15 iterations of a subspace restricted conjugate gradient projection, the choice of 750 and 125 roughly balance total computational cost.

2.1. *Testing and analysis of problem class (Mat, B) without noise*

The data supporting the three main findings for the algorithm behavior on problem class (Mat, B) without noise, Claims 1–3, are generated from problem instances when the problem sizes (k, m, n) tested are selected to best explore each of the properties under consideration. Determining the recovery phase transition behavior underlying Claim 1 is best resolved by focusing the tests near the phase transition. Determining the average time for each algorithm in the region where they are able to recover the measured sparse vector, which informs Claim 2, requires testing throughout their recovery region. The aforementioned tests can provide a wealth of other information about the tested algorithms throughout their recovery regions, such as the mean and standard deviation of: the fraction of true positives of the recovered vector, the number of iterations, and the asymptotic convergence rates. To more concisely convey the data establishing Claim 3 we conduct greater numbers of tests for a few selective values of (k, m, n) in the union of the algorithms' recovery region.

The measured vector x is considered to be *successfully recovered* if the algorithm returns the approximation $\hat{x}$ which differs from x by no more than 10^{-3} in any component:

$$\|x - \hat{x}\|_\infty < 10^{-3}. \tag{4}$$

For the sparse vector ensemble B, (4) guarantees both successful recovery of the support set and causes the algorithm to have sufficiently many iterations to establish the asymptotic convergence rate.

Determination of average case phase transitions for various problem classes The following tests are conducted to identify the average case recovery phase transition for various (Mat, B) problem classes. Results are presented for the largest value of n tested for each matrix ensemble, namely $n = 2^{20}$ for $Mat = DCT$, $n = 2^{18}$ for $Mat = \mathcal{S}_7$, and $n = 2^{14}$ for $Mat = \mathcal{N}$. The problem sizes were selected based on the memory capacity of the GPUs used in the testing. For each value of n, tests were conducted for 30 values of m selected as $m = \lceil \delta \cdot n \rceil$ for each

$$\delta \in \{.001, .002, .004, .006, .008, .01, .02, .04, .06, .08, 0.1, \dots, 0.99\} \tag{5}$$

with 18 additional, linearly spaced values of δ from 0.1 to 0.99. Testing the algorithms at the smallest values of $\delta = m/n$ is unprecedented and only possible here due to n being sufficiently large so that the algorithms successfully recover measured vectors for non-trivial values of $\rho = k/m$ associated with the small values of $\delta = m/n$.

For each m, n pair, the bisection method is used, starting with $k = 1$ and $k = m - 1$, to determine values k_{min} and k_{max} such that the algorithm is observed to have successfully recovered at least nine out of ten problem instances for those values of $k < k_{min}$ and to have successfully recovered at most one out of ten problem instances for those values of $k > k_{max}$. The interval $[k_{min}, k_{max}]$ is an estimate for the recovery phase transition region, which is then sampled by testing ten problem instances for each of 50 independent values of k uniformly spaced in $[k_{min}, k_{max}]$, or every value of k in the interval is tested when $k_{max} - k_{min} < 50$. The experimental setup tests more than 15,000 problem instances for each algorithm, each problem class, and each value of n. Rather than presenting the success probability directly which can have a noisy appearance, the 50% success probability is fit using a logistic regression[§] and presented as a 50% success recovery phase transition curve. Figure 1 is representative of output for determining average case recovery phase transitions.

Determination of algorithm timing and other behavior in recovery region The behavior of the hard thresholding algorithms is explored over the entire (k, m, n) region for the same algorithms, problem classes, and m, n pairs as described for the phase transition evaluation. For each (m, n) pair,

[§] Details of the logistic regression are provided in [37, Sec. S2].

ten problem instances are tested for $k = \lceil jm/50 \rceil$ for $j = 1, 2, 3, \ldots$ continuing until the algorithm fails to successfully recover any of the ten measured k sparse vectors. The data generated includes: relative errors in multiple norms, average time per iteration, average total time, number of iterations, average convergence rate, and true/false support set discovery rates. Throughout the manuscript, the average total time is used to determine an *algorithm selection map* where for each (k, m, n) tested we indicate the algorithm which is able to reliably recover the measured sparse vectors with the least run time. Figure 2 is representative of output for determining algorithm behavior in the recovery region.

Variance of algorithm behavior The data generated from the testing described in the prior paragraph can be used to display numerous properties of an algorithm's behavior. For conciseness we limit our display of data for other performance characteristics to a few representative problem sizes (k, m, n). For each problem class (Mat, B) with $Mat \in (\mathcal{N}, \mathcal{S}_7, DCT)$ we conduct tests for NIHT, HTP, and CSMPSP with n selected as 2^{12} for $Mat = \mathcal{N}$, $n = 2^{16}$ for $Mat = \mathcal{S}_7$, and $n = 2^{18}$ for $Mat = DCT$. For $Mat \in (\mathcal{S}_7, DCT)$ four values of m are selected so that m/n is the value of (5) that is nearest to $\{1/100, 1/20, 1/10, 3/10\}$; for $Mat = \mathcal{N}$, the value of m so that $m/n \approx 1/100$ is omitted. For each m, n pair, one thousand problem instances were tested for each of three values of k selected so that $k = c \cdot \rho^{\star}_{(Mat,B)}(m/n)$ for $c = \{1/2, 3/4, 9/10\}$ where $\rho^{\star}_{(Mat,B)}(m/n)$ is the maximum of the 50% level curves of the logistic regression fit of the recovery probability from the three algorithm's previously generated data. Tables I–III are representative of output for determining the variance of algorithm behavior and include the following data: $\#success$, $\#trials$, and the average and standard deviation for the following performance characteristics: fraction of the true support set contained in the support set of the recovered vector, recovery time, number of iterations, and convergence rate.

2.2. *Testing and analysis of problem classes both with and without noise*

While the discussion in Sec. 2.1 and the data presented in Sec. 3 focus on the binary vector ensemble B at many values of n, the data generated for Secs. 4–5 considers the problem classes (Mat, vec) for $vec \in \{B, U, N, B_\epsilon, U_\epsilon, N_\epsilon\}$ for a single value of n per problem class. For $Mat \in \{S_7, DCT\}$ we fix $n = 2^{17}$ and for $Mat = \mathcal{N}$ we fix $n = 2^{13}$. While the analysis remains the same as described in the Sec. 2.1, the main change is an alternate definition of *successful recovery*. For $vec \in \{U, N\}$, there is a positive probability nonzero entries of the vector x will take values below the success criteria defined by (4). Similarly for problem classes (Mat, vec_ϵ), even when the support set of the vector x is known *a priori* the noise will introduce errors proportional to the noise level ϵ. Therefore, the measured vector x from a randomly drawn problem instance from the problem classes (Mat, vec) with $vec \in \{B, U, N, B_\epsilon, U_\epsilon, N_\epsilon\}$ is considered to be *successfully recovered* if the relative ℓ_2 error is no greater than 10^{-3} plus a twice the noise level ϵ:

$$\frac{\|x - \hat{x}\|_2}{\|x\|_2} < 10^{-3} + 2\epsilon, \tag{6}$$

where 2ϵ was observed to most accurately identify correct support set identification for (Mat, B_ϵ). Notice that this analysis uses the relative ℓ_2 norm even in the noise free case where $\epsilon = 0$. For the problem class (Mat, B), (4) implies (6). When comparing figures from Secs. 3 and 4 it is clear that defining success via (4) or (6) results in very similar phase transition curves $\rho^{alg}_{(Mat,B)}(m/n)$. It is important to note that this difference in the definition of success has no impact on algorithms during the recovery process; determining success is solely an aspect of the analysis and display of the output from the algorithms. Finally, the problem instances (k, m, n) for the tabular data in Sec. 4 were chosen to match the problem instances in the tabular data for the noise-free setting. This permits direct comparison of the tables to infer the impact of noise on the algorithms' performance characteristics. All data displayed in Secs. 4–5 utilize (6).

2.3. *Algorithm Implementation and Computational Environment*

Empirical performance comparisons necessarily depend on both the algorithm implementation and the computational hardware. The algorithm implementations are accelerated through a GPU-parallelization as described in [34] and are available for download at [35]. The major factors in the acceleration are the parallelization of the matrix-vector multiplications and the rapid identification of the current proxy for the support set listed as `DetectSupport` in Algs. 1–3. This GPU implementation also significantly accelerates the generation of the random problem instances. The acceleration available in the software *GAGA* [35] permits large-scale testing at application sized problems.

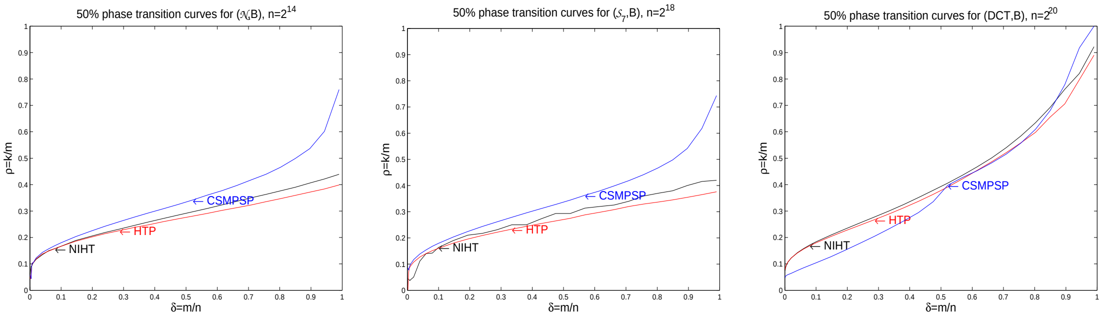
Equally important to empirical performance comparisons is the computational environment. The extensive testing for problem class (Mat, B) in Sec. 3 was conducted on a GPU workstation including dual Intel Xeon X5650 CPUs with 24GB of 1333MHz RAM and a single NVIDIA C2050 GPU. To demonstrate stability across platforms, the testing in Secs. 4–5 for problem classes (Mat, vec) with alternative vector distributions and with additive noise was conducted on multiple GPU workstations equipped with dual Intel Core i5-2500 CPUs with 6GB of 1333MHz RAM and a single NVIDIA GTX480 GPU. Finally, all tabular data in this paper was generated on the most current GPU architecture (Kepler) with a GPU workstation containing dual Intel Xeon E5-2643 CPUs with 64GB of 1600MHz RAM and four NVIDIA K10 GPUs.

2.4. *Supplementary Material*

Only a small portion of the algorithm performance data calculated as described in Sec. 2 is presented in the main body of the manuscript. The supplementary document [37], with section and figure numbers prefixed with the letter S, contains the following additional data. Section S2 presents data showing clear separation between problem instances where the algorithms converge to the correct vector and problem instances where the approximation error is order 1. This information validates the stopping criteria stated in Sec. 2. Section S3.1 presents recovery phase transition curves for (Mat, B) with $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$ at smaller problem sizes than presented in Sec. 3; this data shows the convergence of the recovery phase transition curves as n increases. Section S3.2a includes ratios of algorithm recovery times to the minimum average recovery time that were omitted from Sec. 3.2. Section S3.2b presents algorithm selection maps for (Mat, B) with $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$ for smaller values of n than those presented in Sec. 3.2 depicting consistency of the slection maps for different values of n. Sec. S4.2 presents ratios of the recovery time to the fastest recovery time which are omitted from Sec. 4.2. Section S5 provides additional data for the analysis from Sec. 5. Section S5.1 presents recovery phase transitions with each plot considering a single algorithm and matrix ensemble with $vec \in \{U, N\}$; Sec. S5.2 presents the ratio of an algorithm's average recovery time over the minimum average time for each of the algorithms NIHT, HTP, and CSMPSP for problem classes (Mat, vec) for $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$ and $vec \in \{U_\epsilon, N_\epsilon\}$ for $\epsilon = 1/10$. Section S6 presents recovery phase transitions for $(\mathcal{S}_p, vec)$ for $p = 4$, 7, and 13, for each of $vec \in \{B, U, N, B_\epsilon, U_\epsilon, N_\epsilon\}$ with $\epsilon = 1/10$. These plots show that the gap between the recovery phase transition curves for $p = 13$ and $p = 7$ is much smaller than the gap between $p = 7$ and $p = 4$. Section S6 also provides a full set of empirical results for the matrix ensemble $\mathcal{S}_4$ demonstrating that the algorithm selection maps presented in Sec. 3 are consistent for $(\mathcal{S}_p, B)$ with smaller values of p. These results informed the focus in the main body of the manuscript on $p = 7$ for $Mat = \mathcal{S}_p$.

3. ALGORITHM PERFORMANCE FOR PROBLEM CLASS (Mat, B)

This section presents and analyzes the main features of the data from the testing of problem classes (Mat, B) for $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$ and the hard thresholding algorithms NIHT, HTP, and CSMPSP. Focus on the sparse signed vector ensemble B is due to both it being the vector ensemble with the lowest recovery phase transition, see [32, 33] and Claim 7, and due to the ease by which one can certify identification of the support set via (4). The majority of the results shown are given as figures in order to provide as concise as possible a presentation of the large data set collected.



(a) (b) (c)

Figure 1. 50% recovery probability logistic regression curves for $vec = B$ and algorithms NIHT, HTP, and CSMPSP: (a) $Mat = \mathcal{N}$ with $n = 2^{14}$, (b) $Mat = \mathcal{S}_7$ with $n = 2^{18}$, and (c) $Mat = DCT$ with $n = 2^{20}$.

The plots all have the undersampling ratio $\delta = m/n$ as the horizontal axis and the oversampling ratio $\rho = k/m$ as the vertical axis; this is consistent with earlier presentation of similar compressed sensing studies [38, 29, 21, 32, 33] and with theoretical sampling theorems [29, 39].

3.1. *Regions of high probability of recovery:* (Mat, B)

Claim 1 (Recovery regions: sparse binary vectors without noise)
For problem classes (Mat, B) with $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$ and problem instances drawn without noise, for each matrix ensemble the recovery phase transitions for NIHT, HTP, and CSMPSP increasingly approach one another for $m \ll n$, with the exception of CSMPSP for (DCT, B) and NIHT for $(\mathcal{S}_7, B)$ with $\delta < 1/20$ where for both a substantially lower phase transition is observed. For ratios of m/n approaching one CSMPSP has a phase transition that is substantially higher for problem classes $(\mathcal{N}, B)$ and $(\mathcal{S}_7, B)$ and a modestly higher phase transition for (DCT, B); NIHT and HTP are observed to have recovery regions similar to each other. (See Fig. 1.)

Claim 1 is supported by data represented in Fig. 1 and the more detailed statements in Obs. 3.1.2 and 3.1.1. Fig. 1 presents the 50% recovery level curves for each of the problem classes (Mat, B) at the largest value of n tested; the phase transitions for NIHT, HTP, and CSMPSP concentrate with increasing n to values near those shown in Fig. 1. Additional data supporting Claim 1 is available in [37, Sec. S3.1] including the 50% logistic regression fit for the smaller values of n and the concentration of the phase transition gap between the 10% and 90% logistic regression as n increases.

Observation 3.1.1
For the problem classes $(\mathcal{N}, B)$ and $(\mathcal{S}_7, B)$ with $\delta > 1/20$, CSMPSP has a recovery phase transition that is above NIHT, whose phase transition is above HTP. Within each problem class, as m/n approaches zero the phase transitions of these three algorithms approach one another with the exception of NIHT for $(\mathcal{S}_7, B)$ where the phase transition drops substantially for $\delta < 1/20$. As m/n approaches one the phase transition curves for CSMPSP becomes nearly twice as high as those of NIHT and HTP. (See Fig. 1(a-b).)

Observation 3.1.2
For the problem class (DCT, B) with 2^{20}, NIHT and HTP have recovery phase transitions within 0.01 of each other for $m/n < 1/4$ and approach one another as m/n decreases to zero; whereas CSMPSP is observed to have a substantially lower phase transition as m/n decreases below $1/2$. For $m/n > 1/2$ the phase transitions of NIHT remains slightly above that of HTP, with CSMPSP comparable near $m/n = 1/2$ and increasing to have the highest phase transition as m/n approaches one. (See Fig. 1(c).)

Similar recovery phase transition studies to those shown in Fig. 1 were conducted in [32, 33, 22]. Quantifying and comparing the recovery phase transition curves for algorithms is of primary importance for their application as it indicates the problem sizes (k, m, n) where the algorithm will return the measured vector. Observations 3.1.1 and 3.1.2 highlight instances where the phase transitions are substantially different, indicating problem sizes where only one algorithm is able to reliably recover the measured vector. However, these observations also highlight problem sizes in which multiple algorithms are able to recover the measured sparse vector, in particular, as m/n approaches zero. This is the region of greatest practical interest for compressed sensing as $\delta = m/n \to 0$ corresponds to the most substantial undersampling. In regions where more than one algorithm is able to reliably recover the measured vector, further information is needed in order to determine which algorithm to select. To provide this additional information, this manuscript focuses on the average recovery time of the algorithms.

3.2. Average time for recovery and algorithm selection maps: (Mat, B)

Claim 2 (Algorithm selection: sparse binary vectors without noise)
NIHT is able to recover problems from the class (DCT, B) near the observed recovery phase transition (for $\delta < 9/10$) in less time than can HTP or CSMPSP. NIHT is typically able to recover problems from the class $(\mathcal{N}, B)$ for $m \ll n$ in less time than HTP and CSMPSP except for $\rho < 1/15$ where HTP is the fastest; however, for m/n approaching one CSMPSP is preferable due to a substantially higher phase transition. CSMPSP is able to recover problems from the class $(\mathcal{S}_7, B)$ near the observed recovery transition in less time than can NIHT and HTP, though NIHT requires the least time for much of the recovery region. (See Fig. 2.)

Claim 2 is supported by Observations 3.2.1–3.2.3 which follow from the data displayed in Fig. 2. The principal information in the algorithm selection maps are the trends in the data depicted by regions where one algorithm is consistently selected as the fastest; individual points are not intended to portray a certainty. In fact, large-scale testing is sure to have minor changes from one experiment to the next. Re-running the tests will result in algorithm selection plots with very similar layering of regions selecting a particular algorithm while several individual points may vary. The data suggests that NIHT is typically able to recover a problem instance in less time than can HTP and CSMPSP due to its lesser computational cost per iteration, which allows NIHT to more quickly change support sets, without having performed undue computations on a candidate support set. This is most pronounced in early iterations where there is greater uncertainty in the support set, in which cases HTP and CSMPSP spend considerable computational resources determining the nonzero values on support sets which then change.

Fig. 2 presents timing information for NIHT, HTP, and CSMPSP for problem classes (Mat, B) with $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$. The average recovery time for each algorithm is recorded for each (k, m, n) where at least one algorithm's average error satisfies (4). Algorithm selection maps of the phase space are displayed in Fig. 2(a-c) which indicate the algorithm whose minimum recovery time for the given problem class is shown in the plot at a given (k, m, n); consequently, the algorithm selection map indicates which algorithm is recommended at that particular value of (k, m, n). The minimum average recovery time among all algorithms is displayed in Fig. 2(d-f). In Fig. 2(g-i), the ratio of the average recovery time for NIHT compared to the fastest algorithm is presented throughout the NIHT's recovery region. Similar plots depicting the run time ratios against the fastest algorithm are available for HTP and CSMPSP in the supplementary material [37, Sec. S3.2] along with algorithm selection maps for smaller values of n.

Observation 3.2.1
For the problem class $(\mathcal{N}, B)$, the algorithms show a clear layering in k/m of minimum compute time. CSMPSP has a substantially higher phase transition than the other algorithms, particularly as m/n approaches one, with the algorithm selection map marking CSMPSP in that region. Once entering the region where the other algorithms are also able to reliably recover the measured vector,

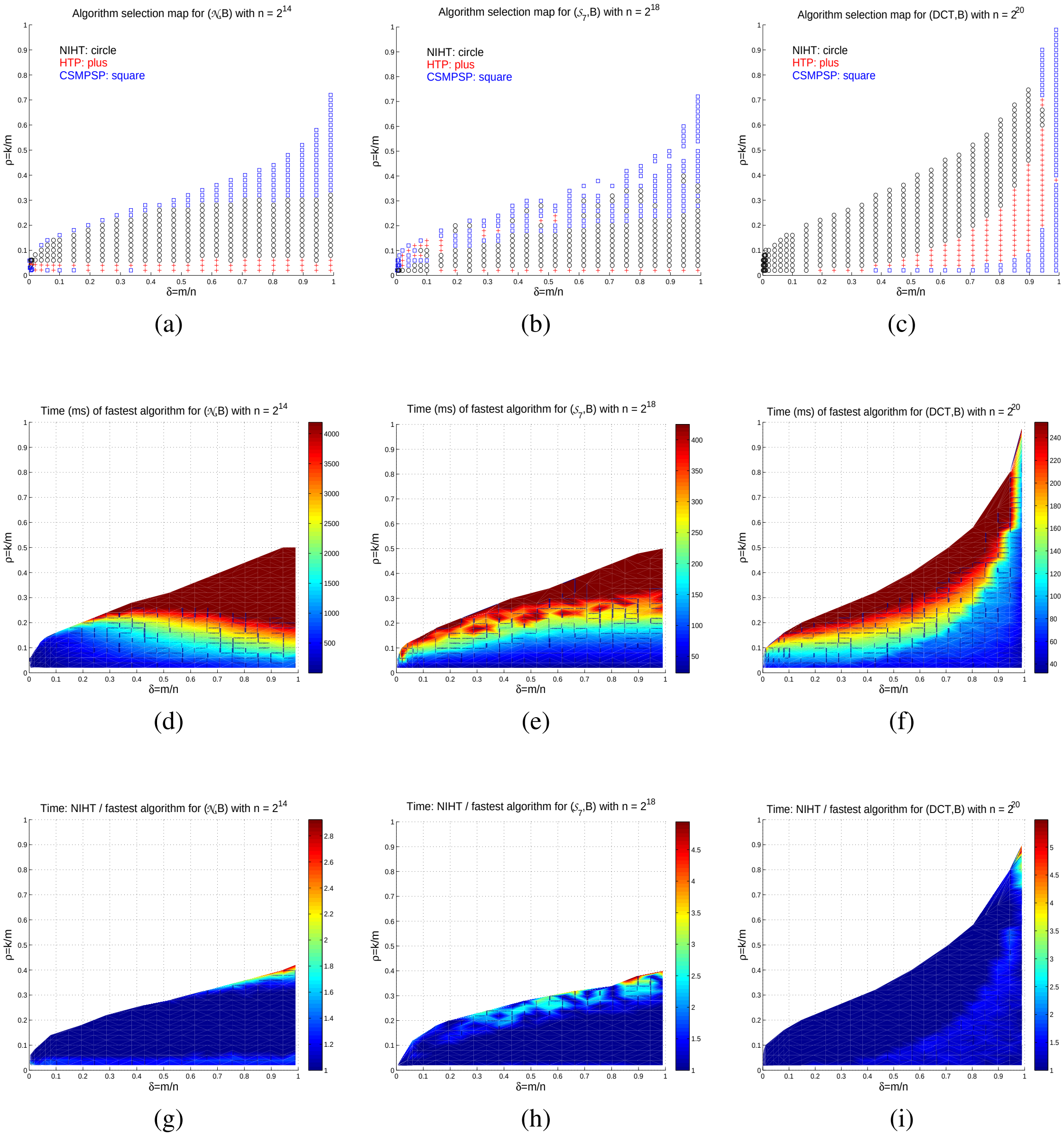


Figure 2. (a-c): Algorithm selection maps. (d-f): Average time for the fastest algorithm. (g-i): Ratio of average time for NIHT over the fastest algorithm. Left panels: $(\mathcal{N}, B)$ with $n = 2^{14}$; Center panels: $(\mathcal{S}_7, B)$ with $n = 2^{18}$; Right panels: (DCT, B) with $n = 2^{20}$.

we observe NIHT requires the least time for moderate values of k/m, and HTP taking the least time as k/m approaches zero. (See Fig. 2(a).) For $m/n < .8$, NIHT's recovery time is always within a multiple of two of the fastest algorithm. (See Fig. 2(g).)

Obs. 3.2.1 encourages the use of CSMPSP for the problem class $(\mathcal{N}, B)$ and the largest values of k/m recoverable, most notably as m/n approaches one. For $m/n \lesssim 1/2$ the recovery time for NIHT is either the least or within about 40% of the fastest; HTP and CSMPSP take modestly longer. Fig. 2(d) shows that for $m/n < 1/10$ the absolute recovery time for problems of size $n = 2^{14}$ in less than 500ms in regions where these algorithms are reliably able to recover the measured vector.

Observation 3.2.2
For the problem class $(\mathcal{S}_7, B)$ the algorithm selection map is divided into two regions. (See

m	k	Algorithm	successes / 1000 tests	fraction of correct support		time (ms)	iterations	convergence rate
				successes	failures	successes	successes	successes
2622	141	NIHT	1000	1(± 0)	–	18.5(± 0.9)	11(± 0.5)	0.391(± 0.011)
		HTP	1000	1(± 0)	–	40.9(± 9.5)	2(± 0.5)	0.007(± 0.011)
		CSMPSP	989	1(± 0)	0.992(± 0.002)	38.2(± 11.9)	2(± 0.5)	0.002(± 0.022)
	211	NIHT	1000	1(± 0)	–	27.0(± 1.2)	17(± 0.7)	0.526(± 0.010)
		HTP	1000	1(± 0)	–	79.5(± 6.1)	4(± 0.3)	0.043(± 0.011)
		CSMPSP	0	–	0.855(± 0.040)	–	–	–
	253	NIHT	1000	1(± 0)	–	38.7(± 4.4)	24(± 2.6)	0.546(± 0.006)
		HTP	1000	1(± 0)	–	126.7(± 21.3)	6(± 0.9)	0.148(± 0.052)
		CSMPSP	0	–	0.631(± 0.053)	–	–	–
15729	1239	NIHT	1000	1(± 0)	–	26.3(± 0.3)	16(± 0.1)	0.459(± 0.004)
		HTP	1000	1(± 0)	–	61.5(± 0.7)	3(± 0.0)	0.006(± 0.002)
		CSMPSP	998	1(± 0)	0.996(± 0.000)	95.0(± 44.6)	4(± 2.0)	0.058(± 0.051)
	1858	NIHT	1000	1(± 0)	–	38.9(± 0.9)	24(± 0.5)	0.562(± 0.005)
		HTP	1000	1(± 0)	–	91.3(± 6.9)	4(± 0.3)	0.028(± 0.013)
		CSMPSP	0	–	0.855(± 0.012)	–	–	–
	2229	NIHT	1000	1(± 0)	–	53.3(± 1.8)	32(± 1.0)	0.615(± 0.004)
		HTP	1000	1(± 0)	–	165.2(± 11.9)	7(± 0.5)	0.166(± 0.026)
		CSMPSP	0	–	0.686(± 0.017)	–	–	–
26215	2401	NIHT	1000	1(± 0)	–	29.8(± 0.7)	18(± 0.4)	0.469(± 0.002)
		HTP	1000	1(± 0)	–	63.9(± 0.9)	3(± 0.0)	0.006(± 0.002)
		CSMPSP	1000	1(± 0)	–	108.5(± 15.5)	5(± 0.7)	0.049(± 0.013)
	3601	NIHT	1000	1(± 0)	–	44.1(± 0.8)	26(± 0.5)	0.597(± 0.003)
		HTP	1000	1(± 0)	–	108.8(± 9.3)	5(± 0.4)	0.051(± 0.018)
		CSMPSP	0	–	0.866(± 0.009)	–	–	–
	4321	NIHT	1000	1(± 0)	–	61.4(± 1.7)	36(± 0.9)	0.649(± 0.002)
		HTP	1000	1(± 0)	–	185.5(± 12.2)	8(± 0.5)	0.179(± 0.026)
		CSMPSP	0	–	0.710(± 0.013)	–	–	–
75332	10451	NIHT	1000	1(± 0)	–	39.1(± 0.7)	21(± 0.3)	0.537(± 0.003)
		HTP	1000	1(± 0)	–	73.9(± 0.5)	3(± 0.0)	0.004(± 0.000)
		CSMPSP	1000	1(± 0)	–	89.9(± 12.5)	4(± 0.5)	0.010(± 0.005)
	15676	NIHT	1000	1(± 0)	–	61.1(± 0.4)	33(± 0.1)	0.664(± 0.001)
		HTP	1000	1(± 0)	–	129.1(± 5.1)	5(± 0.2)	0.046(± 0.007)
		CSMPSP	11	1(± 0)	0.928(± 0.003)	132.3(± 17.4)	5(± 0.6)	0.050(± 0.024)
	18811	NIHT	1000	1(± 0)	–	85.1(± 1.3)	46(± 0.6)	0.717(± 0.001)
		HTP	1000	1(± 0)	–	222.3(± 8.8)	8(± 0.4)	0.173(± 0.017)
		CSMPSP	0	–	0.806(± 0.007)	–	–	–

Table I. Average and standard deviation of performance characteristics for NIHT, HTP, and CSMPSP for (DCT, B) with $n = 2^{18}$

Fig. 2(b).) CSMPSP has a substantially higher phase transition than the other algorithms, particularly as m/n approaches one. CSMPSP has the least recovery time, or within a small multiple, throughout its recovery region. NIHT has a modestly smaller recovery time than CSMPSP in the majority of its recovery region, with HTP having the least recovery time for the smallest value of k/m. For $m/n < 1/2$, NIHT's recovery time is typically within 2.5 times that of the fastest algorithm and never more than 4.5 times the fastest recovery time. (See Fig. 2(h).)

Obs. 3.2.2 encourages the use of CSMPSP for the problem class $(\mathcal{S}_7, B)$; with average recovery times for problems of size $n = 2^{18}$ in under 400ms as depicted in Fig. 2(e). When in the regime well below the phase transition, Obs. 3.2.2 encourages the use of NIHT.

Observation 3.2.3
For the problem class (DCT, B), NIHT is able to reliably successfully recover the measured vector in less time than HTP and CSMPSP for approximately $\frac{1}{2}\rho^{niht}_{(DCT,B)}(\delta) \lesssim k/m \lesssim \rho^{niht}_{(DCT,B)}(\delta)$ and $\delta = m/n \lesssim 4/5$. HTP has a smaller recovery time for lower values of values of k/m, and CSMPSP has an even lower recovery time as m/n approaches one or k/m approaches zero. (See Fig. 2 (c).) Throughout the recovery region the algorithms have recovery times that are typically within a multiple of five of each other, and often much closer. (See [37, Fig. S7].) For $m/n < .8$, NIHT's recovery time is always within a multiple of two of the fastest algorithm. (See Fig. 2 (i).)

Obs. 3.2.3 encourages the use of NIHT for the problem class (DCT, B), with the computational time scaling approximately proportional to n. NIHT has its greatest recovery time at its phase transition, but even there Fig. 2(f) shows NIHT reliably recovers the measured vector with $n = 2^{20}$ with an average recovery time under 250ms.

m	k	Algorithm	successes / 1000 tests	fraction of correct support		time (ms)	iterations	convergence rate
				successes	failures	successes	successes	successes
246	19	NIHT	1000	1(± 0)	–	8.6(± 0.9)	14(± 1.4)	0.442(± 0.037)
		HTP	1000	1(± 0)	–	13.3(± 3.7)	2(± 0.5)	0.002(± 0.004)
		CSMPSP	1000	1(± 0)	–	13.2(± 2.2)	2(± 0.3)	0.001(± 0.002)
	28	NIHT	984	1(± 0)	0.507(± 0.073)	13.2(± 1.8)	22(± 2.8)	0.531(± 0.029)
		HTP	990	1(± 0)	0.546(± 0.084)	28.5(± 10.0)	4(± 1.4)	0.051(± 0.058)
		CSMPSP	998	1(± 0)	0.411(± 0.076)	21.8(± 4.5)	3(± 0.7)	0.011(± 0.020)
	34	NIHT	696	1(± 0)	0.511(± 0.106)	18.9(± 3.8)	31(± 6.0)	0.590(± 0.033)
		HTP	718	1(± 0)	0.525(± 0.106)	53.8(± 37.4)	8(± 5.2)	0.180(± 0.127)
		CSMPSP	860	1(± 0)	0.497(± 0.100)	34.5(± 13.6)	6(± 2.1)	0.106(± 0.090)
410	37	NIHT	1000	1(± 0)	–	12.9(± 0.9)	17(± 1.2)	0.486(± 0.020)
		HTP	1000	1(± 0)	–	22.5(± 4.9)	3(± 0.5)	0.005(± 0.005)
		CSMPSP	1000	1(± 0)	–	20.7(± 4.1)	2(± 0.5)	0.001(± 0.002)
	56	NIHT	992	1(± 0)	0.585(± 0.111)	21.4(± 2.9)	28(± 3.6)	0.597(± 0.024)
		HTP	991	1(± 0)	0.605(± 0.071)	50.7(± 19.7)	5(± 2.0)	0.091(± 0.073)
		CSMPSP	999	1(± 0)	0.571(± 0.000)	35.7(± 6.4)	5(± 0.8)	0.052(± 0.035)
	67	NIHT	577	1(± 0)	0.577(± 0.088)	32.4(± 6.9)	42(± 8.6)	0.654(± 0.020)
		HTP	567	1(± 0)	0.584(± 0.090)	111.5(± 73.9)	12(± 7.4)	0.287(± 0.132)
		CSMPSP	896	1(± 0)	0.568(± 0.072)	60.5(± 27.7)	7(± 2.9)	0.161(± 0.101)
1178	153	NIHT	1000	1(± 0)	–	38.2(± 1.6)	24(± 1.0)	0.597(± 0.015)
		HTP	1000	1(± 0)	–	70.4(± 9.8)	3(± 0.4)	0.010(± 0.009)
		CSMPSP	1000	1(± 0)	–	68.2(± 5.7)	3(± 0.5)	0.012(± 0.012)
	230	NIHT	1000	1(± 0)	–	68.4(± 5.8)	43(± 3.5)	0.715(± 0.010)
		HTP	997	1(± 0)	0.707(± 0.035)	171.5(± 43.8)	7(± 1.8)	0.172(± 0.069)
		CSMPSP	1000	1(± 0)	–	100.5(± 9.0)	5(± 0.6)	0.053(± 0.024)
	275	NIHT	330	1(± 0)	0.691(± 0.057)	129.1(± 24.0)	80(± 14.7)	0.764(± 0.008)
		HTP	175	1(± 0)	0.682(± 0.059)	495.2(± 237.0)	21(± 9.9)	0.444(± 0.062)
		CSMPSP	994	1(± 0)	0.679(± 0.051)	169.4(± 114.7)	8(± 4.9)	0.161(± 0.083)

Table II. Average and standard deviation of performance characteristics for NIHT, HTP, and CSMPSP for $(\mathcal{N}, B)$ with $n = 2^{12}$

m	k	Algorithm	successes / 1000 tests	fraction of correct support		time (ms)	iterations	convergence rate
				successes	failures	successes	successes	successes
656	32	NIHT	831	1(± 0)	0.367(± 0.115)	13.5(± 4.9)	20(± 7.5)	0.458(± 0.038)
		HTP	1000	1(± 0)	–	19.5(± 5.0)	3(± 0.7)	0.009(± 0.014)
		CSMPSP	996	1(± 0)	0.164(± 0.174)	17.7(± 2.4)	3(± 0.7)	0.009(± 0.021)
	48	NIHT	1	1(± 0)	0.201(± 0.114)	33.4(± 0.0)	51(± 0.0)	0.493(± 0.000)
		HTP	999	1(± 0)	0.604(± 0.000)	45.0(± 11.5)	6(± 1.5)	0.128(± 0.070)
		CSMPSP	998	1(± 0)	0.375(± 0.442)	28.0(± 4.4)	5(± 1.1)	0.079(± 0.049)
	58	NIHT	0	–	0.127(± 0.094)	–	–	–
		HTP	754	1(± 0)	0.568(± 0.115)	74.6(± 23.8)	10(± 3.2)	0.291(± 0.107)
		CSMPSP	916	1(± 0)	0.560(± 0.095)	44.6(± 19.6)	7(± 2.8)	0.164(± 0.098)
3933	305	NIHT	992	1(± 0)	0.736(± 0.118)	49.9(± 16.5)	70(± 22.9)	0.513(± 0.024)
		HTP	1000	1(± 0)	–	39.7(± 9.3)	5(± 1.1)	0.056(± 0.040)
		CSMPSP	1000	1(± 0)	–	31.3(± 4.8)	4(± 1.3)	0.043(± 0.054)
	458	NIHT	921	1(± 0)	0.656(± 0.071)	72.0(± 33.1)	100(± 45.8)	0.623(± 0.017)
		HTP	1000	1(± 0)	–	64.9(± 9.7)	8(± 1.1)	0.176(± 0.049)
		CSMPSP	994	1(± 0)	0.149(± 0.212)	55.8(± 10.2)	12(± 5.2)	0.333(± 0.229)
	549	NIHT	445	1(± 0)	0.613(± 0.068)	93.9(± 24.2)	130(± 33.5)	0.668(± 0.013)
		HTP	486	1(± 0)	0.674(± 0.053)	176.3(± 60.9)	21(± 7.1)	0.444(± 0.041)
		CSMPSP	997	1(± 0)	0.047(± 0.030)	77.0(± 16.6)	14(± 6.5)	0.398(± 0.264)
6554	589	NIHT	1000	1(± 0)	–	34.2(± 4.7)	47(± 6.3)	0.559(± 0.022)
		HTP	1000	1(± 0)	–	37.4(± 6.2)	5(± 0.7)	0.038(± 0.024)
		CSMPSP	1000	1(± 0)	–	43.5(± 9.3)	8(± 4.8)	0.203(± 0.224)
	883	NIHT	997	1(± 0)	0.696(± 0.074)	75.3(± 20.3)	103(± 27.4)	0.663(± 0.013)
		HTP	1000	1(± 0)	–	71.1(± 7.4)	8(± 0.8)	0.191(± 0.037)
		CSMPSP	989	1(± 0)	0.135(± 0.229)	65.3(± 11.6)	13(± 6.4)	0.422(± 0.300)
	1060	NIHT	424	1(± 0)	0.667(± 0.052)	111.0(± 23.0)	150(± 30.8)	0.710(± 0.011)
		HTP	312	1(± 0)	0.693(± 0.043)	230.3(± 76.4)	26(± 8.6)	0.455(± 0.012)
		CSMPSP	1000	1(± 0)	–	87.5(± 17.5)	17(± 6.9)	0.494(± 0.273)
18833	2460	NIHT	1000	1(± 0)	–	43.8(± 4.5)	57(± 5.7)	0.659(± 0.013)
		HTP	1000	1(± 0)	–	45.8(± 2.7)	5(± 0.3)	0.053(± 0.015)
		CSMPSP	998	1(± 0)	0.103(± 0.119)	43.0(± 4.1)	6(± 1.8)	0.111(± 0.072)
	3690	NIHT	1000	1(± 0)	–	85.0(± 13.5)	111(± 17.2)	0.759(± 0.008)
		HTP	999	1(± 0)	1.000(± 0.000)	103.5(± 6.1)	11(± 0.7)	0.291(± 0.025)
		CSMPSP	989	1(± 0)	0.200(± 0.343)	56.5(± 2.8)	8(± 1.1)	0.162(± 0.047)
	4428	NIHT	86	1(± 0)	0.730(± 0.033)	200.0(± 24.2)	258(± 31.1)	0.804(± 0.006)
		HTP	0	–	0.689(± 0.017)	–	–	–
		CSMPSP	1000	1(± 0)	–	80.4(± 5.5)	10(± 1.6)	0.250(± 0.059)

Table III. Average and standard deviation of performance characteristics for NIHT, HTP, and CSMPSP for $(\mathcal{S}_7, B)$ with $n = 2^{16}$

3.3. *Variance of algorithm recovery behavior:* (Mat, B)

Claim 3 (Algorithm variability: binary vectors without noise)
Within the recovery region for each of NIHT, HTP, and CSMPSP, and the problem class (Mat, B) *with* $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$*, at any* (k, m, n) *with* m *and* n *fixed, the variability of the following*

algorithm performance characteristics decreases as $\rho = k/m$ decreases: fraction of support set identified, time for recovery, iterations, and asymptotic convergence rate. (See Tabs. I–III.)

The low variability of the asymptotic convergence rate is to be expected from the concentration of eigenvalues of the random matrix ensembles tested. The low variability of the number of iterations and fraction of correct support set suggests that the algorithms tested are likely to be identifying approximately the same fraction of the correct support set per iteration. This behavior is established by testing the algorithms for large numbers of problem instances for a given problem size (k, m, n). In particular, for m, n fixed, variability decreases as k decreases. Recall from Sec. 2.1, for each problem size (k, m, n) one thousand problem instances were tested for each line of Tabs. I–III.

For problem classes (DCT, B), $(\mathcal{N}, B)$, and $(\mathcal{S}_7, B)$ Tabs. I, II, and III respectively present the average and standard deviation of algorithm characteristics: fraction of correct support set identified for both successes and failures, and recovery time, number of iterations, and convergence rates for successes. Some apparent discrepancies exist when the algorithm has succeeded on less than 1000 of the 1000 problem instances; this indicates the problem size (k, m, n) is near the algorithm's phase transition. For all three algorithms, the concentration about the mean of each of the performance characteristics intensifies as $\rho = k/m$ decreases below the phase transition.

The low variance of the algorithm characteristics in regions where they recover the measured vector allow for highly reliable conclusions to be drawn on the performance of the tested hard thresholding algorithms in their regions of applicability. This consistent behavior is an important aspect of the reliability of the claims and observations in this manuscript. Moreover, they inform a practitioner that the algorithm can be expected to perform consistently for problem instances from the same class and similar problem sizes.

4. ALGORITHM PERFORMANCE FOR PROBLEM CLASS (Mat, B_ϵ)

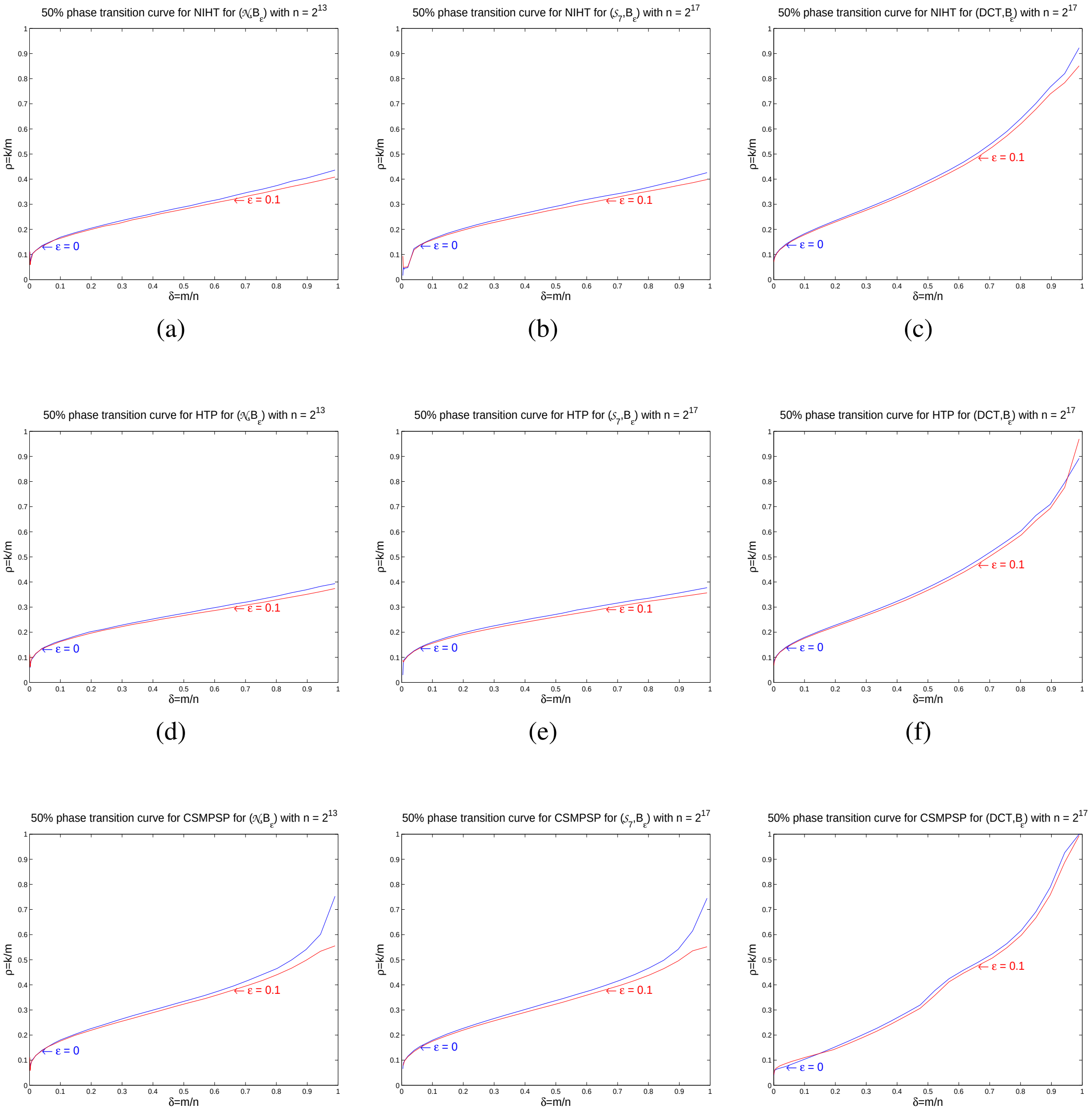
This section expands the empirical investigation to consider the effects of noise by studying problem classes (Mat, B) and (Mat, B_ϵ). Secs. 4.1, 4.2, and 4.3 are aligned with Secs. 3.1, 3.2, and 3.3, respectively. Sec. 5 extends the analysis further to the vector ensembles $\{U_\epsilon, N_\epsilon\}$.

Recall from Sec. 2.2 that this section considers the slightly relaxed definition of success given by (6), rather than (4) considered in the previous section. For $Mat \in \{\mathcal{S}_7, DCT\}$, the tests are conducted with $n = 2^{17}$ and for $Mat = \mathcal{N}$ with $n = 2^{13}$. Though various other noise levels were tested, this manuscript studies the moderate noise level $\epsilon = 1/10$ as representative of the impact on the algorithms of non adversarial, additive noise. For smaller noise levels, the recovery phase transition curves are nearly identical to the recovery phase transition curves without noise. For somewhat larger noise levels, the success condition (6) fails to adequately identify when an algorithm identifies a suitable fraction of the support set.

4.1. Regions of high probability of recovery in the presence of noise: (Mat, B_ϵ)

Claim 4 (Algorithm recovery: effects of moderate noise ($\epsilon = 1/10$))
For $m \ll n$, the recovery phase transitions for algorithms NIHT, HTP, and CSMPSP decrease insubstantially for moderate levels of noise, $\epsilon = 1/10$ for all but δ near one, across all problem classes (Mat, vec) with $Mat \in \{DCT, \mathcal{N}, \mathcal{S}_7\}$ and $vec \in \{B, B_\epsilon\}$. The phase transition curves for problem classes (Mat, B_ϵ) and (Mat, B) are similar throughout the phase space. The relative ℓ_2 error of the recovered vectors scales with the noise level. (See Fig. 3.)

In this section we study the effect of noise on the greedy algorithms by studying the problem classes (Mat, B) and (Mat, B_ϵ) for all matrix ensembles. As detailed in Sec. 2.2 the algorithms attempt to find the vector x from the information $y = Ax + e$ where e is drawn uniformly from the sphere of radius $\epsilon\|Ax\|_2$. For vector ensemble B we set $\epsilon = 0$ to eliminate additive noise and for vector ensemble B_ϵ we set $\epsilon = 1/10$. In other words, the vector ensemble B_ϵ consists of noisy measurements of binary signals with a signal-to-noise ratio equal to 10. This section performs a parallel analysis to that in Sec. 3.1 by describing the effect of noise on the region of successful



(a) (b) (c)

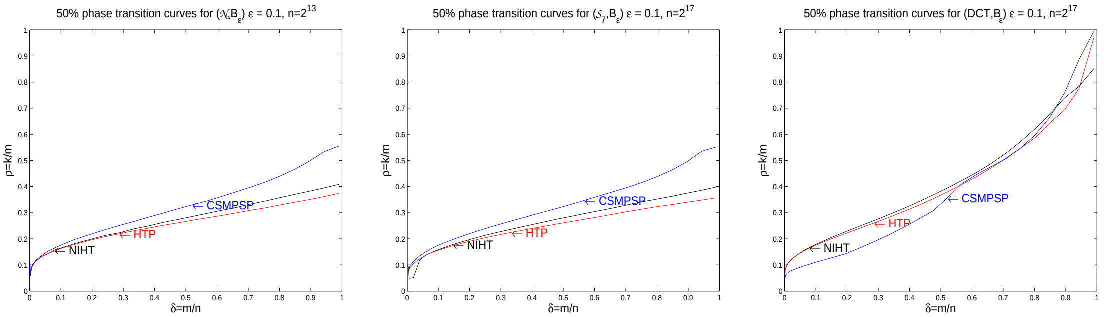
(d) (e) (f)

(g) (h) (i)

Figure 3. 50% recovery probability logistic regression curves for $vec = B_\epsilon$ with $\epsilon = 0$ and $\epsilon = 1/10$. Algorithms: (a-c) NIHT, (d-f) HTP, and (g-i) CSMPSP. Left Panels: $Mat = \mathcal{N}$ with $n = 2^{13}$; Center Panels: $Mat = \mathcal{S}_7$ with $n = 2^{17}$; Right panels: $Mat = DCT$ with $n = 2^{17}$.

recovery. Claim 4 is supported by Obs. 4.1.1 and 4.1.2. The modest reduction in the recovery phase transitions show that the algorithms are stable to noise when using the success condition (6) suggested by the analysis of the worst case behavior of these algorithms. The aforementioned prior analysis is valid only for k/m much smaller that the values presented here, with Claim 4 stating its wider observed applicability.

Observation 4.1.1
With $\epsilon = 1/10$, $alg \in \{$NIHT,HTP,CSMPSP$\}$, and $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$ the phase transition curves $\rho^{alg}_{(Mat,B_\epsilon)}(\delta)$ and $\rho^{alg}_{(Mat,B)}(\delta)$ are within 0.01 of each other for $\delta = m/n < 1/3$. The region of successful recovery for problem class (Mat, B_ϵ) is the overwhelming majority of the successful recovery regions for problem class (Mat, B). (See Fig. 3.)



(a) (b) (c)

Figure 4. 50% recovery probability logistic regression curves for $vec = B_\epsilon$ with $\epsilon = 1/10$ and algorithms NIHT, HTP, and CSMPSP: (a) $Mat = \mathcal{N}$ with $n = 2^{13}$, (b) $Mat = \mathcal{S}_7$ with $n = 2^{17}$, and (c) $Mat = DCT$ with $n = 2^{17}$.

Observation 4.1.1 describes what is commonly referred to as a "stability to noise" of the recovery phase transition. Each of these algorithms has a theoretical sufficient condition based on the restricted isometry property which guarantees the relative ℓ_2 recovery error is bounded by a factor scaling with the noise level ϵ. While these sufficient conditions are incredibly pessimistic in the values of k for which they are applicable [29, 30], Obs. 4.1.1 demonstrates all three algorithms' recovery errors are proportional to the noise level ϵ in the majority of problem instances within the noiseless recovery region.

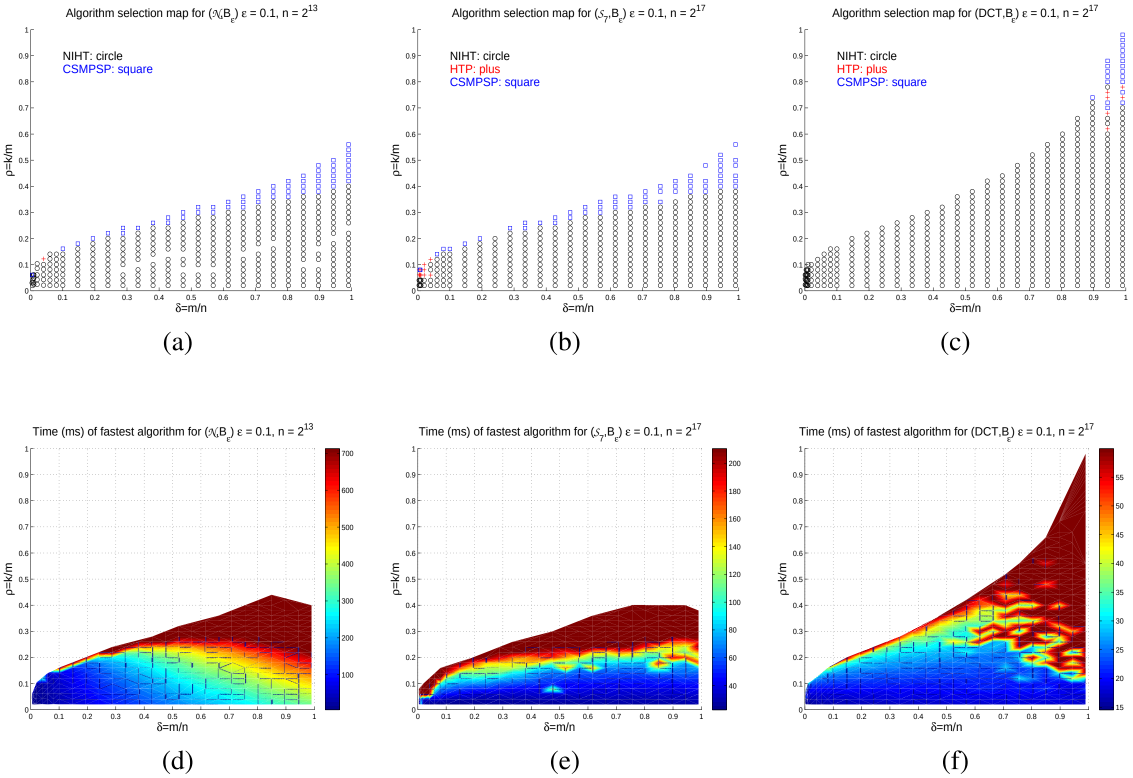
Observation 4.1.2
For $\epsilon = 1/10$ and $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$, the relationships between the phase transition curves for NIHT, HTP, and CSMPSP for problem class (Mat, B_ϵ) are consistent with Observations 3.1.2 and 3.1.1. (See Figs. 1 and 4.)

As the recovery phase transition curves for (Mat, B_ϵ) track closely with the related recovery phase transition from (Mat, B), the relative positioning of the moderate noise recovery phase transitions remain the same as the noise free case. Observation 4.1.2 states that the algorithms' regions of successful recovery in the presence of noise share the same inclusion relationships as those without noise. While analogous results on the locations of the recovery phase transition curves for greedy algorithms have been reported in previous work [32, 33] and for noisy measurements with the Approximate Message Passing algorithm [20], this appears to be the first report on recovery phase transitions for greedy algorithms for noisy measurements.

4.2. *Average time for recovery and algorithm selection maps in the presence of noise:* (Mat, B_ϵ)

Claim 5 (Algorithm selection: sparse binary vectors with moderate noise ($\epsilon = 1/10$))
Across the problem classes (Mat, B_ϵ) contaminated by moderate noise ($\epsilon = 1/10$) and $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$, NIHT recovers vectors in less time than HTP or CSMPSP throughout the overwhelming majority of the phase space. (See Fig. 5.) Moreover, when NIHT is not the fastest algorithm and $m/n < 1/2$, NIHT never requires more than 10% more time than that of HTP or CSMPSP. (See [37, Sec. S4.2].)

Though the relationships between the recovery regions change little with the inclusion of moderate noise, the noise has a profound effect on the relationships of other performance characteristics of these algorithms. Claim 5 states that, in terms of average recovery time, the projection based methods HTP and CSMPSP are more significantly degraded by the presence of noise than the steepest descent based method NIHT. Plots of ratios of recovery times presented in [37] show that HTP and CSMPSP often take more than 10 times longer than NIHT to recover the



(a) (b) (c)

(d) (e) (f)

Figure 5. (a-c): Algorithm selection maps with moderate noise $\epsilon = 1/10$. (d-f): Average time for the fastest algorithm. Left panels: $(\mathcal{N}, B_\epsilon)$ with $n = 2^{13}$; Center panels: $(\mathcal{S}_7, B_\epsilon)$ with $n = 2^{17}$; Right panels: (DCT, B_ϵ) with $n = 2^{17}$.

noise corrupted vectors. On the other hand, in the most interesting region of the phase space for compressed sensing, namely $\delta = m/n < 1/2$, the timing ratio plots show that NIHT never requires more than 1.1 times as long as the fastest recovery algorithm. Claim 5 is supported by the more detailed Obs. 4.2.1.

Observation 4.2.1
For the problem class (Mat, B_ϵ) with $\epsilon = 1/10$ and $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$, NIHT is able to reliably successfully recover the measured vector in the least time throughout the region where multiple algorithms are able to reliably recover the measured vectors. Due to its higher recovery phase transition, CSMPSP is the fastest algorithm throughout the region where it is the only algorithm able to successfully recover the measured vector. For the problem class $(\mathcal{S}_7, B_\epsilon)$, as m/n approaches zero HTP recovers the vector in the least time. (See Fig. 5.) Moreover, NIHT is often 10 to 40 times faster than HTP and CSMPSP, and for $m/n < 1/2$ the recovery time for NIHT is never more than 2 times that of the fastest algorithm. (See [37, Sec. S4.2].)

Observation 4.2.1 encourages the use of NIHT for problems where noise is present. In specific cases where the problem dimensions fall outside of the recovery region for NIHT, CSMPSP is recommended. Furthermore, Fig. 5(d-f) indicate that noisy measurements of a signal from $vec = B_\epsilon$ with $\epsilon = 1/10$ and $n = 2^{17}$ can be recovered in under 60ms for $Mat = DCT$ and under 200ms for $Mat = \mathcal{S}_7$. When using the dense matrices from $Mat = \mathcal{N}$, a noisy signal from B_ϵ with $n = 2^{13}$ can be recovered by NIHT in less than 200ms when $m/n < 0.3$ and $k/m < \rho^{niht}_{(\mathcal{N},B_\epsilon)}(m/n)$.

m	k	Algorithm	successes /	fraction of correct support		time (ms)	iterations
			1000 tests	successes	failures	successes	successes
2622	141	NIHT	1000	1(± 0)	–	22.9(± 1.6)	14(± 1.0)
		HTP	1000	1(± 0)	–	429.3(± 11.8)	18(± 0.5)
		CSMPSP	1000	1(± 0.000919)	–	434.5(± 13.8)	19(± 0.6)
	211	NIHT	1000	1(± 0)	–	30.7(± 1.6)	19(± 1.0)
		HTP	1000	1(± 0)	–	466.6(± 5.4)	20(± 0.2)
		CSMPSP	0	–	0.840(± 0.043)	–	–
	253	NIHT	996	1(± 0)	0.520(± 0.041)	44.0(± 6.2)	27(± 3.7)
		HTP	1000	1(± 0)	–	524.6(± 33.1)	22(± 1.4)
		CSMPSP	0	–	0.609(± 0.053)	–	–
15729	1239	NIHT	1000	1(± 0)	–	26.7(± 1.4)	16(± 0.9)
		HTP	1000	1(± 0)	–	459.0(± 2.6)	19(± 0.0)
		CSMPSP	1000	0.999(± 0.001)	–	467.3(± 10.5)	20(± 0.4)
	1858	NIHT	1000	1(± 0)	–	38.1(± 1.6)	23(± 1.0)
		HTP	1000	1(± 0)	–	500.5(± 11.2)	21(± 0.5)
		CSMPSP	0	–	0.842(± 0.012)	–	–
	2229	NIHT	1000	1(± 0)	–	53.4(± 2.7)	32(± 1.6)
		HTP	1000	1(± 0)	–	586.0(± 19.4)	24(± 0.8)
		CSMPSP	0	–	0.664(± 0.017)	–	–
26215	2401	NIHT	1000	1(± 0)	–	28.3(± 1.4)	17(± 0.9)
		HTP	1000	1(± 0)	–	471.7(± 2.8)	19(± 0.0)
		CSMPSP	1000	0.998(± 0.001)	–	484.0(± 4.4)	20(± 0.1)
	3601	NIHT	1000	1(± 0)	–	41.5(± 1.7)	24(± 1.0)
		HTP	1000	1(± 0)	–	521.4(± 3.0)	21(± 0.0)
		CSMPSP	0	–	0.853(± 0.009)	–	–
	4321	NIHT	1000	1(± 0)	–	60.3(± 2.8)	35(± 1.6)
		HTP	1000	1(± 0)	–	627.6(± 19.8)	25(± 0.8)
		CSMPSP	0	–	0.690(± 0.012)	–	–
75332	10451	NIHT	1000	1(± 0)	–	33.5(± 1.5)	18(± 0.8)
		HTP	1000	1(± 0)	–	525.1(± 13.6)	19(± 0.5)
		CSMPSP	1000	1(± 0)	–	568.1(± 32.7)	21(± 1.2)
	15676	NIHT	1000	1(± 0)	–	50.4(± 1.9)	26(± 1.0)
		HTP	1000	1(± 0)	–	585.6(± 60.0)	21(± 2.2)
		CSMPSP	0	–	0.919(± 0.003)	–	–
	18811	NIHT	1000	1(± 0)	–	76.5(± 2.7)	40(± 1.4)
		HTP	1000	1(± 0)	–	744.3(± 144.4)	27(± 5.3)
		CSMPSP	0	–	0.779(± 0.007)	–	–

Table IV. Average and standard deviation of performance characteristics for NIHT, HTP, and CSMPSP for (DCT, B_ϵ) for $\epsilon = 1/10$ with $n = 2^{18}$

4.3. *Variance of algorithm recovery behavior in the presence of noise:* (Mat, B_ϵ)

Claim 6 (Algorithm variability: sparse binary vectors with moderate noise ($\epsilon = 1/10$))
Within the recovery region for each of NIHT, HTP, and CSMPSP, and the problem class (Mat, B_ϵ) *with* $\epsilon = 1/10$ *and* $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$*, at any* (k, m, n) *with* m *and* n *fixed, the variability of the following algorithm performance characteristics decreases as* $\rho = k/m$ *decreases: fraction of support set identified, time for recovery, and iterations. (See Tabs. IV–VI.)*

Claim 6 is supported by the data in Tabs. IV–VI. While the noise effects the magnitudes of the timings and iterations, the algorithms continue to exhibit a predictable, repeated behavior for successful recovery for a specific problem instance (Mat, vec) and (k, m, n). Larger standard deviations relative to the mean are observed when the algorithm has failed to recovery some of the 1000 problems tested indicating the problem size is near the recovery phase transition. The variability of these performance characteristics will decrease as $\rho = k/m$ decreases. Recall from Sec. 2.2 that the problem sizes (k, m, n) in Tabs. IV–VI were chosen to match those in Tabs. I–III and do not necessarily fall into the recovery regions for moderate noise.

5. ALGORITHM PERFORMANCE FOR PROBLEM CLASSES (Mat, N_ϵ) AND (Mat, U_ϵ)

The empirical investigation now expands to include the sparse vector ensembles $\{B, U, N, B_\epsilon, U_\epsilon, N_\epsilon\}$. For this set of vector ensembles, an analysis similar to that performed in Secs. 3 and 4 will produce recovery phase transitions, algorithm selection maps, ratios of recovery time to the fastest algorithm, and information about the concentration of performance characteristics for the algorithms. The total combinations result in more information than can be readily presented in a single article. Here we focus on the case of measurements corrupted by

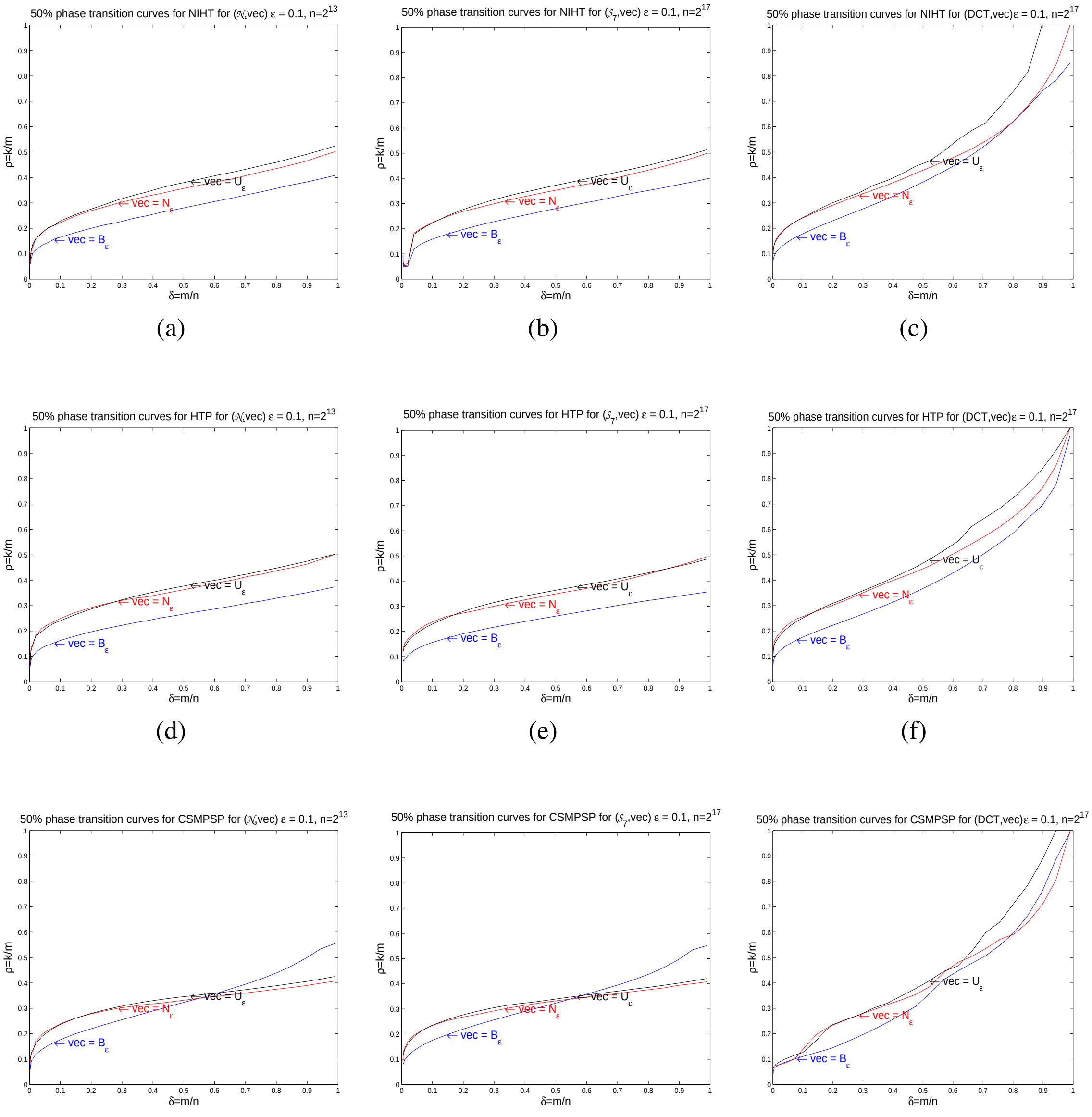
m	k	Algorithm	successes /	fraction of correct support		time (ms)	iterations
			1000 tests	successes	failures	successes	successes
246	19	NIHT	1000	1(± 0)	–	9.5(± 1.0)	16(± 1.7)
		HTP	1000	1(± 0)	–	131.6(± 4.0)	18(± 0.5)
		CSMPSP	1000	1(± 0)	–	134.1(± 11.3)	19(± 1.6)
	28	NIHT	990	1(± 0)	0.518(± 0.117)	14.0(± 2.2)	23(± 3.4)
		HTP	986	1(± 0)	0.531(± 0.085)	148.6(± 18.1)	21(± 2.5)
		CSMPSP	997	1(± 0)	0.548(± 0.055)	160.5(± 71.5)	23(± 10.2)
	34	NIHT	632	1(± 0)	0.525(± 0.103)	19.3(± 3.8)	31(± 6.0)
		HTP	674	1(± 0)	0.535(± 0.108)	178.5(± 41.8)	25(± 5.8)
		CSMPSP	835	1(± 0)	0.511(± 0.101)	199.2(± 114.7)	29(± 16.4)
410	37	NIHT	1000	1(± 0)	–	13.2(± 1.1)	17(± 1.4)
		HTP	1000	1(± 0)	–	185.2(± 5.1)	19(± 0.5)
		CSMPSP	1000	1(± 0)	–	198.0(± 61.6)	20(± 6.3)
	56	NIHT	990	1(± 0)	0.609(± 0.081)	21.4(± 2.8)	28(± 3.5)
		HTP	984	1(± 0)	0.604(± 0.093)	219.0(± 31.8)	22(± 3.2)
		CSMPSP	999	1(± 0)	0.589(± 0.000)	274.2(± 193.5)	28(± 19.9)
	67	NIHT	487	1(± 0)	0.581(± 0.093)	32.0(± 6.8)	41(± 8.4)
		HTP	463	1(± 0)	0.576(± 0.095)	279.1(± 73.5)	28(± 7.3)
		CSMPSP	827	1(± 0)	0.569(± 0.080)	743.0(± 459.6)	76(± 46.9)
1178	153	NIHT	1000	1(± 0)	–	35.4(± 2.1)	22(± 1.3)
		HTP	1000	1(± 0)	–	462.7(± 12.1)	19(± 0.5)
		CSMPSP	1000	1(± 0)	–	864.7(± 768.4)	37(± 32.5)
	230	NIHT	998	1(± 0)	0.739(± 0.012)	64.8(± 7.5)	40(± 4.6)
		HTP	995	1(± 0)	0.764(± 0.035)	586.2(± 70.7)	25(± 2.9)
		CSMPSP	1000	1(± 0)	–	2579.7(± 876.7)	109(± 37.1)
	275	NIHT	145	1(± 0)	0.679(± 0.058)	124.9(± 24.8)	77(± 15.2)
		HTP	45	1(± 0)	0.668(± 0.058)	1026.3(± 296.8)	43(± 12.5)
		CSMPSP	964	1(± 0)	0.681(± 0.045)	2962.0(± 271.5)	126(± 11.5)

Table V. Average and standard deviation of performance characteristics for NIHT, HTP, and CSMPSP for $(\mathcal{N}, B_\epsilon)$ for $\epsilon = 1/10$ with $n = 2^{12}$

m	k	Algorithm	successes /	fraction of correct support		time (ms)	iterations
			1000 tests	successes	failures	successes	successes
656	32	NIHT	966	1(± 0)	0.454(± 0.075)	11.8(± 3.5)	19(± 5.4)
		HTP	1000	1(± 0)	–	139.5(± 4.0)	19(± 0.5)
		CSMPSP	572	1(± 0)	0.000(± 0.000)	153.3(± 59.2)	21(± 8.0)
	48	NIHT	4	1(± 0)	0.354(± 0.082)	18.9(± 4.2)	30(± 6.6)
		HTP	997	1(± 0)	0.632(± 0.048)	160.5(± 11.2)	22(± 1.5)
		CSMPSP	549	1(± 0)	0.001(± 0.026)	239.4(± 217.5)	33(± 29.7)
	58	NIHT	0	–	0.234(± 0.067)	–	–
		HTP	683	1(± 0.00066)	0.569(± 0.123)	194.3(± 25.2)	26(± 3.4)
		CSMPSP	487	1(± 0)	0.086(± 0.204)	348.3(± 292.3)	48(± 39.9)
3933	305	NIHT	978	1(± 0)	0.695(± 0.041)	52.7(± 21.5)	75(± 30.1)
		HTP	1000	1(± 0)	–	177.9(± 10.4)	21(± 1.2)
		CSMPSP	996	1(± 0.000542)	0.000(± 0.000)	521.0(± 405.9)	61(± 47.7)
	458	NIHT	837	1(± 0)	0.630(± 0.065)	71.8(± 31.1)	101(± 43.3)
		HTP	1000	1(± 0)	–	208.1(± 11.4)	24(± 1.3)
		CSMPSP	997	1(± 0.000351)	0.000(± 0.000)	957.6(± 300.9)	112(± 35.0)
	549	NIHT	266	1(± 0)	0.609(± 0.069)	92.3(± 22.9)	129(± 31.8)
		HTP	221	1(± 0)	0.657(± 0.055)	346.3(± 77.6)	40(± 9.0)
		CSMPSP	993	1(± 0.000389)	0.179(± 0.306)	1059.9(± 171.4)	123(± 19.8)
6554	589	NIHT	1000	1(± 0)	–	34.0(± 6.5)	47(± 8.8)
		HTP	1000	1(± 0)	–	180.2(± 6.5)	21(± 0.7)
		CSMPSP	998	1(± 0)	0.002(± 0.002)	702.6(± 436.1)	80(± 49.4)
	883	NIHT	979	1(± 0)	0.705(± 0.072)	75.6(± 24.2)	103(± 32.8)
		HTP	1000	1(± 3.58e-05)	–	220.4(± 8.4)	25(± 0.9)
		CSMPSP	990	1(± 0.000148)	0.019(± 0.016)	1066.2(± 239.1)	120(± 26.8)
	1060	NIHT	148	1(± 0)	0.640(± 0.051)	107.7(± 21.5)	146(± 29.0)
		HTP	38	1(± 0)	0.662(± 0.043)	425.0(± 83.4)	48(± 9.3)
		CSMPSP	995	1(± 0.000284)	0.323(± 0.317)	1103.1(± 168.9)	123(± 18.8)
18833	2460	NIHT	1000	1(± 0)	–	40.6(± 5.3)	54(± 6.9)
		HTP	1000	1(± 0)	–	201.6(± 4.7)	22(± 0.5)
		CSMPSP	1000	1(± 0)	–	1057.7(± 336.9)	113(± 36.0)
	3690	NIHT	1000	1(± 0)	–	81.2(± 15.1)	106(± 19.4)
		HTP	1000	1(± 0)	–	278.0(± 9.7)	30(± 1.0)
		CSMPSP	992	1(± 0)	0.039(± 0.078)	1032.5(± 373.8)	109(± 39.5)
	4428	NIHT	0	–	0.679(± 0.025)	–	–
		HTP	0	–	0.647(± 0.014)	–	–
		CSMPSP	995	1(± 7.16e-06)	0.285(± 0.256)	1141.6(± 243.7)	120(± 25.6)

Table VI. Average and standard deviation of performance characteristics for NIHT, HTP, and CSMPSP for $(\mathcal{S}_7, B_\epsilon)$ for $\epsilon = 1/10$ with $n = 2^{16}$

additive noise which is more likely to present itself in applications and present recovery phase transition curves and algorithm selection maps. Additional data is presented in the supplementary material [37, Sec. S5].



(a) (b) (c)

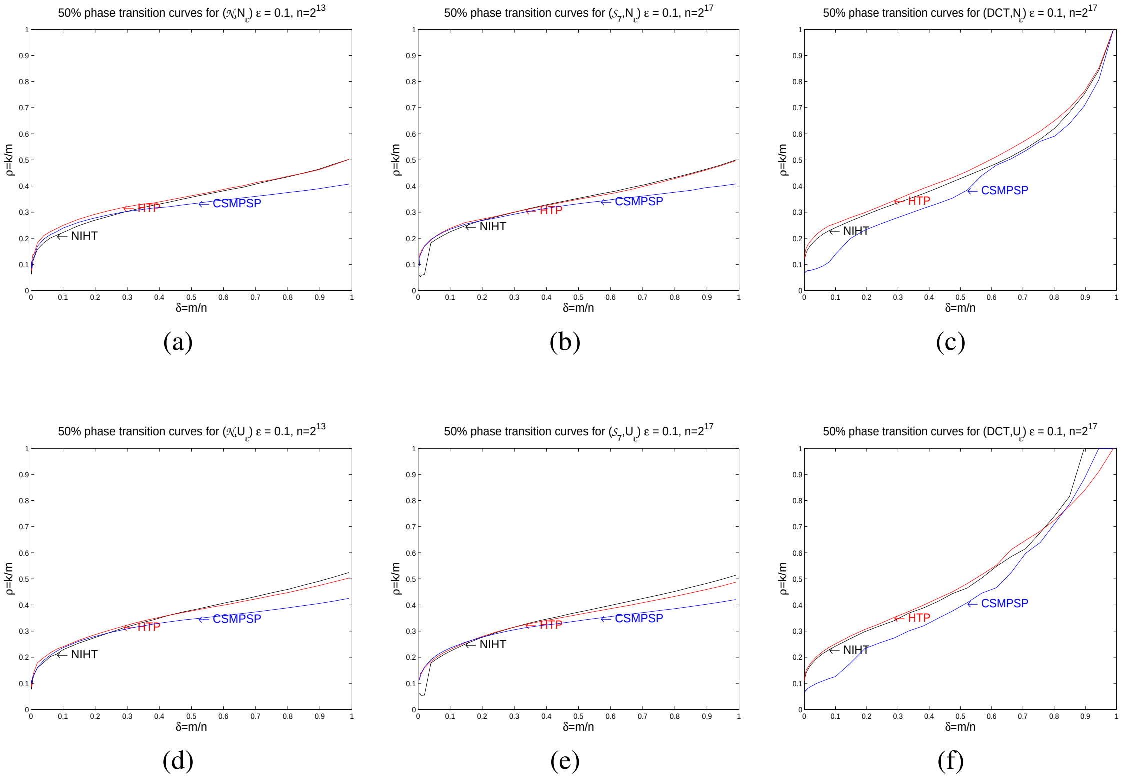
(d) (e) (f)

(g) (h) (i)

Figure 6. 50% recovery probability logistic regression curves for $vec \in \{B_\epsilon, U_\epsilon, N_\epsilon\}$ with $\epsilon = 1/10$ for (a-c) NIHT, (d-f) HTP, and (g-i) CSMPSP. Left panels: $Mat = \mathcal{N}$ with $n = 2^{13}$; Center panels: $Mat = \mathcal{S}_7$ with $n = 2^{17}$; Right panels: $Mat = DCT$ with $n = 2^{17}$.

Claim 7 (Recovery regions: alternate vector ensembles)
For NIHT, HTP, and CSMPSP with $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$, the recovery phase transition for problem classes (Mat, U_ϵ) and (Mat, N_ϵ) are substantially higher than the phase transitions for the problem class (Mat, B_ϵ) for $m/n < 0.5$. (See Fig. 6.) For problem classes (Mat, vec) with $vec \in \{N_\epsilon, U_\epsilon\}$, the recovery phase transition curves for NIHT and HTP are similar to each other with at least one superior to CSMPSP. (See Fig. 7.)

There have been several cases [31, 32, 33] where it has been shown that of all problem classes $(\mathcal{N}, vec)$, $vec = B$ is the most challenging vector ensemble for greedy algorithms. These previous results are extended to the setting of noisy measurements and the matrix ensembles DCT and $\mathcal{S}_7$ through Claim 7. Claim 7 is supported by Observation 5.0.1–5.0.3 and the data presented in Figs. 6



(a) (b) (c)

(d) (e) (f)

Figure 7. 50% recovery probability logistic regression curves for problem classes (Mat, vec) for $vec \in \{N_\epsilon, U_\epsilon\}$ with noise $\epsilon = 1/10$ and algorithms NIHT, HTP, and CSMPSP. (a-c) $vec = N_\epsilon$, (d-f) $vec = U_\epsilon$. Left panels: $Mat = \mathcal{N}$ with $n = 2^{13}$; Center panels: $Mat = \mathcal{S}_7$ with $n = 2^{17}$; Right panels: $Mat = DCT$ with $n = 2^{17}$.

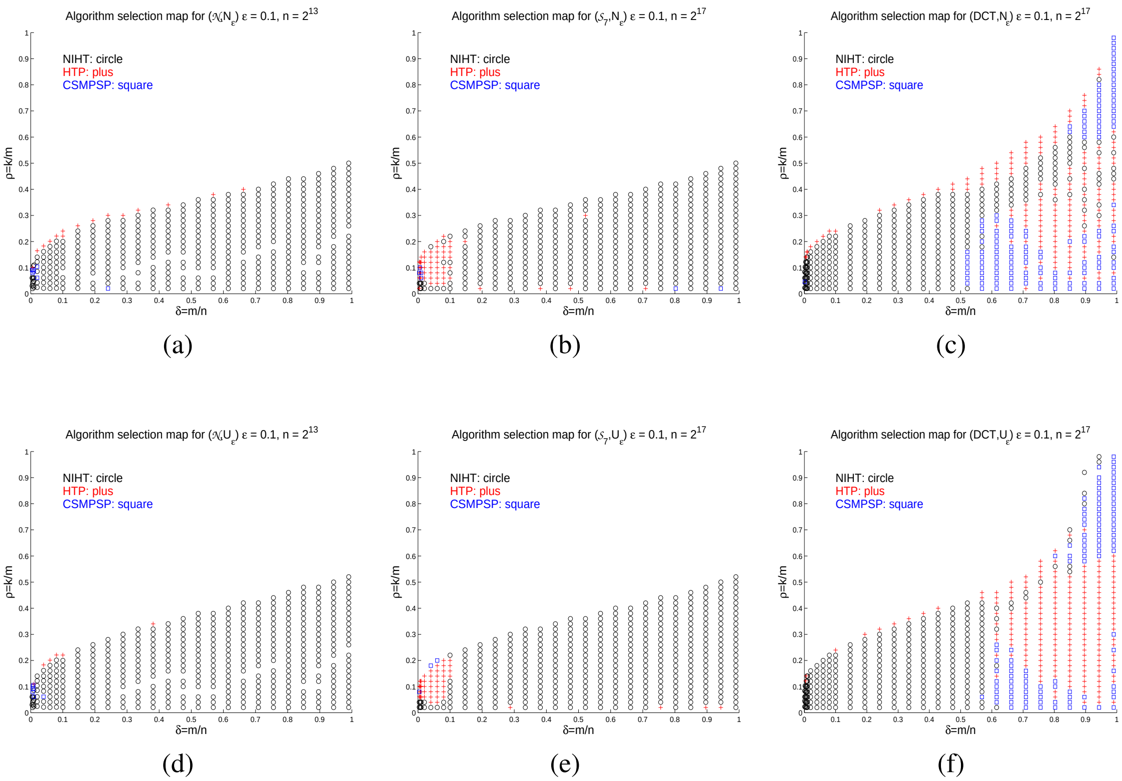
and 7. Figure 6 depicts this behavior for the three matrix ensembles where the recovery phase transition curves for NIHT, HTP, and CSMPSP are plotted for vec from the sparse vector ensembles $\{B_\epsilon, U_\epsilon, N_\epsilon\}$.

Observation 5.0.1
For moderate noise $\epsilon = 1/10$ and across all problem classes (Mat, vec) with $vec \in \{B_\epsilon, U_\epsilon, N_\epsilon\}$, the recovery phase transitions for the sparse vector ensemble B_ϵ is typically lower than for U_ϵ and N_ϵ. Specifically: for all $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$, $\rho^{alg}_{(Mat,U_\epsilon)}(\delta) \approx \rho^{alg}_{(Mat,N_\epsilon)}(\delta) > \rho^{alg}_{(Mat,B_\epsilon)}(\delta)$ for NIHT and HTP and with $\delta = m/n \in (0,1)$ and for CSMPSP with $\delta \in (0, 1/2)$. (See Fig. 6.)

The relative positions of the recovery phase transitions of NIHT, HTP, and CSMPSP for the problem classes (Mat, vec) for $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$ and $vec \in \{U_\epsilon, N_\epsilon\}$ are identified in Obs. 5.0.2 and 5.0.3 and displayed in Fig. 7. Relationships between the recovery regions similar to those depicted in Fig. 7 persist for the noiseless setting $vec \in \{U, N, \}$ [37].

Observation 5.0.2
For moderate noise $\epsilon = 1/10$ and problem classes (Mat, N_ϵ) with $Mat \in \{\mathcal{N}, \mathcal{S}_7\}$, and $\delta \in [1/20, 1/2]$, the recovery phase transitions for NIHT and HTP have phase transition curves within 0.02 of each other. HTP has a noticeably higher phase transition curve than NIHT for the problem class (DCT, N_ϵ). (See Fig. 7(a-c).)

Observation 5.0.3
For moderate noise $\epsilon = 1/10$ and problem classes (Mat, U_ϵ) with $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$, and $\delta \in$



(a) (b) (c)

(d) (e) (f)

Figure 8. Algorithm selection maps for alternate vector distributions with noise $\epsilon = 1/10$. (a-c) $vec = N_\epsilon$, (d-f) $vec = U_\epsilon$. Left panels: $Mat = \mathcal{N}$ with $n = 2^{13}$; Center panels: $Mat = \mathcal{S}_7$ with $n = 2^{17}$; Right panels: $Mat = DCT$ with $n = 2^{17}$.

$[1/20, 1/2]$, the recovery phase transitions for NIHT and HTP have phase transition curves within 0.02 of each other. For $(\mathcal{N}, U_\epsilon)$ and $(\mathcal{S}_7, U_\epsilon)$ and $\delta \in [1/20, 1/4]$, the recovery phase transitions for NIHT, HTP, and CSMPSP have phase transition curves within 0.02 of each other, but as δ exceeds approximately $1/2$ the recovery phase transition for CSMPSP falls below those of NIHT and HTP. (See Fig. 7(d-f).)

Important information omitted from identifying the location of the recovery phase transitions is resolved by identifying which algorithm boasts the smallest time for successful recovery. Claim 8 summarizes these selection maps and is supported by Obs. 5.0.6–5.0.5.

Claim 8 (Algorithm selection maps: alternate vector ensembles in the presence of noise)
For the problem classes (Mat, vec) with $vec \in \{U_\epsilon, N_\epsilon\}$, NIHT is able to reliably recover the measured vector in less time than HTP or CSMPSP through the majority of the recovery region. For the matrix ensemble $\mathcal{S}_7$, HTP is able to reliably recovery the measured vector in less time than NIHT or CSMPSP in the extreme undersampling regime of $\delta = m/n < 1/10$. For the matrix ensemble DCT, NIHT requires the least time for $\delta < 1/2$ while HTP and CSMPSP require less time for $\delta > 1/2$. (See Fig. 8.)

Summary information regarding algorithm selection is given by Obs. 5.0.4, 5.0.5, and 5.0.6 for each matrix ensemble $\mathcal{N}$, $\mathcal{S}_7$, and DCT, respectively. The algorithm selection maps for problem classes (Mat, vec) with $vec \in \{U_\epsilon, N_\epsilon\}$ appear in Fig. 8. Plots of the fastest recovery time and ratios of the algorithms recovery time to the fastest recovery time are available in the supplementary material [37].

Observation 5.0.4
For the problem classes $(\mathcal{N}, U_\epsilon)$ or $(\mathcal{N}, N_\epsilon)$ with $\epsilon = 1/10$, NIHT reliably recovers the measured vector in less time than HTP or CSMPSP throughout the overwhelming majority of the phase space with two exceptions. HTP has the least recovery as $\rho = k/m$ approaches the recovery phase transition for HTP, $\rho^{htp}_{(\mathcal{N},vec_\epsilon)}(\delta)$. CSMPSP recovers the measured vector in the least time for $\delta = m/n \lesssim .002$. Throughout its recovery region, NIHT never requires more than 1.4 times the fastest recovery time. (See Fig. 8 (Left Panels)) and [37, Sec. S5.2].)

Observation 5.0.5
For the problem classes $(\mathcal{S}_7, U_\epsilon)$ or $(\mathcal{S}_7, N_\epsilon)$ with $\epsilon = 1/10$, HTP reliably recovers the measured vector in the least amount of time for $k/m \lesssim \rho^{htp}_{(\mathcal{S}_7,vec_\epsilon)}(\delta)$ with $\delta \lesssim 0.1$. NIHT reliably recovers the measured vector in the least time for $\delta \gtrsim 0.1$ and never requires more than 2 times the fastest recovery time for $\delta \in (0, 1)$. (See Fig. 8 (Center Panels) and [37, Sec. S5.2].)

Observation 5.0.6
For the problem classes (DCT, U_ϵ) or (DCT, N_ϵ) with $\epsilon = 1/10$, NIHT reliably recovers the measured vector in the least amount of time for $k/m \lesssim \rho^{niht}_{(DCT,vec_\epsilon)}(\delta)$ with $\delta \lesssim \frac{1}{2}$, and never requires more than 3 times the fastest recovery time. For $\delta \lesssim \frac{1}{2}$, HTP and CSMPSP have mean recovery times 5 to 10 times larger than the average recovery time for NIHT; though for $\delta > 1/2$ HTP and CSMPSP require the least time. (See Fig. 8 (Right Panels) and [37, Sec. S5.2].)

Observations 5.0.4–5.0.6 coupled with Obs. 4.2.1 provide convincing evidence that when recovering vectors from measurements contaminated with a moderate level of noise, NIHT is the algorithm of choice. Restricting attention to the region of the phase space with at least two times undersampling, $m/n < 1/2$, NIHT generally recovers a vector from noise contaminated measurements faster than the other two algorithms. Importantly, the exceptions to this rule are clearly identified in Obs. 5.0.4–5.0.6 and 4.2.1. In particular, for matrix ensemble $\mathcal{S}_7$ and $m/n < 0.1$, HTP is the algorithm of choice when recovering vectors from measurements contaminated with a moderate level of noise.

6. CONCLUSION

In compressed sensing one seeks to identify both a measurement process of a k sparse vector that requires very few measurements and a computationally efficient recovery algorithm which will identify the measured vector. In this paper, previous empirical studies identifying the recovery regions for a particular problem class have been extended to include alternate vector ensembles and the effects of noise. As the problem instances approach the recovery phase transition curve, the convergence of the iterative algorithms can be arbitrarily slow. Therefore, a practitioner needs to identify the number of measurements required to push the ratio k/m into the algorithms' recovery regions. Often, this results in a problem size where multiple algorithms will reliably return the correct solution. This paper focuses on distinguishing the performance of the algorithms, particularly in the regions where multiple algorithms typically succeed. The algorithm selection maps included in this paper inform practitioners and theorists of the algorithm that is able to reliably recover the measured vector in the least computation time. Furthermore, the concentration of performance characteristics for each algorithm tells practitioners that an algorithm will have consistent, predictable behavior and provides a foundation for the observations and claims made by this article.

The algorithm selection maps also highlight possible areas of future algorithm development. The algorithms NIHT, HTP, and CSMPSP have a high fraction of their recovery regions in common, with this more pronounced for smaller values of m/n. In these regions where all of the algorithms are able to recover the measured vector, it is typically NIHT that is able to do so in the least time. The data suggests this is a consequence of early iterations in HTP and CSMPSP performing excessive computations to determine the nonzero values for a currently estimated support, followed by abruptly changing the support set. However, the data also suggests this inefficiency of the early iterations of HTP and CSMPSP is nearly overcome by their fast asymptotic rate, which follows in part from their implentation with the conjugate gradient method. These observations suggest the possibility of an algorithm that would be superior to all of the algorithms tested here by better balancing the low per iteration complexity of NIHT and the fast asymptotic rates of HTP and CSMPSP.

This manuscript is quantitative where precise statements provide valuable information and is qualitative for statements where further precision is of limited value. Highly quantitative statements can be readily obtained by performing a large number of tests for a given problem instance, and the software, GAGA [35], is designed to handle such large-scale testing. However, many of the most interesting observations in this paper would not be substantially more informative when made more quantitative. The algorithm selection maps identify the fastest algorithm and the associated recovery time ratios provide a general idea of how each algorithm compares to the fastest; a precise identification of this ratio is of secondary importance. Claims 1–8 provide a general summary of the substantial amount of data contained in this manuscript. These claims are informed by the more specific observations from Secs. 3–5 and the supplementary material [37]. Conducting additional performance comparisons for other algorithms and formally testing hypotheses regarding algorithm behavior can be readily accomplished using GAGA [34, 35] and is slated for future work.

ACKNOWLEDGEMENTS

The authors thank the anonymous referees for helping improve this manuscript. JDB was supported by NSF DMS 1112612 and NSF OISE 0854991. JT was supported by an Nvidia Professor Partnership.

REFERENCES

1. Donoho DL. Compressed sensing. *IEEE Transactions on Information Theory* 2006; **52**(4):1289–1306.
2. Candès EJ. Compressive sampling. *International Congress of Mathematicians. Vol. III.* European Mathematical Society, Zürich, 2006; 1433–1452.
3. APS M. Mosek optimization software. `http://www.mosek.com`.
4. CVX Research, Inc. CVX: Matlab software for disciplined convex programming, version 2.0 beta. `http://cvxr.com/cvx` Sep 2012.
5. ILOG CPLEX. Mathematcial programming system. `http://www.cplex.com`.
6. Figueiredo MAT, Nowak RD, Wright SJ. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal Selected Topics in Signal Processing* 2007; **1**(4):586–597.
7. Berg Evd, Friedlander MP. Probing the pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing* 2008; **31**(2):890–912.
8. Wright SJ, Nowak RD, Figueiredo MAT. Sparse reconstruction by separable approximation. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing* 2008; .
9. Yin W, Osher S, Goldfarb D, Darbon J. Bregman iterative algorithms for ℓ^1-minimization with applications to compressed sensing. *SIAM Journal on Imaging Science* 2008; **1**(1):143–168.
10. Donoho DL. Neighborly polytopes and sparse solution of underdetermined linear equations 2004. Technical Report, Department of Statistics, Stanford University.
11. Donoho DL. High-dimensional centrally symmetric polytopes with neighborliness proportional to dimension. *Discrete and Computational Geometry* 2006; **35**(4):617–652.
12. Xu W, Hassibi B. Precise stability phase transitions for ℓ_1 minimization: A unified geometric framework. *IEEE Transactions on Information Theory* 2011; **57**(10):6894 –6919.
13. Donoho DL, Tanner J. Neighborliness of randomly projected simplices in high dimensions. *Proceedings of the National Academy of Sciences USA* 2005; **102**(27):9452–9457 (electronic).
14. Donoho DL, Tanner J. Sparse nonnegative solution of underdetermined linear equations by linear programming. *Proceedings of the National Academy of Sciences USA* 2005; **102**(27):9446–9451 (electronic).
15. Donoho DL, Tanner J. Counting faces of randomly projected polytopes when the projection radically lowers dimension. *Journal of the American Mathematical Society* 2009; **22**(1):1–53.
16. Donoho DL, Tanner J. Counting the faces of randomly-projected hypercubes and orthants, with applications. *Discrete and Computational Geometry* 2010; **43**(3):522–541.
17. Recht B, Xu W, Hassibi B. Null space conditions and thresholds for rank minimization. *Mathematical Programming. Series B.* 2011; **127**:175–211.
18. Stojnic M. Optimization in block-sparse compressed sensing and its strong thresholds. *Selected Topics in Signal Processing, IEEE Journal of* 2010; **4**(2):350–357.
19. Donoho DL, Maleki A, Montanari A. Message-passing algorithms for compressed sensing. *Proceedings of the National Academy of Sciences* 2009; **106**(45):18 914–18 919.
20. Donoho DL, Maleki A, Montanari A. The noise-sensitivity phase transition in compressed sensing. *IEEE Transactions on Information Theory* 2011; **57**(10):6920–6941.

21. Donoho D, Tanner J. Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing. *Philosophical Transactions of the Royal Society A* 2009; **367**(1906):4273–4293. With electronic supplementary materials available online.
22. Monajemi H, Jafarpour S, Gavish M, Stat 330/CME 362 Collaboration, Donoho DL. Deterministic matrices matching the compressed sensing phase transitions of gaussian random matrices. *Proceedings of the National Academy of Sciences* 2013; **110**(4):1181–1186.
23. Needell D, Tropp J. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis* 2009; **26**(3):301–321.
24. Dai W, Milenkovic O. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Transactions on Information Theory* 2009; **55**(5):2230–2249.
25. Blumensath T, Davies ME. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis* 2009; **27**(3):265–274.
26. Blumensath T, Davies ME. Normalised iterative hard thresholding; guaranteed stability and performance. *IEEE Selected Topics in Signal Processing* 2010; **4**(2):298–309.
27. Foucart S. Hard thresholding pursuit: an algorithm for compressive sensing. *SIAM Journal on Numerical Analysis* 2011; **49**(6):2543–2563.
28. Candes EJ, Tao T. Decoding by linear programming. *IEEE Transactions on Information Theory* 2005; **51**(12):4203–4215.
29. Blanchard JD, Cartis C, Tanner J, Thompson A. Phase transitions for greedy sparse approximation algorithms. *Applied and Computational Harmonic Analysis* 2011; **30**(2):188–203.
30. Cartis C, Thompson A. A new and improved quantitative recovery analysis for iterative hard thresholding algorithms in compressed sensing 2013. Submitted, [Online] http://eprints.maths.ox.ac.uk/1752/.
31. Donoho D, Johnstone I, Montanari A. Accurate prediction of phase transitions in compressed sensing via a connection to minimax denoising 2013. URL `http://front.math.ucdavis.edu/1111.1041`.
32. Maleki A, Donoho D. Optimally tuned iterative reconstruction algorithms for compressed sensing. *Selected Topics in Signal Processing, IEEE Journal of* april 2010; **4**(2):330 –341.
33. Sturm B. Sparse vector distributions and recovery from compressed sensing 2011. ArXiv:1103.6246v2.
34. Blanchard JD, Tanner J. GPU accelerated greedy algorithms for compressed sensing. *Mathematical Programming Computation* 2013; **5**(3):267–304.
35. Blanchard JD, Tanner J. GAGA: GPU Accelerated Greedy Algorithms 2012. URL `www.gaga4cs.org`, version 1.0.0.
36. Hestenes M, Stiefel E. Methods of conjugate gradients for solving linear systems. *Joutnal of Research of the National Bureau of Standards* 1952; **49**(6):409–436.
37. Blanchard JD, Tanner J. Performance comparisons of greedy algorithms in compressed sensing: supplementary material 2014. [Available]: Online.
38. Blanchard JD, Cartis C, Tanner J. Compressed Sensing: How sharp is the restricted isometry property? *SIAM Review* 2011; **53**(1):105–125.
39. Donoho DL, Tanner J. Precise undersampling theorems. *Proceedings of the IEEE* 2010; **98**(6):913–924.

Supplementary Material for "Performance Comparison of Greedy Algorithms for Compressed Sensing" by J.D. Blanchard and J. Tanner

Jeffrey D. Blanchard[1]* and Jared Tanner[2]†

[1] *Department of Mathematics and Statistics, Grinnell College, Grinnell, IA 50112*
[2] *Mathematics Institute, University of Oxford, 24-29 St Giles', Oxford OX1 3LB, UK*

S1. INTRODUCTION TO SUPPLEMENTARY MATERIAL

This manuscript contains supplementary material for the article *"Performance Comparisons of Greedy Algorithms for Compressed Sensing"* by J.D. Blanchard and J. Tanner. The sections in this supplementary document are aligned with the sections in the main article; in some cases, section numbers are skipped in order to maintain the alignment with the main article. The sections, figures, and observations in the supplementary material are prefixed with the letter S. All references to numbers without the S prefix refer to the main article. This supplementary document is outlined as follows.

S2. In Sec. 2, the stopping criteria, definition of success, and logistic regression for defining the recovery phase transition curves are described. This supplementary section provides additional information supporting the choices made for the experimental set-up including justification of the stopping criteria and measure of success, and a definition of the logistic regression employed to determine the recovery phase transitions.

S3. This supplementary section provides supporting information to Sec. 3.

S3.1. In Sec. 3, data is presented for the largest value of n tested for each matrix ensemble. Section S3.1 includes additional information showing that these findings are consistent for smaller values of n and depicts the concentration of the phase transitions as n increases.

S3.2. In Sec. 3, Fig. 2 contains algorithm selection maps, fastest recovery times, and example timing ratio plots for the largest value of n tests. The omitted algorithm timing ratios at the largest values of n are presented here in Sec. S3.2a. Section S3.2b includes algorithm selection maps with fastest recovery times for smaller values of n which depict a consistent algorithm selection as n changes.

S4. This supplementary section provides supporting information for Sec. 4.

S4.2. Section 4.2 presents algorithm selection maps and fastest recovery time plots for the problem class (Mat, B_ϵ). This supplementary section presents ratios of the recovery time for NIHT, HTP, and CSMPSP to the fastest recovery time.

S5. This supplementary section provides supporting information for Sec. 5.

S5.1. In Sec. 5 data is presented for the sparse vector ensembles U_ϵ and N_ϵ with $\epsilon = 1/10$. Section S5.1 provides additional observations and a similar performance analysis for noiseless setting with sparse vector ensembles U and N.

S5.2. This supplementary section includes the ratios of the recovery times for NIHT, HTP, and CSMPSP versus the fastest recovery time for the problem classes (Mat, vec) with $vec \in \{U_\epsilon, N_\epsilon\}$ which were omitted from Sec. 5.

S6. This supplementary section investigates the role of the number of nonzeros per column in the sparse matrix ensemble $\mathcal{S}_p$. The main article focuses on the sparse matrix ensemble $\mathcal{S}_7$. This section discusses the selection of $p = 7$ including comparisons of recovery phase transition curves for the matrix ensembles $\mathcal{S}_p$ with $p = 4, 7, 13$ and information demonstrating that consistent performance is observed for the problem classes $(\mathcal{S}_4, vec)$ with $vec \in \{B, U, N, B_\epsilon, U_\epsilon, N_\epsilon\}$.

*JDB was supported by NSF DMS 1112612 and NSF OISE 0854991.
†JT was supported by an Nvidia Professor Partnership.

S2. EXPERIMENTAL SET-UP

Justification of Stopping Criteria The stopping criteria described in Sec. 2 were chosen based on extensive testing. Figure S1 depicts the ℓ_∞ error for every problem instance tested on the problem class (Mat, B). This includes the errors for every algorithm (NIHT, HTP, and CSMPSP) and every matrix ensemble ($Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$) for the sparse vector ensemble B, a total of more than 1.5 million tests. This plot of ℓ_∞-error versus iterations indicates that the stopping criteria and the definition of success (4) combine to provide a clear separation of $\|\hat{x} - x\|_\infty$ in problem instances between 10^{-3} and 1. This lack of intermediate values indicated the stopping criteria has not prematurely stopped problem instances where the error has been reduced from that of the initial guess of the zero vector. More precisely, less than 1 in 1,300 problem instances shown in Fig. S1 have $\|\hat{x} - x\|_\infty$ between 10^{-3} and 1. In fact, for this set of more than 1.5 million tests, there are no problem instances with $\|\hat{x} - x\|_\infty$ between $10^{-2.6}$ and $10^{-0.6}$.

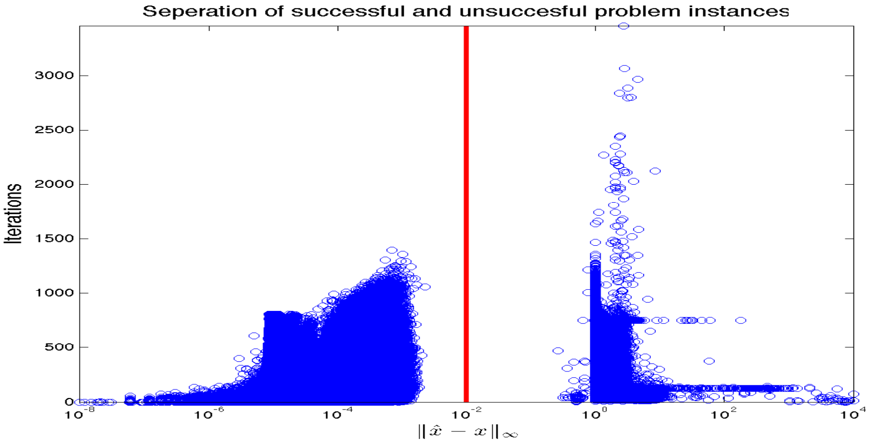


Figure S1. Separation of $\|\hat{x} - x\|_\infty$ error vs iteration of NIHT, HTP, and CSMPSP for all, over 1.5 million, tests of problem classes (Mat, B) for $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$ presented in Sec.3.

Determination of the average case phase transition: logistic regression The experimental setup for the large-scale testing reported in this article is described in Sec. 2. The identification of the 50% recovery phase transitions as described in Sec. 2.1 includes testing more than 15,000 problem instances for each algorithm, each problem class, and each value of n. To present this data as concisely as possible, for each m, n pair, we compute a logistic regression of the data

$$\hat{\pi}(k/m) = \frac{1}{1 + \exp(-b_0(1 - b_1 k/m))} \tag{7}$$

where the fit parameters b_0 and b_1 are calculated to minimize

$$\sum_k \left| \hat{\pi}(k/m) - \frac{\#success(k,m,n)}{\#trials(k,m,n)} \right| \tag{8}$$

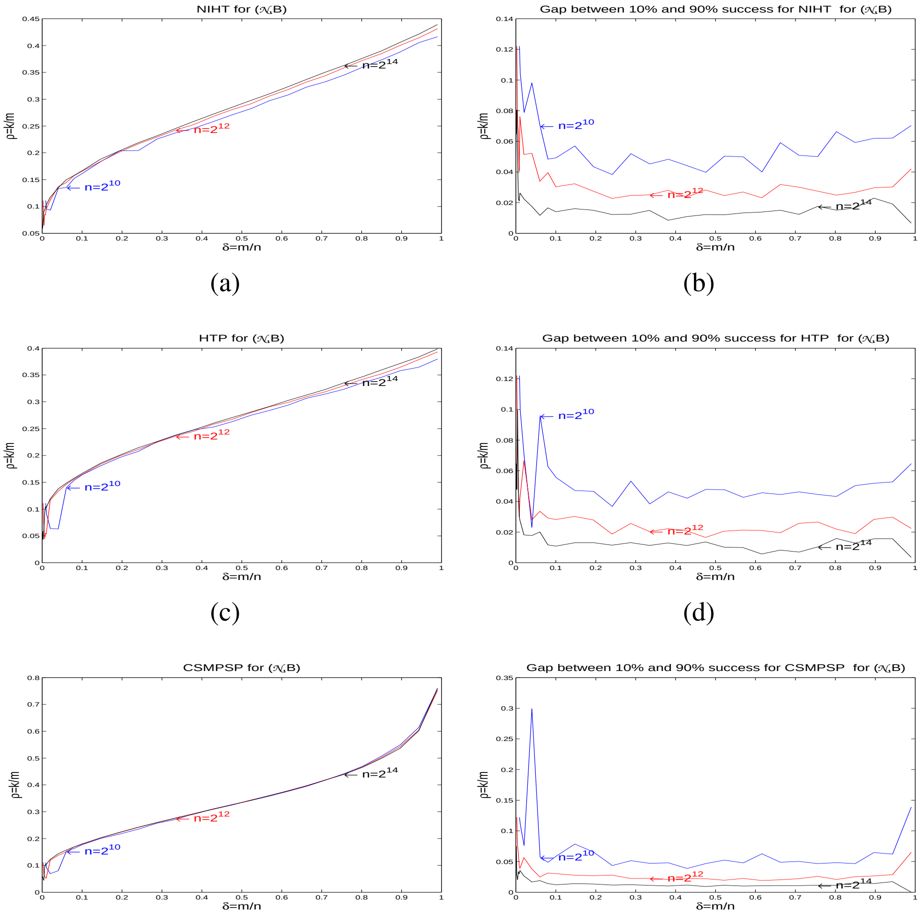
The definition of success given in (4) yields a clear separation of successes and failures. In the computation of the logistic regression, the success criteria is relaxed to $\|\hat{x} - x\|_\infty < 10^{-2}$, where 10^{-2} could be replaced by any value in the region of separation from Fig. S1.

Fig. 1 displays the curve $\rho = \rho^{alg}_{(Mat,B)}(\delta)$ such that $\hat{\pi}(\rho) = 1/2$ for NIHT, HTP, and CSMPSP with $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$ and $n = 2^{20}$ for $Mat = DCT$, $n = 2^{18}$ for $Mat = \mathcal{S}_7$, and $n = 2^{14}$ for $Mat = \mathcal{N}$. Figs. S2-S4 display the logistic regression phase transitions for some other values of n tested as well as the width of the transition regions given by $\rho_{0.1} - \rho_{0.9}$ where $\hat{\pi}(\rho_c) = c$. Unless otherwise stated, all phase transition curves presented are for the 50% recovery level curve defined by $\hat{\pi}(k/m) = 1/2$.

S3. ALGORITHM PERFORMANCE FOR PROBLEM CLASS (Mat, B)

S3.1. Regions of high probability of recovery: (Mat, B)

Concentration of recovery phase transitions for (Mat, B) Sec. 3.1 discusses the recovery phase transitions of the problem classes (Mat, B) for $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$, but only presents data for the largest value of n tested for each problem class. This supplementary section includes further data on the recovery phase transition curves for this problem class at smaller values of n. Along with presenting the 50% recovery phase transitions in the left panels of each figure, the right panels include the gap between the 10% and 90% recovery curves of the logistic regressions. The plots show quantitative convergence of the recovery phase transitions toward those shown in Fig. 1.

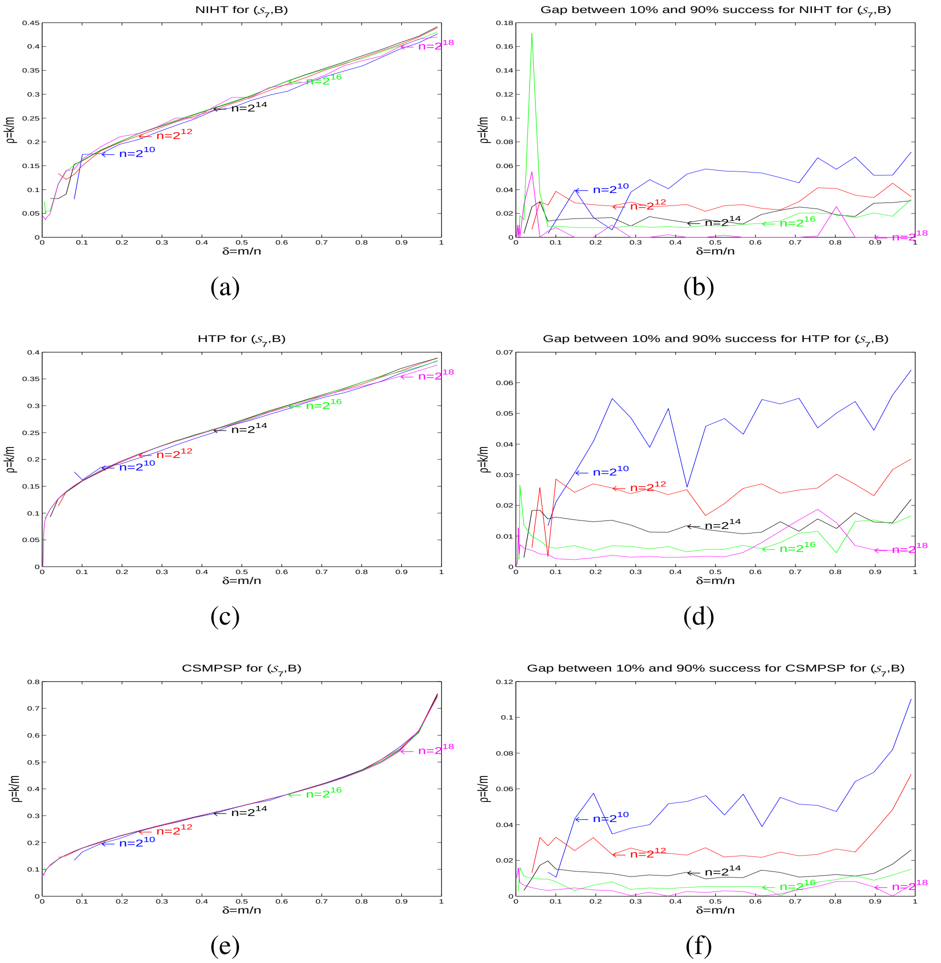


(a) (b)

(c) (d)

(e) (f)

Figure S2. Left panels: 50% recovery probability logistic regression curves for $(\mathcal{N}, B)$ and $n = 2^j$ with $j = 10, 12, 14$. Right panels: gap between 10% and 90% recovery probability curves. Results shown for the algorithms: NIHT (a-b), HTP (c-d), and CSMPSP (e-f).

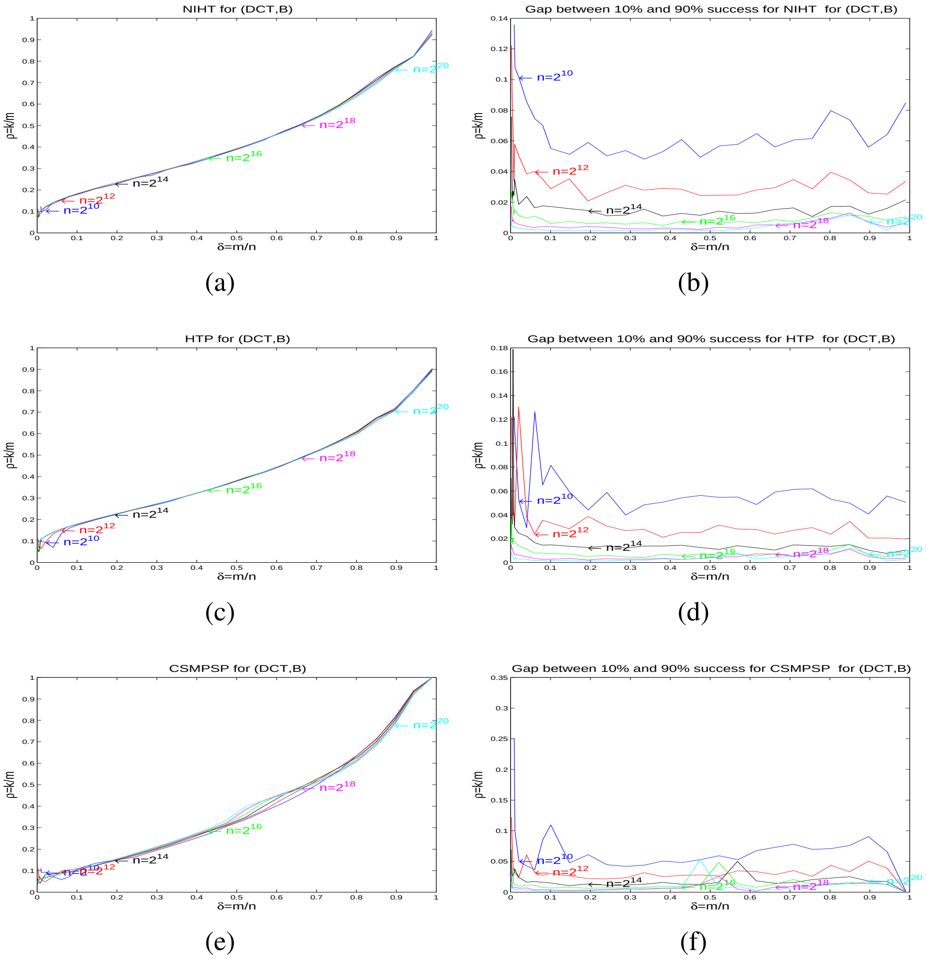


(a) (b)

(c) (d)

(e) (f)

Figure S3. Left panels: 50% recovery probability logistic regression curves for $(\mathcal{S}_7, B)$ and $n = 2^j$ with $j = 10, 12, 14, 16, 18$. Right panels: gap between 10% and 90% recovery probability curves. Results shown for the algorithms: NIHT (a-b), HTP (c-d), and CSMPSP (e-f).



(a) (b)

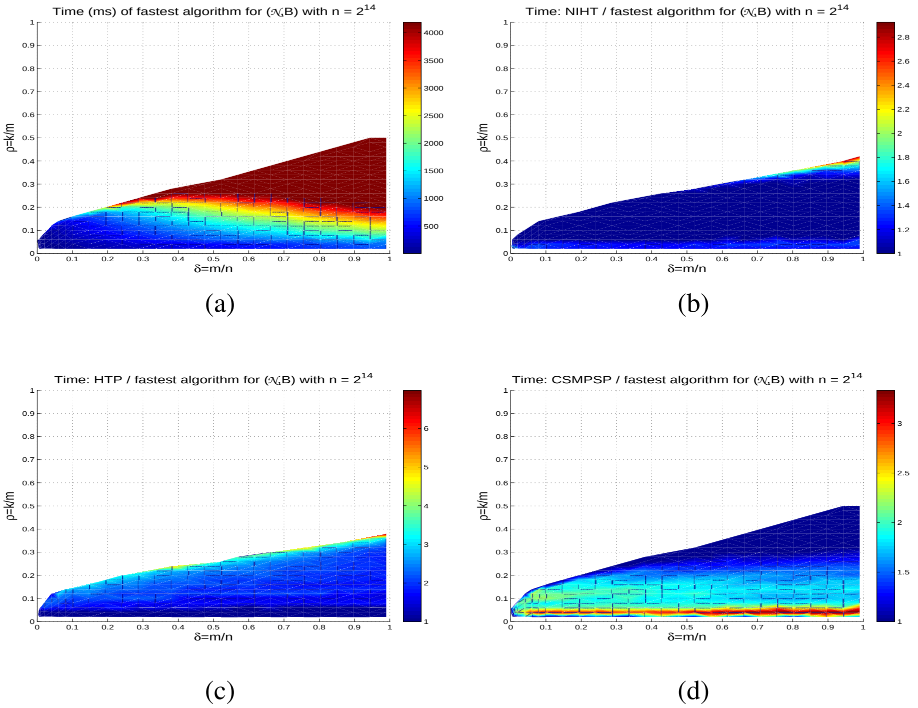
(c) (d)

(e) (f)

Figure S4. Left panels: 50% recovery probability logistic regression curves for (DCT, B) and $n = 2^j$ with $j = 10, 12, 14, 16, 18, 20$. Right panels: gap between 10% and 90% recovery probability curves. Results shown for the algorithms: NIHT (a-b), HTP (c-d), and CSMPSP (e-f).

S3.2. Average Time for recovery and algorithm selection maps: (Mat, B)

S3.2a. Ratios of recovery time to fastest recovery time for (Mat, B) *with large* n This subsection contains the ratios of each algorithms' recovery time to the fastest recovery time of all algorithms. The ratios of NIHT to the fastest recovery time appear in Fig. 2(g-i) as examples. Several Claims and Observations in Sec. 3 state that certain algorithms require less than a specified multiple of the fastest recovery time. Such observations are informed by the following figures.



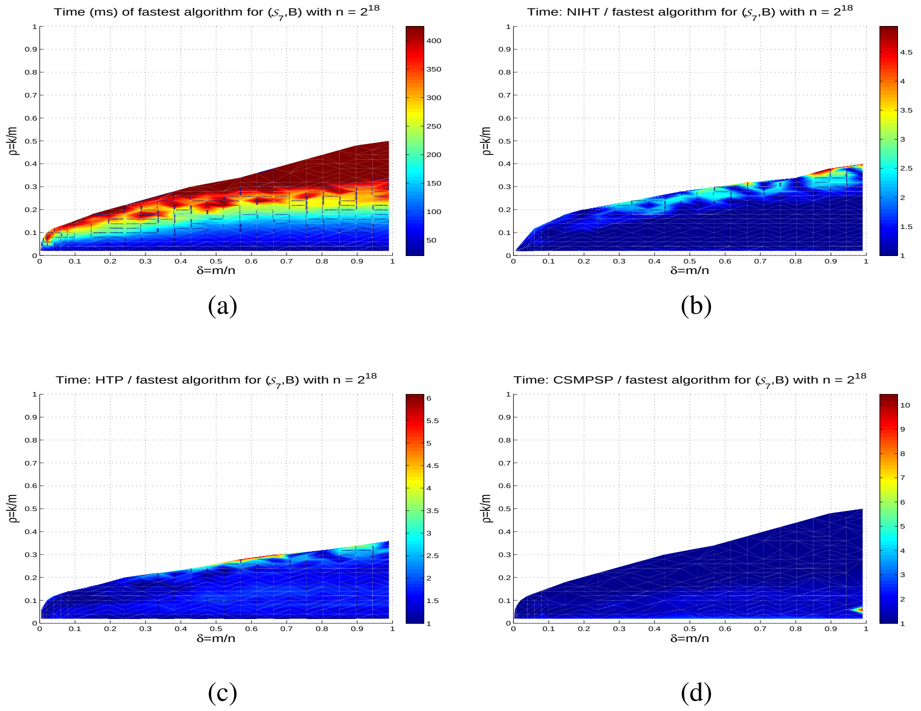
(a)

(b)

(c)

(d)

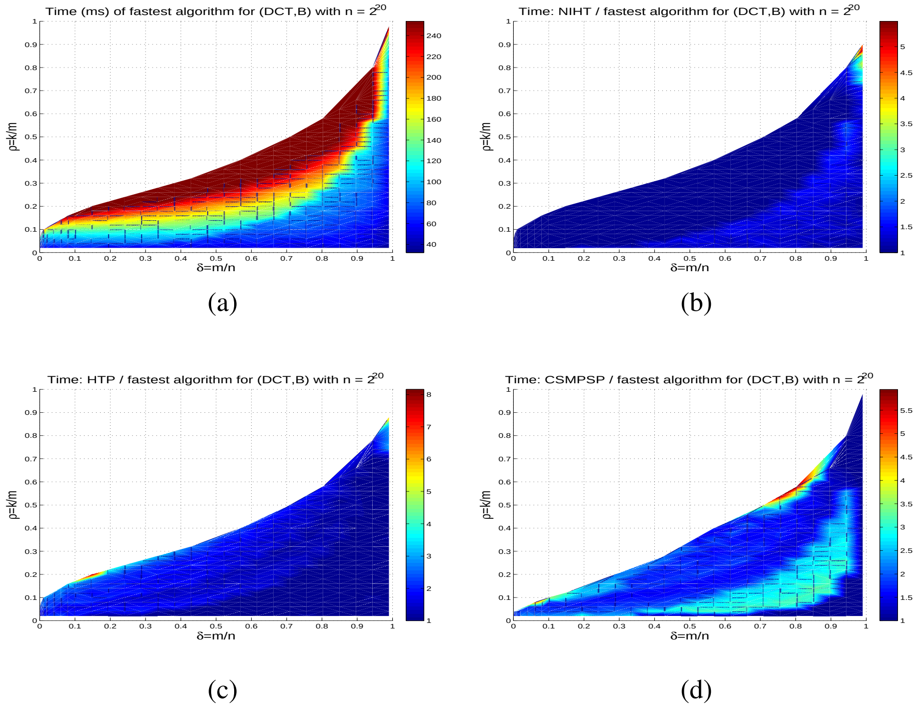
Figure S5. Panel (a), time for fastest algorithm for $(\mathcal{N}, B)$ with $n = 2^{14}$. Ratio of average time for NIHT, HTP, and CSMPSP over the fastest algorithm in (b-d) respectively.



(a) (b)

(c) (d)

Figure S6. Panel (a), time for fastest algorithm for $(\mathcal{S}_7, B)$ with $n = 2^{18}$. Ratio of average time for NIHT, HTP, and CSMPSP over the fastest algorithm in (b-d) respectively.

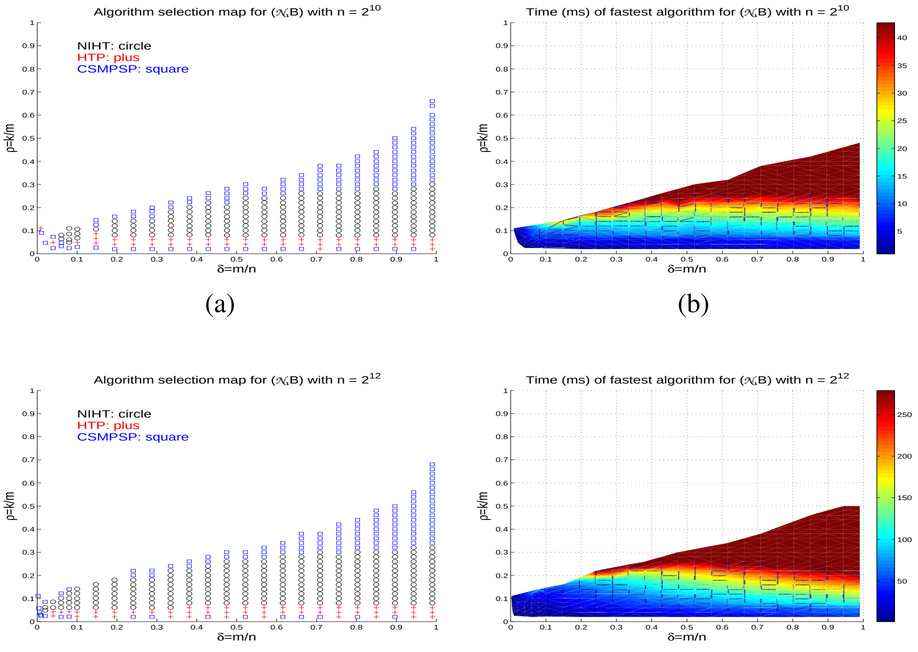


(a) (b)

(c) (d)

Figure S7. Panel (a), time for fastest algorithm for (DCT, B) with $n = 2^{20}$. Ratio of average time for NIHT, HTP, and CSMPSP over the fastest algorithm in (b-d) respectively.

S3.2b. Algorithm selection maps for (Mat, B) *at smaller values of* n Sec. 3.2 presents algorithm selection maps and average recovery times for the problem classes (Mat, B) for $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$, but only presents data for a single value of n per problem class. This supplementary section includes similar algorithm selection maps for smaller values of n, showing that the selection maps are consistent for n sufficiently large.



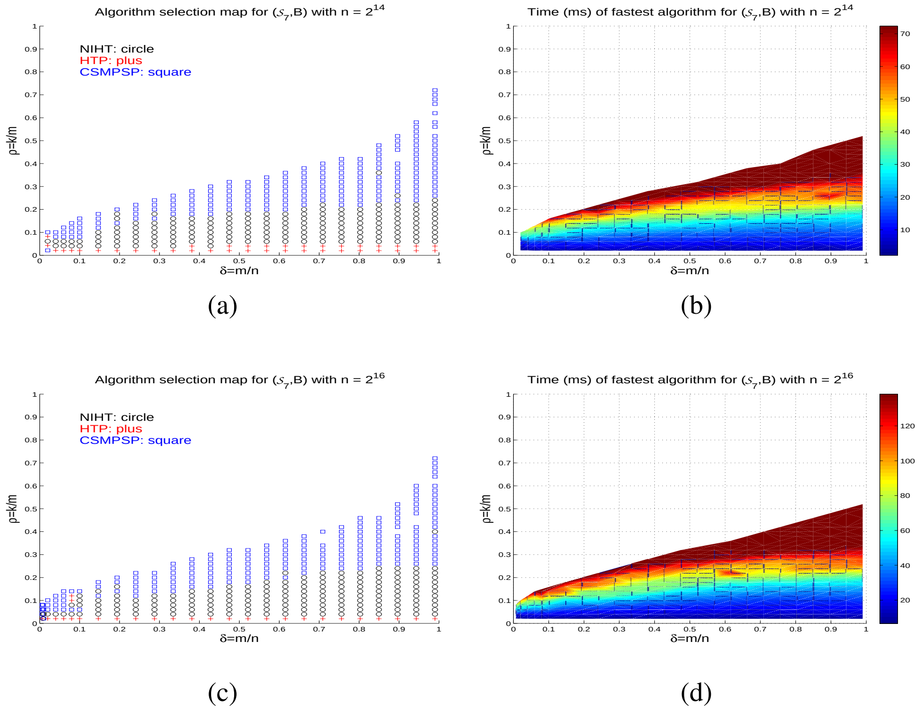
(a)

(b)

(c)

(d)

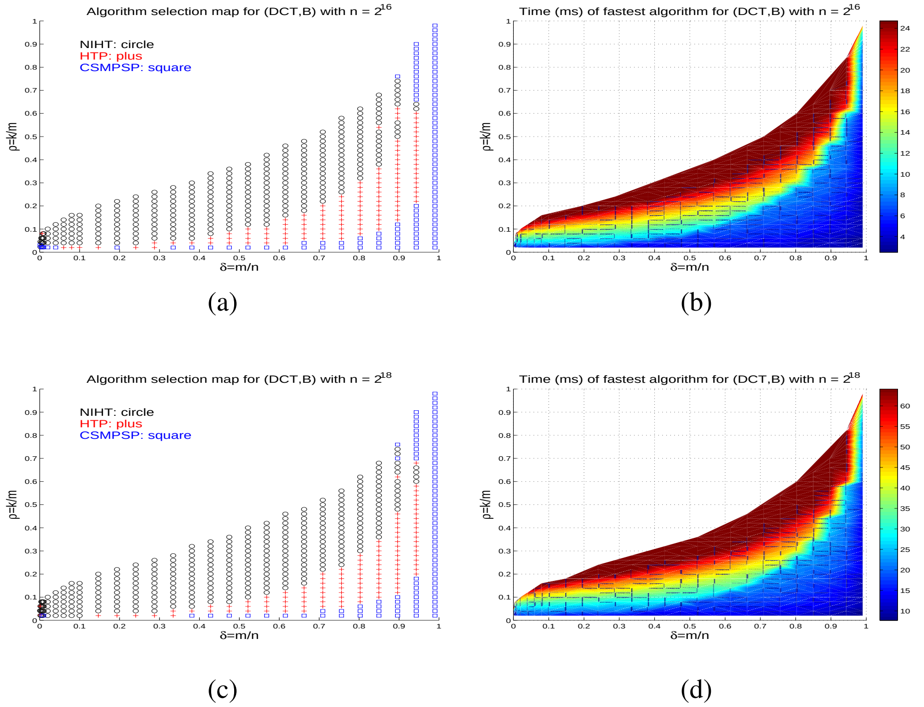
Figure S8. Left panels: Algorithm selection maps for problem class $(\mathcal{N}, B)$. Right panels: Average time for the fastest algorithm. Panels (a-b) with $n = 2^{10}$ and (c-d) with $n = 2^{12}$.



(a) (b)

(c) (d)

Figure S9. Left panels: Algorithm selection maps for problem class $(\mathcal{S}_7, B)$. Right panels: Average time for the fastest algorithm. Panels (a-b) with $n = 2^{14}$ and (c-d) with $n = 2^{16}$.



(a) (b)

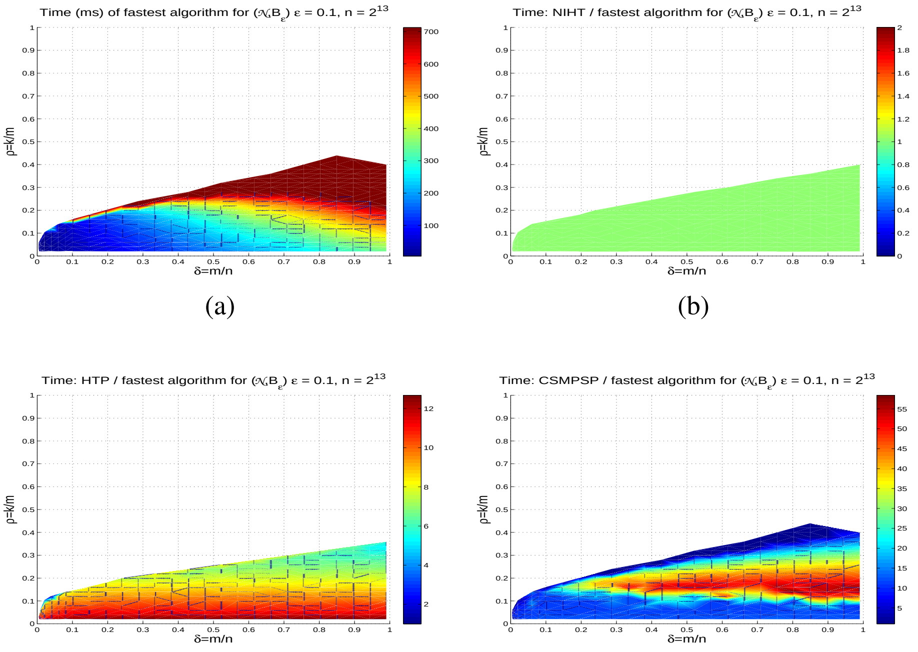
(c) (d)

Figure S10. Left panels: Algorithm selection maps for problem class (DCT, B). Right panels: Average time for the fastest algorithm. Panels (a-b) with $n = 2^{16}$ and (c-d) with $n = 2^{18}$.

S4. ALGORITHM PERFORMANCE FOR PROBLEM CLASS (Mat, B_ϵ)

S4.2. Average recovery time and algorithm selection maps in the presence of noise: (Mat, B_ϵ)

Ratios of recovery time to fastest recovery time in the presence of noise: (Mat, B_ϵ) Section 4.2 presents algorithm selection maps and fastest recovery time plots for the problem class (Mat, B_ϵ). This supplementary section presents ratios of the recovery time for NIHT, HTP, and CSMPSP to the fastest recovery time. The information contained in this section informs Claim 5 and Obs. 4.2.1.



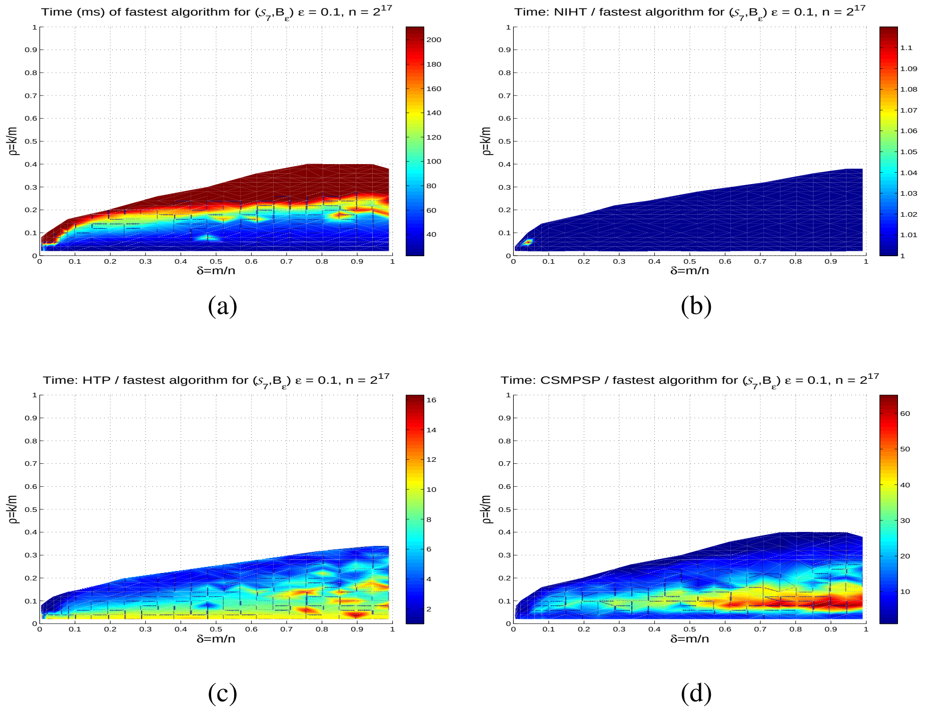
(a)

(b)

(c)

(d)

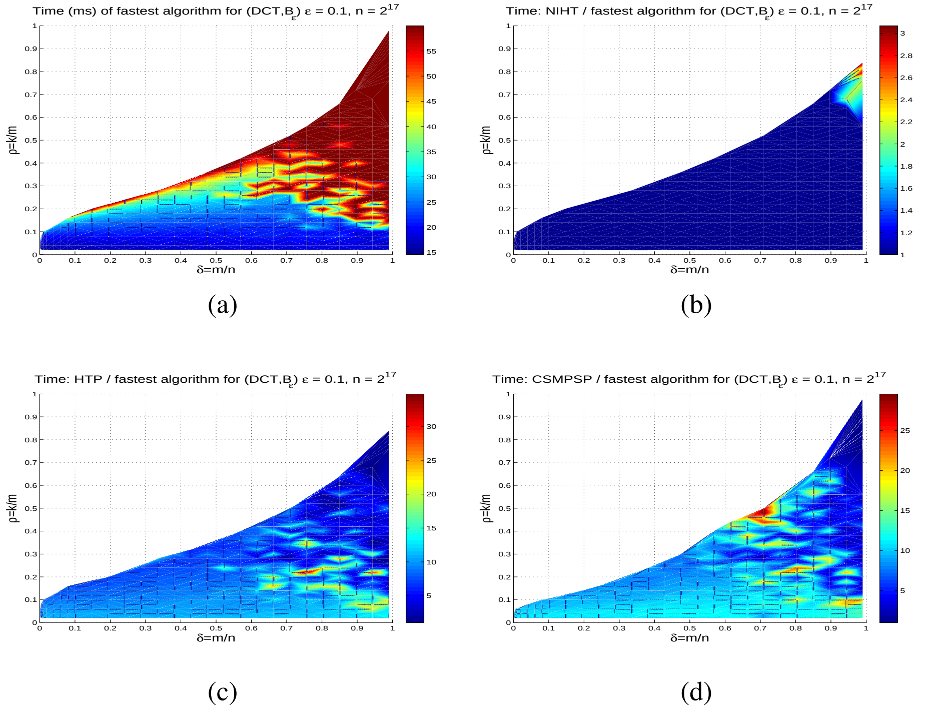
Figure S11. Panel (a), time for fastest algorithm for $(\mathcal{N}, B_\epsilon)$ with $\epsilon = 1/10$ and $n = 2^{13}$. Ratio of average time for NIHT, HTP, and CSMPSP over the fastest algorithm in (b-d) respectively.



(a) (b)

(c) (d)

Figure S12. Panel (a), time for fastest algorithm for $(\mathcal{S}_7, B_\epsilon)$ with $\epsilon = 1/10$ and $n = 2^{17}$. Ratio of average time for NIHT, HTP, and CSMPSP over the fastest algorithm in (b-d) respectively.



(a) (b)

(c) (d)

Figure S13. Panel (a), time for fastest algorithm for (DCT, B_ϵ) with $\epsilon = 1/10$ and $n = 2^{17}$. Ratio of average time for NIHT, HTP, and CSMPSP over the fastest algorithm in (b-d) respectively.

S5. PERFORMANCE ANALYSIS ON OTHER VECTOR DISTRIBUTIONS

S5.1. Recovery Phase Transitions for Alternate Vector Ensembles

Section 5 includes a discussion of the recovery phase transitions for the sparse vector ensembles U_ϵ and N_ϵ. This supplementary section contains the representations of the data used for Obs. S5.1–S5.4 focused on the sparse vector ensembles U and N. For some problem classes, the recovery phase transition curves lack smoothness and for $(\mathcal{N}, N)$ the recovery phase transition curve is not monotonically increasing. Observation S5.1 and Fig. S14 are useful when interpreting the data presented in Fig. S16 and are explicitly referenced in Obs. S5.2 and S5.4.

Observation S5.1
For the problem class (Mat, vec) with $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$ and $vec \in \{U, N\}$, the phase transition region for each algorithm can be much wider than observed for $vec = B$. In some cases, the phase transition region is large enough to cause $\rho^{alg}_{(Mat,vec)}(\delta)$ to lack the expected smoothness and monotonicity. (See Fig. S14.)

As stated in Obs. S5.1, the greedy algorithms' transition regions can be much wider[‡] when recovering vectors from the alternate distributions, even in the noiseless setting. From Sec. S3.1 the transition region widths depicted in Figs. S2–S4 are essentially bounded above by 0.02 for problem classes (Mat, B) with $n \geq 2^{16}$ for $Mat \in \{\mathcal{S}_p, DCT\}$ and $n \geq 2^{12}$ for $Mat = \mathcal{N}$. For comparable problem sizes on the alternate vector ensembles $vec \in \{U, N\}$, the transition regions can be 5 to 20 times as wide. Figure S14 contains three representative recovery phase transitions $\rho^{alg}_{(Mat,vec)}(\delta)$ (a,c,e) together with the associated transition region width (b,d,f). In each of these cases, $\rho^{alg}_{(Mat,vec)}(\delta)$ lacks the now familiar smoothness and monotonicity of recovery phase transitions in sparse approximation.

Similar to Obs. 5.0.1 in Sec. 5, Obs. S5.2 states that, in the noiseless setting, the recovery phase transition for NIHT, HTP, and CSMPSP is lowest for the sparse vector ensemble B. Observations S5.3–S5.4 for vector ensembles U and N parallel the similar observations for the moderate noise setting with vector ensembles U_ϵ and N_ϵ, namely Obs. 5.0.2–5.0.3.

Observation S5.2
For the problem classes (Mat, vec) with $vec \in \{B, U, N\}$, the recovery phase transitions for the sparse vector ensemble B is typically lower than for U and N. Specifically: for all $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$, $\rho^{alg}_{(Mat,U)}(\delta) \approx \rho^{alg}_{(Mat,N)}(\delta) > \rho^{alg}_{(Mat,B)}(\delta)$ for NIHT, HTP, and CSMPSP and with $\delta = m/n \in (0,1)$, except for NIHT on the problem class $(\mathcal{N}, N)$ where $\rho^{niht}_{(Mat,N)}(\delta) < \rho^{niht}_{(Mat,B)}(\delta) < \rho^{niht}_{(Mat,U)}(\delta)$ (Obs. S5.1) for $\delta \in (0.3, 0.8)$. (See Fig. S15.)

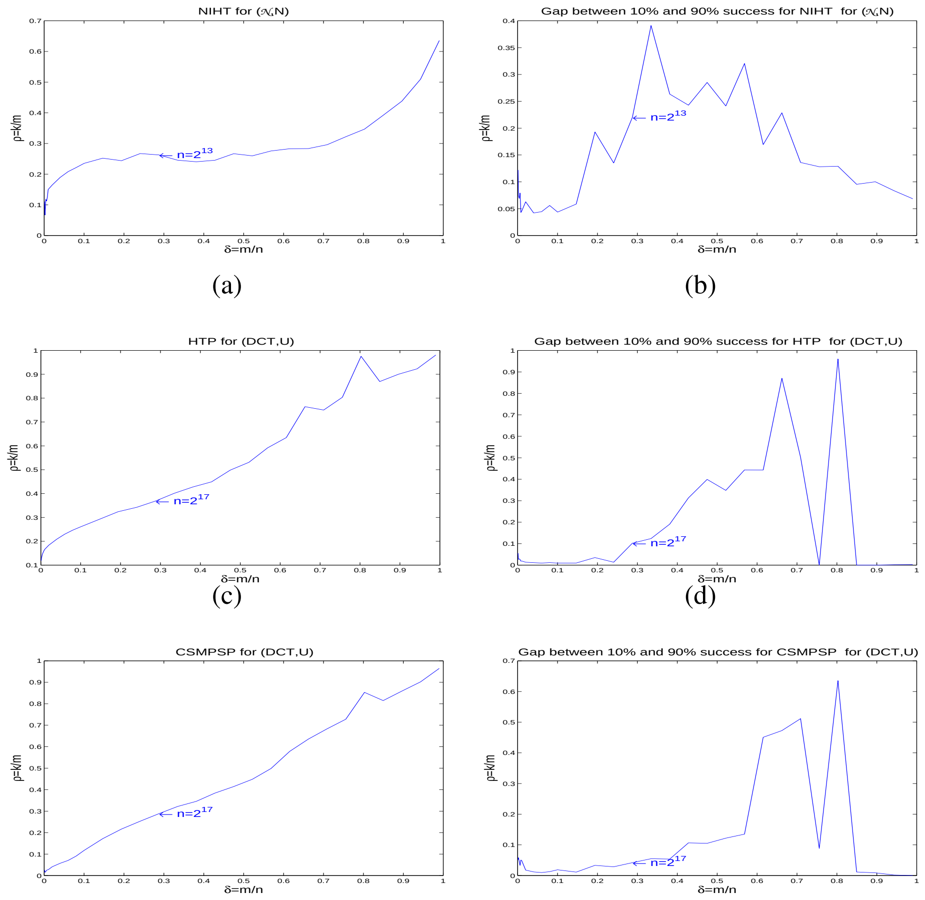
Observation S5.3
For problem classes (Mat, U) with $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$, and $\delta \in [1/20, 1/2]$, the recovery phase transitions for NIHT and HTP have phase transition curves within 0.02 of each other. For (DCT, U) CSMPSP has the lowest PT curve but has the highest phase transition curve for $(\mathcal{N}, U)$ and $(\mathcal{S}_p, U)$. (See Fig. S16(d-f).)

Observation S5.4
For problem classes (Mat, N), the recovery phase transition curves for the algorithms have qualitatively similar behavior to the associated phase transition curves for vector ensembles U as detailed in Obs. S5.3 and 5.0.3. (See Fig. S16.) The notable exceptions are

(i) $\rho^{niht}_{(\mathcal{N},N)}(\delta)$ which suffers from a large phase transition region when both the matrix ensemble and vector ensemble are normally distributed (Obs. S5.1);

(ii) HTP has a noticeably higher phase transition curve than NIHT for problem class $(\mathcal{N}, N)$. (See Fig. S16(a).)

[‡] This phenomenon is likely due to logistic regression and definition of success tailored for the sparse ensemble B. Tuning the logistic regression to a specific problem class may alter the pahse transition curves. This was avoided in the analysis for consistency.

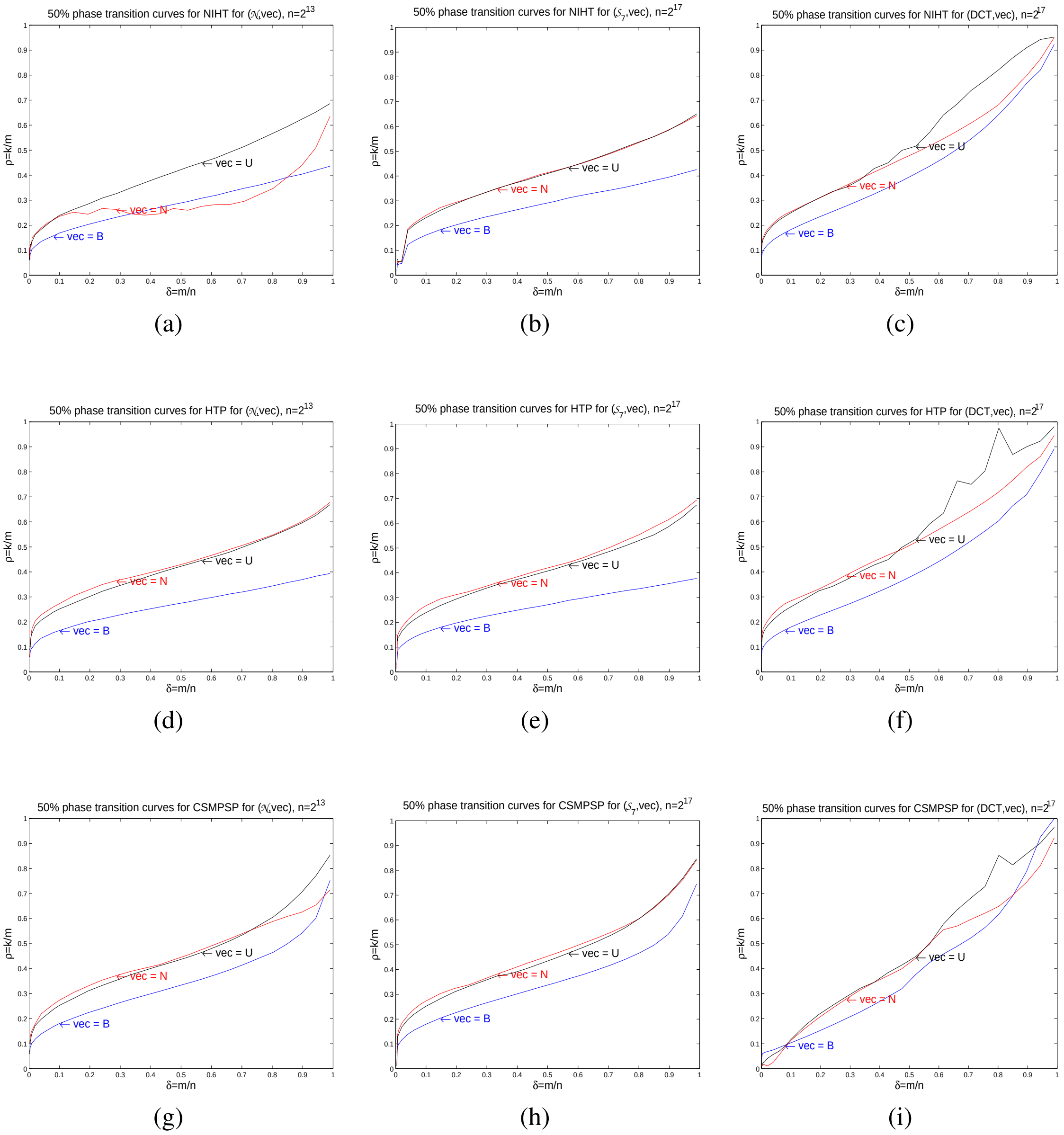


(a) (b)

(c) (d)

(e) (f)

Figure S14. Left panels: 50% recovery probability logistic regression curves. Right panels: gap between 10% and 90% recovery probability curves. (a-b) NIHT with $(\mathcal{N}, U)$ with $n = 2^{13}$; (c-d) HTP for (DCT, U) with $n = 2^{17}$; (e-f) CSMPSP for (DCT, U) with $n = 2^{17}$.



50% phase transition curves for NIHT for ($\mathcal{N}$,vec), n=2^{13}

50% phase transition curves for NIHT for ($\mathcal{S}_7$,vec), n=2^{17}

50% phase transition curves for NIHT for (DCT,vec), n=2^{17}

(a) (b) (c)

50% phase transition curves for HTP for ($\mathcal{N}$,vec), n=2^{13}

50% phase transition curves for HTP for ($\mathcal{S}_7$,vec), n=2^{17}

50% phase transition curves for HTP for (DCT,vec), n=2^{17}

(d) (e) (f)

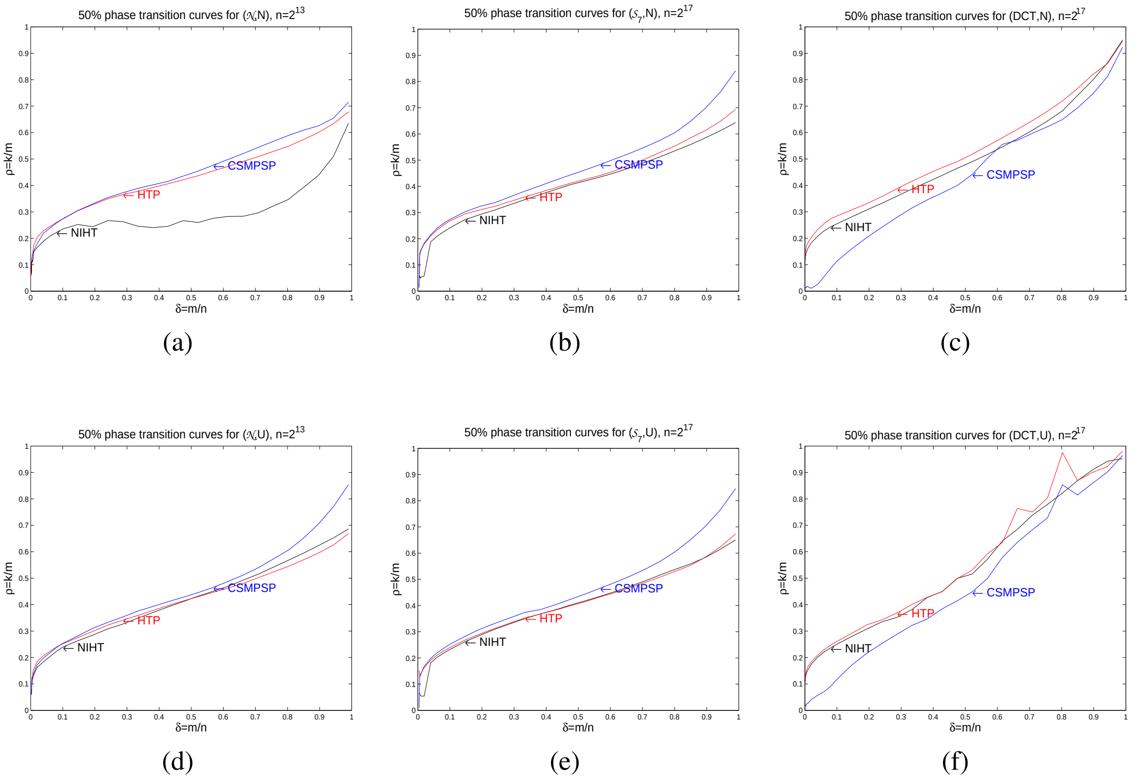
50% phase transition curves for CSMPSP for ($\mathcal{N}$,vec), n=2^{13}

50% phase transition curves for CSMPSP for ($\mathcal{S}_7$,vec), n=2^{17}

50% phase transition curves for CSMPSP for (DCT,vec), n=2^{17}

(g) (h) (i)

Figure S15. 50% recovery probability logistic regression curves for $vec \in \{B, U, N\}$ for (a-c) NIHT, (d-f) HTP, and (g-i) CSMPSP. Left panels: $Mat = \mathcal{N}$ with $n = 2^{13}$; Center panels: $Mat = \mathcal{S}_7$ with $n = 2^{17}$; Right panels: $Mat = DCT$ with $n = 2^{17}$.



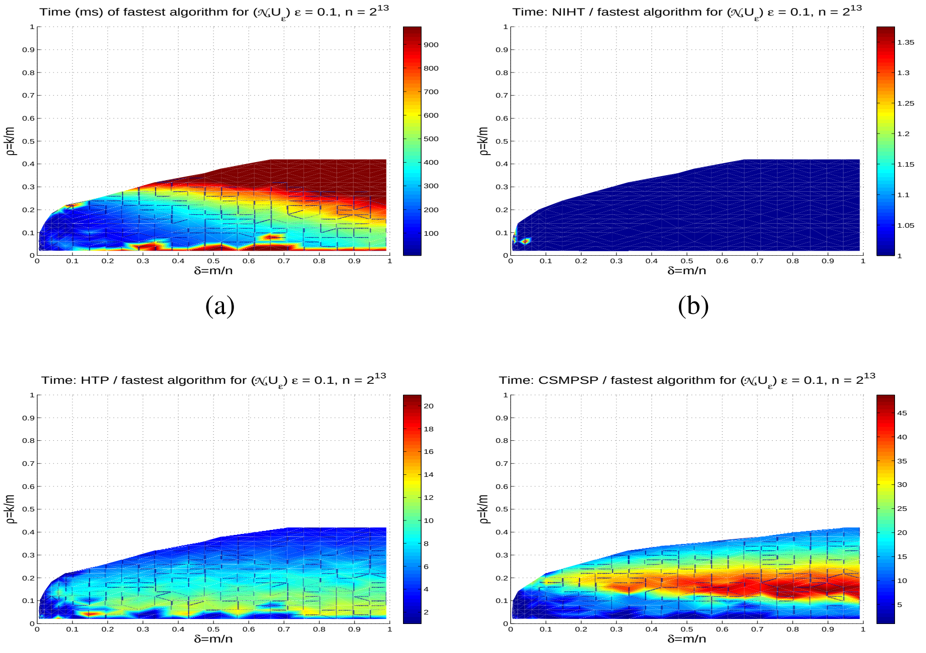
(a) (b) (c)

(d) (e) (f)

Figure S16. 50% recovery probability logistic regression curves for problem classes (Mat, vec) with $vec \in \{N, U\}$ and for algorithms NIHT, HTP, and CSMPSP. (a-c) $vec = N$, (d-f) $vec = U$. Left panels: $Mat = \mathcal{N}$ with $n = 2^{13}$; Center panels: $Mat = \mathcal{S}_7$ with $n = 2^{17}$; Right panels: $Mat = DCT$ with $n = 2^{17}$.

S5.2. Algorithm Selection Maps for Alternate Vector Ensembles

In Sec. 5, algorithm selection maps are presented for (Mat, vec) with $vec \in \{U_\epsilon, N_\epsilon\}$. This supplementary section contains the representations of data informing Obs. 5.0.4–5.0.6. Figures S17–S22 present the ratio of the recovery time for each algorithm to the fastest recovery time in the union of the recovery regions for problem classes (Mat, U_ϵ) and (Mat, N_ϵ). Note the ratio of recovery time for NIHT to the fastest algorithm is consistently small and almost always one for the matrix ensembles $\mathcal{N}$ and $\mathcal{S}_7$.



Time (ms) of fastest algorithm for $(\mathcal{N},U_\epsilon)$ ε = 0.1, n = 2^{13}

(a)

Time: NIHT / fastest algorithm for $(\mathcal{N},U_\epsilon)$ ε = 0.1, n = 2^{13}

(b)

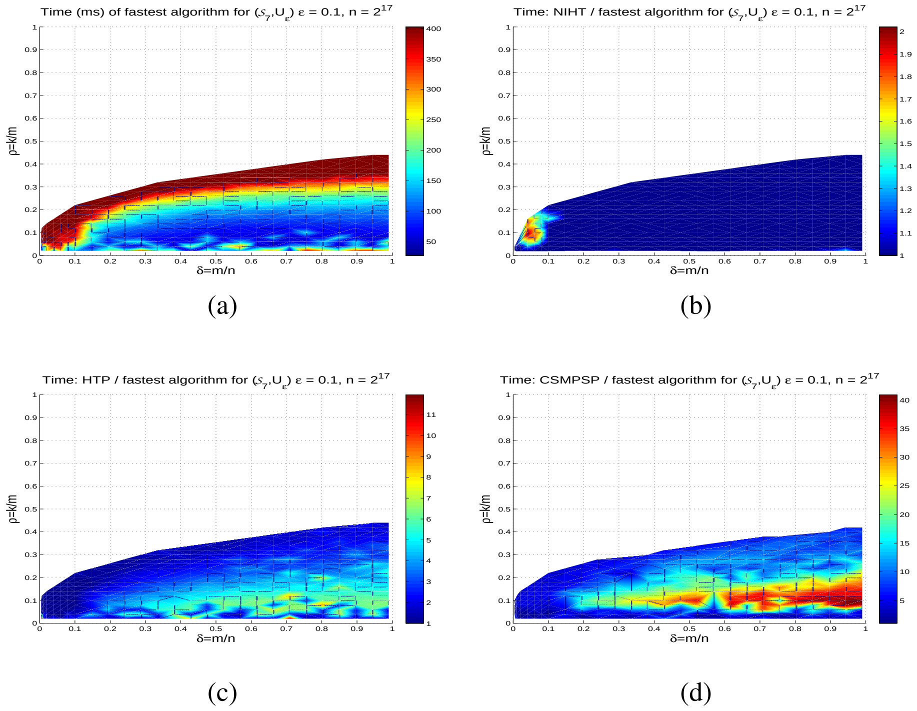
Time: HTP / fastest algorithm for $(\mathcal{N},U_\epsilon)$ ε = 0.1, n = 2^{13}

(c)

Time: CSMPSP / fastest algorithm for $(\mathcal{N},U_\epsilon)$ ε = 0.1, n = 2^{13}

(d)

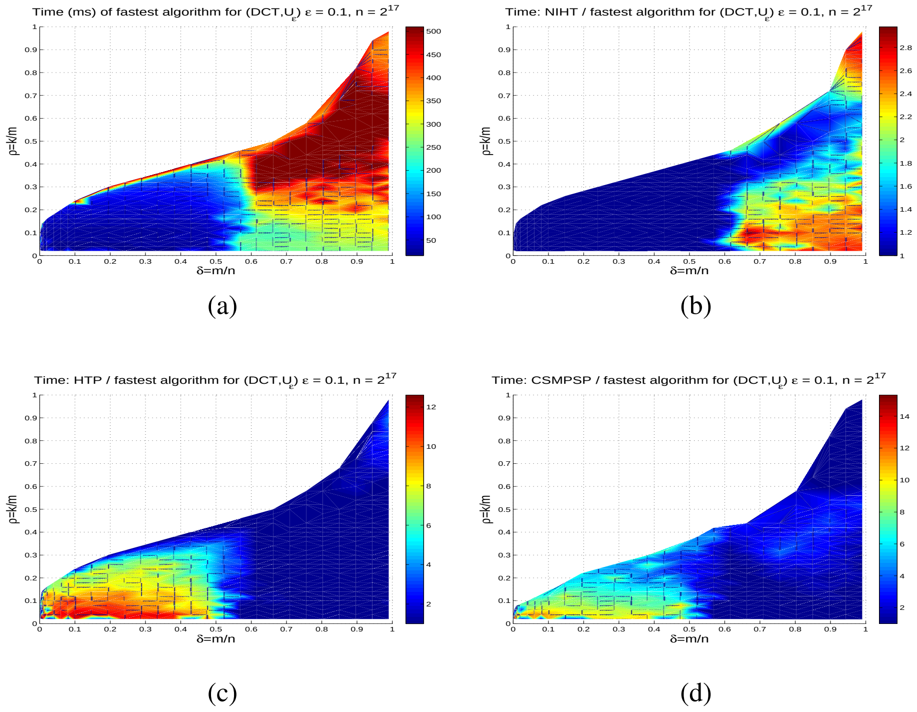
Figure S17. Panel (a), time for fastest algorithm for $(\mathcal{N}, U_\epsilon)$ with $\epsilon = 1/10$ and $n = 2^{13}$. Ratio of average time for NIHT, HTP, and CSMPSP over the fastest algorithm in (b-d) respectively.



(a) (b)

(c) (d)

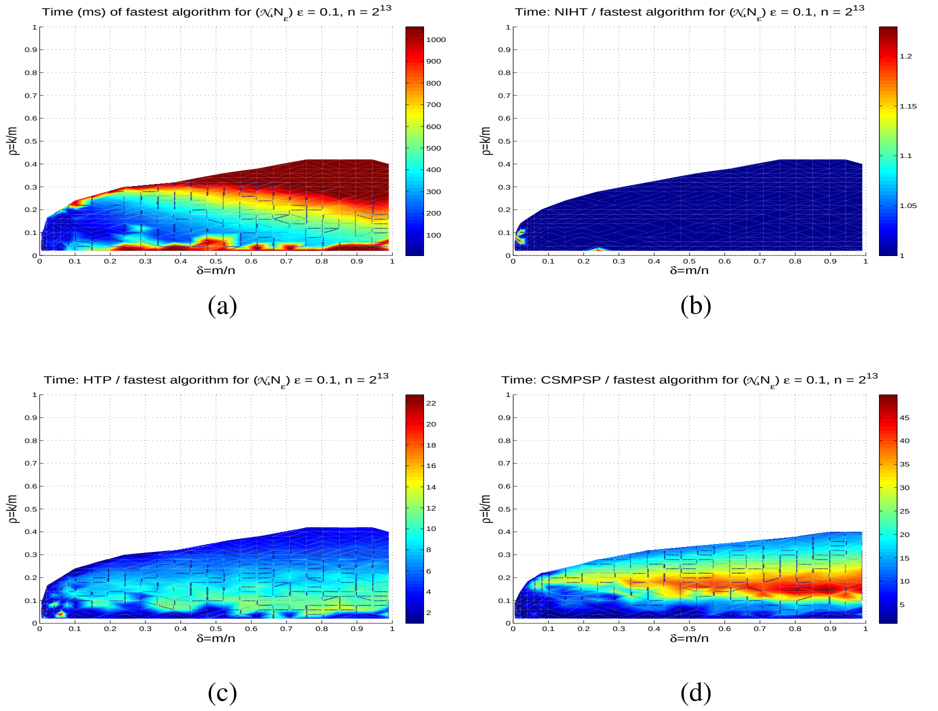
Figure S18. Panel (a), time for fastest algorithm for $(\mathcal{S}_7, U_\epsilon)$ with $\epsilon = 1/10$ and $n = 2^{17}$. Ratio of average time for NIHT, HTP, and CSMPSP over the fastest algorithm in (b-d) respectively.



(a) (b)

(c) (d)

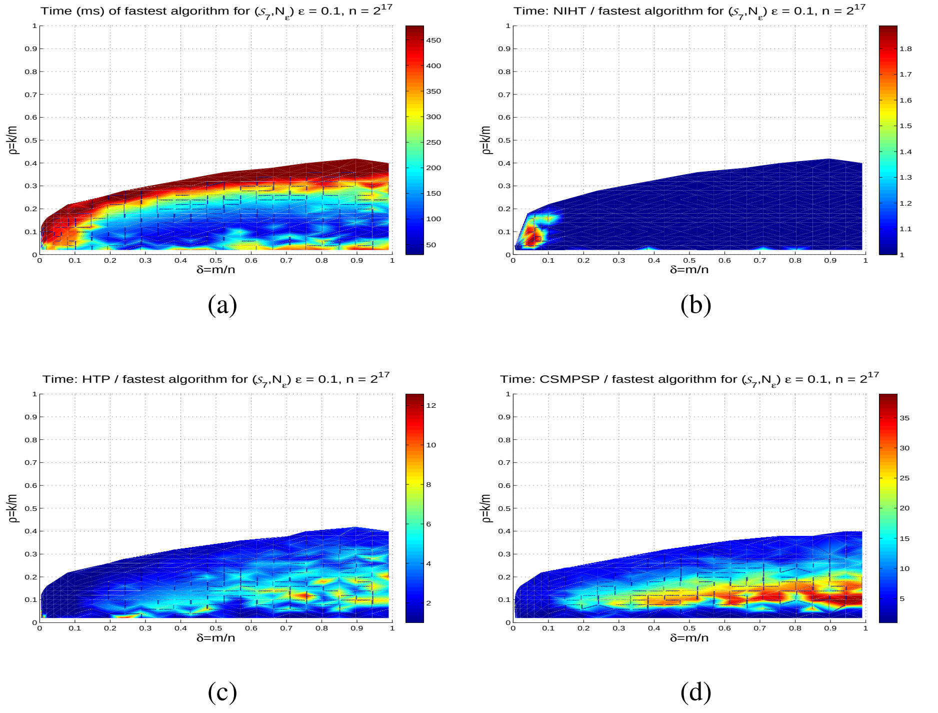
Figure S19. Panel (a), time for fastest algorithm for (DCT, U_ϵ) with $\epsilon = 1/10$ and $n = 2^{17}$. Ratio of average time for NIHT, HTP, and CSMPSP over the fastest algorithm in (b-d) respectively.



(a) (b)

(c) (d)

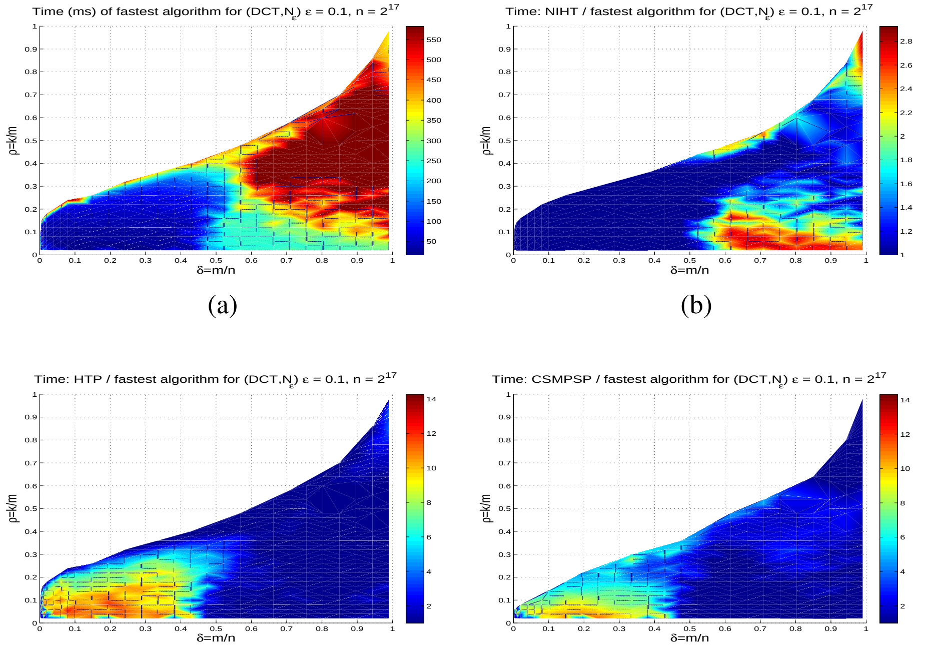
Figure S20. Panel (a), time for fastest algorithm for $(\mathcal{N}, N_\epsilon)$ with $\epsilon = 1/10$ and $n = 2^{13}$. Ratio of average time for NIHT, HTP, and CSMPSP over the fastest algorithm in (b-d) respectively.



(a) (b)

(c) (d)

Figure S21. Panel (a), time for fastest algorithm for $(\mathcal{S}_7, N_\epsilon)$ with $\epsilon = 1/10$ and $n = 2^{17}$. Ratio of average time for NIHT, HTP, and CSMPSP over the fastest algorithm in (b-d) respectively.



(a) (b)

(c) (d)

Figure S22. Panel (a), time for fastest algorithm for (DCT, N_ϵ) with $\epsilon = 1/10$ and $n = 2^{17}$. Ratio of average time for NIHT, HTP, and CSMPSP over the fastest algorithm in (b-d) respectively.

S6. ROLE OF NONZEROS PER COLUMN ON PROBLEM CLASSES $(\mathcal{S}_p, vec)$

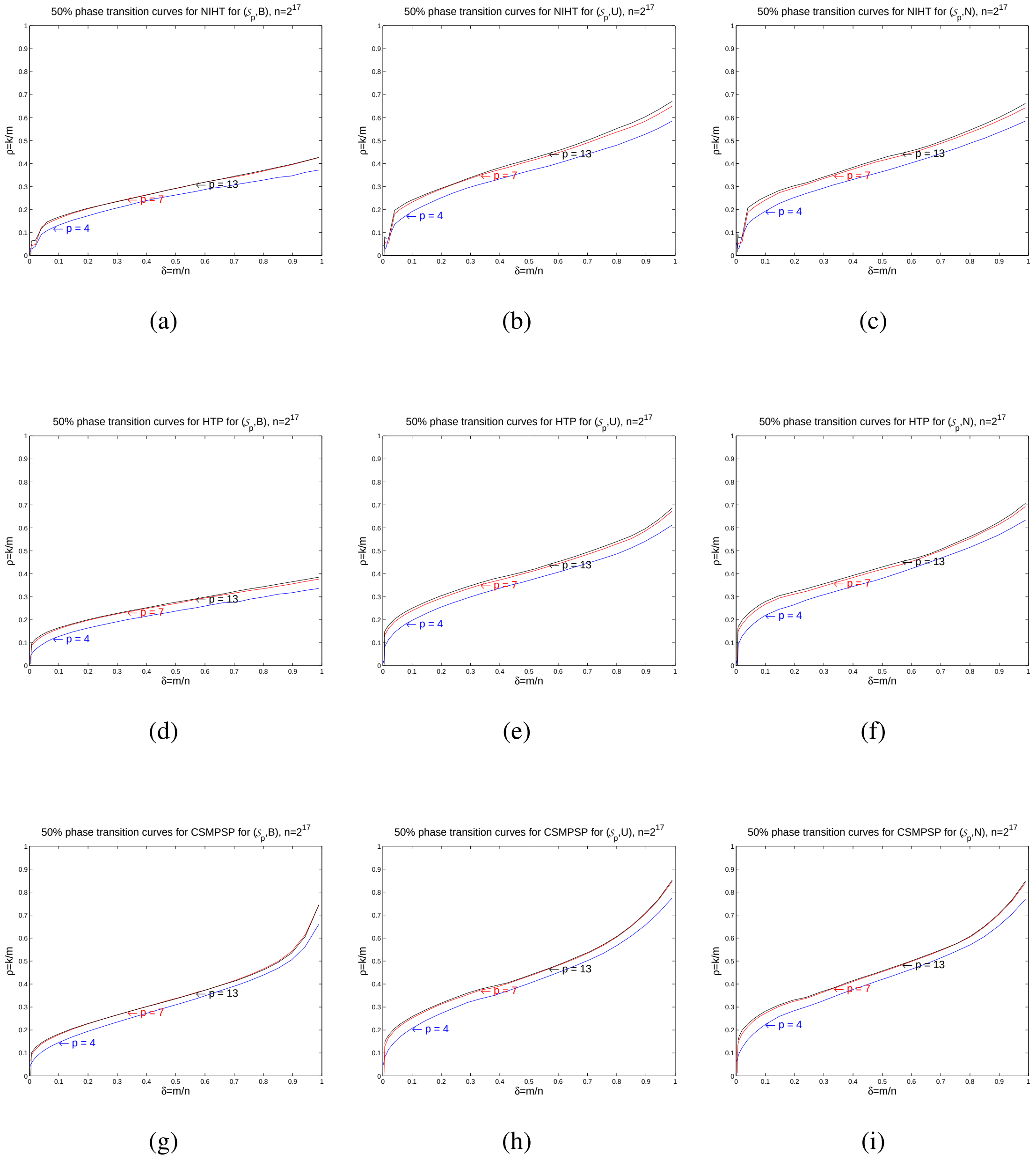
This supplementary section briefly investigates the role of the number of nonzeros per column in the sparse matrix ensemble $\mathcal{S}_p$. For computational reasons, using very few nonzeros per column provides a relatively fast matrix multiplication. Sections 3 and 4 focus on $p = 7$ due to diminishing gains in the phase transition for larger values of p and its computational efficiency. Increasing the number of nonzeros from $p = 7$ to $p = 13$ requires roughly 1.85 times as many computations, yet there is only a minor increase in the recovery phase transition. Figures S23 and S24 present the recovery phase transitions of a fixed algorithm for the problem classes $(\mathcal{S}_p, vec)$, $p = 4, 7, 13$, displaying the three recovery phase transition curves together on a single plot for a fixed vector ensemble $vec \in \{B, U, N, B_\epsilon, U_\epsilon, N_\epsilon\}$. This data supports[§] our focus on $\mathcal{S}_7$, though similar behaviour would be observed for values of p near 7.

Observation S6.1
For problem classes $(\mathcal{S}_p, vec)$ with $vec \in \{B, U, N, B_\epsilon, U_\epsilon, N_\epsilon\}$ increasing the number of nonzeros per column from $p = 4$ to $p = 7$ increases the area under the recovery phase transition curve for NIHT, HTP, and CSMPSP from between 8% and 17%; in contrast, further increasing the number of nonzeros per column from $p = 7$ to $p = 13$ never increases the area under the recovery phase transition by more than 3%. (See Figs. S23 and S24.)

The following demonstrates that very few nonzeros per column ($p = 4$) can still lead to a reasonably sized recovery region. Figures S23–S28 provide plots for recovery phase transitions, concentration of the phase transition, algorithm selection maps, fastest recovery times, and algorithm performance in the presence of noise for the problem class $(\mathcal{S}_4, vec)$ for $vec \in \{B, U, N, B_\epsilon, U_\epsilon, N_\epsilon\}$. While the recovery phase transition curves for matrix ensemble $\mathcal{S}_p$ decrease for sparse vector ensembles as the number of nonzeros per column p decreases, the overall behavior of the algorithms is consistent. This information, summarized in Obs. S6.2, informs practitioners that the measurement and recovery process can be accelerated by using fewer nonzeros provided the problem instance falls within the recovery region for the lower value of p.

Observation S6.2
For all problem classes $(\mathcal{S}_p, vec)$ with $vec \in \{B, U, N, B_\epsilon, U_\epsilon, N_\epsilon\}$ and algorithms NIHT, HTP, and CSMPSP, the following remain consistent for $p = 4$ and $p = 7$: relationships between recovery phase transitions, algorithm selection maps, and concentration of the recovery phase transition. In the presence of moderate noise, the matrix ensemble demonstrates stability for $\delta = m/n \to 0$, but is more susceptible to noise than for larger numbers of nonzeros per column as $\delta \to 1$. (See Figs. S25–S28.)

[§] Other values of p were also tested. The computational time increases smoothly with p along with slight gains in algorithm recovery regions.



(a)

(b)

(c)

(d)

(e)

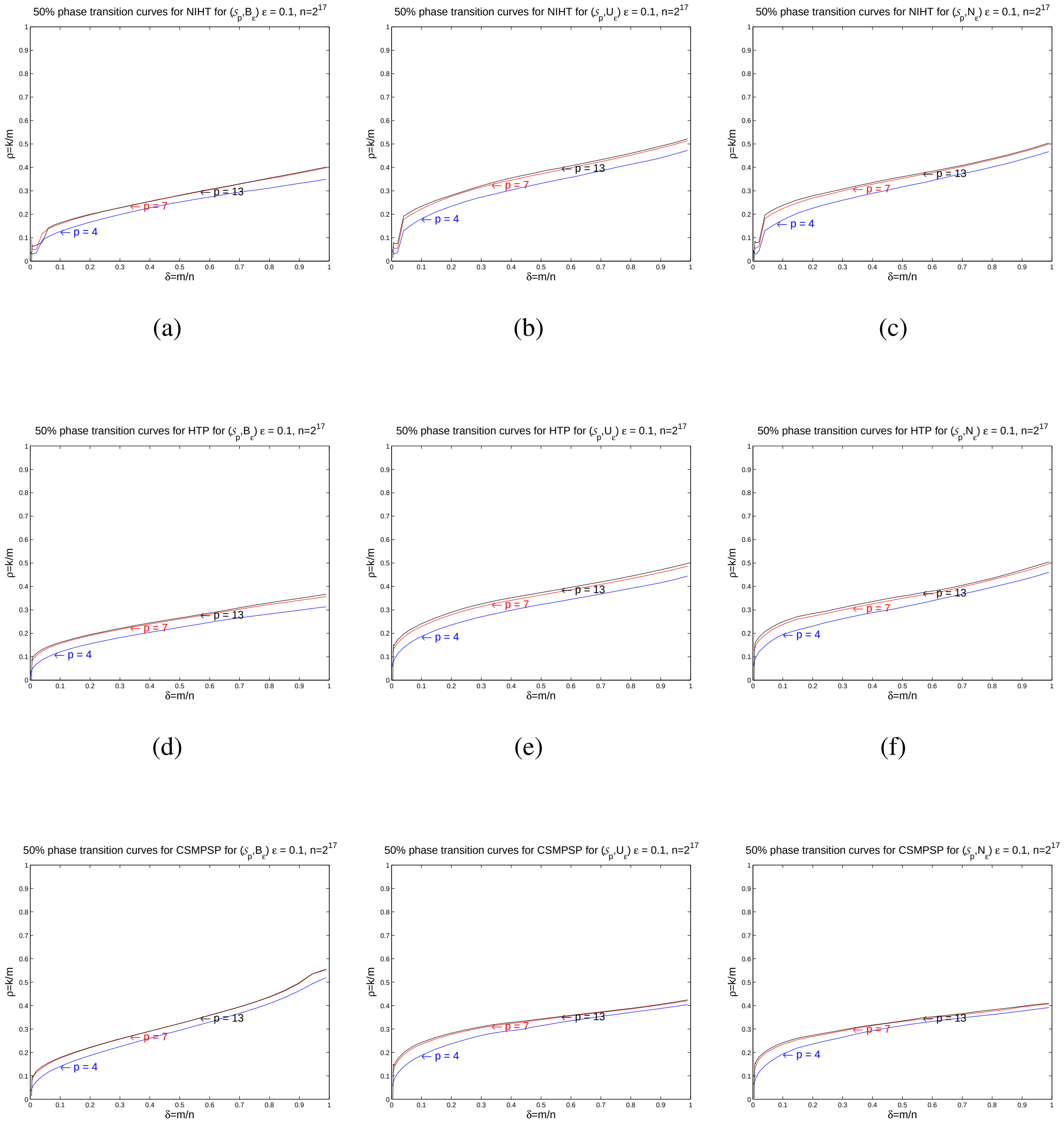
(f)

(g)

(h)

(i)

Figure S23. 50% recovery probability logistic regression curves for $Mat = \mathcal{S}_p$ with $n = 2^{17}$ and for $p = 4$, 7, and 13 in each plot. Left, center, and right panels are $vec = B$, $vec = U$, and $vec = N$ respectively. Algorithms: NIHT (a-c), HTP (d-f), and CSMPSP (g-i).

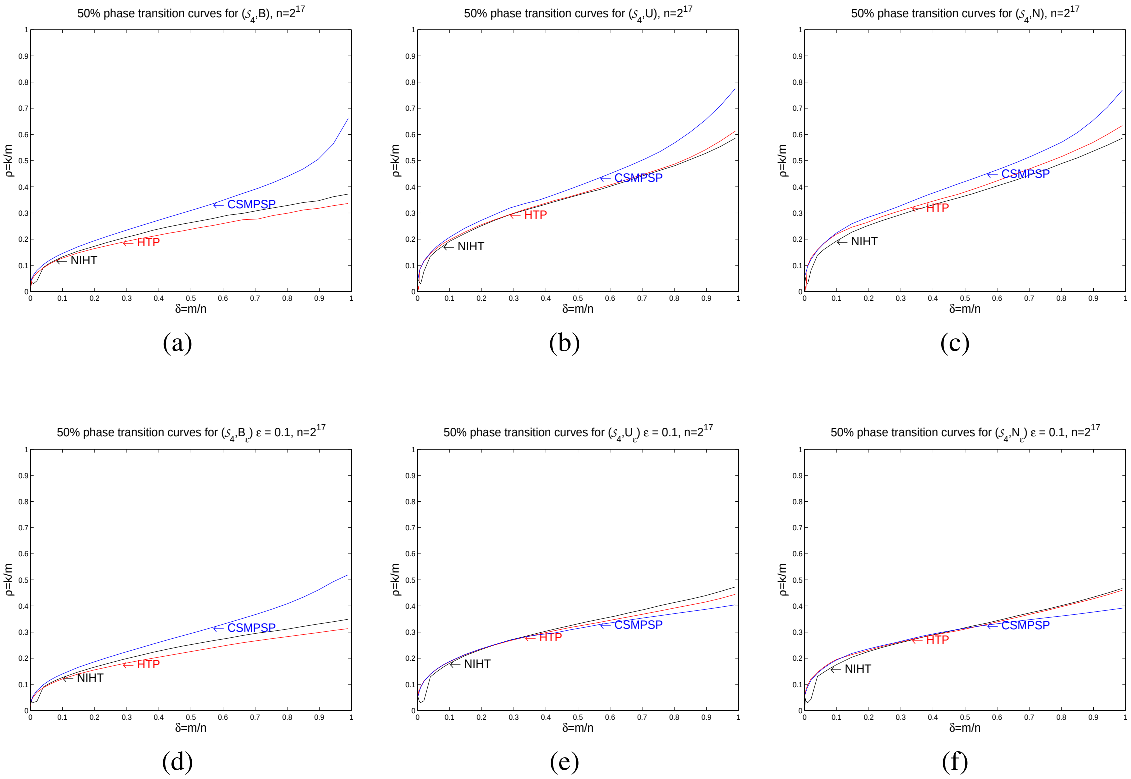


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

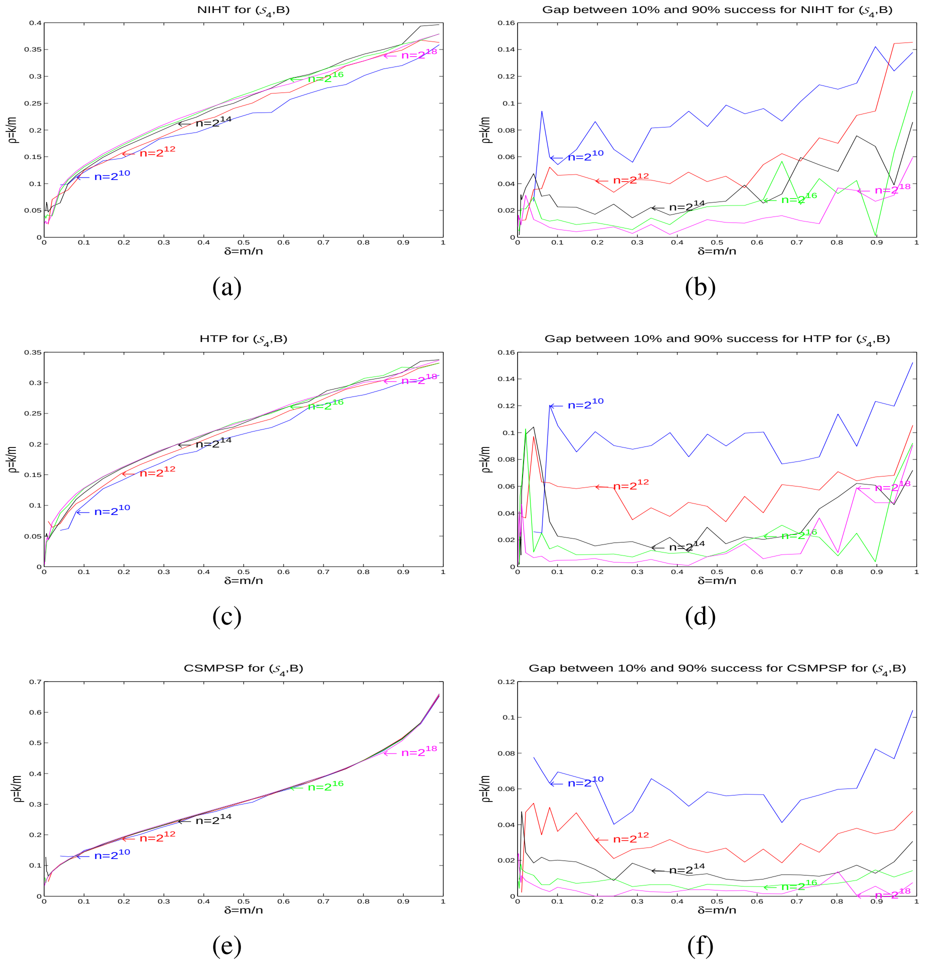
Figure S24. 50% recovery probability logistic regression curves for $Mat = \mathcal{S}_p$ with $n = 2^{17}$ and for $p = 4$, 7, and 13 in each plot. Left, center, and right panels are $vec = B_\epsilon$, $vec = U_\epsilon$, and $vec = N_\epsilon$ respectively with $\epsilon = 1/10$. Algorithms: NIHT (a-c), HTP (d-f), and CSMPSP (g-i).



(a) (b) (c)

(d) (e) (f)

Figure S25. 50% recovery phase transition curves of NIHT, HTP, and CSMPSP for $n = 2^{17}$ and problem class $(\mathcal{S}_4, vec)$ with (a-c) $vec \in \{B, U, N\}$ and (d-f) $vec \in \{B_\epsilon, U_\epsilon, N_\epsilon\}$ with $\epsilon = 1/10$. Left panels: sparse binary vectors; Center panels: sparse uniform vectors; Right panels: sparse normal vectors.

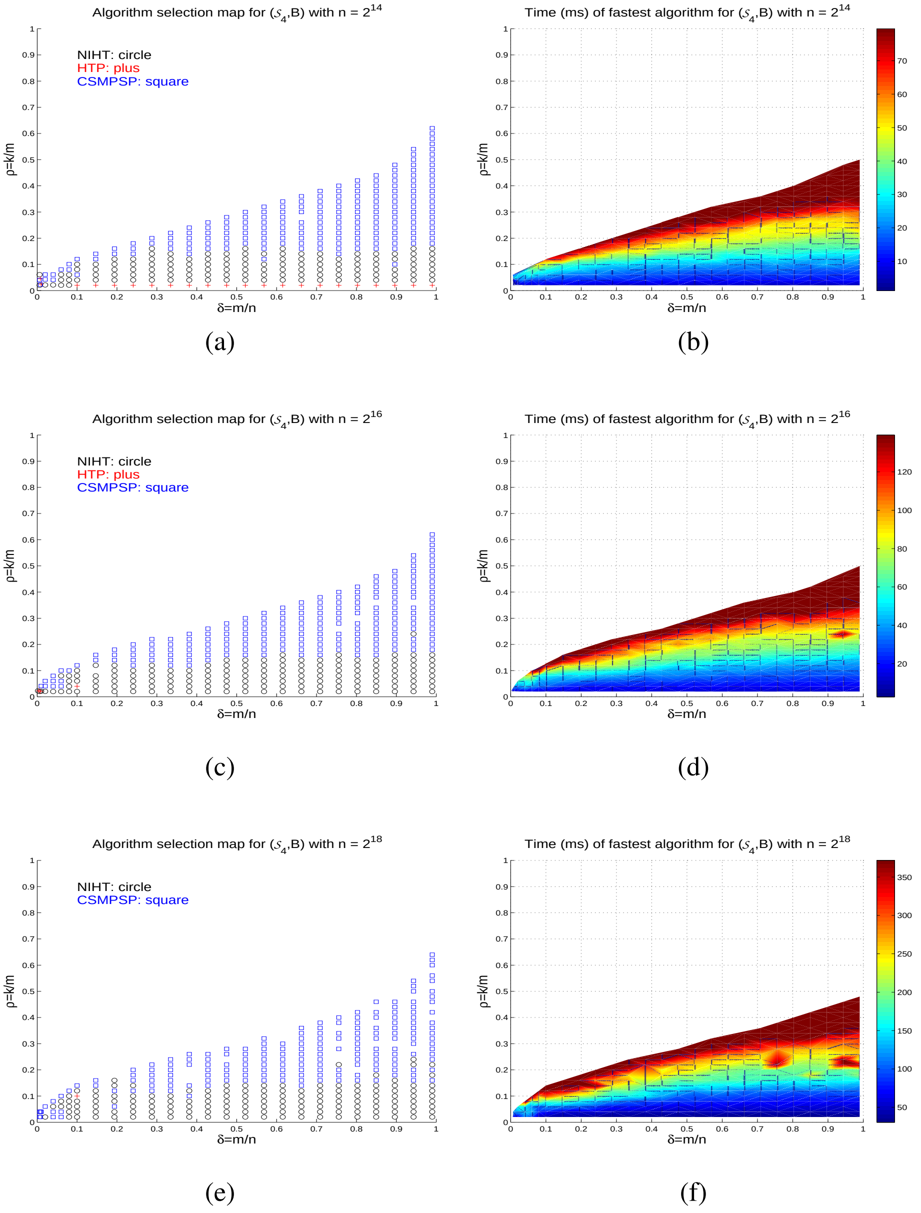


(a) (b)

(c) (d)

(e) (f)

Figure S26. Left panels: 50% recovery probability logistic regression curves for $(\mathcal{S}_4, B)$ and $n = 2^j$ with $j = 10, 12, 14, 16, 18$. Right panels: gap between 10% and 90% recovery probability curves. Results shown for the algorithms: NIHT (a-b), HTP (c-d), and CSMPSP (e-f).

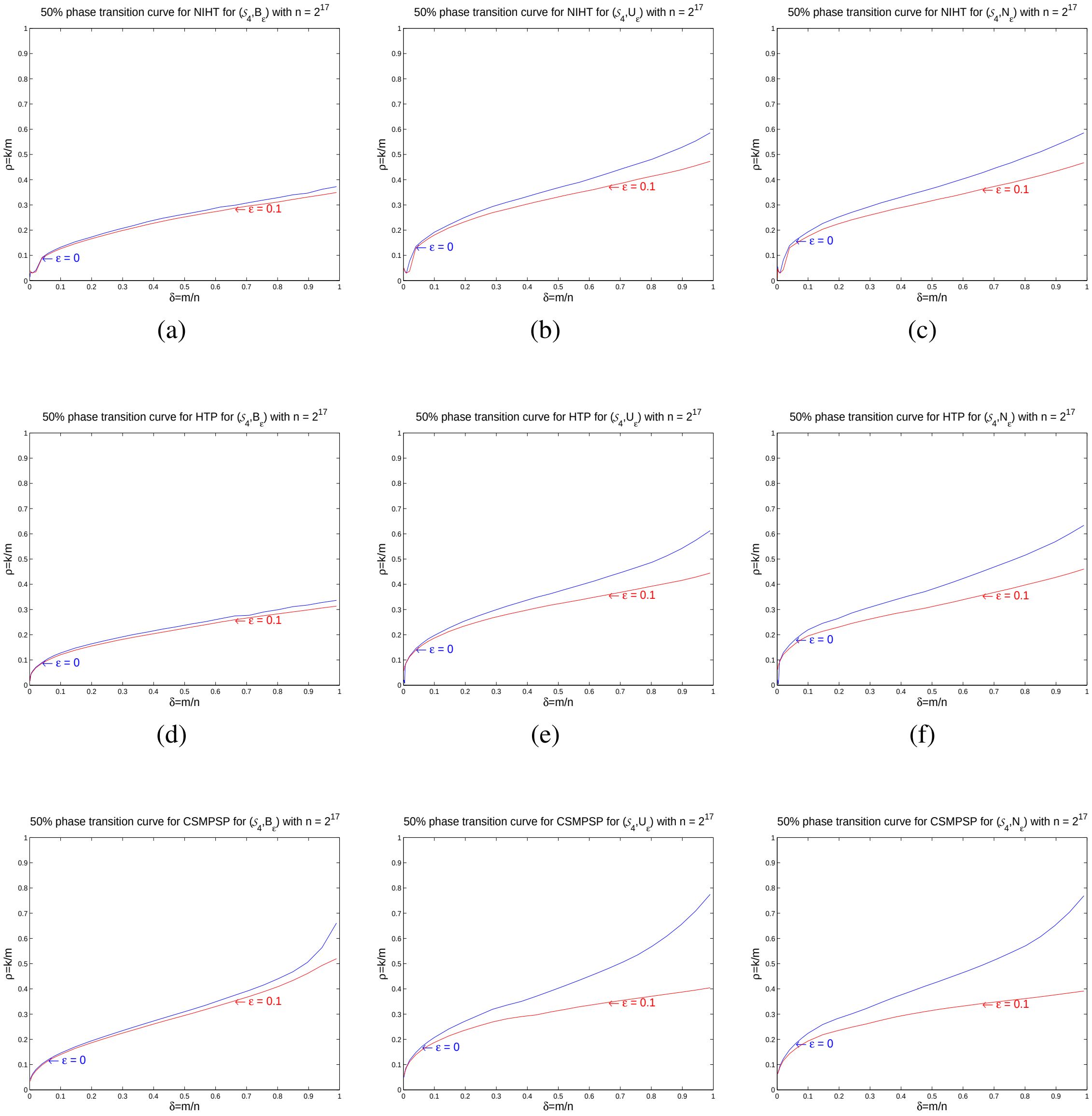


(a) (b)

(c) (d)

(e) (f)

Figure S27. Left panels: Algorithm selection maps for problem class $(\mathcal{S}_4, B)$. Right panels: Average time for the fastest algorithm. Panels (a-b) with $n = 2^{14}$, (c-d) with $n = 2^{16}$, and (e-f) with $n = 2^{18}$.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure S28. 50% recovery probability logistic regression curves for problem class $(\mathcal{S}_4, vec)$ with $n = 2^{17}$ and $vec \in \{B, U, N, B_\epsilon, U_\epsilon, N_\epsilon\}$ with $\epsilon = 0$ and $\epsilon = 1/10$. Algorithms: (a-c) NIHT, (d-f) HTP, and (g-i) CSMPSP. Left Panels: sparse binary vectors; Center Panels: sparse uniform vectors; Right panels: sparse normal vectors.