The In-Crowd Algorithm for Fast Basis Pursuit Denoising

Patrick R. Gill, Albert Wang, *Student Member, IEEE*, and Alyosha Molnar, *Member, IEEE*

***Abstract*—We introduce a fast method, the "in-crowd" algorithm, for finding the exact solution to basis pursuit denoising problems. The in-crowd algorithm discovers a sequence of subspaces guaranteed to arrive at the support set of the final solution of l_1-regularized least squares problems. We provide theorems showing that the in-crowd algorithm always converges to the correct global solution to basis pursuit denoising problems. We show empirically that the in-crowd algorithm is faster than the best alternative solvers (homotopy, fixed point continuation and spectral projected gradient for l_1 minimization) on certain well- and ill-conditioned sparse problems with more than 1000 unknowns. We compare the in-crowd algorithm's performance in high- and low-noise regimes, demonstrate its performance on more dense problems, and derive expressions giving its computational complexity.**

***Index Terms*—Algorithms, computation time, optimization methods, tomography.**

I. Introduction

A. Basis Pursuit Denoising

FINDING the best sparse representation for high-dimensional data is an important step for many applications in signal processing [1]–[3] and statistics [4], [5]. Solving the underdetermined system of linear equations $\mathbf{y} = \mathbf{A}\mathbf{x}$ (where $\mathbf{A}$ is an $M \times N$ matrix and $M < N$) subject to a sparsifying regularizer is effective in identifying such representations. Regularization can be thought of as a mathematical implementation of Occam's Razor: in the face of many possibilities, all of which are plausible, favor the simplest candidate solutions. In this context, $\mathbf{y}$ is the $M \times 1$ vector of observed data, $\mathbf{A}$ is a transform matrix composed of N atoms, and $\mathbf{x}$ is the $N \times 1$ solution vector.

One way to ensure maximum sparsity in $\mathbf{x}$ is to solve the problem:

$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \text{ subject to } \mathbf{y} = \mathbf{A}\mathbf{x} \tag{1}$$

where $\|\mathbf{x}\|_0$ is simply the number of nonzero components of $\mathbf{x}$. Unfortunately, solving (1) usually involves a combinatorial search, making it computationally intractable. Minimization using the l_1 norm, which often delivers the same solution as the l_0 norm [6], [7], is frequently substituted:

$$\min_{\mathbf{x}} \|\mathbf{x}\|_1 \text{ subject to } \mathbf{y} = \mathbf{A}\mathbf{x}. \tag{2}$$

The above minimization problem is also known as basis pursuit [8]. Although the l_1 norm is weaker than l_0 in ensuring sparsity, l_1-regularized optimization is a convex problem and admits efficient solution via linear programming techniques.

In many applications (see Section II) it is desirable to trade off exact congruence of $\mathbf{A}\mathbf{x}$ and $\mathbf{y}$ in exchange for a sparser $\mathbf{x}$. In these cases, a more appropriate formulation is basis pursuit denoising (BPDN) [3], [9]. BPDN involves solving the following problem:

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1. \tag{3}$$

BPDN [closely related to LASSO regression [4] and see (6)] is simply least-squares minimization with an l_1 regularizer to penalize complex solutions. The regularization parameter $\lambda > 0$ establishes the cost of complexity relative to the least-squares error $\frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$.

Notation

We introduce some notation for the problem in (3) that will be useful later. Let $\mathbf{x}$ be candidate solutions of (3). Let $\mathbf{r}$ be the residual $\mathbf{r} \equiv \mathbf{y} - \mathbf{A}\mathbf{x}$. $f(\mathbf{x})$ is the total error to minimize, such that $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{r}\|_2^2 + \lambda\|\mathbf{x}\|_1$. This total error can be divided into a reconstruction imperfection term $\frac{1}{2}\|\mathbf{r}\|_2^2$ and the regularizer $\lambda\|\mathbf{x}\|_1$. The imperfection term $\frac{1}{2}\|\mathbf{r}\|_2^2$ penalizes the deviation of the expected observations $\mathbf{A}\mathbf{x}$ from the actual observations $\mathbf{y}$, while $\lambda\|\mathbf{x}\|_1$ penalizes solutions where the sum of $|x_k|$ is high. The dot product of two vectors is denoted with angle brackets $\langle , \rangle$ and element-by-element vector multiplication is $*$. Let p be the number of nonzero components of the $\mathbf{x}$ that solves (3). We assume that the l_2 norm of the columns A_i of $\mathbf{A}$ is 1, and that $\mathbf{y}$ lies within the subspace spanned by these columns.[1]

B. Organization of the Manuscript

In Section II, we describe applications of BPDN, including the 3-D imaging application [10], [11] that motivates us to look for fast BPDN solvers. Section III outlines prior work towards obtaining solutions to (3). The in-crowd algorithm is given in Section IV along with convergence proofs (also see Appendix A) and iteration bounds. Section V compares the accuracy of BPDN reconstructions to those of some alternative sparse solvers. Section VI benchmarks the in-crowd algorithm

Manuscript received May 28, 2010; revised October 08, 2010, February 16, 2011, and May 10, 2011; accepted June 20, 2011. Date of publication July 05, 2011; date of current version September 14, 2011. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Arie Yeredor. This work was funded in part by the NIH under R21 Grant EB 009841-01.

The authors are with the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14853 USA (e-mail: prg56@cornell.edu; aw383@cornell.edu; am699@cornell.edu).

Digital Object Identifier 10.1109/TSP.2011.2161292

[1]Relaxing this constraint would put a portion of $\mathbf{y}$ orthogonal to any possible $\mathbf{A}\mathbf{x}$ and therefore irrelevant to the optimization problem.

against alternatives, and investigates the effect of changing λ. Section VII investigates a regime of dense BPDN problems where alternative solvers are faster than the in-crowd algorithm. Finally, Section VIII derives an expression for the computational complexity of an iteration of the in-crowd algorithm and provides avenues of exploration that may yield faster variants of the in-crowd algorithm for different problem scales.

II. Motivation and Background

A. Applications of BPDN

There are two broad categories of application in which solving (2) will not recover a useful, sparse $\mathbf{x}$; specifically:

Category 1: $\mathbf{y} = (\mathbf{A}\mathbf{x})*(1+\boldsymbol{\eta})$ where $\mathbf{x}$ may be sparse, but $\boldsymbol{\eta}$ is $M \times 1$ noise with a large enough magnitude that solving (2) exactly would constitute unacceptable overfitting. For these problems, λ is set to be high enough that the effect of the regularizer in (3) is at least as large as the effect of $\boldsymbol{\eta}$.

Category 2: $\mathbf{y}$ was not generated by the linear matrix multiplication of an $\mathbf{A}$ matrix with a sparse $\mathbf{x}$.

Problems falling into Category 1 include the following:

i) computer vision and reconstruction problems [12];
ii) recovering sparse, noisy signals or images [13], [14].

Problems falling into Category 2 include the following:

i) lossy image or video compression or encoding with overcomplete dictionaries [15]–[18];
ii) reconstructing the few sparse, strongest components of a dense $\mathbf{x}$, typical in compressed sensing [19] imaging applications;
iii) image denoising with an overcomplete basis [20].

The denoising form of basis pursuit is therefore used for many real-world problems.

B. Imaging With ASPs and BPDN

Finding fast solutions to large-scale, ill-conditioned BPDN problems is needed for a 3-D imaging application described in the remainder of Section II.

Determining the three-dimensional arrangement of light sources is an important component of many biological assays and studies [21], [22]. As static optical sensors can only measure information at a fixed two-dimensional plane, the recovery of three-dimensional structure is intrinsically an underdetermined problem. The majority of contemporary techniques rely on scanning or complex optical systems to overcome this measurement deficiency [22]. We have recently demonstrated a new class of angle sensitive pixel (ASP) based imager which directly recovers sufficient information to permit 3-D reconstructions of sparse light sources [10], [11]. An ASP observes a signal which follows the relation:

$$R = I_0(1 - m\cos(b\theta + \phi))F(\theta)(1 + \eta) \tag{4}$$

where R is the readout of the ASP, I_0 is a constant proportional to the light flux at the ASP, θ is the incident angle along the optically sensitive axis, b is the angular sensitivity (designed to be in the 7–50 range), m is the modulation depth of the ASPs (designed to be maximal; typical values of m are approximately 0.6), ϕ is a designable phase offset, $F(\theta)$ is a slowly varying aperture function and η is multiplicative noise. Compared to traditional intensity-sensitive pixels, the outputs of heterogeneous arrays of ASPs have a more independent output when exposed to out-of-focus light sources. We search for a sparse set of candidate light sources that account for the observed signal by posing a BPDN problem that reconstructs the location and intensity of several nearby light sources, as follows.

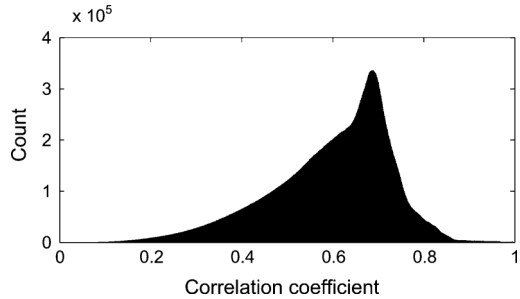


Fig. 1. Distribution of correlation coefficients of the columns of $\mathbf{A}$ for a sample imaging problem with $N = 12\,500$ and $M = 625$.

Assume that there is some volume of interest in which we wish to determine the location and magnitude of a variety of light sources. We parcel the volume into N individual subregions, and form the vector $\mathbf{x} = [x_1, x_2, \ldots x_N]^T$, where the i^{th} component x_i represents light source intensity at the i^{th} subregion. Using (4), we determine the response of the M ASPs for unit intensity at each individual subregion. The normalized individual responses to light at one location define one column A_i of the matrix $\mathbf{A}$. The system is linear, so for a given arrangement of sources in space $\mathbf{x}_0$, the product $\mathbf{A}\mathbf{x}_0$ predicts the sensor outputs $\mathbf{y}_0$ we would observe. Therefore, for a given observed set of outputs $\mathbf{y}$, the solution to (3) provides a reasonable guess at the true structure of the few luminous sources.

Two features that are readily apparent are the scale of the optimization problem and the ill-conditioned nature of the matrix $\mathbf{A}$. Dividing a volume of one cubic millimeter into parcels 10 microns on a side results in an N of 10^6, while a small imager might have $M \approx 10^4$–10^5 sensors. Furthermore, spatially adjacent sources are likely to produce very similar responses. This results in a high mutual coherence [23] for $\mathbf{A}$ (see Fig. 1 for the distribution of off-diagonal entries of $\mathbf{A}^T\mathbf{A}$ for a sample problem with 12 500 spatial locations and 625 sensors). Therefore, any BPDN solver appropriate for this reconstruction problem must be able to handle poorly conditioned matrices and very large problem scales. These two requirements drove our development of the in-crowd algorithm.

III. Solving BPDN

In this section, we review established approaches to solving (3) quickly for sparse $\mathbf{x}$. One popular approach to solving BPDN is based on homotopy [24]. Homotopy methods trace the global optimal solution path over changing λ. For $\lambda = \infty$, the optimal solution is trivial: $\mathbf{x} = \mathbf{0}$. Relaxing λ from ∞ causes the optimal solution path to leave the origin and introduce nonzero components. Further decreases may introduce new nonzero components or drive existing components to zero; hence, homotopy methods return not only a solution for a given λ, but also the solution trajectory $\mathbf{x}(\lambda)$ illustrating the optimal solution for a

whole range of λ. By virtue of their reliance on the overall optimum solution as components enter and exit the active set, these optimizers are very efficient for sparse $\mathbf{x}$ (see [25] and the associated implementation at [26]). Homotopy is the fastest alternative to the in-crowd algorithm for solutions to (3) on our imaging application (see Section II-B), as will be shown in Section IV-B.

Another fast method which can be made to converge to the solution to (3) is the spectral projected gradient for L1 minimization (SPGL1) [27], [28]. This method probes the shape of the trade-off curves between solutions of l_1 minimization of $\mathbf{x}$ for a constrained $\|\mathbf{r}\|_2{}^2$ [see (5)] and l_2 minimization of $\mathbf{r}$ for a constrained $\|\mathbf{x}\|_1$ [(6), also known as the LASSO problem [4]]. This method is faster than homotopy for problems where columns of $\mathbf{A}$ are nearly orthogonal, as will be shown in Section VI-A.

Implementations of SPGL1 do not directly solve (3), rather they solve either

$$\min_{\mathrm{x}} \|\mathbf{x}\|_1 \text{ subject to} \|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2 \leq \sigma \tag{5}$$

or

$$\min_{\mathrm{x}} \|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2 \text{ subject to } \|\mathbf{x}\|_1 \leq \tau. \tag{6}$$

For benchmarking purposes, in this paper we determine the time taken for SPGL1 to reach the exact solution of (3) by first computing $\tau = \|\mathbf{x}\|_1$ where $\mathbf{x}$ is the homotopy BPDN solution, then solving (6) with SPGL1.

Interior point methods solving (3) as a general convex problem also solve BPDN. A method using the preconditioned conjugate gradients algorithm to compute a search direction has shown itself to perform well on large problems [29]. However for the problems presented in this paper we found the homotopy implementation to be faster than these methods, so we will not cover them in detail.

Fixed-point continuation (FPC) [30], both using Barzilai-Borwein steps [31] and with an active set implementation [32], is also potentially an attractive BPDN solver. As will be shown in Section VI-A, for sparse problems the active set FPC method's run times compare with homotopy's and for more dense problems FPC is the fastest BPDN solver (see Section VII). However, FPC is known to produce incorrect solutions in some hard cases [30]. In fact, on the imaging problem Section VI-B), we will show active set FPC routinely fails to converge to a solution close to the correct $\mathbf{x}$ (see Fig. 5). For no problem was the speed of the Barzilai-Borwein FPC method [31] competitive, as will be shown in Section VI-A and Section VI-B. FPC can take a value of λ as an input so that its target is to solve (3); we have provided this λ as an input throughout the paper.

One additional class of interest for smaller problems is gradient projection for sparse reconstruction (GPSR) [33]. Although we investigated this class of BPDN solver, under no circumstances did it deliver a solution faster than some alternative solver (be it homotopy, active set FPC or SPGL1), so we do not report its results.

IV. In-Crowd Optimization

In this section, we introduce the in-crowd algorithm, an iterative method for solving BPDN that is effective especially for large scale sparse problems, and prove its convergence. The flavor of the in-crowd algorithm can be summarized as follows: *think globally, but not too often.* The computational complexity of solving (3) with sufficiently large N and small p is often dominated by searching the dictionary of N possible atoms for appropriate additions to the active set I. The in-crowd algorithm is partially insulated from the size of the global problem by consulting the full dictionary only rarely. Very often, candidates for additions to I remain viable even after adding other candidates. Performing an entirely new search over the N possible additions after each addition to I thus can be computationally wasteful. Instead, the in-crowd algorithm admits a whole group of L atoms to I, where L is a fixed small integer, before referring to the full $\mathbf{A}$ matrix again.

A. The In-Crowd Algorithm

The following is the procedure for in-crowd optimization.

Step 1) Declare $\mathbf{x}$ to be $\mathbf{0}$, so $\mathbf{r} = \mathbf{y}$.
Step 2) Declare the active set I to be the empty set.
Step 3) Calculate the "usefulness" $u_j \equiv |\langle \mathbf{r}, A_j \rangle| \forall j \in I^c$, where I^c denotes the complement of I.
Step 4) If on I^c no $u_j > \lambda$, then terminate.
Step 5) Otherwise, add the L components with the largest u_j to I, but do not add any component for which $u_j \leq \lambda$.
Step 6) Solve (3) exactly on the subspace spanned by all of the components in I. Use current values of $\mathbf{x}$ to warm-start the solver. This subproblem is expected to be dense.
Step 7) Take any zero-valued members of the exact solution of Step 6) out of I.
Step 8) Set all components of $\mathbf{x}$ to be 0 except for the components in I; set these to the value found by the exact solution of Step 6).
Step 9) Update $\mathbf{r} = \mathbf{y} - \mathbf{A}\mathbf{x}$; n.b. $\mathbf{A}\mathbf{x}$ can be found during the subproblem of Step 6) since $\forall\, j \in I^c, x_j = 0$.
Step 10) Go to Step 3).

In Step 5), the best choice of L depends on the relative speed of Steps 6) and 3) for a specific problem and computer architecture. We find $L = 25$ to be reasonable for most problems and we have used it exclusively throughout this paper. However, for different choices of matrix $\mathbf{A}$ or for problems (such as partial Fourier observations) with an implicit fast method for finding $\mathbf{A}\mathbf{x}$, other fixed or adaptive choices for L can provide advantages in computation time (see Section VIII-C). We used Matlab's built-in quadprog function [34], [35] for our subproblem solver (see Section VIII-B) in Step 6) when the cardinality of I was $< M$ and an alternative quadratic programming solver[2] [36] when the cardinality of I was $\geq M$.

The stopping criterion in Step 4) is equivalent to terminating when I does not change from one iteration to the next. In practice, subproblem solvers may not fully optimize in Step 6),

[2]Matlab's built-in quadprog does not always respect the boundary conditions given when the cardinality of $I \geq M$, necessitating an alternative exact solver. In our benchmarks, this alternative solver was almost never used since the cardinality of I usually remained well below M, but was needed as a patch especially for larger problems with small λ.

so this alternative stopping criterion can be more numerically robust.

B. Convergence of the Algorithm

To show that the in-crowd algorithm always terminates at the global optimum, it suffices to prove the following.

- Item 1) The error $f(\mathbf{x})$ calculated in Step 6) is completely specified by I.
- Item 2) There are a finite number of possible sets I.
- Item 3) Step 6) will always decrease the error relative to the previous round; i.e., $f(\mathbf{x}_{t+1}) < f(\mathbf{x}_t)$.
- Item 4) At termination, nonzero components x_k of $\mathbf{x}$ satisfy $\langle \mathbf{r}, A_k \rangle = \mathrm{sgn}(x_k)\lambda$, and zero components x_j of $\mathbf{x}$ satisfy $|\langle \mathbf{r}, A_j \rangle| \leq \lambda$.

Items 1) and 2) are trivial, but Items 3) and 4) are subtle enough to warrant a proof. Item 3) is proven in Theorem A.1; moreover, under certain circumstances, the error $f(\mathbf{x})$ is proved to decay exponentially with iteration count (see Appendix A). Step 6) will find the solution that minimizes $f(\mathbf{x})$ on I and may not end up assigning a nonzero value to all of the newly added components, but the final error $f(\mathbf{x})$ is guaranteed by Theorem A.1 to be lower than during the previous execution of Step 6).

The in-crowd algorithm therefore cannot retrace its own path, eliminating the potential for cycles. Moreover, it makes use of the best possible choices for additions to I given local knowledge of $\mathbf{r}$ by adding components to I with the highest u. Since the in-crowd algorithm never retraces its path and traverses a finite number of elements, it must terminate. We now prove that the in-crowd algorithm terminates only at the exact BPDN solution [see Item 4)] by pointing out a feature of the in-crowd algorithm's stopping condition that we will tie to general convex optimization theory in Appendix B.

Theorem 4.1: When the in-crowd algorithm terminates, all components x_i of $\mathbf{x}$ are either equal to zero and $|\langle \mathbf{r}, A_i \rangle| \leq \lambda$, or alternatively are nonzero and satisfy $\langle \mathbf{r}, A_i \rangle = \mathrm{sgn}(x_i)\lambda$.

Proof: By Step 4), the in-crowd algorithm does not terminate if for any $j \in I^c$, $u_j \equiv |\langle \mathbf{r}, A_j \rangle| > \lambda$. Regarding the nonzero components $\{x_k\}, k \in I$, their values have been optimized by the exact solver of Step 6). On $\{x_k\}$, the gradient of $f(\mathbf{x})$ exists (since there are no discontinuities in the gradient of $f(\mathbf{x})$ except where $x_i = 0$), and (by force of the exact optimization) must equal $\mathbf{0}$, since any nonzero gradient would imply a better solution than the one found by the exact subproblem solver. By (8) of Appendix B, the gradient being $\mathbf{0}$ implies that $\langle \mathbf{r}, A_k \rangle = \mathrm{sgn}(x_k)\lambda$. ∎

As established by general convex optimization theory, the conditions established in Theorem 4.1 are both necessary and sufficient for the optimality of the solution $\mathbf{x}$ (see Appendix B) generated by the in-crowd algorithm. Therefore, the in-crowd algorithm halts only at the global minimizer to (3).

C. Lower Iteration Bound

A lower bound on the iteration count, $\lceil \frac{p}{L} \rceil + 1$, is given by the fact that at least $\lceil \frac{p}{L} \rceil$ iterations are required increase the cardinality of I to p, and one partial iteration [terminating at Step 4)] is required to confirm the optimality of $\mathbf{x}$ (i.e., to check that the subdifferential of f contains $\mathbf{0}$, see Appendix B).

There is no guarantee that the same few elements will be not be added then pruned multiple times, however by Items 1) and 3) above, we are guaranteed that each temporary addition to I must be made with distinct combinations of components in I, each with lower associated $f(\mathbf{x})$ than all previous combinations, limiting the number of possible cyclic additions and deletions of all atoms. As will be shown in Section VIII-A and the insets of Figs. 3, 4, 6, and 7, the lower bound given here is in fact reasonably tight.

V. Greedy Solvers and BPDN

A. Greedy Solvers

As $f(\mathbf{x})$ is a convex function, improved solutions can be found using only local knowledge of f around $\mathbf{x}$. Several existing heuristics take advantage of this property to find sparse $\mathbf{x}$ where $\mathbf{y} \approx \mathbf{A}\mathbf{x}$. Examples include orthogonal matching pursuit (OMP) [37] and least angle regression (LARS) [38]. These approaches build their solution element by element, choosing the best available atom at each iteration. Every iteration adds this A_i to an active set of columns.

Recent solutions in a similar spirit to the in-crowd algorithm are CoSaMP [23] and subspace pursuit [39]. Both of these add the ability to prune unnecessary elements from the current active set of columns, and both solve a least squares problem on their active sets at each iteration. Indeed, the primary difference between the in-crowd algorithm and these two procedures is that with the in-crowd algorithm, an l_1 regularizer (with a λ equal to that of the global problem) is included in the subproblem.

It should be noted that unlike the greedy solvers mentioned above, the in-crowd algorithm solves (3) exactly.

B. BPDN Yields Better Imaging Reconstructions

For our imaging application, the accuracy of reconstructions is higher for BPDN than for the other sparse solutions found by OMP, CoSaMP and subspace pursuit. To quantify the benefit of BPDN, we took the imaging problem whose $\mathbf{A}$ matrix generated Fig. 1 and judged reconstruction of sources based on a noisy signal. In more detail, S true sources with intensity[3] in $[-1, 1]$ were chosen randomly to generate $\mathbf{x}_{\mathrm{ideal}}$, and noiseless data $\mathbf{y}_{\mathrm{ideal}} = \mathbf{A}\mathbf{x}_{\mathrm{ideal}}$ were generated. We presented the solvers with $\mathbf{y} = \mathbf{y}_{\mathrm{ideal}} * (1 + \boldsymbol{\eta})$, where $\boldsymbol{\eta}$ is $M \times 1$ Gaussian noise such that the signal-to-noise ratio of $\mathbf{y}$ is 10. Fig. 2 then plots the mean correlation coefficient between $\mathbf{x}$ and $\mathbf{x}_{\mathrm{ideal}}$ over 100 runs for solutions of (3) with $\lambda = 0.2$ along with those from OMP, subspace pursuit and CoSaMP.

Subspace pursuit and CoSaMP[4] fare particularly poorly at this task, since the pseudoinverse tends to partition power equally among components of $\mathbf{x}$ with highly correlated corresponding A_k (see Fig. 1).

[3]We allowed negative light sources to compensate for the fact that the implementations of some of the methods we wished to benchmark did not allow us to restrict solutions to having exclusively nonnegative components.

[4]CoSaMP is usually implemented with an approximate least squares solver that is only accurate when columns of $\mathbf{A}$ are nearly orthogonal. For our problem we used an exact pseudoinverse since many columns were highly correlated.

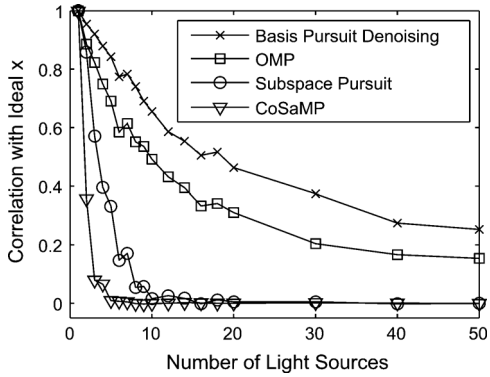


Fig. 2. Accuracy of four reconstruction methods for the imaging problem with $M = 625$, $N = 12\,500$ and various numbers of light sources S.

TABLE I
PROBLEM DEFINITIONS

Problem	N	M	S	p_G	p_I
1	1000	200	20	1.52	7.42
2	4000	200	20	1.48	8.02
3	4000	800	20	14.8	21.7
4	4000	800	80	61.9	46.1
5	10000	500	25	14.4	23.2
6	10000	500	50	32.7	30.9
7	10000	1000	25	20.1	30.0
8	10000	1000	100	91.4	69.1
9	30000	1000	25	19.3	35.3
10	30000	1000	50	39.6	54.1
11	30000	1000	100	118.2	77.1
12	100000	1000	25	20.1	35.6
13	100000	1000	50	39.7	61.1
14	100000	1000	100	196.1	80.5
15	200000	1000	25	19.8	38.1
16	200000	1000	50	41.4	56.7
17	200000	1000	100	269.1	82.0

VI. OBSERVED PERFORMANCE OF THE IN-CROWD ALGORITHM

A. Matrices With Small Correlations

To the extent that it is possible to control $\mathbf{A}$, for nearly all applications it is advantageous to make the columns of $\mathbf{A}$ as orthogonal as possible. As a consequence, many applications that use BPDN employ $\mathbf{A}$ matrices with small-magnitude correlations between rows. Although when $M < N$ it is impossible to have columns totally orthogonal, by the central limit theorem the expected magnitude of correlation coefficients between columns of a random i.i.d. Gaussian $\mathbf{A}$ goes as $\frac{1}{\sqrt{M}}$. Benchmarking solutions to (3) with a random $\mathbf{A}$ matrix therefore gives a reasonable approximation to the expected running times of applications where the columns of $\mathbf{A}$ are nearly orthogonal.

We generated a sequence of problems (see Table I) defined by a spherical ($\|A_k\|_2 = 1$) Gaussian random matrix $\mathbf{A}$ with various M and N. We generated synthetic data $\mathbf{y}$ in a manner similar to that of Section V-B, i.e., by starting with a vector $\mathbf{x}_{\text{ideal}}$ with S nonzero components pulled from a random uniform distribution over $[-1, 1]$, applying $\mathbf{A}$ and injecting multiplicative Gaussian noise to arrive at an SNR of 10. Solving (3) with $\lambda = 0.2$ results in a solution with a number of nonzeros p that may be greater or smaller than S, and the average p was also found to depend on the problem type (Gaussian p_G or Imaging p_I—see Section VI-B). To avoid favoring methods using

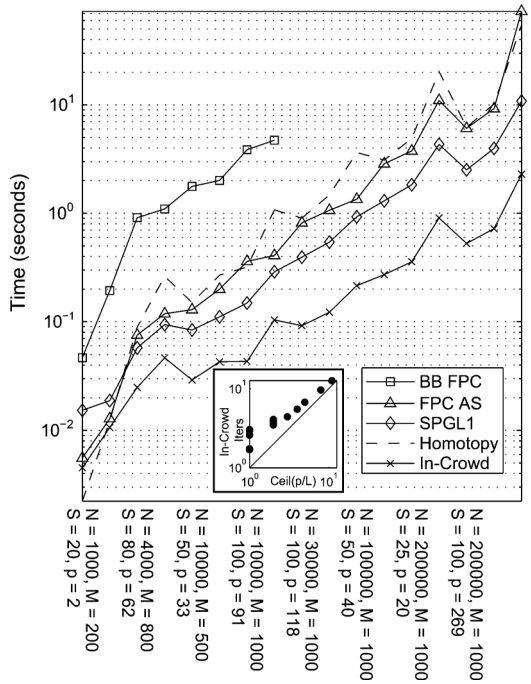


Fig. 3. Running time for random Gaussian matrices. For clarity, every second problem size is labeled; for a full list of problem sizes see Table I. Inset: mean in-crowd iteration count as a function of Ceil ($\frac{\text{mean}(p)}{L}$); log scale used.

TABLE II
RUNNING TIMES IN SECONDS FOR GAUSSIAN RANDOM PROBLEMS

Problem	In-Crowd	Homotopy	SPGL1	FPC AS	BB FPC
1	0.00453	0.00223	0.0153	0.00557	0.0467
2	0.0106	0.0123	0.0189	0.0129	0.194
3	0.025	0.0893	0.0575	0.0753	0.911
4	0.0468	0.262	0.094	0.119	1.1
5	0.0292	0.15	0.0838	0.129	1.77
6	0.043	0.269	0.111	0.2	2.01
7	0.0434	0.32	0.149	0.36	3.86
8	0.104	1.07	0.289	0.408	4.72
9	0.0918	0.903	0.393	0.818	long
10	0.122	1.47	0.542	1.07	long
11	0.215	3.64	0.93	1.35	long
12	0.272	3.09	1.3	2.87	long
13	0.357	4.91	1.83	3.76	long
14	0.906	20.3	4.33	11.0	long
15	0.527	6.2	2.52	6.09	long
16	0.723	10.2	3.97	9.24	long
17	2.29	54.4	10.8	72.9	long

Matlab's built-in functions, we translated all methods into c++ via Matlab's mcc command and compiled the c++ code with gcc 4.1.1 with all optimizations possible. We timed the execution of 50 problems on a 2.4 GHz Core i7 system with turbo boost disabled to obtain the benchmarks for in-crowd, the fastest homotopy solver we found for these problems [26], the spectral projected gradient for L1 minimization (SPGL1) [27], [28] and two implementations of FPC: Barzilai–Borwein FPC (BB FPC) [31] and active set FPC (FPC AS) [32]. Mean running times in seconds are plotted in Fig. 3 and listed in Table II. The inset of Fig. 3 shows that the number of iterations the in-crowd algorithm takes is close to the lower bound derived in Section IV-C.

As can be seen in Fig. 3 and Table II, the in-crowd method is up to 5.5 times faster than all other solvers for $N \geq 4000$. Its closest rival (SPGL1) in fact solves LASSO [see (6)] rather than BPDN; the fastest alternative BPDN solver of problem 17

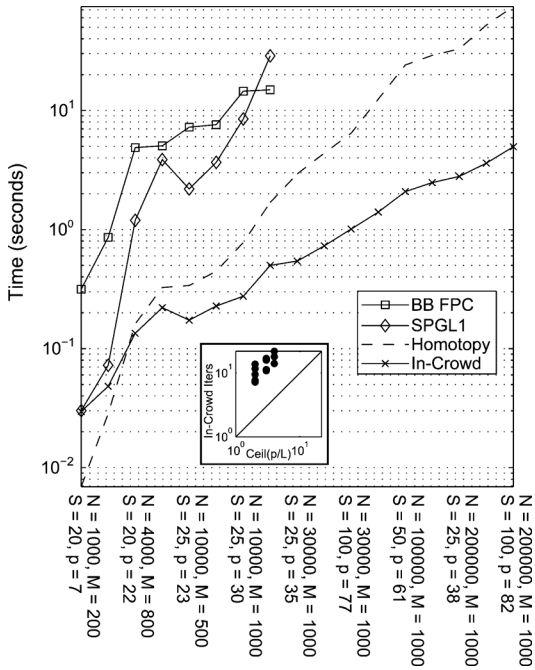


Fig. 4. Running time for the imaging problem. For clarity, every second problem size is labeled; for a full list of problem sizes see Table I. Inset: mean in-crowd iteration count as a function of Ceil ($\frac{\text{mean}(p)}{L}$); log scale used.

TABLE III
RUNNING TIMES IN SECONDS FOR IMAGING PROBLEMS

Problem	In-Crowd	Homotopy	SPGL1	BB FPC
1	0.0297	0.00692	0.0301	0.315
2	0.0482	0.0284	0.0725	0.859
3	0.134	0.161	1.2	4.88
4	0.221	0.326	3.87	5.05
5	0.173	0.338	2.19	7.25
6	0.228	0.446	3.67	7.6
7	0.275	0.778	8.48	14.5
8	0.5	1.69	28.7	14.9
9	0.543	2.94	long	long
10	0.729	4.38	long	long
11	1.01	6.45	long	long
12	1.4	12.5	long	long
13	2.08	24.1	long	long
14	2.48	29.1	long	long
15	2.8	32.9	long	long
16	3.61	52.3	long	long
17	4.96	74.8	long	long

(homotopy) is more than 23 times slower than the in-crowd algorithm.

The maximum l_1 deviation of the in-crowd and homotopy solutions (i.e., $\|\mathbf{x}_{\text{in-crowd}} - \mathbf{x}_{\text{homotopy}}\|_1$) for any[5] run was within the range of deviations entirely accountable by finite machine precision: $5{*}10^{-13}$. SPGL1 converges iteratively to the exact LASSO solution. Using the default SPGL1 stop criterion [28] (optimality tolerance $= 10^{-4}$), SPGL1 halted relatively close to the solution found by the other solvers. SPGL1's maximum l_1 deviation was 0.004, which, while larger than that of the solution delivered by homotopy, results in only a minuscule increase in final $f(\mathbf{x})$.

[5]Excluding runs with $p = 0$, where the homotopy implementation we used [26] takes a single initial step.

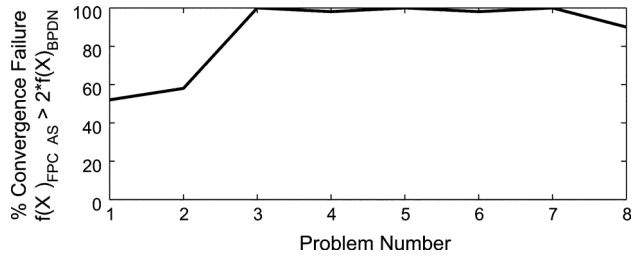


Fig. 5. Proportion of problems where active set FPC failed to discover a solution with an $f(\mathbf{x})$ within a factor of 2 of the correct BPDN solution as a function of problem number (see Table I).

B. 3-D ASP Imaging Application

We ran a suite of test problems similar to those of Section IV-A but with $\mathbf{A}$ based on models of our imaging application (see Section II-B); the timing results can be found in Fig. 4 and Table III (n.b. the FPC variant mentioned here is Barzilai–Borwein). The inset of Fig. 4 shows that the number of iterations taken by the in-crowd algorithm is a factor of 2.56–6.86 greater than the lower iteration bound derived in Section IV-C. The increased number of iterations compared to the previous set of benchmarks is due to the difficulty selecting the appropriate atom out of a collection of atoms that are highly correlated. Still, the ratio of iterations taken by the in-crowd algorithm to the lower bound is small, indicating that the in-crowd algorithm is relatively robust in the face of poorly conditioned problems.

On these problems, we discovered that SPGL1, the fastest exact solver aside from the in-crowd for nearly orthogonal $\mathbf{A}$ problems, does not converge quickly, so we ceased our benchmarks of this method for large-scale problems. SPGL1 still seems to converge exponentially, but the exponential constant is small.

We found the active set implementation of FPC [32] did not converge well even for our smallest problems. In the cases where the active set method did converge to the correct solution, the discrepancy between its solution and that of other BPDN solvers was of the order expected by finite machine precision; however especially for problems larger than problem 2, for the large majority of problems the active set method had a catastrophically large terminal $f(\mathbf{x})$: more than twice that of the other solvers[6] (see Fig. 5). In general, FPC is known to fail to converge for some difficult problems [30], although solutions delivered by Barzilai-Borwein FPC were more reasonable. As such we benchmarked only the Barzilai-Borwein FPC variant.

Although the l_1 divergence between homotopy and the in-crowd solutions was higher than for the Gaussian case ($3 * 10^{-7}$ for problem 17), the corresponding median difference in $f(\mathbf{x})$ was within machine precision: $2 * 10^{-14}$. This discrepancy is possible because level surfaces of $f(\mathbf{x})$ around the minimum will be elongated whenever two nonzero, nearly parallel atoms exist as part of the true solution. As before, for this problem type the in-crowd algorithm was the fastest solver by a factor of 15 for the largest-scale problems.

[6]We were careful to allow a sufficiently high maximum number of iterations to the active set FPC solver, and exactly the same code was used to analyze imaging problems as was used in Gaussian problems where this method returns the correct answer. We speculate that the coherent $\mathbf{A}$ matrix of imaging problems derails active set FPC.

As is evident from p_I shown in Table I, BPDN did not in general return the solution corresponding to the exact light source locations. From Fig. 2, we see reconstruction accuracy is approximately 40% when $\frac{M}{S}$ is 20. However, due to the high correlations of the columns representing adjacent spatial locations, often the discrepancy between ideal and observed $\mathbf{x}$ corresponded to blurring, omitting dim sources or small shifts in the inferred source location, and the reconstructed sources are overall in a configuration similar to their actual locations. Our imaging application [10], [11] introduced in Section II-B motivates our interest in high-noise, very sparse, very underdetermined, ill-conditioned problems; the in-crowd algorithm excels in this regime.

C. Changing λ and Noise Levels

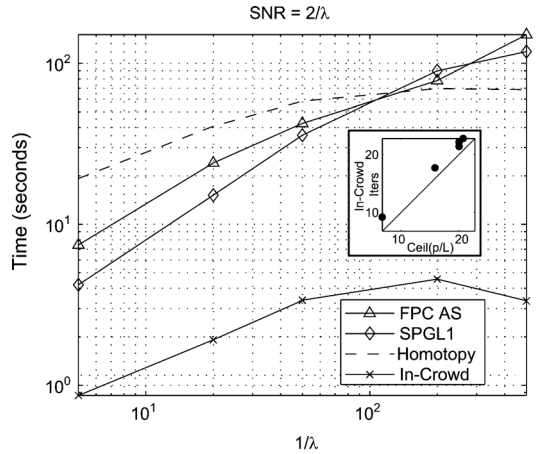
Thus far, all benchmarks discussed were run with $\lambda = 0.2$ and SNR = 10, meaning that the average influence of noise on the observed $\mathbf{y}$ is relatively large and the strength of the regularizer is about twice the strength of the noise. With a large enough λ, the cardinality of I never grows to be too large, so the dense subproblem of Step 6) can be solved relatively quickly. With a smaller λ, it is possible that not only more components will be involved with every subproblem (making individual iterations slower), but also that the total number of iterations taken by the in-crowd algorithm increases as the residual becomes more influenced by noise than by correct choices for additions to I.

To observe the effect of decreasing λ we performed two numerical experiments,[7] both based on Gaussian problem 14 of Table I. In one, noise and λ are scaled down together [see Fig. 6(a)] and in the other, noise is held with a constant SNR of 10 while λ is scaled down [see Fig. 6(b)] The insets in these figures plot the number of iterations taken by the in-crowd algorithm as a function of the lower iteration bound (see Section IV-C). Except where noise is much stronger than λ [to the right of Fig. 6(b)], the lower iteration bound is relatively tight.

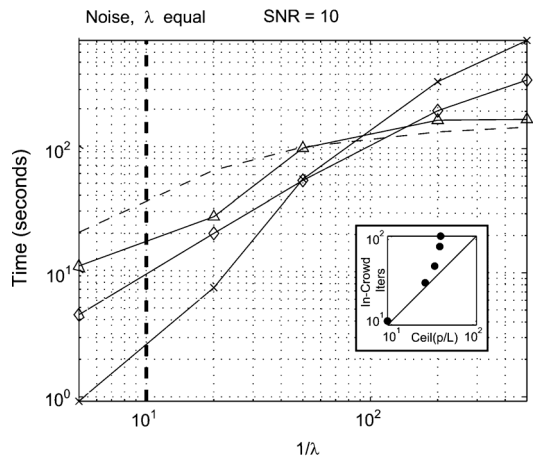
The vertical dashed line in Fig. 6(b) divides problems with choices of λ appropriate for denoising (to the left of the line) from those with λ insufficient to denoise the signal (right of the line). Most real problems of interest, where noise suppression is desirable and overfitting of noise is not, are expected to use λ greater than noise. If high levels of noise are combined with a low λ, the result is a high in-crowd iteration count [see inset of Fig. 6(b)] and longer running times than the other methods. Outside the zone where λ is too small and for the entirety of Fig. 6(a) (which plots performance where λ is proportional to noise), the in-crowd algorithm is still the fastest BPDN solver for this class of problem.

We also tested the effect of lowering λ on imaging problem 14. As before, we tested the cases both where the signal-to-noise ratio (SNR) equaled 10 and when the SNR was set to $\frac{2}{\lambda}$. As shown in Table IV, running times increase modestly with decreasing λ.

[7]As before, homotopy and the in-crowd algorithm yielded solutions almost exactly congruent with each other. The maximum l_1 deviation of the in-crowd and homotopy solutions was less than $1.4{*}10^{-9}$ for any problem.



(a)



(b)

Fig. 6. Performance of algorithms with alternative λs for Gaussian problem 14. **a** Mean running times for changing λ with the noise-to-λ ratio fixed at 0.5. Inset: mean in-crowd iteration count as a function of Ceil $\left(\frac{\text{mean}(p)}{L}\right)$; log scale used. **b** Mean running times for changing λ with the signal-to-noise ratio fixed at 0.1. Problems to the right of the dashed line constitute fitting noise. Inset: mean in-crowd iteration count as a function of Ceil $\left(\frac{\text{mean}(p)}{L}\right)$; log scale used.

TABLE IV
RUNNING TIMES IN SECONDS FOR IMAGING PROBLEM 14 WITH CHANGING λ

λ	SNR	In-Crowd	Homotopy
0.20	10	2.37	27.4
0.05	40	7.21	65.4
0.05	10	7.58	66.3
0.02	100	17.4	120.0
0.02	10	18.0	122.0

VII. LOW-NOISE, DENSE PROBLEMS

The imaging problems that inspired work on the in-crowd algorithm (see Section II-B) are of a scale with $\frac{M}{p} \approx 10-40$: less sparse than some interesting BPDN problems, but sparser than others. Typical video compression problems with an overcomplete basis [17] have $\frac{M}{p} \approx 50-20$; a relatively high-magnitude $\|\mathbf{r}\|_2$ is acceptable and a correspondingly large λ is used. In addition to sparse, noisy problems, Fig. 6(a) demonstrates that the in-crowd algorithm is even better suited to very sparse, very underdetermined problems with low noise and λ, and thus small final $\|\mathbf{r}\|_2$.

Compressed sensing problems [1], [2], [19] give rise to a class of very low-noise BPDN problems that are less sparse, with

TABLE V
DENSE PROBLEM DEFINITIONS; $\lambda = 0.0005$, SNR = 10 000

Problem	N	M	S	p_G	Accuracy
18	5000	2000	500	856	97.7%
19	10000	5000	1000	1322	97.5%
20	20000	5000	1000	1912	98.7%
21	50000	10000	1500	2354	99.3%
22	50000	10000	2000	4557	99.1%

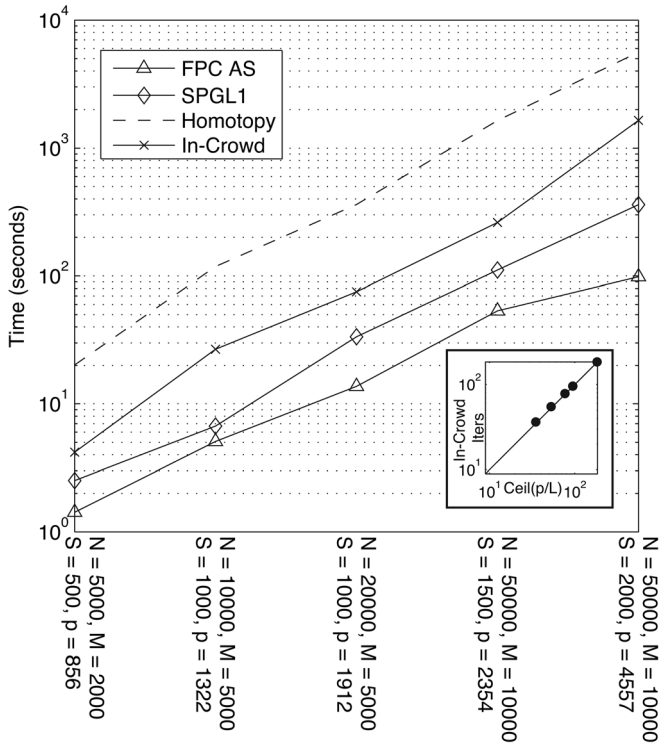


Fig. 7. Running time for random Gaussian matrices with low-noise, dense problems. More problem details are found in Table V. Inset: mean in-crowd iteration count as a function of Ceil $\left(\frac{\text{mean}(p)}{L}\right)$; log scale used.

$\frac{M}{p} \approx 5$ and $\frac{N}{M} \approx 3$. Compared to the BPDN problems presented thus far, in these applications noise is small and there are many more observations per unknown. Given the additional high-fidelity data available, the underlying signal $\mathbf{x}_{\text{ideal}}$ is expected to be reconstructed more or less exactly unless p is overwhelmingly large. We ran a set of dense problems defined in Table V with results plotted in Fig. 7 and Table VI to characterize in-crowd performance on dense problems. Despite the deceptively high p_G (which includes many almost-zero terms), for every problem in this suite the correlation coefficient between the recovered $\mathbf{x}$ and $\mathbf{x}_{\text{ideal}}$ was high (mean correlation given in % in Table V) and never less than 96% for any problem. While the iterative methods (the active set FPC method and SPGL1) both perform well on these problem scales, it is worth noting that the in-crowd algorithm maintains a run-time advantage over homotopy. Moreover, the fact that the number of iterations taken by the in-crowd algorithm was almost exactly equal to the lower iteration bound (shown by the proximity of points to the diagonal in the inset of Fig. 7) indicates that extremely few false steps were ever taken by the in-crowd algorithm. Perhaps a choice of $L = 25$ is overly cautious for this type of low-noise, nearly fully determined problem; see Section VIII-C for a discussion of potential modifications to the in-crowd algorithm that could make it faster on dense problems.

TABLE VI
RUNNING TIMES IN SECONDS FOR EXACT SOLUTIONS OF DENSER GAUSSIAN PROBLEMS

Problem	In-Crowd	Homotopy	SPGL1	FPC AS
18	4.18	20.1	2.52	1.42
19	26.7	118.5	6.69	5.08
20	75.3	361.7	33.5	13.7
21	262.1	1642.7	111.3	53.5
22	1650.1	5606.9	361.0	98.3

VIII. COMPUTATIONAL COMPLEXITY

A. Outer Loop Complexity

Here we provide an analysis of the algorithmic complexity of the in-crowd algorithm. Step 3) dominates the complexity of the outer loop [i.e., Steps 3)–10), excluding Step 6)]. It requires a matrix multiplication of $\mathbf{A}$ (which is $M \times N$) with the $M \times 1$ residual, requiring MN operations. A lower limit to the number of times the outer loop is run is $\lceil \frac{p}{L} \rceil + 1$, derived in Section IV-C. As with investigations into the number of homotopy iterations needed to find a solution [40], it is advantageous to empirically investigate the ratio of the actual number of iterations to this lower limit, henceforth denoted by $k_{\text{out}} \geq 1$. k_{out} ranged from 1.007–3.06 for all Gaussian random problems we investigated (see insets of Figs. 3, 6, and 7) and from 2.56 to 6.86 for all our imaging problems (see inset of Fig. 4). Although it is tempting to declare k_{out} to be effectively a constant, k_{out} might scale with M or N for certain types of problems, so we include it in our expression of computational complexity of the outer loop: $MN\left(\lceil \frac{pk_{\text{out}}}{L} \rceil + 1\right)$.

B. Inner Loop Complexity

The subproblem solver used in Matlab's quadprog routine [34], [35] uses an active-set strategy that alternates between using a least-squares solver on the nonzero components of $\mathbf{x}$ and a method of determining which components should be added or subtracted to this active set. Constraints on the associated quadratic programming problem (which enforce $|x_i| \geq 0$) act on the complement of the active sets; since the problem handed to the subproblem solver is dense and with fewer than $L + 1$ zero-valued components, the number of active constraints is also small and the computational cost of running the subproblem solver is dominated instead by the complexity of the iterated least-squares problem. The first iteration of this problem has a complexity of $O(q^3)$, where q is the current cardinality of I. Subsequent solutions to the least squares problem are accelerated by a Cholesky update to the initial problem as atoms enter and leave the active set,[8] with a

[8]In principle, the Cholesky factorization could be passed from the termination of one call to the subproblem solver to the initialization of the next subproblem solver, requiring a computationally faster update to the Cholesky factorization of complexity $O(q^2 L)$. However, in practice, a fresh factorization is usually faster. With $L = 25$, computing the *de novo* Cholesky factorization is faster than 25 incremental updates when $q < 15\,000$.

computational complexity $O(q^2 n)$, where n atoms are added or subtracted; often n is 1 or 2.

C. Potential Speed Improvements for Future Investigation

We can see several avenues for further speed improvements to the in-crowd algorithm going forward. These include the following.

1) Adaptively choosing the value of L and the type of subproblem solver based on the type and scale of the problem being solved,
2) Warm-starting the outer loop, the subproblem solver or both, based on the guess of a good dense solver (like SPGL1 or active set FPC) iterated only a few times.
3) Using the fact that the theorems proving in-crowd optimality are quite permissive, such that the subproblem solver does not need to solve BPDN *per se*, as long as the final value of $f(\mathbf{x})$ is smaller than that of the initial guess, and that for all nonzeros $\{x_k\}$ in the subsolver solution, $\langle \mathbf{r}, A_k \rangle = \operatorname{sgn}(x_k)\lambda$.

Overall, there are many freedoms permitted by in-crowd optimization that we have yet to explore. The conjunction of this flexibility with the already-encouraging numerical results presented in this paper mean that the in-crowd algorithm will be of considerable industrial and academic interest as a fast BPDN solver.

IX. Conclusion

We have presented a new algorithm for solving BPDN which we observe to be approximately 15 times faster (see Fig. 4 and Table III) than the best available alternative method (homotopy) for the types of imaging problems we encounter. It is approximately 5 times faster than SPGL1 on large mostly orthogonal problems (see Fig. 3 and Table II). The in-crowd algorithm performs well with a range of λ provided that λ is high enough to remove noise in the system (see Fig. 6). We expect it to perform well for sparse, real-world, noisy, large-scale BPDN problems; therefore it may be of great usefulness to applications such as overcomplete video codecs and underdetermined model selection. Section VII reveals that alternative BPDN solvers are faster on dense problems, but Section VIII suggests that an alternative subproblem solver that scales more gracefully with larger p may be possible. However, even without modification the in-crowd algorithm is of immediate practical utility on sparse problems, and potentially provides the groundwork for a family of specialized algorithms able to scale well for most BPDN problems with many unknowns.

Appendix A
Error Decay Bound

Here, we will show that $f(\mathbf{x})$ strictly decreases with iteration under the in-crowd algorithm, and for arbitrary[9] $\mathbf{y}$ and $\mathbf{A}$, where $\|A_i\|_2 = 1 \forall i$, this decrease is initially exponential.

Define the *minimum projection property* $\Theta(\mathbf{A})$ of $\mathbf{A}$:

$$\Theta(\mathbf{A}) \equiv \min_{\mathbf{z} \in \operatorname{Span}(A_i), \|\mathbf{z}\|_2 = 1} \left(\max_k |\langle \mathbf{z}, A_k \rangle| \right).$$

[9]$\mathbf{y}$ is trivially restricted to the span of the columns of $\mathbf{A}$, as mentioned in Section I-A

Remark: For all matrices $\mathbf{A} \neq \mathbf{0}$, $\Theta(\mathbf{A}) > 0$.

This is clear by contradiction. If $\Theta(\mathbf{A}) = 0$, then there must be some $\mathbf{z}$ such that $\langle \mathbf{z}, A_i \rangle = 0 \ \forall i$. Such a $\mathbf{z}$ must then lie outside the span of $\{A_i\}$.

Remark: For $N \times N$ orthonormal matrices, $\Theta(\mathbf{A}) = \frac{1}{\sqrt{N}}$, and adding additional columns to any $\mathbf{A}$ can only increase $\Theta(\mathbf{A})$. Finding $\Theta(\mathbf{A})$ in general is a nonconvex problem.

With this definition in place, we have the following theorem on the convergence of the in-crowd algorithm.

Theorem A.1: The error $f(\mathbf{x})$ under the in-crowd algorithm always decreases with iteration, and converges at least exponentially in iteration count for matrices where $\|A_i\|_2 = 1$ while $\|\mathbf{r}\|_2 \geq \frac{2\lambda}{\Theta(\mathbf{A})}$ and $\|\mathbf{r}\|_2^2 \geq 2\lambda \|\mathbf{x}\|_1$.

Proof: At iteration t of the algorithm, suppose that we have a current residual $\mathbf{r}_t = \mathbf{y} - \mathbf{A}\mathbf{x}_t$. For the algorithm to step through an additional iteration, at least one new component must be added to I. This implies that there must exist some nonempty set of indexes i where the usefulness $u_{t+1,i} \equiv |\langle \mathbf{r}_t, A_i \rangle|$ must be larger than λ. Let $k = \operatorname{argmax}_i(u_{t+1,i})$. The optimizer has available to it the candidate solution $\mathbf{x}_t + \epsilon \mathbf{e}_k$ where $\mathbf{e}_k$ is the unit vector in the k direction. Therefore, the optimizer will converge on a solution with error at most equal to $f(\mathbf{x}_t + \epsilon \mathbf{e}_k)$ for the choice of ϵ that minimizes f at the end of iteration $t+1$. Hence,

$$\begin{aligned} f(\mathbf{x}_{t+1}) &\leq f(\mathbf{x}_t + \epsilon \mathbf{e}_k) \\ &\leq \frac{1}{2} \|\mathbf{y} - \mathbf{A}(\mathbf{x}_t + \epsilon \mathbf{e}_k)\|_2{}^2 + \lambda \|\mathbf{x}_t + \epsilon \mathbf{e}_k\|_1. \end{aligned}$$

Using properties of the inner product,

$$\begin{aligned} f(\mathbf{x}_{t+1}) &\leq \frac{1}{2} \|\mathbf{r}_t\|_2{}^2 - \epsilon \langle \mathbf{r}_t, A_k \rangle + \frac{1}{2}\epsilon^2 + \lambda \|\mathbf{x}_t\|_1 + \lambda |\epsilon| \\ &\leq f(\mathbf{x}_t) + \epsilon(\lambda \operatorname{sgn}(\epsilon) - \langle \mathbf{r}_t, A_k \rangle) + \frac{1}{2}\epsilon^2. \end{aligned}$$

Without loss of generality,[10] consider only the case $\epsilon > 0$ and $\langle \mathbf{r}_t, A_k \rangle > \lambda$. Define $\Phi(\epsilon) \equiv \epsilon(\lambda - \langle \mathbf{r}_t, A_k \rangle) + \frac{1}{2}\epsilon^2$; this represents the change to f over a single iteration of the in-crowd algorithm if error can be improved only[11] by modifying x_k. Choosing $\epsilon = (\langle \mathbf{r}_t, A_k \rangle - \lambda)$ makes $\Phi(\epsilon) = -\frac{1}{2}\epsilon^2 < 0$, so $f(\mathbf{x}_{t+1}) \leq \left(f(\mathbf{x}_t) - \frac{1}{2}\epsilon^2\right) < f(\mathbf{x}_t)$, proving the first part of Theorem A.1, that f always decreases with iteration count. Solving for $\Phi(\epsilon') \leq -\frac{\|\mathbf{r}_t\|_2{}^2 \Theta^2(\mathbf{A})}{8}$ is feasible for a range of ϵ' when the following quadratic has real roots:

$$\epsilon'^2 + 2\epsilon'(\lambda - \langle \mathbf{r}_t, A_k \rangle) + \frac{\|\mathbf{r}_t\|_2^2 \Theta^2(\mathbf{A})}{4} = 0.$$

The requirement therefore is that

$$4(\lambda - |\langle \mathbf{r}_t, A_k \rangle|)^2 \geq \|\mathbf{r}_t\|_2^2 \Theta^2(\mathbf{A}).$$

By the fact that the algorithm continued at Step 4), $\langle \mathbf{r}_t, A_k \rangle - \lambda > 0$, so it follows that

$$2(|\langle \mathbf{r}_t, A_k \rangle| - \lambda) \geq \|\mathbf{r}_t\|_2 \Theta(\mathbf{A}).$$

[10]True since $\operatorname{sgn}(\epsilon) = \operatorname{sgn}(\langle \mathbf{r}_t, A_k \rangle)$ so $\epsilon(\lambda \operatorname{sgn}(\epsilon) - \langle \mathbf{r}_t, A_k \rangle)$ is always negative and equal to $-|\epsilon|(|\langle \mathbf{r}_t, A_k \rangle| - \lambda)$.

[11]This is an unlikely worst case scenario for the in-crowd algorithm, only attained when all the elements added to I are almost parallel so that only one is made nonzero, but orthogonal to the existing I such that no co-optimization with existing members of I is possible.

The choice of k implies that $|\langle \mathbf{r}_t, A_k\rangle| \geq |\langle \mathbf{r}_t, A_i\rangle|\ \forall i \neq k$. By the definition of $\Theta(\mathbf{A})$ and since $\|A_k\|_2 = 1$, there exist j such that $|\langle A_j, \mathbf{r}_t\rangle| \geq \Theta(\mathbf{A})\|\mathbf{r}_t\|_2$. It follows that $|\langle \mathbf{r}_t, A_k\rangle| \geq \Theta(\mathbf{A})\|\mathbf{r}_t\|_2$. A stricter requirement on the existence of a real ϵ' is therefore

$$2(\Theta(\mathbf{A})\|\mathbf{r}_t\|_2 - \lambda) \geq \|\mathbf{r}_t\|_2\Theta(\mathbf{A})$$
$$\|\mathbf{r}_t\|_2 \geq \frac{2\lambda}{\Theta(\mathbf{A})}$$

which is the first condition on $\|\mathbf{r}\|_2$ of Theorem A.1. As long as $\|\mathbf{r}_t\|_2 > \frac{2\lambda}{\Theta(\mathbf{A})}$, we can find an ϵ' where $\Phi(\epsilon') = -\frac{\|\mathbf{r}_t\|_2^2\Theta^2(\mathbf{A})}{8}$. Thus,

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{\|\mathbf{r}_t\|_2^2\Theta^2(\mathbf{A})}{8}$$
$$\leq \frac{1}{2}\|\mathbf{r}_t\|_2^2\left(1 - \frac{\Theta^2(\mathbf{A})}{4}\right) + \lambda\|\mathbf{x}_t\|_1.$$

However, we desire a bound showing an exponential decrease of the total error, not just the residual imperfection. Splitting the $\left(1 - \frac{\Theta^2(\mathbf{A})}{4}\right)$ term into two equal portions,

$$f(\mathbf{x}_{t+1}) \leq \frac{1}{2}\|\mathbf{r}_t\|_2^2\left(1 - \frac{\Theta^2(\mathbf{A})}{8}\right) + \lambda\|\mathbf{x}_t\|_1 - \frac{\|\mathbf{r}_t\|_2^2\Theta^2(\mathbf{A})}{16}$$

which implies

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t)\left(1 - \frac{\Theta^2(\mathbf{A})}{8}\right)$$

when $\|\mathbf{r}\|_2{}^2 \geq 2\lambda\|\mathbf{x}\|_1$, which is the second condition on $\|\mathbf{r}\|_2$ of Theorem A.1.

Appendix B
Convergence Criteria for BPDN

Prior work (for example, [41]) has shown that a convex function $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ attains its global minimum at $\mathbf{x}$ if and only if the zero vector $\mathbf{0}$ is an element of the subdifferential of g. For a given λ, the subdifferential of $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1$ is

$$\partial f(\mathbf{x}) = -\mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x}) + \lambda\partial\|\mathbf{x}\|_1. \tag{7}$$

For the l_1 norm, the subdifferential is the set

$$\partial\|\mathbf{x}\|_1 = \begin{cases} & v_i = 1 \quad x_i > 0 \\ \mathbf{v} \in \mathbb{R}^n & v_i = -1 \quad x_i < 0 \\ & v_i \in [-1, 1] \quad x_i = 0. \end{cases}$$

Let I be the support of $\mathbf{x}$. For those indexes $k \in I$, the requirement that $\mathbf{0} \in \partial f(\mathbf{x})$ implies that

$$A_k^T(\mathbf{y} - \mathbf{A}\mathbf{x}) = \langle A_k, \mathbf{r}\rangle = v_k\lambda = \mathrm{sgn}(x_k)\lambda. \tag{8}$$

For those indexes $j \in I^c$, $x_j = 0$, we have that

$$\left|A_j^T(\mathbf{y} - \mathbf{A}\mathbf{x})\right| = |\langle A_j, \mathbf{r}\rangle| = |v_j|\lambda \text{ for some } v_j \in [-1, 1].$$

Equivalently,

$$|\langle A_j, \mathbf{r}\rangle| \leq \lambda. \tag{9}$$

To summarize, there are two necessary and sufficient criteria for the minimizer $\mathbf{x}$ of the function $f(\mathbf{x}) = \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1$. On the support of $\mathbf{x}$, the correlation between the residual and the columns of $\mathbf{A}$ must equal exactly $\mathrm{sgn}(x_k)\lambda$. Off the support, the correlation must have magnitude less than or equal to λ. To show that these two criteria together are equivalent to finding the solution of (3), consider a candidate solution $\mathbf{x}$. Since zero-valued components j have $|\frac{\partial\frac{1}{2}\|\mathbf{r}_t\|_2^2}{\partial x_j}|_{x_j=0} = |\langle \mathbf{r}, A_j\rangle| \leq \lambda$, no change in x_j for any j can result in a lower total error. All nonzero components satisfy $\langle \mathbf{r}, A_k\rangle = \mathrm{sgn}(x_k)\lambda$, thus error is locally stationary. Hence, it is impossible to alter any component x_i of $\mathbf{x}$ to decrease total error. Therefore, $\mathbf{x}$ is a local minimum. Since the problem is convex, the minimum is global.

Acknowledgment

The authors would like to thank W. Xu, M. Wang, and A. K. Tang for their thoughtful comments on the algorithm, the field, and the manuscript. They also would like to thank M. Salman Asif for his timely code contributions, and the two anonymous reviewers for extensive help polishing this work and pointing out items of interests that we initially overlooked. The authors would also like to thank the NIH, who helped fund this work.

References

[1] M. Lustig, D. Donoho, J. Santos, and J. Pauly, "Compressed sensing MRI," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 72–82, 2008.
[2] U. Gamper, P. Boesiger, and S. Kozerke, "Compressed sensing in dynamic MRI," *Magn. Reson. Med.*, vol. 59, no. 2, pp. 365–373, 2008.
[3] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, no. 1, pp. 129–159, 2001.
[4] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Stat. Soc. B (Method.)*, vol. 58, no. 1, pp. 267–288, 1996.
[5] P. Ravikumar, G. Raskutti, M. Wainwright, and B. Yu, "Model selection in Gaussian graphical models: High-dimensional consistency of l1-regularized MLE," *Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 21, pp. 1329–1336, 2008.
[6] J. Bergh and J. Löfström, *Interpolation Spaces : An Introduction*. New York: Springer-Verlag, 1976.
[7] R. Gribonval and M. Nielsen, "Beyond sparsity: Recovering structured representations by minimization and greedy algorithms," *Adv. Comput. Math.*, vol. 28, no. 1, pp. 23–41, 2008.
[8] S. Chen and D. Donoho, *Basis Pursuit*, vol. 1, pp. 41–44, Oct. 1994.
[9] S. Becker, J. Bobin, and E. Candès, "NESTA: A fast and accurate first-order method for sparse recovery," vol. 904, 2009 [Online]. Available: http://arxiv.org/abs/0903.1443
[10] A. Wang, P. Gill, and A. Molnar, "Light field image sensors based on the Talbot effect," *Appl. Opt.*, vol. 48, no. 31, pp. 5897–5905, 2009.
[11] A. Wang, P. R. Gill, and A. Molnar, "Fluorescent imaging and localization with angle sensitive pixel arrays in standard CMOS," presented at the IEEE Sensors Conf., Waikoloa, Big Island, HI, Nov. 3, 2010.
[12] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proc. IEEE*, vol. 98, no. 6, pp. 1031–1044, 2010.
[13] A. Bruckstein, D. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Rev.*, vol. 51, no. 1, pp. 34–81, 2009.
[14] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, 2006.
[15] B. Olshausen and D. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1?," *Vis. Res.*, vol. 37, no. 23, pp. 3311–3325, 1997.

[16] J. Ye and M. van der Schaar, "Fully scalable 3-D overcomplete wavelet video coding using adaptive motion compensated temporal filtering," U.S. Patent App. 10/531,195, Oct. 8, 2003.
[17] Y. Andreopoulos, M. Van der Schaar, A. Munteanu, J. Barbarien, P. Schelkens, and J. Cornelis, "Complete-to-overcomplete discrete wavelet transforms for scalable video coding with MCTF," in *Vis. Commun. Image Process.*, Citeseer, pp. 719–731.
[18] B. Olshausen and D. Field, "Sparse coding of sensory inputs," *Current Opinion Neurobiol.*, vol. 14, no. 4, pp. 481–487, 2004.
[19] D. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.
[20] K. Barthel, H. Cycon, and D. Marpe, "Image denoising using fractal and wavelet-based methods," in *Proc. SPIE*, Citeseer, 2003, vol. 5266, pp. 10–18.
[21] J. Pawley and B. Masters, "Handbook of biological confocal microscopy," *J. Biomed. Optics*, vol. 13, 2008.
[22] A. Diaspro *et al., Confocal and two-photon microscopy: foundations, applications, and advances*. New York: Wiley-Liss, 2002.
[23] D. Needell and J. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comput. Harmon. Anal.*, vol. 26, no. 3, pp. 301–321, 2009.
[24] D. Malioutov, M. Cetin, and A. Willsky, "Homotopy continuation for sparse signal representation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2005, vol. 5.
[25] M. S. Asif and J. K. Romberg, "Dynamic updating for 11 minimization," 2009 [Online]. Available: http://arxiv.org/abs/0903.1443
[26] M. S. Asif, L1 homotopy 2009 [Online]. Available: http://users.ece.gatech.edu/~sasif/homotopy/index.html
[27] E. van den Berg and M. P. Friedlander, "Probing the Pareto frontier for basis pursuit solutions," *SIAM J. Sci. Comput.*, vol. 31, no. 2, pp. 890–912, 2008.
[28] E. van den Berg and M. P. Friedlander, SPGL1: A Solver for Large-Scale Sparse Reconstruction, Jun. 2007 [Online]. Available: http://www.cs.ubc.ca/labs/scl/spg11
[29] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale 11-regularized least squares," *IEEE J. Sel. Topics Signal Process.*, vol. 1, no. 4, pp. 606–617, Dec. 2007.
[30] E. Hale, W. Yin, and Y. Zhang, "Fixed-point continuation applied to compressed sensing: Implementation and numerical experiments," *J. Comput. Math.*, vol. 28, no. 2, pp. 170–194, 2010.
[31] E. Hale, W. Yin, and Y. Zhang, Fixed-Point Continuation (FPC) for Large-Scale Image and Data Processing Applications of 11-Minimization, Jun. 2008 [Online]. Available: http://www.caam.rice.edu/ optimization/L1/fpc/
[32] Z. Wen and W. Yin, FPC as a Matlab Solver for 11-Regularization Problems, Jul. 2010 [Online]. Available: http://www.caam.rice.edu/optimization/L1/FPC_AS/
[33] M. Figueiredo, R. Nowak, and S. Wright, "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *IEEE J. Sel. Topics Signal Process.*, vol. 1, no. 4, pp. 586–597, 2007.
[34] MathWorks, Quadratic Programming: Optimization Algorithms and Examples (Optimization Toolbox), Feb. 2011 [Online]. Available: http://www.mathworks.com/help/toolbox/optim/ug/brnox7l.html#brozzpo
[35] P. Gill, W. Murray, and M. Wright, *Practical Optimization*. New York: Academic, 1981.
[36] A. Wills, SPM: QPC–Quadratic Programming in C–HOME, Jan. 2010 [Online]. Available: http://sigpromu.org/quadprog/
[37] J. Tropp and A. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, p. 4655, 2007.
[38] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Ann. Stat.*, vol. 32, no. 2, pp. 407–451, 2004.
[39] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2230–2249, 2009.
[40] I. Drori and D. Donoho, "Solution of l_1 minimization problems by LARS/homotopy methods," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2006, vol. 3, pp. 636–640.
[41] P. Garrigues and L. Ghaoui, "An homotopy algorithm for the Lasso with online observations," *Adv. Neural Inf. Process. Systems*, vol. 21, pp. 489–496, 2009.

Patrick R. Gill received the B.Sc. (Hons.) degree from the University of Toronto, Canada, in 2001 and the Ph.D. degree from the University of California at Berkeley in 2007.

He is currently a Postdoctoral Researcher in the School of Electrical and Computer Engineering and the Department of Psychology at Cornell University, Ithaca, NY. His research interests are in computational and theoretical neuroscience, with specialization in the fields of sensory neuroscience and neural network behaviors.

Albert Wang (S'07) received the B.A. degree from Harvard University, Cambridge, MA, in 2005. He is currently working towards the Ph.D. degree in electrical engineering at Cornell University, Ithaca, NY.

His research interests include analog and mixed signal circuits and hardware based signal processing.

Alyosha Molnar (S'06–M'07) received the Hon. B.Sc.Eng. degree from Swarthmore College in 1997, the M.Sc. University of California at Berkeley in 2003, and the Ph.D. degree from the University of California at Berkeley in 2007.

From 1998 to 2002, he was with the RF IC Group at Conexant Systems, Inc. (now Skyworks Solutions, Inc.), Newport Beach, CA, where he developed their first-generation GSM direct conversion receiver. He is currently a member of the faculty of the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY.