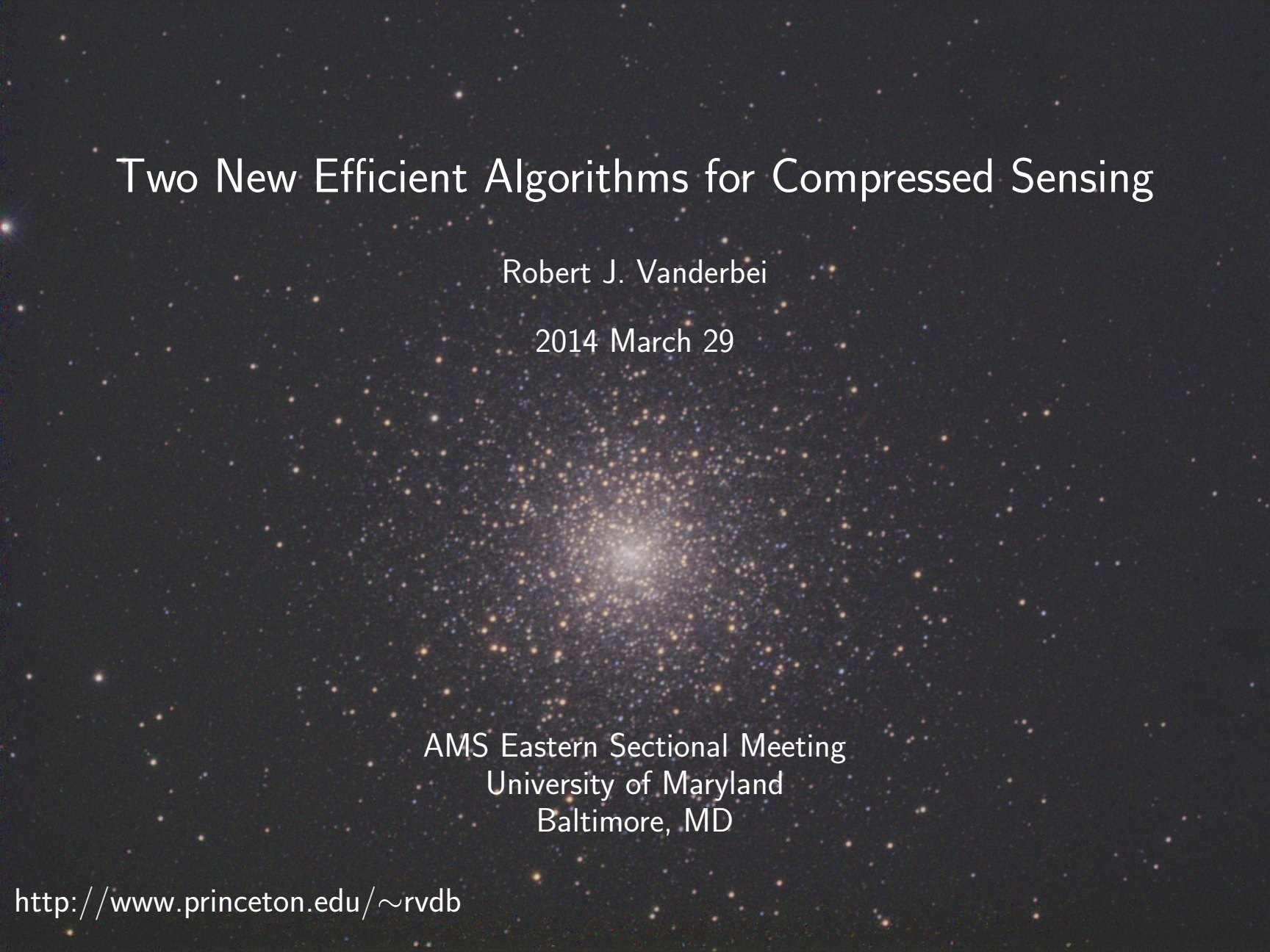


Two New Efficient Algorithms for Compressed Sensing

Robert J. Vanderbei

2014 March 29

AMS Eastern Sectional Meeting
University of Maryland
Baltimore, MD

http://www.princeton.edu/∼rvdb

Compressed Sensing

Compressed sensing aims to recover a sparse signal from a small number of measurements.

Let $\mathbf{x}^0 = (x_1^0, \ldots, x_n^0)^T \in \mathbb{R}^n$ denote a signal to be recovered.

We assume n is large and that $\mathbf{x}^0$ is sparse.

Let $\mathbf{A}$ be a given (or chosen) $m \times n$ matrix with $m \ll n$.

The *compressed sensing problem* is to recover $\mathbf{x}^0$ assuming only that we know $\mathbf{y} = \mathbf{A}\mathbf{x}^0$ and that $\mathbf{x}^0$ is sparse.

Since $\mathbf{x}^0$ is a sparse vector, one can hope that it is the sparsest solution to the underdetermined linear system and therefore can be recovered from $\mathbf{y}$ by solving

$$(\mathrm{P}_0) \qquad \min_{\mathbf{x}} \|\mathbf{x}\|_0 \text{ subject to } \mathbf{A}\mathbf{x} = \mathbf{y},$$

where

$$\|\mathbf{x}^0\|_0 = \#\{i : x_i \neq 0\}.$$

This problem is NP-hard due to the nonconvexity of the 0-pseudo-norm.

Basis Pursuit

Basis pursuit is one way to circumvent NP-hardness.

Replace $\|\mathbf{x}\|_0$ with $\|\mathbf{x}\|_1 = \sum_j |x_j|$:

$$(\mathrm{P}_1) \qquad \min_{\mathbf{x}} \|\mathbf{x}\|_1 \text{ subject to } \mathbf{A}\mathbf{x} = \mathbf{y}.$$

Various conditions exist under which the solutions to (P_0) and (P_1) are unique.

One key question is: under what conditions are the solutions to (P_0) and (P_1) the same?

Various sufficient conditions have been discovered.

Let $\mathbf{A}_{*S}$ denote the submatrix of $\mathbf{A}$ with columns from a subset $S \subset \{1, \ldots, n\}$.

We say that $\mathbf{A}$ has the *k-restricted isometry property (k-RIP)* with constant δ if for any S with cardinality k,

$$(1-\delta)\|\mathbf{v}\|_2^2 \leq \|\mathbf{A}_{*S}\mathbf{v}\|_2^2 \leq (1+\delta)\|\mathbf{v}\|_2^2 \text{ for any } \mathbf{v} \in \mathbb{R}^k,$$

where $\|\mathbf{v}\|_2 = \sqrt{\sum_j v_j^2}$.

Let $\delta_k(\mathbf{A})$ denote the smallest value of δ for which the matrix $\mathbf{A}$ has the k-RIP property.

Theorem. *Let $k = \|\mathbf{x}^0\|_0$. Suppose that $k \ll n$, $\mathbf{A}$ satisfies the k-RIP condition, and $\delta_k(\mathbf{A}) < 1/3$. Then the solutions to (P_0) and (P_1) are the same.*

Two "New" Ideas

The *first idea* is motivated by the fact that the desired solution is *sparse* and therefore should require only a relatively small number of simplex pivots to find, starting from an appropriately chosen starting point—the zero vector.

Using the *parametric simplex method* it is easy to take the zero vector as the starting basic solution.

The *second idea* requires a modified sensing scheme:

> stack the signal vector $\mathbf{x}$ into a matrix $\mathbf{X}$ and then multiply the matrix signal on both the left and the right sides to get a compressed matrix signal.

This idea allows one to formulate the linear programming problem in such a way that the constraint matrix is very sparse and therefore the problem can be solved very efficiently.

An Equivalent Parametric Formulation

Consider the following parametric perturbation to (P_1):

$$\begin{aligned} \widehat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \ & \mu\|\mathbf{x}\|_1 + \|\boldsymbol{\epsilon}\|_1 \\ \text{subject to} \ & \mathbf{A}\mathbf{x} + \boldsymbol{\epsilon} \ = \ \mathbf{y}. \end{aligned} \tag{1}$$

For large values of μ, the optimal solution has $\widehat{\mathbf{x}} = \mathbf{0}$ and $\widehat{\boldsymbol{\epsilon}} = \mathbf{y}$.
For values of μ close to zero, the situation reverses: $\widehat{\boldsymbol{\epsilon}} = \mathbf{0}$.

We refer to this problem as the *vector compressed sensing problem*.

Dealing with Absolute Values

One way to reformulate the optimization problem (1) as a linear programming problem is to split each variable into a difference between two nonnegative variables,

$$\mathbf{x} = \mathbf{x}^+ - \mathbf{x}^- \quad \text{and} \quad \boldsymbol{\epsilon} = \boldsymbol{\epsilon}^+ - \boldsymbol{\epsilon}^-,$$

where the entries of $\mathbf{x}^+, \mathbf{x}^-, \boldsymbol{\epsilon}^+, \boldsymbol{\epsilon}^-$ are all nonnegative.

The next step is to replace $\|\mathbf{x}\|_1$ with $\mathbf{1}^T(\mathbf{x}^+ + \mathbf{x}^-)$ and to make a similar substitution for $\|\boldsymbol{\epsilon}\|_1$.

In general, the sum does not equal the absolute value but it is easy to see that it does at optimality.

Linear Programming Formulation

Here is the reformulated linear programming problem:

$$\begin{aligned} \min_{\mathbf{x}^+,\mathbf{x}^-,\boldsymbol{\epsilon}^+,\boldsymbol{\epsilon}^-} \quad & \mu \mathbf{1}^T(\mathbf{x}^+ + \mathbf{x}^-) + \mathbf{1}^T(\boldsymbol{\epsilon}^+ + \boldsymbol{\epsilon}^-) \\ \text{subject to} \quad & \mathbf{A}(\mathbf{x}^+ - \mathbf{x}^-) + (\boldsymbol{\epsilon}^+ - \boldsymbol{\epsilon}^-) = \mathbf{y} \\ & \mathbf{x}^+, \mathbf{x}^-, \boldsymbol{\epsilon}^+, \boldsymbol{\epsilon}^- \geq 0. \end{aligned}$$

For μ large, the *optimal solution* has $\mathbf{x}^+ = \mathbf{x}^- = 0$, and $\boldsymbol{\epsilon}^+ - \boldsymbol{\epsilon}^- = \mathbf{y}$.

And, given that these latter variables are required to be nonnegative, it follows that

$$\begin{aligned} y_i > 0 \quad &\Longrightarrow \quad \epsilon_i^+ > 0 \text{ and } \epsilon_i^- = 0 \\ y_i < 0 \quad &\Longrightarrow \quad \epsilon_i^+ = 0 \text{ and } \epsilon_i^- > 0 \\ y_i = 0 \quad &\Longrightarrow \quad \epsilon_i^+ = 0 \text{ and } \epsilon_i^- = 0 \end{aligned}$$

With these choices, the solution is feasible for all μ and is optimal for large μ.

Furthermore, declaring the nonzero variables to be *basic* variables and the zero variables to be *nonbasic*, we see that this optimal solution is also a basic solution and can therefore serve as a starting point for the *parametric simplex method*.

Kronecker Compressed Sensing

Unlike the vector compressed sensing problem, Kronecker compressed sensing is used for sensing multidimensional signals (e.g., matrices or tensors).

For example, given a sparse matrix signal $\mathbf{X}^0 \in \mathbb{R}^{n_1 \times n_2}$, we can use two sensing matrices $\mathbf{A} \in \mathbb{R}^{m_1 \times n_1}$ and $\mathbf{B} \in \mathbb{R}^{m_2 \times n_2}$ and try to recover $\mathbf{X}^0$ from knowledge of $\mathbf{Y} = \mathbf{A}\mathbf{X}^0\mathbf{B}^T$ by solving the *Kronecker compressed sensing* problem:

$$(\mathrm{P}_2) \qquad \widehat{\mathbf{X}} = \operatorname{argmin} \|\mathbf{X}\|_1 \ \text{ subject to } \ \mathbf{A}\mathbf{X}\mathbf{B}^T = \mathbf{Y}.$$

When the signal is multidimensional, Kronecker compressed sensing is more natural than classical vector compressed sensing.

Kroneckerizing Vector Problems

Sometimes, even when facing vector signals, it is beneficial to use Kronecker compressed sensing due to its added computational efficiency.

More specifically, even though the target signal is a vector $\mathbf{x}^0 \in \mathbb{R}^n$, if we assume that n can be factored into $n_1 \times n_2$, then we can first stack $\mathbf{x}^0$ into a matrix $\mathbf{X}^0 \in \mathbb{R}^{n_1 \times n_2}$ by putting each length n_1 sub-vector of $\mathbf{x}^0$ into a column of $\mathbf{X}^0$.

We then multiply the matrix signal $\mathbf{X}^0$ on both the left and the right by sensing matrices $\mathbf{A}$ and $\mathbf{B}$ to get a compressed matrix signal $\mathbf{Y}^0$.

We will show that we are able to solve this Kronecker compressed sensing problem much more efficiently than the corresponding vector compressed sensing problem.

Vectorizing the Kroneckerization

Standard LP solvers expect to see a vector of variables, not a matrix.

The naive vectorization goes like this...

Let $\mathbf{x} = \mathrm{vec}(\mathbf{X})$ and $\mathbf{y} = \mathrm{vec}(\mathbf{Y})$, where, as usual, the $\mathrm{vec}(\cdot)$ operator takes a matrix and concatenates its elements column-by-column to build one large column-vector containing all the elements of the matrix.

In terms of $\mathbf{x}$ and $\mathbf{y}$, problem (P_2) can be rewritten as an equivalent *vector compressed sensing* problem:

$$\mathrm{vec}(\widehat{\mathbf{X}}) = \mathrm{argmin} \, \|\mathbf{x}\|_1 \;\; \text{subject to} \;\; \mathbf{U}\mathbf{x} = \mathbf{y},$$

where $\mathbf{U}$ is given by the $(m_1 m_2) \times (n_1 n_2)$ Kronecker product of $\mathbf{A}$ and $\mathbf{B}$:

$$\mathbf{U} = \mathbf{B} \otimes \mathbf{A} = \begin{bmatrix} \mathbf{A}b_{11} & \cdots & \mathbf{A}b_{1n_2} \\ \vdots & \ddots & \vdots \\ \mathbf{A}b_{m_2 1} & \cdots & \mathbf{A}b_{m_2 n_2} \end{bmatrix}.$$

The matrix $\mathbf{U}$ is fully dense. *This is bad.*

Sparsifying the Constraint Matrix

The key to an *efficient* algorithm for solving the linear programming problem associated with the Kronecker sensing problem lies in noting that the dense matrix $\mathbf{U}$ can be factored into a product of two very sparse matrices:

$$\mathbf{U} = \begin{bmatrix} \mathbf{A}b_{11} & \cdots & \mathbf{A}b_{1n_2} \\ \vdots & \ddots & \vdots \\ \mathbf{A}b_{m_2 1} & \cdots & \mathbf{A}b_{m_2 n_2} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{A} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A} \end{bmatrix} \begin{bmatrix} b_{11}\mathbf{I} & b_{12}\mathbf{I} & \cdots & b_{1n_2}\mathbf{I} \\ b_{21}\mathbf{I} & b_{22}\mathbf{I} & \cdots & b_{2n_2}\mathbf{I} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m_2 1}\mathbf{I} & b_{m_2 1}\mathbf{I} & \cdots & b_{m_2 n_2}\mathbf{I} \end{bmatrix} =: \mathbf{VW},$$

where $\mathbf{I}$ denotes an $n_1 \times n_1$ identity matrix and $\mathbf{0}$ denotes an $m_1 \times m_1$ zero matrix.

The constraints on the problem are

$$\mathbf{Ux} + \boldsymbol{\epsilon} = \mathbf{y}.$$

Exploiting the Sparsification

The matrix $\mathbf{U}$ is usually completely dense.

But, it is a product of two very sparse matrices: $\mathbf{V}$ and $\mathbf{W}$.

Hence, introducing some new variables, call them $\mathbf{z}$, we can rewrite the constraints like this:

$$\begin{aligned} \mathbf{z} - \mathbf{W}\mathbf{x} &= 0 \\ \mathbf{V}\mathbf{z} + \boldsymbol{\epsilon} &= \mathbf{y}. \end{aligned}$$

And, as before, we can split $\mathbf{x}$ and $\boldsymbol{\epsilon}$ into a difference between their positive and negative parts to convert the problem to a linear program:

$$\begin{aligned} \min_{\mathbf{x}^+,\mathbf{x}^-,\boldsymbol{\epsilon}^+,\boldsymbol{\epsilon}^-} \quad & \mu\mathbf{1}^T(\mathbf{x}^+ + \mathbf{x}^-) + \mathbf{1}^T(\boldsymbol{\epsilon}^+ + \boldsymbol{\epsilon}^-) \\ \text{subject to} \quad \mathbf{z} - \mathbf{W}(\mathbf{x}^+ - \mathbf{x}^-) &= 0 \\ \mathbf{V}\mathbf{z} + (\boldsymbol{\epsilon}^+ - \boldsymbol{\epsilon}^-) &= \mathbf{y} \\ \mathbf{x}^+, \mathbf{x}^-, \boldsymbol{\epsilon}^+, \boldsymbol{\epsilon}^- &\geq 0. \end{aligned}$$

This formulation has more variables and more constraints.

But, the constraint matrix is *very sparse*.

For linear programming, sparsity of the constraint matrix is the key to algorithm efficiency.

Numerical Results

For the *vector* sensor, we generated random problems using $m = 1{,}122 = 33 \times 34$ and $n = 20{,}022 = 141 \times 142$.

We varied the number of nonzeros k in signal $\mathbf{x}^0$ from 2 to 150.

We solved the straightforward linear programming formulations of these instances using an interior-point solver called LOQO.

We also solved a large number of instances of the parametrically formulated problem using the parametric simplex method as outlined above.

We followed a similar plan for the *Kronecker* sensor.

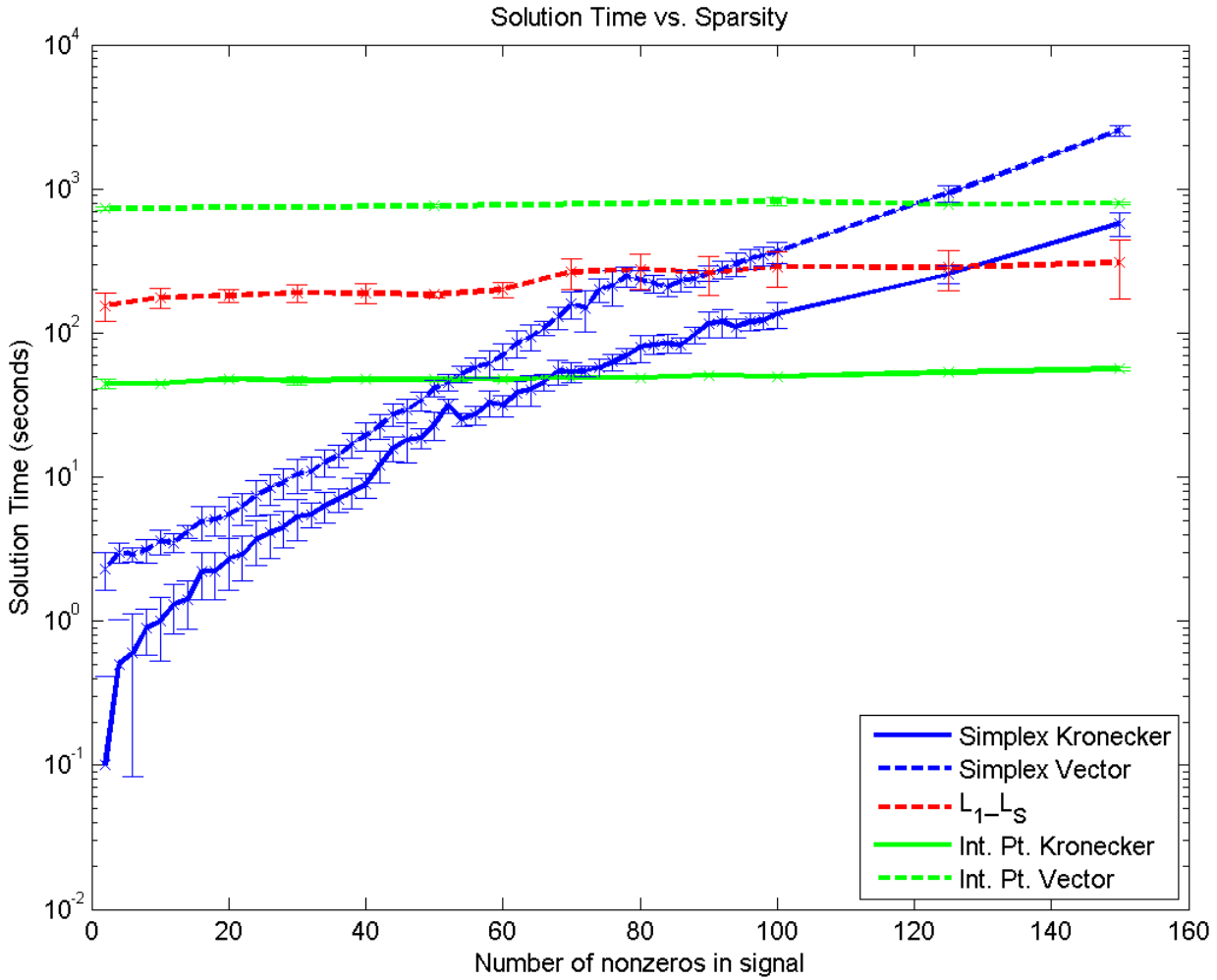
For these problems, we used $m_1 = 33$, $m_2 = 34$, $n_1 = 141$, $n_2 = 142$, and various values of k.

Again, the straightforward linear programming problems were solved by LOQO and the parametrically formulated versions were solved by a custom developed parametric simplex method.

For the Kronecker sensing problems, the matrices $\mathbf{A}$ and $\mathbf{B}$ were generated so that their elements are independent standard Gaussian random variables.

For the vector sensing problems, the corresponding matrix $\mathbf{U}$ was used.

We also ran the publicly-available, state-of-the-art $l_1\_l_s$ code by Boyd et al.



The error bars have lengths equal to one standard deviation based on the multiple trials.

Conclusions

The interior-point solver (LOQO) applied to the Kronecker sensing problem is uniformly faster than both $l_1\_l_s$ and the interior-point solver applied to the vector problem (the three horizontal lines in the plot).

For very sparse problems, the parametric simplex method is best.

In particular, for $k \leq 70$, the parametric simplex method applied to the Kronecker sensing problem is the fastest method.

It can be two or three orders of magnitude faster than $l_1\_l_s$.

But, as explained earlier, the Kronecker sensing problem involves changing the underlying problem being solved.

If one is required to stick with the vector problem, then it too is the best method for $k \leq 80$ after which the $l_1\_l_s$ method wins.

Instructions for downloading and running the various codes/algorithms described herein can be found at:

http://www.orfe.princeton.edu/~rvdb/tex/CTS/kronecker_sim.html

Reference

The work described here is joint work with Han Liu and Kevin Lin.

The paper describing the work can be found at

http://orfe.princeton.edu/~rvdb/tex/CTS/cts_MPC5.pdf

Moral

"The simplex method is 200 times faster than the simplex method." – John Forrest

Thank You!