Fast Blind MIMO Decoding through Vertex Hopping

Jonathan Perlstein*, Thomas Dean†, Mary Wootters*† and Andrea Goldsmith*
*Department of Computer Science, †Department of Electrical Engineering,
Stanford University, Stanford, CA 94305
Email: jrperl@cs.stanford.edu, trdean@stanford.edu, marykw@stanford.edu, andrea@ee.stanford.edu

***Abstract*—We present an algorithm that efficiently performs blind decoding of MIMO signals. That is, given no CSI at either the transmitter or receiver, our algorithm takes a block of samples and returns an estimate of the underlying data symbols. In prior work, the problem of blind decoding was formulated as a non-convex optimization problem. In this work, we present an algorithm that efficiently solves this non-convex problem. This algorithm leverages concepts of linear and mixed-integer linear programming. Empirically, we show that our technique has an error performance close to that of zero-forcing with perfect CSI at the receiver. Initial estimates of the runtime of the algorithm presented in this work suggest that the real-time blind decoding of MIMO signals is possible for even modest sized MIMO systems.**

***Index Terms*—MIMO, Multiuser detection, Blind source separation, Optimization**

I. Introduction

In this work we propose an efficient method to blindly estimate MIMO channels and decode the underlying transmissions. A previous work, [1], has shown that as long as the channel gain matrix is non-singular, the geometry of the constellation can be exploited to recover the underlying data up to some small amount of remaining ambiguity. In [1], the authors formulate the problem of blind MIMO decoding as a non-convex optimization problem and provide theoretical guarantees as to when the interior point algorithm, with a logarithmic barrier function, correctly solves this problem.

As the size of the MIMO system grows, interior-point based algorithms given in [1] become ineffective due to both an increasing proportion of spurious optima as well as numerical instability. In this work, we propose an algorithm inspired by techniques commonly used to solve linear and mixed-integer programs that allow us to perform blind decoding on higher order MIMO systems. More importantly, this approach is far more computationally efficient than the approach in [1], such that real-time decoding is possible.

A blind decoding algorithm that is realistic in terms of both sample size requirements and computational complexity has important applications. Current trends in wireless system design are moving towards systems with shorter wavelengths, higher user mobility and more antennas per user [2]–[4]. Thus, channel state information (CSI) is increasingly rapidly varying and difficult to acquire. Additionally, new systems being proposed demand increased reliability and decreased latency [5]. These factors combined imply that reducing resource overhead from channel estimation will become even more important in the development of future wireless systems.

An efficient blind decoding algorithm can enable high-rate communications in environments where the channel changes too rapidly to be measured or where communication occurs in short bursts. Capacity of channels with CSI unknown at the receiver have been considered extensively in the literature (e.g. [6]–[8] and references therein). While it is theoretically possible to achieve reliable, high-rate communications in these conditions, few schemes exist that do so practically.

A number of previous works have considered blind decoding of MIMO systems, notably [9], [10]. See [1] for a more complete overview of these works. In this work, we compare the performance of our new approach only to [1]. This is because previous approaches have prohibitively large sample size requirements, often growing exponentially in the number of transmit antennas. To our knowledge, [1] is the only blind MIMO decoding method that has sample size requirements that are less than the block length of modern wireless systems.

We now highlight several notable features of the algorithm presented in this work:

- For an $n \times n$ MIMO system we can typically decode a block of k channel uses in $O\left(n^4 k\right)$ operations which is the best-case runtime of our algorithm. We also characterize the scenarios where the number of operations for decoding exceeds this bound.
- At high SNR, given appropriate inputs described in [1], our approach solves the blind decoding problem with a success rate approaching 1 for systems as large as $n = 12$. In comparison, the success rate of the approach in [1] was bounded below 1 beyond $n = 5$ and was 0 beyond $n = 8$.
- We implement the proposed algorithm in the Rust programming language and show that the run time of our algorithm is several orders of magnitude faster than the approach given in [1]. For $n \leq 8$ our implementation is fast enough to enable real-time blind decoding of data streams on the order of several Mbps without specialized hardware. We note that $n \leq 8$ captures nearly all MIMO systems in use today [11], [12].
- At low SNR, our approach has BER performance nearly matching zero-forcing with perfect CSI. At high SNR, our technique appears to have an error floor near 10^{-4}. In all cases, we outperform maximum-likelihood decoding

when the CSI estimate at the receiver has as little as 1% estimation error.

The remainder of this paper is organized as follows. Section II describes the system model and notation used throughout this work. Section III revisits several relevant theoretical results presented in [1]. Section IV provides a high-level description of our new algorithm. Empirical results are presented in Section V and conclusions are offered in Section VI. Any implementation details and optimizations not contained in the work appear in [13].

II. System Model and Notation

In this work we consider $n \times n$ real-valued channel gain matrices, denoted $\mathbf{A}$, where each entry is i.i.d. and normally distributed with zero mean and unit variance, $\mathcal{N}(0,1)$. We assume AWGN, drawn from $\mathcal{N}\left(0,\sigma^2\right)$, is present in the channel. Finally, consider a block fading model where the channel gain matrix is constant over a period of k channel uses after which it is redrawn independently. This work focuses on the transmission of BPSK signals over such channels. [1]

The vector $\mathbf{x} \in \{-1,+1\}^n$ denotes the symbols transmitted in a single channel use and the matrix $\mathbf{X} \in \{-1,+1\}^{n\times k}$ denotes the set of symbols transmitted over a single block of k channel uses. Likewise, single observations and blocks of symbols at the receiver are denoted as $\mathbf{y}$ and $\mathbf{Y}$ respectively.

We assume that *no CSI* is available at either the transmitter or the receiver. Without the aid of pilot symbols or any knowledge of the underlying data symbols, the receiver attempts to recover an estimate of $\mathbf{X}$, denoted $\hat{\mathbf{X}}$. However, as discussed in [1], without additional side information, the receiver is only capable of recovering $\mathbf{X}$ up to an acceptable transform matrix (ATM), meaning that within each block, $\hat{\mathbf{X}}$ is correct up to permutation and negation of the rows. In the high SNR limit, we say that an algorithm solves the blind decoding problem if given only $\mathbf{Y}$ as input, it returns $\hat{\mathbf{X}}$ correctly up to an ATM.

III. Fitting a Parallelepiped

In [1], the authors formulate the blind decoding problem as the following non-convex optimization problem:

$$\underset{\mathbf{U}}{\text{maximize}} \quad \log|\det \mathbf{U}| \tag{1}$$

$$\text{subject to} \quad \|\mathbf{U}\mathbf{y}_i\|_\infty \leq 1 + c\cdot\sigma,\ i = 1,\ldots,k, \tag{2}$$

where c is some margin which is chosen based on the noise variance. The set of optimal $\mathbf{U}$ are equivalent to $\mathbf{A}^{-1}$ up to an ATM. Geometrically, this problem can be interpreted as fitting the minimum volume parallelepiped that matches the observed samples. In [1], the authors show that despite the fact that the problem is not convex, under certain assumptions, gradient descent returns the correct solution with high probability. Here, we briefly recount several important theoretical facts proven in [1] about the problem given by (1)–(2). We initially focus on the noiseless case ($\sigma = 0$), and return to the case $\sigma \neq 0$ in Section IV-D. We also assume that $\mathbf{Y}$ is full rank; if it is not then (1)–(2) is not a well-posed problem.

In (2), each $\mathbf{y}_i$ imposes two linear constraints on each row $\mathbf{u}^{(j)}$, that is $-1 \leq \left\langle \mathbf{u}^{(j)}, \mathbf{y}_i \right\rangle \leq 1$, for all i, j. The feasible region is thus an n^2-dimensional polytope. We say that a given $\mathbf{U}$ is at a vertex of this polytope if $\mathbf{U}\mathbf{Y} \in \{-1,+1\}^{n\times k}$. Note that the objective function is not defined at all vertices of the feasible region; if $\mathbf{U}\mathbf{Y}$ is not full rank, this implies $\mathbf{U}$ is singular and the value of (1) is not defined. If two vertices $\mathbf{U}_1$ and $\mathbf{U}_2$ are adjacent (share an edge of the polytope) then this also implies that the Hamming distance between $\mathbf{U}_1\mathbf{Y}$ and $\mathbf{U}_2\mathbf{Y}$ is 1. We note the following additional facts about the program given in (1)–(2), which are proven in [1]:

- Solutions to the blind decoding problem lie on vertices of the feasible region. When n is such that a Hadamard matrix exists, all optima are strict and lie on vertices. In all cases, optima will only lie on the boundary of the feasible region.
- For $n < 6$, all optima of (1)–(2) are global optima. [1] gives specific values of $\mathbf{X}$ that guarantee all optima of (1)–(2) are solutions to the blind decoding problem.

IV. Algorithm

The program (1)–(2) is not linear, nor is it even convex. One should not necessarily expect tools from convex optimization, let alone linear programming, to work well. However, we adapt such techniques by leveraging two facts about the problem geometry: the fact that the objective function is multilinear in the rows of $\mathbf{U}$, and that solutions to the blind decoding problem lie on vertices of the feasible region.

The first step of our algorithm is finding a value of $\mathbf{U}$ that is at a non-singular vertex of the feasible region. It turns out that finding such a vertex is a non-trivial task and is often the majority of the work in solving the problem. This procedure is described in Section IV-A.

Once we have found an initial non-singular vertex of the feasible region, we efficiently explore neighboring vertices of the feasible region in search of a global optimum. This is accomplished in a similar manner as the simplex algorithm. We form a tableau from the linear constraints that define the feasible region and hop between vertices by performing Gauss-Jordan pivots on this data structure. This procedure is described in greater depth in Section IV-B.

At each step, we hop to the neighboring vertex that has the largest increase in objective function and backtrack if we find a local optimum. We discuss how to identify when we are at a local versus a global optimum in Section IV-C. Local optima are rare and do not exist for $n \leq 5$. In nearly every case the algorithm terminates after at most a few hops; we elaborate on performance and when this is not the case in Section V-A and in greater depth in [13].

[1] While we only consider $n \times n$ real channels, we note that the results in this work can be extended to $n \times n$ complex-valued channels by considering the usual $2n \times 2n$ equivalent real-valued channel gain matrix and can be extended rectangular channels as discussed in [1]. The results in [1] extend to general MPAM constellations; the performance of the algorithms presented here under the presence of higher-order modulation is a topic of on going research.

A. Finding an initial vertex

Linear programs are often solved by the simplex technique which typically starts at the origin. When the origin of a linear program is not feasible, techniques exist to find a suitable feasible solution, often termed a basic feasible solution or BFS [14]. Unfortunately, it is not clear how to leverage standard techniques to solve our problem. In our case, the origin is singular, as are a majority of the vertices of the feasible region, and so in particular the gradient is not defined and we cannot start our simplex technique at these points.

Instead, we describe how to find a suitable BFS for our problem in Algorithm 1. The ℓ_∞ constraints in (2) can be expressed as $n^2 \times 2kn$ linear inequality constraints: let $\mathbf{u} = \text{vec}(\mathbf{U})$, then (2) can be expressed as $\bar{\mathbf{Y}}\mathbf{u} \leq \mathbb{1}$ for some appropriate $\bar{\mathbf{Y}}$. Suppose $\mathbf{U}$ satisfies l constraints with equality. Then we form the $n^2 \times l$ matrix $\mathbf{B}$ by taking the appropriate rows of $\bar{\mathbf{Y}}$.

Algorithm 1 Initialization

Input: An $n \times k$ matrix of received samples $\mathbf{Y}$.

Output: A matrix $\mathbf{U}$ that satisfies at least n^2 linearly independent constraints of (2) with equality.

1: Generate an initial point $\mathbf{U}^{(0)}$ as described in [1]
2: **while** $\mathbf{B}$ is not full rank **do**
3: $\quad\Delta = \left(\mathbf{U}^{-1}\right)^\intercal$.
4: $\quad$Find $\mathbf{N}$, a basis for null($\mathbf{B}$).
5: $\quad$Compute $\Delta = \text{proj}_{\mathbf{N}}\Delta$.
6: $\quad$Find max $t \in \mathbb{R}_+$ such that $\|(\mathbf{U} + t\Delta)\mathbf{Y}\|_\infty = \pm 1$.
7: $\quad$Update $\mathbf{B}$.
8: **end while**
9: **return** $\mathbf{U}$

Algorithm 1 will return a value of $\mathbf{U}$ such that at least n^2 entries of $\mathbf{UY}$ will be equal to ± 1, with at least n of these entries per row. Because the remaining entries of $\hat{\mathbf{X}} = \mathbf{UY}$ are not independent, it is likely that most, if not all, entries equal ± 1. A more concrete discussion on the output of Algorithm 1 is contained in [13].

After running Algorithm 1, the resulting columns of $\hat{\mathbf{X}}$ must be separated into 'good' versus 'bad' columns. Good columns are those that lie in $\{-1, +1\}^n$. We let $\mathbf{Y}_g$ and $\mathbf{Y}_b$ denote the matrices composed of corresponding the good and bad columns of $\mathbf{Y}$ respectively. If $\mathbf{Y}_g$ is not full rank, then we must rerun Algorithm 1 again until a suitable $\hat{\mathbf{X}}$ is obtained. This happens very rarely in the noiseless case. Additional optimizations of Algorithm 1 are discussed in [13].

B. Vertex Hopping

Given an acceptable output of Algorithm 1, we proceed by forming a new program with additional constraints. It is not hard to see (and is proven in [13]) that global optima of the following program will also be global optima of (1)–(2):

$$\begin{aligned} \text{maximize} \quad & |\det \mathbf{U}| && (3) \\ \text{s.t.} \quad & \mathbf{UY}_g = \pm 1 && (4) \\ & \|\mathbf{UY}_b\|_\infty \leq 1. \end{aligned}$$

We can also see that the value of $\mathbf{U}$ returned from Algorithm 1 is a BFS of (3)–(4). This program is effectively a non-linear mixed integer program and we will attempt to optimize it as such. We do so by flipping the signs of individual entries of $\mathbf{UY}_g$ in an attempt to hillclimb towards an optimal value of $\mathbf{U}$ while allowing for backtracking when we reach a local optimum.

We use the tableau data structure, commonly used to implement the simplex algorithm, to efficiently allow us to 'hop' between feasible values of $\mathbf{U}$ and flip a single entry of $\mathbf{UY}_g$. This 'hop' is accomplished by performing a Gauss-Jordan pivot on the appropriate entry of $\mathbf{UY}_g$. A detailed description of the formation of the tableau and the process of pivoting is contained in [13]; however, our approach to constructing a tableau exactly follows the approach in [14].

At each step of the algorithm we have a number of neighboring vertices that can be chosen as a pivot. Each choice will change a single entry of $\mathbf{UY}_g$ and affects only a single row of $\mathbf{U}$. Because the determinant is linear in the rows of $\mathbf{U}$, the resulting change in the objective function is linear. Thus, by considering the gradient, we can easily compute the value of the objective function at adjacent vertices. At each step, we hop to the adjacent vertex that causes the greatest increase in the objective function. After a hop, the gradient of $\mathbf{U}$ can be efficiently updated via the Sherman-Morrison inverse formula [15]. This process is repeated in the manner of a breadth-first search until either we find a global optimum or we have exhausted the search space. We discuss how to detect whether we are at a global versus a local optimum in Section IV-C. This main procedure is described in Algorithm 2.

We note that the additional constraints imposed in (4) may create additional local optima that are not present in (1)–(2). While our solver has the ability to backtrack away from local optima, we do not allow it to visit to vertices where $\det \mathbf{U} = 0$ as the gradient is undefined at these points. It is possible that we may complete a breadth-first search without finding a global optimum. Empirically, the odds that this occurs are dependent on the values of (n, k) and typically occurs in less than 2% of initial choices of $\mathbf{U}$ for $n \leq 8$. When this occurs we simply rerun Algorithm 1 and attempt another search.

C. Stopping Criteria

In this subsection, we consider how to efficiently determine whether a value of $\mathbf{U}$ is at a local optimum or a global optimum. Since we do not know the value of $\mathbf{A}$, the value of the objective function without any additional information is difficult to interpret. However, the determinant of any square matrix with values ± 1 can only take on a discrete set of values (see [16]). As a result, the only values that $|\det \mathbf{U}|$ can take is simply this spectrum scaled by some unknown constant, namely $\det \mathbf{A}^{-1}$. In other words, (1)–(2) is maximized when the following quantity obtains the maximum determinant for any ± 1-valued matrix of dimension n:

$$\begin{aligned} \max_{\tilde{\mathbf{Y}}} \quad & |\det \mathbf{U}\tilde{\mathbf{Y}}| && (5) \\ \text{s.t.} \quad & \text{cols}(\tilde{\mathbf{Y}}) \subseteq \text{cols}(\mathbf{Y}). && (6) \end{aligned}$$

Algorithm 2 Main Proceedure

Input: $\mathbf{U}$ such that $\mathbf{UY}_g$ is full rank.
Output: $\hat{\mathbf{X}}$ or error.

```
Form simplex tableau using procedure in [13].
repeat
    Compute objective function at all neighboring vertices
    if optimum reached then
        if global optimum then
            return UY
        else
            backtrack to first vertex with unvisited neighbor
        end if
    end if
    perform a pivot to the largest unvisited neighbor
    update gradient of objective function
until state space exhausted
return error
```

Solving this problem seems like a difficult combinatorial optimization problem so we do not rely on solving directly. However, when we are at an optimum, we can often determine the value of (5) by inspecting the value of the objective function at neighboring vertices. In fact, in [13], we prove that for $n \leq 6$ the neighbors of a global optimum have a unique signature. We conjecture that this remains true up to $n \leq 10$, meaning that we can easily distinguish between local and global optima in these cases.

D. Adapting to AWGN

So far, we have only considered solving the blind decoding problem in the limit of high SNR. We now turn our attention to how to robustly solve the blind decoding problem in the presence of AWGN. We make no claims that the technique provided here is optimal in terms of BER performance; it almost certainly is not. However, our technique works well empirically and is computationally efficient.

In order to understand how to adapt to noise, we consider the output of Algorithm 1 in more detail. Empirically, for any (n, k), when Algorithm 1 terminates, the vast majority of entries of $\mathbf{UY}$ are contained in the set $\{-1, +1\}$. A small proportion of entries are 0-valued and an even smaller proportion appear uniformly distributed on $(-1, +1)$. When AWGN is present, the behavior of Algorithm 1 changes drastically because the columns of $\mathbf{Y}$ no longer have such strong linear dependence. With near certainty, we will have only n^2 entries that are exactly $\{-1, +1\}$, and, as k grows, this means that $\mathbf{Y}_g$ will almost never be full rank and we will be unable to proceed with Algorithm 2.

To address this issue, we incorporate a simple rounding procedure to Algorithm 1. If entires of $\mathbf{UY}$ are within some tolerance, δ, of $\{-1, 0, 1\}$, we simply adjust $\mathbf{Y}$ to effectively round off the corresponding entry of $\mathbf{UY}$. The optimal value of δ is primarily a function of the SNR of the system which we find empirically and report in [13]. The performance of this approach is treated more thoroughly in [13], and the empirical BER performance is given in Section V.

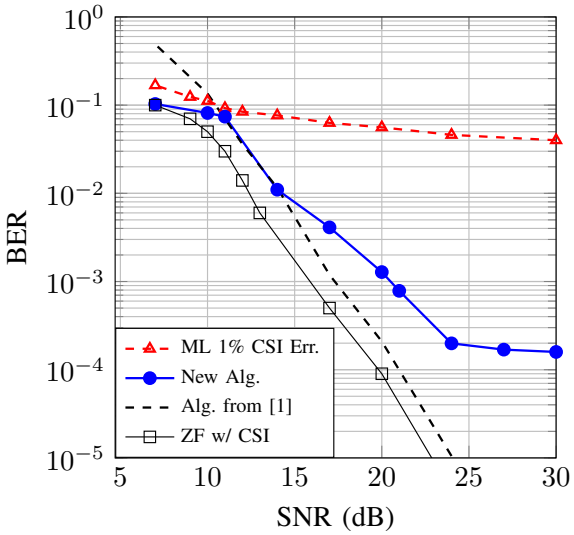

Fig. 1. BER performance of our algorithm with AWGN. Here $n = 4, k = 30$ with i.i.d. Gaussian channels. Also shown is the performance of zero-forcing, assuming perfect CSI, the blind decoding algorithm given in [1], as well as maximum-likelihood decoding assuming a CSI error that is 1% of the noise variance.

TABLE I
RUNTIME ON A SINGLE CORE OF A 2.2GHZ INTEL I7

		New Alg.		Alg. of [1]	
n	k	Success	Time(s)	Success	Time(s)
2	8	0.99	4.71e-5	0.99	3.01e-2
3	13	0.98	2.09e-4	0.99	6.33e-2
4	18	0.97	6.18e-4	0.99	0.13
5	18	0.95	2.71e-3	0.97	0.30
6	22	0.96	1.12e-2	0.93	0.59
8	30	0.97	9.48e-2	0.80	3.5
10	45	0.95	1.72	0	-
12	60	0.94	44.7	0	-

V. NUMERICAL RESULTS

In Figure 1, we show empirical results where we compare our algorithm to zero-forcing with perfect CSI as well as to the algorithm given in [1]. We see that at low SNR, our algorithm nearly matches zero-forcing. At high SNR we encounter an error floor near 10^{-4}. We also compare our algorithm to maximum-likelihood decoding with imperfect CSI. We outperform ML decoding with an estimation error modeled as AWGN with variance as low 1% of the channel noise variance.

In Figure 2 we give the success performance for various values of n in the limit of high SNR. Here, $\mathbf{X}$ is generated uniformly at random, and success a measure of how often a random $\mathbf{X}$ has the correct theoretical guarantees provided in [1]. For $n \leq 5$, the success probability of our algorithm nearly exactly matches that of the algorithm given in [1]. We outperform the method in [1] beyond $n = 5$.

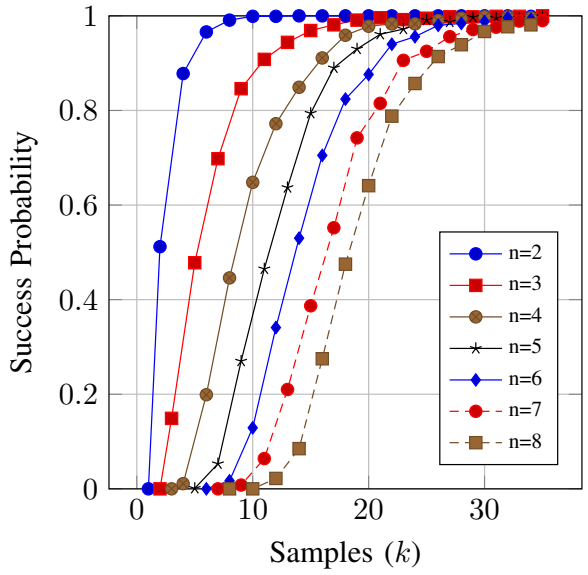

Fig. 2. The success rate of approach $\mathbf{X}$ is drawn uniformly at random for various values of n. Our algorithm succeeds with near 100% success when $\mathbf{X}$ has the theoretical properties to ensure recovery, given in [1].

A. Runtime Performance

Obtaining a expression for the average-case asymptotic time complexity of our algorithm appears difficult. We note, however, that in the best case, Algorithm 1 will directly solve the blind decoding problem. This happens with non-negligible probably for $n \leq 4$. A number of optimizations for each step in Algorithm 1 are presented in [13], where it is shown that Algorithm 1 has a time complexity of $O\left(n^4 k\right)$.

Empirically, for $n \leq 8$, in most cases the solver will require at most one or two pivots before it finds a global optimum. For all values of n, we occasionally encounter a 'trap' case as described in Section IV-B. In these cases the solver enumerates a large subspace before exiting, thus inflating the average runtime. Beyond $n = 8$, the runtime of our algorithm begins to noticeably increase. We note that the spectrum of possible determinants of ± 1-valued matrices grows rapidly as n grows (see [16]) and hence we conjecture that this rapid growth of possible determinants leads to the increased runtime of our algorithm.

We report the runtime of an implementation of our algorithm written in the Rust programming language; details of this implementation are contained in [13]. Table I reports the average time per trial on a single core of an Intel i7 2.2GHz processor. These numbers are preliminary as there are many more optimizations remaining to be implemented in our solver. For comparison, the runtime Algorithm 1 of [1], implemented using MATLAB's `fmincon` solver, is also reported.

VI. Conclusion

We have shown that, for values as large as $n = 8$, we can efficiently and reliably perform blind decoding of BPSK MIMO symbols in the presence of AWGN. The technique presented in this work still works beyond $n = 8$, but the underlying non-convex optimization problem appears to become computationally difficult as n grows. For small n, our technique is both efficient and practical in terms of sample and computational complexity. At low SNR, our technique performs comparably to zero-forcing decoding and at all SNRs outperforms ML decoding with as little as 1% CSI error.

The algorithm presented in this work motivates a suite of future research topics. Many of these topics are related to understanding and improving the performance of our algorithm in a variety of operating conditions. Examples include: developing optimizations to exploit the structure present in complex channel gain matrices, studying the performance of our algorithm under higher modulation orders, and considering square channels. Further, more research is warranted at the wireless-systems level to understand how to best use our algorithm to build high-rate, reliable MIMO systems that operate in environments with rapidly changing CSI.

Acknowledgements

T. Dean is supported by the Fannie and John Hertz Foundation. This work was supported in part under grant NSF-CCF-0939370.

References

[1] T. Dean, M. Wootters, and A. Goldsmith, "Blind joint MIMO estimation and decoding," *IEEE Trans. Inf. Theory*, In review, preliminary version at https://arxiv.org/abs/1802.01049.

[2] J. G. Andrews, S. Buzzi, W. Choi, *et al.*, "What will 5G be?," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, 2014.

[3] F. Boccardi, R. W. Heath, A. Lozano, *et al.*, "Five disruptive technology directions for 5G," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 74–80, 2014.

[4] T. S. Rappaport, S. Sun, R. Mayzus, *et al.*, "Millimeter wave mobile communications for 5G cellular: It will work!," *IEEE Access*, vol. 1, pp. 335–349, 2013.

[5] A. Osseiran, F. Boccardi, V. Braun, *et al.*, "Scenarios for 5G mobile and wireless communications: the vision of the METIS project," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 26–35, 2014.

[6] E. Telatar, "Capacity of multi-antenna gaussian channels," *Trans. on Emerging Telecommunications Technologies*, vol. 10, no. 6, pp. 585–595, 1999.

[7] A. Goldsmith, S. A. Jafar, N. Jindal, and S. Vishwanath, "Capacity limits of MIMO channels," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 5, pp. 684–702, 2003.

[8] L. Zheng and D. N. C. Tse, "Communication on the grassmann manifold: A geometric approach to the noncoherent multiple-antenna channel," *IEEE Trans. Inf. Theory*, vol. 48, no. 2, pp. 359–383, 2002.

[9] S. Talwar, M. Viberg, and A. Paulraj, "Blind separation of synchronous co-channel digital signals using an antenna array. I. algorithms," *IEEE Trans. Signal Process.*, vol. 44, no. 5, pp. 1184–1197, 1996.

[10] L. K. Hansen and G. Xu, "A hyperplane-based algorithm for the digital co-channel communications problem," *IEEE Trans. Inf. Theory*, vol. 43, no. 5, pp. 1536–1548, 1997.

[11] 3GPP, "TS 36.213 evolved universal terrestrial radio access (E-UTRA); physical layer procedures," *ETSI Tech. Rep., 2010*.

[12] IEEE Computer. Society, "Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," *Std 802.11ac-2013*, 2013.

[13] T. Dean, J. Perlstein, M. Wootters, and A. Goldsmith, "Efficient, blind MIMO decoding, extended version," in preparation.

[14] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[15] J. Sherman and W. J. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," *The Annals of Mathematical Statistics*, vol. 21, no. 1, pp. 124–127, 1950.

[16] W. P. Orrick and B. Solomon, "Spectrum of the determinant function," http://www.indiana.edu/∼maxdet/spectrum.html.