Fast and Near–Optimal Matrix Completion via Randomized Basis Pursuit

Zhisu Zhu[*] Anthony Man–Cho So[†] Yinyu Ye[‡]

June 4, 2009

Abstract

Motivated by the philosophy and phenomenal success of compressed sensing, the problem of reconstructing a matrix from a sampling of its entries has attracted much attention recently. Such a problem can be viewed as an information–theoretic variant of the well–studied matrix completion problem, and the main objective is to design an *efficient* algorithm that can reconstruct a matrix by inspecting only a *small* number of its entries. Although this is an impossible task in general, Candès and co–authors have recently shown that under a so–called *incoherence* assumption, a rank r $n \times n$ matrix can be reconstructed using semidefinite programming (SDP) after one inspects $O(nr \log^6 n)$ of its entries. In this paper we propose an alternative approach that is much more efficient and can reconstruct a larger class of matrices by inspecting a significantly smaller number of the entries. Specifically, we first introduce a class of matrices, which we call *stable* matrices, and show that it includes all those that satisfy the incoherence assumption. Then, we propose a randomized basis pursuit (RBP) algorithm and show that it can reconstruct a stable rank r $n \times n$ matrix after inspecting $O(nr \log n)$ of its entries. Our sampling bound is only a logarithmic factor away from the information–theoretic limit and is essentially optimal. Moreover, the runtime of the RBP algorithm is bounded by $O(nr^2 \log n + n^2 r)$, which compares very favorably with the $\Omega(n^4 r^2 \log^{12} n)$ runtime of the SDP–based algorithm. Perhaps more importantly, our algorithm will provide an *exact* reconstruction of the input matrix in *strongly polynomial* time if the matrix entries are rational. By contrast, the SDP–based algorithm can only provide an *approximate* one in polynomial time.

Keywords: Matrix Completion, Stable Matrices, Randomized Basis Pursuit, Randomized Algorithms, Analysis of Algorithms

MSC2000 Subject Classification: 65F30, 68Q25, 68W20

[*]Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA 94305. E–mail: `zhuzhisu@stanford.edu`

[†]Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, N. T., Hong Kong. E–mail: `manchoso@se.cuhk.edu.hk`. This research is supported by Hong Kong Research Grants Council (RGC) General Research Fund (GRF) Project No. CUHK 2150603.

[‡]Department of Management Science and Engineering, Stanford University, Stanford, CA 94305. E–mail: `yinyu-ye@stanford.edu`

1 Introduction

A fundamental problem that arises frequently in many disciplines is that of reconstructing a matrix with certain properties from some partial information. Typically, such a problem is motivated by the desire to deduce global structure from a (small) number of local observations. For instance, consider the following applications:

- **Covariance Estimation.** In areas such as statistics, machine learning and wireless communications, it is often of interest to find the *maximum likelihood estimate* of the covariance matrix $\Sigma \in \mathbb{C}^{m\times m}$ of a random vector $v \in \mathbb{C}^m$. Such an estimate can be used to study the relationship among the variables in v, or to give some indication on the performance of certain systems. Usually, extra information is available to facilitate the estimation. For instance, we may have a number of independent samples that are drawn according to the distribution of v, as well as some structural constraints on Σ (e.g., certain entries of Σ^{-1} have prescribed values [9], the matrix Σ has a Toeplitz structure and some of its entries have prescribed values [11], etc.). Thus, the estimation problem becomes that of completing a partially specified matrix so that the completion satisfies the structural constraints and maximizes certain likelihood function.

- **Graph Realization.** It is a trivial matter to see that given the coordinates of n points in $\mathbb{R}^k$, the distance between any two points can be computed efficiently. However, the inverse problem — given a subset of interpoint distances, find the coordinates of points (called a *realization*) in $\mathbb{R}^k$ (where $k \geq 1$ is fixed) that fit those distances — turns out to be anything but trivial (see, e.g., [33, 35, 34]). Such a problem arises in many different contexts, such as sensor network localization (see, e.g., [1, 36]) and molecular conformation (see, e.g, [18, 8]), and is equivalent to the problem of completing a partially specified matrix to an *Euclidean distance matrix* that has a certain rank (cf. [24, 25]).

- **Recovering Structure from Motion.** A fundamental problem in computer vision is to reconstruct the structure of an object by analyzing its motion over time. This problem, which is known as the *Structure from Motion (SfM) Problem* in the literature, can be formulated as that of finding a low–rank approximation to certain measurement matrix (see, e.g., [7]). However, due to the presence of occlusion or tracking failures, the measurement matrix often has missing entries. When one takes into account such difficulties, the reconstruction problem becomes that of completing a partially specified matrix to one that has a certain rank (see, e.g., [7]).

- **Recommendation Systems.** Although electronic commerce has offered great convenience to customers and merchants alike, it has complicated the task of tracking and predicting customers' preferences. To cope with this problem, various *recommendation systems* have been developed over the years (see, e.g., [15, 32, 10]). Roughly speaking, those systems maintain a matrix of preferences, where the rows correspond to users and columns correspond to items. When an user purchases or browses a subset of the items, she can specify her preferences for those items, and those preferences will then be recorded in the corresponding entries of the matrix. Naturally, if an user has not considered a particular item, then the corresponding entry of the matrix will remain unspecified. Now, in order to predict users' preferences for the unseen items, one will have to complete a partially specified matrix so that the completion maximizes certain performance measure (such as each individual's utility [23]).

Note that in all the examples above, we are forced to take whatever information is given to us. In particular, we cannot, for instance, specify which entries of the unknown matrix to examine. As a result, the reconstruction problem can be ill–posed (e.g., there may not be a unique or even any solution that satisfies the given criteria). This is indeed an important issue. However, we shall not address it in this paper (see, e.g., [20, 24, 19, 25] for related work). Instead, we take a different approach and consider the information–theoretic aspects of the reconstruction problem. Specifically, let $A \in \mathbb{R}^{m\times n}$ be the rank r matrix that we wish to reconstruct. For the sake of simplicity, suppose that r is known. Initially, *no* information about A (other than its rank) is available. However, we are allowed to inspect *any* entry of A and inspect as many entries as we desire in order to complete the reconstruction. Of course, if we inspect all mn entries of A, then we will be able to reconstruct A exactly. Thus, it is natural to ask whether we can inspect only a *small* number of entries and still be able to reconstruct A in an *efficient* manner. Besides being a theoretical curiosity, such a problem does arise in practical applications. For instance, in the sensor network localization setting [36], the aforementioned problem is tantamount to asking which of the pairwise distances are needed in order to guarantee a successful reconstruction of the network topology. It turns out that if the number of required pairwise distances is small, then we will be able to efficiently reconstruct the network topology by performing just a few distance measurements and solving a small semidefinite program (SDP) [40].

To get an idea of what we should aim for, let us first determine the degrees of freedom available in specifying the rank r matrix $A \in \mathbb{R}^{m\times n}$. This will give us a lower bound on the number of entries of A we need to inspect in order to guarantee an exact reconstruction. Towards that end, consider the singular value decomposition (SVD) $A = U\Sigma V^T$, where $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ have orthonormal columns, and $\Sigma \in \mathbb{R}^{r\times r}$ is a diagonal matrix. Clearly, there are r degrees of freedom in specifying Σ. Now, observe that for $i = 1, 2, \ldots, r$, the i–th column of U must be orthogonal to all of the previous $i-1$ columns, and that it must have unit length. Thus, there are $m-i$ degrees of freedom in specifying the i–th column of U, which implies that there are $\sum_{i=1}^{r}(m-i) = r(2m-r-1)/2$ degrees of freedom in specifying U. By the same argument, there are $\sum_{i=1}^{r}(n-i) = r(2n-r-1)/2$ degrees of freedom in specifying V. Hence, we have a total of

$$\Delta \equiv r + \frac{r(2m-r-1)}{2} + \frac{r(2n-r-1)}{2} = r(m+n-r)$$

degrees of freedom in specifying the matrix A. In particular, this implies that we need to inspect at least Δ entries of A, for otherwise there will be infinitely many matrices that are consistent with the observations, and we will not be able to reconstruct A exactly.

Now, a natural question arises whether it is possible to reconstruct A by inspecting just $\Theta(\Delta)$ of its entries. A moment of thought reveals that the answer is no, as the information that is crucial to the reconstruction of A may concentrate in only a few entries. For instance, consider the rank one $n \times n$ matrix $A = e_1 e_1^T$, where $e_1 = (1, 0, \ldots, 0) \in \mathbb{R}^n$. This is a matrix with only one non–zero entry, and it is clear that if we do not inspect that entry, then there is no way we can reconstruct A exactly.

From the above example, we see that our ability to reconstruct A depends not only on the *number* of entries we inspect, but also on *which* entries we inspect and on the *structure* of A. This motivates the following question: are there matrices for which exact reconstruction is possible after inspecting only $\Theta(\Delta)$ of the entries? More generally, is there any tradeoff between the "niceness" of the structure of A and the number of entries we need to inspect in order to reconstruct A?

1.1 Related Work

In a recent work [5], Candès and Recht studied the above questions and proposed a solution that is based on ideas from compressed sensing and convex optimization. They first defined a notion called *coherence*, which can be viewed as a measure of the niceness of a matrix and is motivated by a similar notion in the compressed sensing literature [3]. Informally, a matrix has low coherence if the information that is crucial to its reconstruction is well–spread (cf. the case where $A = e_1 e_1^T$). Then, they proposed the following algorithm for reconstructing any $m \times n$ matrix A:

<u>The Candès–Recht Algorithm</u>

1. Let Γ be a uniformly random subset of $\{1, \ldots, m\} \times \{1, \ldots, n\}$ with given cardinality $|\Gamma| \geq 1$. Inspect the (i, j)–th entry of A if $(i, j) \in \Gamma$, thus obtaining a set of values $\{A_{ij} : (i, j) \in \Gamma\}$.

2. Output an optimal solution to the following optimization problem:

$$\begin{array}{ll} \text{minimize} & \|X\|_* \\ \text{subject to} & X_{ij} = A_{ij} \quad \text{for } (i,j) \in \Gamma \\ & X \in \mathbb{R}^{m \times n} \end{array} \tag{1}$$

   Here, $\|X\|_*$ is the so–called *nuclear norm* of X and is defined as the sum of all the singular values of X.

Candès and Recht showed that if A has low coherence, then whenever $|\Gamma| = \Omega(N^{5/4} r \log N)$, where $N = \max\{m, n\}$ and $r = \text{rank}(A)$, the solution to problem (1) will be unique and equal to A with high probability. In other words, by inspecting $O(N^{5/4} r \log N)$ randomly chosen entries of A and then solving the optimization problem (1), one can reconstruct A exactly with high probability. Note that problem (1) can be formulated as a SDP; see, e.g., [13, Chapter 5]. As such, it can be solved to any desired accuracy in polynomial time (see, e.g., [16, 37]). However, if one uses standard SDP solvers, then the runtime of the Candès–Recht algorithm is at least on the order of $\max\{N^{9/2} r^2 \log^2 N, N^{15/4} r^3 \log^3 N\}$ (see, e.g., [37, 26]), which severely limits its use in practice. Although specialized algorithms are being developed to solve the SDP associated with problem (1) more efficiently (see, e.g., [2, 17, 26, 40]), they either do not have any theoretical time bound, or their runtimes can still be prohibitively high when N is large (at least on the order of $N^{9/2}r^2 \log^2 N$).

Subsequent to the work of Candès and Recht, improvements have been made by various researchers on both the sampling and runtime bounds for the problem. In [22], Keshavan et al. proposed a reconstruction algorithm that is based on the SVD and a certain manifold optimization procedure. They showed that if the input matrix A has low coherence *and* low rank, then by sampling $|\Gamma| = \Omega(Nr \max\{\log N, r\})$ entries of A uniformly at random, their algorithm will produce a sequence of iterates that converges to A with high probability. Note that the sampling complexity of Keshavan et al.'s algorithm is just a polylogarithmic factor away from the information–theoretic minimum Δ and hence is almost optimal. However, their result applies only when the rank of A is bounded above by $N^{1/2}$, and the ratio between the largest and smallest singular values is bounded. Moreover, there is no theoretical time bound for their algorithm. Around the same time, Candès and Tao [6] refined the analysis in [5] and showed that the sampling complexity of the Candès–Recht algorithm can be reduced to $|\Gamma| = \Omega(Nr \log^6 N)$ when the input matrix A has low coherence (but not necessarily low rank). Again, this is just a polylogarithmic factor away from the information–theoretic minimum Δ. However, the runtime of the algorithm remains high (at least on the order of $N^4 r^2 \log^{12} N$).

1.2 Our Contribution

From the above discussion, we see that it is desirable to design a reconstruction algorithm that can work for a large class of matrices and yet still has low sampling and computational complexities. In this paper we make a step towards that goal. Specifically, our contribution is twofold. First, we introduce the notion of *k–stability*, which again can be viewed as a measure of the niceness of a matrix. Roughly speaking, an $m \times n$ matrix is (column) k–stable if its rank remains unchanged after the removal of *any* of its k columns. Intuitively, if a low–rank matrix has high stability (i.e. when k is large), then the information that is crucial to its reconstruction is present in many small subsets of the columns, and hence the matrix should be more amenable to exact reconstruction. As it turns out, the notion of k–stability is related to the so–called *Maximum Distance Separable (MDS) codes* in coding theory [27, Chapter 11]. Moreover, from the above informal definition, we see that k–stability is a *combinatorial* property of matrices, which should be contrasted with the more *analytic* nature of the notion of coherence as defined in [5]. Nevertheless, there is a strong connection between those two notions. More precisely, we show that if a matrix has low coherence, then it must have high stability. Such a connection opens up the possibility of comparing our results to those in [5, 22, 6].

Secondly, we propose a randomized basis pursuit (RBP) algorithm for the reconstruction problem. Our algorithm differs from that of Candès and Recht [5] and Keshavan et al. [22] in three major aspects:

1. We do not sample the matrix *entries* in a uniform fashion. Instead, we first sample the *columns* of the matrix uniformly and then use the sampled columns to guide our sampling

of the appropriate rows. In particular, our sampling strategy is *adaptive* in the sense that it exploits the information accumulated during the sampling process, and hence it can be viewed as some sort of *importance sampling* procedure. By contrast, the sampling strategy used in [5, 22, 6] is neither adaptive nor can be easily extended to one that is. We note that our column sampling strategy is reminiscent of that used for constructing low–rank approximations to a given matrix [14, 12, 31]. However, there is one crucial difference, namely, our strategy does not require any a priori knowledge of the input matrix. By contrast, the strategy used in [14, 12, 31] assumes that the norm of each column of the input matrix is known. We also note that in [5], the authors mentioned that if one inspects r rows and r columns of the input matrix (where r is the rank of the input matrix) and if the $r \times r$ sub–matrix formed by the intersection of those inspected rows and columns happens to be invertible, then the input matrix can be exactly reconstructed via a Schur complement–type calculation. However, they did not discuss how such an invertible sub–matrix can be found in an efficient manner.

2. In practice we often need to reconstruct a matrix in which some of the entries are given and the others are missing. In other words, we may not be able to inspect every entry of the matrix we wish. Under such a setting, one can still attempt to reconstruct the entire matrix using the algorithms in [5, 22] (for instance, one can still solve the optimization problem (1) using the given set of entries as data). By contrast, using our algorithm, one may only be able to reconstruct a sub–matrix of the input matrix if the given set of entries does not contain a complete row or column. However, it should be emphasized that unless the given set of entries constitutes a sample from the required distribution and its cardinality exceeds the required threshold, there is *no* guarantee that the algorithms in [5, 22] will produce the correct reconstruction of the matrix.

3. Our algorithm does not involve any optimization procedure and will produce an *exact* reconstruction in *strongly polynomial* time if the entries of the input matrix are rational. This should be contrasted with (i) the SDP–based algorithm of Candès and Recht [5], which can only return an *approximate* solution in polynomial time (see [30] for discussions on the complexity of solving SDPs); and (ii) the spectral method of Keshavan et al. [22], which is known to *converge* to an exact solution but has no theoretical time bound.

Regarding the performance of our algorithm, we show that if the input matrix A has high stability (in particular, this includes the case where A has low coherence), then by sampling $O(Nr \log N)$ entries of A using our row–column sampling procedure, we can reconstruct A exactly with high probability. Furthermore, we show that the runtime of our algorithm is bounded above by $O(Nr^2 \log N + N^2 r)$. Thus, on both the sampling and computational complexities, our bounds yield substantial improvement over those in [22, 6]. Moreover, our sampling bound is essentially optimal, as the extra $\log N$ factor can be attributed to the coupon collecting phenomenon [28, Chapter 3] (see [5, 22, 6] for related discussions).

1.3 Outline of the Paper

In Section 2 we first introduce the notion of a k–stable matrix and derive some of its properties. Then, we study the relationship between the notion of k–stability and the notion of coherence defined in [5]. Afterwards, we study some constructions of k–stable matrices and show that they are in fact quite ubiquitous. In Section 3 we propose a randomized basis pursuit (RBP) algorithm for the matrix reconstruction problem and analyze its sampling and computational complexities. Then, we consider some variants of the RBP algorithm that could be of practical interest. Finally, we summarize our results and discuss some possible future directions in Section 4.

2 The Class of k–Stable Matrices

As mentioned in the Introduction, our ability to reconstruct a matrix depends in part on its structure. In this paper we shall focus on the class of k–stable matrices, which is defined as follows:

Definition 1 *A rank r matrix $A \in \mathbb{R}^{m\times n}$ is said to be k–*stable *for some $k \in \{0, 1, \ldots, n-r\}$ if*

1. *every $m \times (n-k)$ sub–matrix of A has rank r; and*
2. *there exists an $m \times (n-k-1)$ sub–matrix of A with rank equal to $r-1$.*

In other words, the rank of a k–stable matrix A remains unchanged after removing any of its k columns. We use $\mathcal{M}^{m\times n}(k,r) \subset \mathbb{R}^{m\times n}$ to denote the set of all k–stable rank r $m \times n$ matrices, and use $\mathcal{M}^{m\times n}(k) \subset \mathbb{R}^{m\times n}$ to denote the set of all k–stable $m \times n$ matrices.

Note that the notion of k–stability is defined with respect to the columns of a matrix. Of course, we may also define it with respect to the rows. However, unlike the notions of row rank and column rank — which are equivalent — a column k–stable matrix may not be row k–stable. Unless otherwise stated, we shall refer to a *column* k–stable matrix simply as a k–stable matrix in the sequel.

As we shall see, the notion of k–stability has many nice properties. For instance, it generalizes the notions of coherence defined in [5, 6]. Moreover, a matrix with high stability (i.e. $k = \Theta(n)$) can be reconstructed by a simple and efficient randomized algorithm with high probability. Before we give the details of these results, let us first take a look at some (deterministic) constructions of k–stable matrices.

Example 1 *Let $k \geq 0$ be an integer, and let $a = (a_1, \ldots, a_n) \in \mathbb{R}^n$ be any vector with k zeroes. Consider the $m \times n$ matrix A whose first row is equal to a^T and all other entries are zeroes. Then, A is an $(n-k-1)$–stable rank one matrix, as there is always at least one non–zero column left after removing $n-k-1$ columns from A.*

Example 2 *Let m, n be integers with $n \geq m \geq 1$. Suppose that $u = (u_1, \ldots, u_m) \in \mathbb{R}^m$ and $v = (v_1, \ldots, v_n) \in \mathbb{R}^n$ are given. Consider the $m \times n$ matrix A defined by $A_{ij} = (u_i + v_j)^{-1}$, where $1 \leq i \leq m$ and $1 \leq j \leq n$. The matrix A is known as a* Cauchy *matrix and plays an important role in the construction of Maximum Distance Separable (MDS) codes [27, Chapter 11]. It is well–known that if the u_i's are distinct, the v_j's are distinct, and $u_i + v_j \neq 0$ for all $1 \leq i \leq m$ and $1 \leq j \leq n$, then every square sub–matrix of A is non–singular. In particular, this implies that A is an $(n-m)$–stable rank m matrix.*

Example 3 *Consider the $m \times n$ grid $\mathcal{G} = \{(i,j) \in \mathbb{R}^2 : i = 1, \ldots, m;\ j = 1, \ldots, n\}$ in $\mathbb{R}^2$, with $m \geq n \geq 2$. Suppose that we want to select a small subset of points from $\mathcal{G}$ so that together they span $\mathbb{R}^2$. This is an instance of the anchor selection problem in sensor network localization [40], where the goal is to select a small subset of points from a given set $\mathcal{S}$ of $l \geq d+1$ points in $\mathbb{R}^d$, so that together they span the ambient space $\mathbb{R}^d$. Note that in general, one cannot simply select any $d+1$ points from $\mathcal{S}$, as the points in $\mathcal{S}$ may not be in general position. Now, for the grid $\mathcal{G}$, observe that each of its rows and columns contains at most m points. Thus, any $m+1$ points from $\mathcal{G}$ will span $\mathbb{R}^2$, or equivalently, the rank two $2 \times mn$ matrix A defined by:*

$$A = \begin{bmatrix} 1 & 1 & \cdots & 1 & 2 & \cdots & m & \cdots & m \\ 1 & 2 & \cdots & n & 1 & \cdots & 1 & \cdots & n \end{bmatrix}$$

is $(mn - m - 1)$–stable.

Example 4 *Let $f_1, \ldots, f_m : \mathbb{R}^n \to \mathbb{R}$ be functions of the form:*

$$f_i(x) = g_i(x_{\pi(1)} + \cdots + x_{\pi(k)}, x_{\pi(k+1)} + \cdots + x_{\pi(n)}) \quad \text{for } i = 1, \ldots, m$$

where π is a permutation on $\{1, \ldots, n\}$, and $g_1, \ldots, g_m : \mathbb{R}^2 \to \mathbb{R}$ are some differentiable functions. Suppose that $g_1, \ldots, g_m$ are known, but π is unknown. Now, consider the problem of estimating the Jacobian matrix $J = [(\partial f_i / \partial x_j)_{ij}] \in \mathbb{R}^{m \times n}$. By the chain rule, we have:

$$\frac{\partial f_i}{\partial x_j} = \begin{cases} \dfrac{\partial g_i}{\partial(x_{\pi(1)} + \cdots + x_{\pi(k)})} & \text{if } j \in \{\pi(1), \ldots, \pi(k)\} \\ \dfrac{\partial g_i}{\partial(x_{\pi(k+1)} + \cdots + x_{\pi(n)})} & \text{if } j \in \{\pi(k+1), \ldots, \pi(n)\} \end{cases}$$

Thus, the Jacobian matrix J has rank at most two and is at least $(\min\{k, n-k\} - 1)$–stable. Note that in estimating J, it is natural to first compute $\partial f_i / \partial x_j$ with respect to some fixed $j = 1, \ldots, n$ for all $i = 1, \ldots, m$. This corresponds to the column sampling strategy we mentioned earlier.

Example 5 *Let $p^1, p^2, \ldots, p^l \in \mathbb{R}^n$ be linearly independent vectors such that $p^i \geq \mathbf{0}$ and $e^T p^i = 1$ for $i = 1, \ldots, l$. Let $P \in \mathbb{R}^{n \times ln}$ be the column stochastic matrix whose columns consist of n copies each of the vectors $p^1, \ldots, p^l$. In the context of the Markov Decision Problem, one is interested in finding a set of basis columns in P [39]. Again, one cannot simply select any n columns from P, as they may be linearly dependent. However, observe that the rank l matrix P is $(n-1)$–stable, implying that any $(l-1)n + 1$ columns of P must contain l basis columns.*

2.1 Relation to the Notion of Coherence

In all previous work on the matrix reconstruction problem [5, 22, 6], the notion of coherence is used to measure the niceness of a matrix. This immediately raises the question of whether k–stability and coherence are comparable notions. It turns out that the former can be viewed as a generalization of the latter. Before we formalize this statement, let us first recall the definition of coherence [5]:

Definition 2 *Let $U \subset \mathbb{R}^n$ be a subspace of dimension $r \geq 1$, and let P_U be the orthogonal projection onto U. Then, the* coherence *of U is defined as:*

$$\mu(U) \equiv \frac{n}{r} \max_{1 \leq i \leq n} \|P_U e_i\|_2^2$$

where $e_i \in \mathbb{R}^n$ is the i–th standard basis vector, for $i = 1, \ldots, n$.

A simple consequence of this definition is the following:

Proposition 1 *Let $U \in \mathbb{R}^{n \times r}$ be a matrix with orthonormal columns. When viewed as a subspace of $\mathbb{R}^n$ (i.e. the subspace spanned by the columns of U), the coherence of U is given by:*

$$\mu(U) = \frac{n}{r} \max_{1 \leq i \leq n} \|u_i\|_2^2$$

where u_i is the i–th row of U, for $i = 1, \ldots, n$.

Proof Let $v_1, \ldots, v_r \in \mathbb{R}^n$ be the columns of U. Since U has orthonormal columns, we have:

$$P_U e_i = \sum_{j=1}^{r} (e_i^T v_j) v_j \quad \text{for } i = 1, \ldots, n$$

whence:

$$\|P_U e_i\|_2^2 = \sum_{j=1}^{r} (e_i^T v_j)^2 = \|u_i\|_2^2$$

as desired. □

The following invariance property of k–stable matrices will be useful for establishing the relationship between k–stability and coherence:

Proposition 2 *Let $A \in \mathbb{R}^{m \times n}$ be an arbitrary matrix, and let $U \in \mathbb{R}^{p \times m}$ be a matrix with linearly independent columns (in particular, we have $p \geq m$). Then, for any $k \in \{0, 1, \ldots, n-r\}$, A is a k–stable rank r matrix iff UA is so.*

Proof Let $a_1, \ldots, a_n \in \mathbb{R}^m$ be the columns of A. Then, the columns of UA are given by $Ua_1, \ldots, Ua_n \in \mathbb{R}^p$. Now, for any $l = 1, \ldots, n$, the number of linearly independent vectors in the collection $\{a_{i_1}, \ldots, a_{i_l}\}$ is the same as that in the collection $\{Ua_{i_1}, \ldots, Ua_{i_l}\}$, since U has full column rank. This completes the proof. □

We are now ready to state our first main result:

Theorem 1 *Let $A \in \mathbb{R}^{m\times n}$ be a rank r matrix whose SVD is given by $A = U\Sigma V^T$, where $U \in \mathbb{R}^{m\times r}$, $V \in \mathbb{R}^{n\times r}$ and $\Sigma \in \mathbb{R}^{r\times r}$. For any non–negative integer $k \le n - r$, if the coherence of V satisfies $\mu(V) \le \mu_0$ for some $\mu_0 \in (0, \frac{n}{kr})$, then A is column s–stable for some $s \ge k$. Similarly, for any non–negative integer $k' \le m - r$, if $\mu(U) \le \mu_0$ for some $\mu_0 \in (0, \frac{m}{k'r})$, then A is row s–stable for some $s \ge k'$.*

Proof By Proposition 2, it suffices to show that V^T is a column k–stable rank r matrix, since $U\Sigma \in \mathbb{R}^{m\times r}$ is a matrix with linearly independent columns. Now, consider the following cases:

Case 1: $r = 1$

Let $V = (v_1, \dots, v_n) \in \mathbb{R}^n$. Then, by Proposition 1 and the fact that $\mu(V) \le \mu_0$, we have:

$$\mu(V) = n \max_{1\le i\le n} v_i^2 \le \mu_0 < \frac{n}{k}$$

It follows that $v_i^2 < 1/k$ for $i = 1, \dots, n$. Since $\sum_{i=1}^n v_i^2 = 1$, we conclude that V must have at least $k + 1$ non–zero entries. It follows that V^T is a column s–stable rank one matrix for some $s \ge k$, since the removal of any k columns from V^T does not change its rank.

Case 2: $r \ge 2$

Suppose that V^T is only an l–stable rank r matrix for some $0 \le l \le k - 1$. Then, by definition, there exist $l + 1$ columns whose removal will result in a rank $r - 1$ sub–matrix of V^T. Without loss of generality, suppose that those $l + 1$ columns are the last $l + 1$ columns of V^T. Then, we may write $V^T = [RQ\ \ N]$, where $R \in \mathbb{R}^{r\times(r-1)}, Q \in \mathbb{R}^{(r-1)\times(n-l-1)}, N \in \mathbb{R}^{r\times(l+1)}$, and Q has orthonormal rows. Since V^T has orthonormal rows, we have:

$$I_r = V^T V = RR^T + NN^T = [R\ \ N][R\ \ N]^T$$

which means that the matrix $[R\ \ N] \in \mathbb{R}^{r\times(r+l)}$ also has orthonormal rows. In particular, we have $\|R\|_F^2 \le r - 1$ (here, $\|\cdot\|_F$ is the Frobenius norm). Moreover, since $\|[R\ \ N]\|_F^2 = r$, we have:

$$\|N\|_F^2 = \|[R\ \ N]\|_F^2 - \|R\|_F^2 \ge 1 \tag{2}$$

On the other hand, observe that:

$$\|N\|_F^2 \le \frac{(l+1)r\mu_0}{n} \le \frac{kr\mu_0}{n} < 1$$

which contradicts (2). Thus, we conclude that V^T (and hence A) is a column s–stable rank r matrix for some $s \ge k$.

The statement about the row stability of A can be established by considering $A^T = V\Sigma U^T$ and following the above argument. □

One of the consequences of Theorem 1 is that if both the factors U and V in the SVD of A have small coherence relative to $\min\{m, n\}/r$ (which is the case of interest in the work [5, 22, 6]), then A has high row and column stability. Now, a natural question arises whether the converse also holds. Curiously, as the following proposition shows, the answer is no.

Proposition 3 *Let $k \in \{1, \ldots, n-1\}$ be arbitrary. Then, there exist $n \times n$ rank one matrices A that are both row and column k–stable, and yet the corresponding SVDs $A = \sigma uv^T$ satisfy $\min\{\mu(u), \mu(v)\} = \Theta(n)$.*

Proof Let $\epsilon \in (0, 1/2)$ be fixed. Define $u \in \mathbb{R}^n$ as follows:

$$u_i = \begin{cases} \sqrt{1-\epsilon} & \text{if } i = 1 \\ \sqrt{\epsilon/k} & \text{if } 2 \le i \le k+1 \\ 0 & \text{otherwise} \end{cases}$$

By construction, we have $\|u\|_2^2 = 1$. Now, consider the rank one matrix $A = uu^T$. Since u^T has $k+1$ non–zero entries, it is column k–stable. Thus, by Proposition 2 and the symmetry of A, we conclude that A is both row and column k–stable. On the other hand, using Proposition 1, we compute:

$$\mu(u) = n \max_{1 \le i \le n} u_i^2 = (1-\epsilon)n$$

This completes the proof. □

As can be seen in the proof of Proposition 3, the coherence of a matrix can be very sensitive to the actual values in its entries. This can be partly attributed to the fact that coherence is an analytic notion. By contrast, the notion of k–stability is more combinatorial in nature and hence is not as sensitive to those values.

Theorem 1 and Proposition 3 together show that the notion of k–stability can be regarded as a generalization of the notions of coherence defined in [5, 6]. In particular, various constructions of low–coherence matrices proposed in [5, 6] can be transferred to the high–stable case. However, it would be nice to have some more direct constructions of high–stable matrices. In the next section, we will show that matrices with high stability are actually quite ubiquitous.

2.2 Ubiquity of k–Stable Matrices

Let $A \in \mathbb{R}^{r \times n}$ be a matrix with full row rank (in particular, we have $r \le n$). Then, it is clear that the maximum stability of A is $n - r$, and that the maximum can be attained. It turns out that such a situation is typical. More precisely, we have the following:

Theorem 2 *Let r, n be integers with $n \ge r \ge 1$. Then, the set $\mathcal{S} \equiv \mathbb{R}^{r \times n} \backslash \mathcal{M}^{r \times n}(n-r, r)$ has Lebesgue measure zero when considered as a subset of $\mathbb{R}^{rn}$.*

The proof of Theorem 2 relies on the following well–known result:

Proposition 4 *Let $f : \mathbb{R}^l \to \mathbb{R}$ be a polynomial function that is not identically equal to zero. Then, the solution set:*

$$f^{-1}(0) \equiv \{x \in \mathbb{R}^l : f(x) = 0\}$$

has Lebesgue measure zero.

A proof of Proposition 4 can be found in [29].

Proof of Theorem 2 Suppose that $A \in \mathbb{R}^{r \times n}$ is not $(n-r)$–stable. Then, one of the $r \times r$ sub–matrices of A must be singular, or equivalently, has determinant zero. Since the determinant of a square matrix is a polynomial function of its entries, and since there are only finitely many $r \times r$ sub–matrices of A, it follows from Proposition 4 that $\mathcal{S}$ has Lebesgue measure zero. $\square$

Thus, by taking a generic $r \times n$ matrix R and an arbitrary $m \times r$ matrix Q whose columns are linearly independent, we may conclude from Proposition 2 and Theorem 2 that the $m \times n$ matrix $A = QR$ has rank r and is column $(n-r)$–stable.

In [5] the authors considered an alternative construction of rank r matrices using the so–called *random orthogonal model*. In that model, one constructs an $m \times n$ matrix A via $A = U\Sigma V^T$, where $V \in \mathbb{R}^{n \times n}$ is a random orthogonal matrix drawn according to the Haar measure on the orthogonal group $\mathcal{O}(n)$, $U \in \mathbb{R}^{m \times m}$ is an arbitrary orthogonal matrix, and $\Sigma \in \mathbb{R}^{m \times n}$ is an arbitrary matrix with the partition:

$$\Sigma = \begin{bmatrix} \Sigma_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} : \Sigma_r \in \mathbb{R}^{r \times r} \text{ diagonal}, \; (\Sigma_r)_{ii} \neq 0 \text{ for } i = 1, \ldots, r$$

By construction, the matrix A has rank r. Now, we claim that A is column $(n-r)$–stable with probability one (with respect to the Haar measure on $\mathcal{O}(n)$). To see this, observe that $A = U_r \Sigma_r V_r^T$, where:

$$U = \begin{bmatrix} U_r & \bar{U}_r \end{bmatrix}, \quad V = \begin{bmatrix} V_r & \bar{V}_r \end{bmatrix}$$

and $U_r \in \mathbb{R}^{m \times r}$, $V_r \in \mathbb{R}^{n \times r}$. Since $U_r \Sigma_r$ has linearly independent columns, by Proposition 2, it suffices to show that V_r^T is column $(n-r)$–stable with probability one. Towards that end, it suffices to show that with probability one, every $r \times r$ sub–matrix of V_r^T has non–zero determinant. It turns out that the last statement is well–known; see, e.g., [21, Lemma 2.2]. Thus, we have proven the following:

Theorem 3 *Let A be a rank r $m \times n$ matrix generated according to the random orthogonal model. Then, A is column $(n-r)$–stable with probability one (with respect to the Haar measure on $\mathcal{O}(n)$).*

In [5] it is shown that the coherence of a rank r $n \times n$ matrix generated according to the random orthogonal model is bounded by $O(\bar{r}/r)$ with probability $1 - o(1)$, where $\bar{r} = \max\{r, \log n\}$. Using Theorem 1, we see that such a matrix is column $(n/\bar{r})$–stable with probability $1 - o(1)$. This should be contrasted with the conclusion of Theorem 3, which is much stronger.

3 The Randomized Basis Pursuit (RBP) Algorithm

Let us now consider the algorithmic aspects of matrix reconstruction, particularly those that are related to the reconstruction of low–rank high–stable matrices. As briefly discussed in the Introduction, if a reconstruction algorithm can only inspect a small number of entries, then it should somehow inspect those that contain the most information. Of course, since there is no a priori information on the input matrix, every algorithm must at some point make a guess at which entries are important. Currently, the best algorithms for the reconstruction problem all pursue an entry–wise uniform sampling strategy [5, 22, 6]. Specifically, they all begin by sampling a uniformly random subset of the entries and inspecting the values in those entries. Such a strategy will certainly perform well when the information that is crucial to the reconstruction is well–spread, but could also fail miserably when those information is highly concentrated. As an illustration, consider the rank one $m \times n$ matrix A from Example 1, which has the form:

$$A = \begin{bmatrix} a^T \\ \mathbf{0} \end{bmatrix} \tag{3}$$

Suppose that $a \in \mathbb{R}^n$ has no zero component. Then, it is clear that there is no hope of reconstructing A if we do not inspect all the entries in its first row. However, if the entry–wise uniform sampling strategy is used, then the probability that l randomly sampled entries of A will include all the entries in the first row is bounded above by:

$$\frac{\begin{pmatrix} mn-n \\ l-n \end{pmatrix}}{\begin{pmatrix} mn \\ l \end{pmatrix}} = \frac{l(l-1)\cdots(l-n+1)}{mn(mn-1)\cdots(mn-n+1)} \le \left(\frac{l}{mn}\right)^n$$

In particular, *no* algorithm that uses the entry–wise uniform sampling strategy will be able to reconstruct A with probability larger than e^{-1} even after sampling $l = mn - m = \Theta(mn)$ of its entries!

The above example shows that the entry–wise uniform sampling strategy may miss the critical structure of a matrix if that structure is localized. On the other hand, observe that the matrix A in the above example can be exactly reconstructed once we inspect its first row and *any* of its columns. In general, one may think of a low–rank matrix as being largely determined by a small number of its rows and columns. Such an intuition motivates the following matrix reconstruction algorithm. Note that the algorithm requires the knowledge of the rank of the input matrix. However, as we shall see in Section 3.3.2, such an assumption can be removed if we can bound the stability of the input matrix.

Randomized Basis Pursuit (RBP) Algorithm

Input: A rank r $m \times n$ matrix A, where r is known.

1. Initialization: Initialize $S \leftarrow \emptyset$ and $T \leftarrow \{1, \ldots, n\}$. The set S will be used to store the column indices that correspond to the recovered basis columns of A.

2. Basis Pursuit Step:

   (a) If $T = \emptyset$, then stop. All the columns of A have been examined, and hence A can be reconstructed directly.

(b) Otherwise, let j be drawn from T uniformly at random, and let $u_j \in \mathbb{R}^m$ be the corresponding column of A. Examine all the entries in u_j. If u_j is spanned by the columns whose indices belong to S, then repeat Step 2b. Otherwise, update:

$$S \leftarrow S \cup \{j\}, \quad T \leftarrow T \backslash \{j\}$$

since u_j is a new basis column. Now, if $|S| = r$, then proceed to Step 3. Otherwise, repeat Step 2.

3. Row Identification: Let A_S be the $m \times r$ sub–matrix of A whose columns are those indexed by S. Find r linearly independent rows in A_S. Let $\bar{S}$ be the corresponding set of row indices, and let $A_{\bar{S},S}$ be the corresponding $r \times r$ matrix.

4. Reconstruction: Examine all the entries in the i–th row of A for all $i \in \bar{S}$. Now, the j–th column of A (where $j \notin S$) can be expressed as a linear combination of the basis columns indexed by S, where the coefficients are obtained by expressing the vector $(a_{ij})_{i \in \bar{S}} \in \mathbb{R}^{|\bar{S}|}$ as a linear combination of the columns of $A_{\bar{S},S}$.

It is not hard to show that when the above algorithm terminates, it will produce an exact reconstruction of the input matrix. To illustrate the flow of the algorithm, let us consider again the rank one matrix A from Example 1 with $k = 0$ (see (3)). Since the first row of A has no zero component, any column selected in Step 2b of the algorithm can be the basis column. Now, suppose that j is the index of the selected column. After inspecting all the entries in the j–th column, the algorithm will identify the 1×1 sub–matrix A_{1j} in Step 3, since A_{1j} is the only non–zero entry in the j–th column. Consequently, the algorithm will examine all the entries in the first row of A in Step 4, thus obtaining all the information that is necessary for the reconstruction of A. Note that in this example, the total number of entries inspected by the algorithm is $m + n - 1$, which is exactly equal to the information–theoretic minimum.

From the description of the RBP algorithm, we see that if the input matrix is of low rank but has many candidate basis columns, then the basis pursuit step will terminate sooner, and hence the number of entries inspected by the algorithm will also be lower. This is indeed the case when the input matrix has high stability. Before we proceed with a formal analysis, let us remark that some additional speed up of the above algorithm is possible. For instance, in Step 2b, once we determine that a column lies in the span of those indexed by S, then we do not need to consider it anymore and hence its index can be removed from T. However, in order to facilitate the analysis, we shall focus on the version presented above.

3.1 Sampling Complexity of the RBP Algorithm

In this section we analyze the sampling complexity of the RBP algorithm. Specifically, our goal is to prove the following:

Theorem 4 *Suppose that the input rank r $m \times n$ matrix A to the RBP algorithm is k–stable for some $k \in \{0, 1, \ldots, n - r\}$, i.e. $A \in \mathcal{M}^{m \times n}(k, r)$. Let $\delta \in (0, 1)$ be given. Then, with probability* *at least $1-r\delta$, the RBP algorithm will terminate with an exact reconstruction of A, and the total number of entries inspected by the algorithm is bounded above by $nr+(k+1)^{-1}mnr(1+\ln(1/\delta))$.*

To gain some intuition on the sampling complexity of the RBP algorithm, observe that the basis pursuit step is simply trying to collect r linearly independent columns from the input matrix. Thus, the problem of determining the number of basis pursuit steps needed in order to obtain r linearly independent columns can be viewed as a coupon collector's problem [28, Chapter 3]. In particular, we expect that the number of basis pursuit steps needed, and hence the number of columns inspected, is on the order of $r\log r$. This is consistent with the estimate stated in Theorem 4.

Before we proceed to prove Theorem 4, let us establish the following simple estimate:

Proposition 5 *Let X be a geometric random variable with parameter $p\in(0,1)$. Then, for any $\delta>0$, we have:*

$$\Pr\left(X>\frac{1+\delta}{p}\right)\le e^{-\delta}$$

Proof We compute:

$$\begin{aligned}\Pr\left(X>\frac{1+\delta}{p}\right) &\le \sum_{j=\lceil(1+\delta)/p\rceil}^{\infty} p(1-p)^{j-1} \\ &= p\cdot(1-p)^{\lceil(1+\delta)/p\rceil-1}\cdot\sum_{j=0}^{\infty}(1-p)^j \\ &\le (1-p)^{\delta/p} \\ &\le e^{-\delta}\end{aligned}$$

This completes the proof. □

Proof of Theorem 4 Observe that Step 2 of the RBP algorithm is the only place where randomization is used, and that once Step 2 is completed, the algorithm will always terminate with an exact reconstruction of A. Thus, it suffices to obtain a high probability bound on the number of times Step 2b is being executed throughout the entire course of the algorithm. Towards that end, let us divide the execution of Step 2 into epochs, where the i–th epoch (for $i=0,1,\ldots,r-1$) begins at the iteration where $|S|=i$ for the first time and ends at the iteration just before the one where $|S|=i+1$. Let p_i be the probability that the column selected in an iteration of the i–th epoch is a basis column. Note that p_i is a random variable that depends on which i basis columns are selected in the previous i epochs. However, since the input matrix is assumed to be k–stable, we have:

$$p_i\ge\frac{k+1}{n-i}\qquad\text{for } i=0,1,\ldots,r-1$$

Now, let Y_i be the number of times Step 2b is being executed in the i–th epoch. Then, Y_i is a geometric random variable with parameter p_i, and the number of times Step 2b is being executed throughout the entire course of the algorithm is given by:

$$Y = \sum_{i=0}^{r-1} Y_i$$

By Proposition 5, we have:

$$\Pr\left(Y > \sum_{i=0}^{r-1} \frac{1+\ln(1/\delta)}{p_i}\right) \le \sum_{i=0}^{r-1} \Pr\left(Y_i > \frac{1+\ln(1/\delta)}{p_i}\right) \le r\delta$$

It follows that with probability at least $1 - r\delta$, the total number of times Step 2b is being executed is bounded above by:

$$\begin{aligned} \sum_{i=0}^{r-1} \frac{1+\ln(1/\delta)}{p_i} &\le \left(1 + \ln\frac{1}{\delta}\right) \sum_{i=0}^{r-1} \frac{n-i}{k+1} \\ &= \frac{r(2n-r+1)}{2(k+1)} \left(1 + \ln\frac{1}{\delta}\right) \\ &\le \frac{nr}{k+1} \left(1 + \ln\frac{1}{\delta}\right) \end{aligned}$$

Note that the above quantity is also an upper bound on the number of *distinct* columns examined by the algorithm. Thus, we see that with probability at least $1 - r\delta$, the total number of entries inspected by the algorithm is bounded above by:

$$nr + \frac{mnr}{k+1}\left(1 + \ln\frac{1}{\delta}\right)$$

and the proof is completed. □

Upon combining the results of Theorem 3 and Theorem 4, we see that if A is a rank r $m \times n$ matrix generated according to the random orthogonal model, then with probability at least $1 - O(n^{-3})$, the RBP algorithm will terminate with an exact reconstruction of A, and the total number of entries inspected by the algorithm is bounded above by $O(n + m \log n)$ when $r = O(1)$, and by $O(n \log n + m \log^2 n)$ when $r = O(\log n)$. This is already a significant improvement over the results in [22, 6]. However, there is an even simpler and more sample–efficient *deterministic* algorithm for reconstructing such matrices. Specifically, recall that by Theorem 3, a rank r $m \times n$ matrix A that is generated according to the random orthogonal model is column $(n - r)$–stable with probability one. In particular, with probability one, the first r columns of such matrices are linearly independent. Thus, we can simply take $S = \{1, \dots, r\}$ and proceed directly to Step 3 of the RBP algorithm. Note that the algorithm we just described is completely deterministic.

Moreover, the total number of entries inspected by the algorithm is $mr+nr-r^2 = r(m+n-r)$, which is exactly equal to the information–theoretic minimum. In fact, the same algorithm can be used to reconstruct *any* column $(n-r)$–stable rank r $m \times n$ matrix. The above observations give the following result, which substantially improves those in [22, 6, 38]:

Proposition 6 *There exists a deterministic matrix reconstruction algorithm with the following property: if the input rank r $m \times n$ matrix A is column $(n-r)$–stable, then the algorithm will terminate with an exact reconstruction of A, and the total number of entries inspected by the algorithm is exactly $r(m+n-r)$.*

3.2 Implementation and Complexity Analysis of the RBP Algorithm

In this section we discuss some of the implementation details of the RBP algorithm and analyze its computational complexity. Clearly, the initialization step can be done using $O(n)$ operations. For the basis pursuit step, we need to determine whether a newly selected column is in the span of the current basis columns (i.e. those indexed by S). This can be achieved via a Gram–Schmidt type process. Specifically, we maintain a set $\mathcal{U}$ of orthonormal vectors with the following property: during the i–th epoch (where $i = 0, 1, \ldots, r-1$), the set $\mathcal{U}$ will contain i orthonormal vectors $w_1, \ldots, w_i \in \mathbb{R}^m$, whose span is equal to that of the columns indexed by S. Now, suppose that the algorithm selects the column $v \in \mathbb{R}^m$. To test whether $v \in \text{span}\{w_1, \ldots, w_i\}$, we first compute:

$$\Pi_i(v) \equiv \sum_{l=1}^{i} (w_l^T v) w_l \in \mathbb{R}^m$$

and then check whether $\Pi_i(v) = v$ (we set $\Pi_0(v) = \mathbf{0}$). If this is the case, then we have $v \in \text{span}\{w_1, \ldots, w_i\}$, whence we can proceed to select another column. Otherwise, v is a new basis column. Thus, we add its index to the set S and add the unit vector $(v-\Pi_i(v))/\|v-\Pi_i(v)\|_2$ to the set $\mathcal{U}$ before continuing to the next instruction.

To determine the time needed by the basis pursuit step, observe that during the i–th epoch, each selected column requires $O(im)$ operations. Since $i \le r-1$, by Theorem 4, we conclude that with probability $1 - r\delta$, the total number of operations executed in the basis pursuit step is bounded by $O((k+1)^{-1} mnr^2 \log(1/\delta))$.

In the row identification step, we need to find r linearly independent rows in the $m \times r$ matrix A_S. This can be achieved by a Gram–Schmidt type process similar to the one outlined above, and the total number of operations required is bounded by $O(mr^2)$.

Finally, in order to carry out the reconstruction step, we can first compute the inverse of the non–singular $r \times r$ matrix $A_{\bar{S},S}$ using $O(r^3)$ operations. Then, for each $j \not\in S$, we can express the vector $(a_{ij})_{i \in \bar{S}} \in \mathbb{R}^{|\bar{S}|}$ as a linear combination of the columns of $A_{\bar{S},S}$ using $O(r^2)$ operations. Afterwards, the j–th column can be reconstructed using $O(mr)$ operations. Since we need to reconstruct at most $n-1$ columns, it follows that the total number of operations required in the reconstruction step is bounded by $O(r^3 + nr^2 + mnr) = O(mnr)$.

To summarize, we have the following:

Theorem 5 *Suppose that the input rank r $m \times n$ matrix A to the RBP algorithm is k–stable for some $k \in \{0, 1, \ldots, n-r\}$, i.e. $A \in \mathcal{M}^{m\times n}(k, r)$. Let $\delta \in (0, 1)$ be given. Then, with probability at least $1 - r\delta$, the total number of operations performed by the RBP algorithm is bounded by $O((k+1)^{-1}mnr^2 \log(1/\delta) + mnr)$.*

Note that Theorem 5 only gives a *probabilistic* bound on the runtime. However, the bound can be made deterministic by suitably modifying the RBP algorithm. Specifically, we can add a counter to keep track of the total number of times Step 2b is being executed. Once that number exceeds $(k+1)^{-1}nr(1 + \ln(1/\delta))$, we stop the algorithm and declare failure. With such modification, the conclusion of Theorem 4 still holds. However, we now have a *deterministic* bound of $O((k+1)^{-1}mnr^2 \log(1/\delta) + mnr)$ on the runtime. We remark that such an idea can also be used to develop a "rank–free" version of the RBP algorithm, i.e. one that does not require the knowledge of the rank of the input matrix. We refer the reader to Section 3.3.2 for details.

The time bound obtained in Theorem 5 compares very favorably with that for the SDP–based algorithm of Candès and Recht [5]. Perhaps more importantly, our algorithm will produce an *exact* reconstruction of the input matrix in *strongly polynomial* time. By contrast, the Candès–Recht algorithm can only produce an *approximate* reconstruction in polynomial time. This is due to the fact that SDPs can only be solved to a fixed level of accuracy in polynomial time. We refer the reader to [30] for further discussion on this issue.

3.3 Variants of the RBP Algorithm

3.3.1 A Non–Adaptive RBP Algorithm

Observe that the RBP algorithm is adaptive in the sense that it takes into account the information contained in the current set of inspected entries before deciding which entries to inspect next. Sometimes, however, it may be desirable to have a non–adaptive reconstruction algorithm, in which the set of entries to be inspected can be fixed *a priori* (e.g., the algorithms in [5, 22] have such a feature). It turns out that a straightforward modification of the original RBP algorithm would yield such an algorithm. Indeed, instead of sampling adaptively in the basis pursuit step, we can simply sample uniformly with replacement $O((k+1)^{-1}nr \log(1/\delta))$ rows and $O((k+1)^{-1}nr \log(1/\delta))$ columns of the input matrix. If the input matrix is both row and column k–stable, then upon following the argument in the proof of Theorem 4, one can show that with probability at least $1 - O(r\delta)$, the intersection of the sampled rows and columns will contain a $r \times r$ invertible sub–matrix. In particular, this implies that the reconstruction step will produce an exact reconstruction of the input matrix. Note that both the sampling and computational complexities of this non–adaptive RBP algorithm are of the same order as that of the original RBP algorithm, and hence there is not much loss in performance by sampling in a non–adaptive fashion.

3.3.2 A Rank–Free RBP Algorithm

Recall that the RBP algorithm introduced earlier assumes that the rank of the input matrix is known. However, in practice, there is very little a priori information on the input matrix. This raises the question of whether one can design a reconstruction algorithm that does not need the rank information. It turns out that this is possible if we modify the RBP algorithm using the idea mentioned at the end of the last sub–section. Specifically, we keep track of the number of attempts made by the algorithm to find the next basis column. If that number reaches a pre–specified threshold, say Λ, then we exit the basis pursuit step and proceed to the row identification step of the algorithm. The idea is that if Λ is sufficiently large and the algorithm fails to find a new basis column after Λ drawings, then it probably has found all the basis columns and hence the input matrix can be exactly reconstructed. To formalize this idea, let us first give a precise description of the proposed algorithm.

Rank–Free Randomized Basis Pursuit (RF–RBP) Algorithm

Input: An $m \times n$ matrix A, stopping threshold $\Lambda \geq 1$.

1. Initialization: Initialize $S \leftarrow \emptyset$, $T \leftarrow \{1, \dots, n\}$ and $\kappa \leftarrow 0$. The set S will be used to store the column indices that correspond to the recovered basis columns of A. The counter κ will be used to keep track of the number of attempts made to find the next basis column.

2. Basis Pursuit Step:

   (a) If $T = \emptyset$, then stop. All the columns of A have been examined, and hence A can be reconstructed directly. Otherwise, reset $\kappa \leftarrow 0$ and proceed to Step 2b.

   (b) Let j be drawn from T uniformly at random, and let $u_j \in \mathbb{R}^m$ be the corresponding column of A. Examine all the entries in u_j.

   If u_j is spanned by the columns whose indices belong to S, then increment $\kappa \leftarrow \kappa + 1$. If $\kappa \geq \Lambda$, then proceed to Step 3. Otherwise, repeat Step 2b.

   If u_j is not spanned by the columns whose indices belong to S, then u_j is a new basis column. Update:

$$S \leftarrow S \cup \{j\}, \quad T \leftarrow T \backslash \{j\}$$

   and repeat Step 2.

3. Row Identification: Let A_S be the $m \times |S|$ sub–matrix of A whose columns are those indexed by S. Find $|S|$ linearly independent rows in A_S. Let $\bar{S}$ be the corresponding set of row indices, and let $A_{\bar{S},S}$ be the corresponding $|S| \times |S|$ matrix.

4. Reconstruction: Examine all the entries in the i–th row of A for all $i \in \bar{S}$. Now, express the j–th column of A (where $j \notin S$) as a linear combination of the basis columns indexed by S, where the coefficients are obtained by expressing the vector $(a_{ij})_{i \in \bar{S}} \in \mathbb{R}^{|\bar{S}|}$ as a linear combination of the columns of $A_{\bar{S},S}$.

Again, we are interested in the sampling complexity of the RF–RBP algorithm. It turns out that if the input matrix is known to be k–stable for some $k \geq 0$, then the sampling complexity of the RF–RBP algorithm is comparable to that of the RBP algorithm. Specifically, we prove the following:

Theorem 6 *Suppose that the input $m \times n$ matrix A to the RF–RBP algorithm is k–stable for some $k \geq k_0$, i.e. $A \in \mathcal{M}^{m\times n}(k)$. Let $\delta \in (0,1)$ be given, and set:*

$$\Lambda = \log\left(\frac{\delta}{\min\{m,n\}}\right) \Big/ \log\left(1 - \frac{k_0+1}{n}\right) \tag{4}$$

Then, with probability at least $1-\delta$, the RF–RBP algorithm will terminate with an exact reconstruction of A, and the total number of entries inspected by the algorithm is bounded above by $nr + m(r+1)\Lambda$, where $r = rank(A)$.

Remarks

1. Since $\log(1-(k_0+1)/n) \leq -(k_0+1)/n$, Theorem 6 guarantees that the total number of entries inspected by the RF–RBP algorithm is bounded by:

$$nr + \frac{mn(r+1)}{k_0+1}\log\left(\frac{\min\{m,n\}}{\delta}\right)$$

   In particular, when δ is inversely proportional to a polynomial in $\min\{m,n\}$, the bound above is of the same order as that obtained for the RBP algorithm (see Theorem 4).

2. To appreciate the power of Theorem 6, consider an $m \times n$ matrix A whose rank r is known to be much smaller than $\min\{m,n\}$, say, $r \leq n/2$. If A is generic, then by Theorem 2, it is k–stable, where $k = n - r \geq n/2$. Hence, by Theorem 6, the matrix A can be exactly reconstructed by the RF–RBP algorithm with high probability, and the number of inspected entries is bounded by $O(nr + mr\log n)$. Note that such a reconstruction is done without the algorithm knowing the exact value of r or k. By contrast, the algorithm of Keshavan et al. [22] is much less flexible, as it needs to know the exact value of r in order to guarantee an exact reconstruction.

Proof of Theorem 6 For $j = 1,2,\ldots,r$, let q_j be the probability that the RF–RBP algorithm finds at least j basis columns before proceeding to Step 3. We claim that:

$$q_j \geq \prod_{i=1}^{j}\left[1 - \left(1 - \frac{k+1}{n-i+1}\right)^{\Lambda}\right] \qquad \text{for } j = 1,\ldots,r \tag{5}$$

The proof of (5) will proceed by induction on j. To facilitate the proof, let us again divide the execution of Step 2 into epochs, where the $(j-1)$–st epoch (for $j = 1,2,\ldots,r$) is defined in exactly the same way as in the proof of Theorem 4. Furthermore, let p_j be the probability that the column selected in an iteration of the $(j-1)$–st epoch is a basis column. Since A is assumed to be k–stable, we have:

$$p_j \geq \frac{k+1}{n-j+1} \qquad \text{for } i = 1, 2, \ldots, r$$

Now, for $j = 1$, we have:

$$q_1 = 1 - (1-p_1)^{\Lambda} \geq 1 - \left(1 - \frac{k+1}{n}\right)^{\Lambda}$$

and hence the base case holds. Suppose that (5) holds for some $j < r$. Then, conditioned on the event that the RF–RBP algorithm finds at least j basis columns before proceeding to Step 3, the probability that the RF–RBP algorithm finds at least $j+1$ basis columns before proceeding to Step 3 is given by:

$$q_{j+1}^{cond} = 1 - (1-p_{j+1})^{\Lambda} \geq 1 - \left(1 - \frac{k+1}{n-j}\right)^{\Lambda}$$

Hence, it follows from the definition of conditional probability and the inductive hypothesis that:

$$q_{j+1} = q_{j+1}^{cond} \cdot q_j \geq \prod_{i=1}^{j+1} \left[1 - \left(1 - \frac{k+1}{n-i+1}\right)^{\Lambda}\right]$$

This completes the proof of (5).

Now, observe that the RF–RBP algorithm will terminate with an exact reconstruction of A iff it finds r basis columns before proceeding to Step 3. Using (5) and the definition of Λ in (4), we see that the probability of such an event is at least:

$$\begin{aligned}\prod_{i=1}^{r} \left[1 - \left(1 - \frac{k+1}{n-i+1}\right)^{\Lambda}\right] &\geq \left[1 - \left(1 - \frac{k+1}{n}\right)^{\Lambda}\right]^{r} \\ &\geq \left(1 - \frac{\delta}{\min\{m,n\}}\right)^{\min\{m,n\}} \\ &\geq 1 - \delta\end{aligned}$$

Moreover, the number of distinct columns inspected by the algorithm is always bounded above by $(r+1)\Lambda$, which implies that the total number of entries inspected by the algorithm is bounded above by $nr + m(r+1)\Lambda$. This completes the proof of Theorem 6. □

Finally, upon following the proof of Theorem 5, one can easily establish the following complexity result for the RF–RBP algorithm:

Theorem 7 *Given an $m \times n$ matrix A and a stopping threshold $\Lambda \geq 1$, the total number of operations performed by the RF–RBP algorithm before it terminates is bounded by $O(mr^2\Lambda + mnr)$, where $r = rank(A)$.*

We remark that the bound in Theorem 7 holds for *arbitrary* input matrices. In the case where the input matrix has rank r and is k–stable, we can set Λ as in (4) and bound the total number of operations by:

$$O\left(\frac{mnr^2}{k+1}\log\left(\frac{\min\{m,n\}}{\delta}\right)+mnr\right)$$

In particular, when δ is inversely proportional to a polynomial in $\min\{m,n\}$, the bound above is of the same order as that obtained for the RBP algorithm (see Theorem 5).

4 Conclusion

In this paper we proposed a randomized basis pursuit (RBP) algorithm for the matrix reconstruction problem. We introduced the notion of a k–stable matrix and showed that the RBP algorithm can reconstruct a k–stable rank r $n\times n$ matrix with high probability after inspecting $O((k+1)^{-1}n^2r\log n)$ of its entries. In addition, we showed that the runtime of the RBP algorithm is bounded by $O((k+1)^{-1}n^2r^2\log n+n^2r)$. Our results yield substantial improvement over those in existing literature ([5, 22, 6]), in the sense that the RBP algorithm can reconstruct a *larger class of matrices* by inspecting a *smaller number of entries*, and it can do so in a *more efficient* manner. We also considered two variants of the RBP algorithm that could be of practical interest. First, we introduced a non–adaptive version of the RBP algorithm, in which the set of entries to be inspected can be fixed a priori. Secondly, we proposed a rank–free version of the RBP algorithm that can reconstruct a matrix without knowing the exact value of its rank. Both of these variants could offer great flexibility in practical settings.

Although the RBP algorithm assumes that the entries in the sampled rows and columns are known — which may not be the case in practical applications — in practice one can still use, say, the algorithms in [5, 22] to first complete a (small) number of columns of the input matrix, and then use our algorithm to complete the remaining entries. Such a hybrid approach can potentially lead to a significant saving in computational cost.

Finally, it would be interesting to study the tradeoff between the sampling complexity and computational complexity of the matrix reconstruction problem. Another interesting direction would be to extend our techniques to handle the case where the sampled entries are *noisy*. Some recent results along this direction, which are established using the techniques of [5, 6], can be found in [4, 38].

References

[1] P. Biswas and Y. Ye. Semidefinite Programming for Ad Hoc Wireless Sensor Network Localization. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN 2004)*, pages 46–54, 2004.

[2] J.-F. Cai, E. J. Candès, and Z. Shen. A Singular Value Thresholding Algorithm for Matrix Completion. Preprint, 2008.

[3] E. Candès and J. Romberg. Sparsity and Incoherence in Compressive Sampling. *Inverse Problems*, 23(3):969–985, 2007.

[4] E. J. Candès and Y. Plan. Matrix Completion with Noise. Preprint, 2009.

[5] E. J. Candès and B. Recht. Exact Matrix Completion via Convex Optimization. To appear in *Foundations of Computational Mathematics*, 2009.

[6] E. J. Candès and T. Tao. The Power of Convex Relaxation: Near–Optimal Matrix Completion. Preprint, 2009.

[7] P. Chen and D. Suter. Recovering the Missing Components in a Large Noisy Low–Rank Matrix: Application to SFM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1051–1063, 2004.

[8] G. M. Crippen and T. F. Havel. *Distance Geometry and Molecular Conformation*, volume 15 of *Chemometrics Series*. Research Studies Press Ltd., Taunton, Somerset, England, 1988.

[9] J. Dahl, L. Vandenberghe, and V. Roychowdhury. Covariance Selection for Non–Chordal Graphs via Chordal Embedding. *Optimization Methods and Software*, 23(4):501–520, 2008.

[10] S. Debnath, N. Ganguly, and P. Mitra. Feature Weighting in Content Based Recommendation System Using Social Network Analysis. In *Proceedings of the 17th International Conference on World Wide Web (WWW 2008)*, pages 1041–1042, 2008.

[11] F. A. Dietrich. *Robust Signal Processing for Wireless Communications*, volume 2 of *Foundations in Signal Processing, Communications and Networking*. Springer–Verlag, Berlin Heidelberg, 2008.

[12] P. Drineas, R. Kannan, and M. W. Mahoney. Fast Monte Carlo Algorithms for Matrices II: Computing a Low–Rank Approximation to a Matrix. *SIAM Journal on Computing*, 36(1):158–183, 2006.

[13] M. Fazel. *Matrix Rank Minimization with Applications*. PhD thesis, Department of Electrical Engineering, Stanford University, Stanford, CA 94305, 2002.

[14] A. Frieze, R. Kannan, and S. Vempala. Fast Monte–Carlo Algorithms for Finding Low–Rank Approximations. *Journal of the ACM*, 51(6):1025–1041, 2004.

[15] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[16] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer–Verlag, Berlin Heidelberg, second corrected edition, 1993.

[17] J. P. Haldar and D. Hernando. Rank–Constrained Solutions to Linear Matrix Equations Using PowerFactorization. To appear in *IEEE Signal Processing Letters*, 2009.

[18] T. F. Havel and K. Wüthrich. An Evaluation of the Combined Use of Nuclear Magnetic Resonance and Distance Geometry for the Determination of Protein Conformations in Solution. *Journal of Molecular Biology*, 182(2):281–294, 1985.

[19] L. Hogben. Graph Theoretic Methods for Matrix Completion Problems. *Linear Algebra and its Applications*, 328(1–3):161–202, 2001.

[20] C. R. Johnson. Matrix Completion Problems: A Survey. In C. R. Johnson, editor, *Matrix Theory and Applications*, volume 40 of *Proceedings of Symposia in Applied Mathematics*, pages 171–198. American Mathematical Society, Providence, Rhode Island, 1980.

[21] T. Kariya and C. F. J. Wu. On the Nonsingularity of Principal Submatrices of a Random Orthogonal Matrix. *Journal of Statistical Planning and Inference*, 12:353–357, 1985.

[22] R. H. Keshavan, A. Montanari, and S. Oh. Matrix Completion from a Few Entries. Preprint, 2009.

[23] R. Kumar, P. Raghavan, S. Rajagopolan, and A. Tomkins. Recommendation Systems: A Probabilistic Analysis. *Journal of Computer and System Sciences*, 63(1):42–61, 2001.

[24] M. Laurent. A Tour d'Horizon on Positive Semidefinite and Euclidean Distance Matrix Completion Problems. In P. M. Pardalos and H. Wolkowicz, editors, *Topics in Semidefinite and Interior–Point Methods*, volume 18 of *The Fields Institute for Research in Mathematical Sciences, Communications Series*, pages 51–76. American Mathematical Society, Providence, Rhode Island, 1998.

[25] M. Laurent. Matrix Completion Problems. In C. A. Floudas and P. M. Pardalos, editors, *The Encyclopedia of Optimization*, volume 3, pages 221–229. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001.

[26] Z. Liu and L. Vandenberghe. Interior–Point Method for Nuclear Norm Approximation with Application to System Identification. Preprint, 2009.

[27] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error–Correcting Codes*, volume 16 of *North–Holland Mathematical Library.* North–Holland Publishing Company, Amsterdam, The Netherlands, 1977.

[28] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, New York, 1995.

[29] M. Okamoto. Distinctness of the Eigenvalues of a Quadratic Form in a Multivariate Sample. *Annals of Statistics*, 1(4):763–765, 1973.

[30] L. Porkolab and L. Khachiyan. On the Complexity of Semidefinite Programs. *Journal of Global Optimization*, 10(4):351–365, 1997.

[31] M. Rudelson and R. Vershynin. Sampling from Large Matrices: An Approach through Geometric Functional Analysis. *Journal of the ACM*, 54(4):Article 21, 2007.

[32] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item–Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW 2001)*, pages 285–295, 2001.

[33] J. B. Saxe. Embeddability of Weighted Graphs in k–Space is Strongly NP–Hard. In *Proceedings of the 17th Allerton Conference in Communication, Control, and Computing*, pages 480–489, 1979.

[34] A. M.-C. So. *A Semidefinite Programming Approach to the Graph Realization Problem: Theory, Applications and Extensions.* PhD thesis, Computer Science Department, Stanford University, Stanford, CA 94305, 2007.

[35] A. M.-C. So and Y. Ye. A Semidefinite Programming Approach to Tensegrity Theory and Realizability of Graphs. In *Proceedings of the 17th Annual ACM–SIAM Symposium on Discrete Algorithm (SODA 2006)*, pages 766–775, 2006.

[36] A. M.-C. So and Y. Ye. Theory of Semidefinite Programming for Sensor Network Localization. *Mathematical Programming, Series B*, 109(2):367–384, 2007.

[37] M. J. Todd. Semidefinite Optimization. *Acta Numerica*, 10:515–560, 2001.

[38] J. Wright, A. Ganesh, S. Rao, and Y. Ma. Robust Principal Component Analysis: Exact Recovery of Corrupted Low–Rank Matrices via Convex Optimization. Preprint, 2009.

[39] Y. Ye. A New Complexity Result on Solving the Markov Decision Problem. *Mathematics of Operations Research*, 30(3):733–749, 2005.

[40] Z. Zhu, A. M.-C. So, and Y. Ye. Measurement Sparsification and Chordal Decomposition for Sensor Network Localization and Graph Realization. Manuscript, 2009.