Algorithmica (2016) 76:426–444
DOI 10.1007/s00453-015-0042-6

Matrix Sparsification and the Sparse Null Space Problem

Lee-Ad Gottlieb[1] · **Tyler Neylon**[2]

Received: 14 February 2013 / Accepted: 24 July 2015 / Published online: 4 August 2015

Abstract We revisit the matrix problems sparse null space and matrix sparsification, and show that they are equivalent. We then proceed to seek algorithms for these problems: we prove the hardness of approximation of these problems, and also give a powerful tool to extend algorithms and heuristics for sparse approximation theory to these problems.

Keywords Matrix sparsification · Sparse null space · Sparse approximation

1 Introduction

In this paper, we revisit the matrix problems sparse null space and matrix sparsification.

The sparse null space problem was first considered by Pothen in 1984 [31]. The problem asks, given a matrix A, to find a matrix N that is a full null matrix for A—that is, N is full rank and the columns of N span the null space of A. Further, N should be sparse, i.e., contain as few nonzero values as possible. The sparse null space problem is motivated by its use to solve Linear Equality Problems (LEPs) [11]. LEPs arise in the solution of constrained optimization problems via generalized gradient

A preliminary version of this paper appeared in the proceedings of Random-Approx '10.

✉ Lee-Ad Gottlieb
leead@ariel.ac.il

Tyler Neylon
tylerneylon@gmail.com

[1] Ariel University, Ariel, Israel

[2] Bynomial, Inc., Berkeley, CA, USA

descent, augmented Lagrangian, and projected Lagrangian methods. Berry et al. [4] consider the sparse null space problem in the context of the dual variable method for the Navier-Stokes equations, or more generally in the context of null space methods for quadratic programming. Gilbert and Heath [18] noted that among the numerous applications of the sparse null space problem arising in solutions of underdetermined system of linear equations, is the efficient solution to the force method (or flexibility method) for structural analysis, which uses the null space to create multiple linear systems. Finding a sparse null space will decrease the run time and memory required for solving these systems. More recently, it was shown [30,42] that the sparse null space problem can be used to find correlations between small numbers of times series, such as financial stocks. The decision version of the sparse null space problem is known to be NP-Complete [11], and only heuristic solutions have been suggested for the minimization problem [4,11,18].

The matrix sparsification problem is similar to sparse null space. One is given a full rank matrix A, and the task is to find a matrix B with as few nonzero values as possible given the constraint that $B = AX$ for some nonsingular matrix X; that is, A and B must have the same column space. Many fundamental matrix operations are greatly simplified by first sparsifying a matrix (see [14]) and the problem has applications in areas such as machine learning [34] and in discovering cycle bases of graphs [23]. But there seem to be only a small number of heuristics for matrix sparsification ([8] for example), or algorithms under limiting assumptions ([20] considers matrices that satisfy the Haar condition), and in fact these algorithms allow an error term (of the form $||B - AX|| < \epsilon$) incommensurate with our formal statement of matrix sparsification. McCormick [26] established that the decision version of this problem is NP-Complete.

For these two classic problems, we wish to investigate potentials and limits of approximation algorithms both for the general problems and for some variants under simplifying assumptions. To this end, we will need to consider the well-known vector problems min unsatisfy and exact dictionary representation (elsewhere called the sparse approximation or highly nonlinear approximation problem [37]).

The min unsatisfy problem is an intuitive problem on linear equations. Given a system $Ax = b$ of linear equations (where A is an integer $m \times n$ matrix and b is an integer m-vector), the problem is to provide a rational n-vector x; the measure to be minimized is the number of equations not satisfied by $Ax = b$. The term "min unsatisfy" was first coined by Arora et al. [2] in a seminal paper on the hardness of approximation, but they claim that the the NP-Completeness of the decision version of this problem is implicit in a 1978 paper of Johnson and Preparata [21]. Arora et al. demonstrated that it is hard to approximate min unsatisfy to within a factor $2^{\log^{.5-o(1)} n}$ of optimal (under the assumption that NP does not admit a quasi-polynomial time deterministic algorithm). This hardness result holds over $\mathbb{Q}$, and stronger results are known for finite fields [12]. For this problem, Berman and Karpinski [3] gave a randomized $\frac{m}{c \log m}$-approximation algorithm (where c is a constant). There has been a large amount of recent literature devoted to heuristics for min unsatisfy, where it has been presented as a problem of decoding in the presence of errors (see for example [7,9,41]).

The exact dictionary representation problem is the fundamental problem in sparse approximation theory (see [27]). In this problem, we are given a matrix of dictionary vectors D and a target vector s, and the task is to find the smallest set $D' \subset D$ such that a linear combination of the vectors of D' is equal to s. This problem and its variants have been well studied. According to Temlyakov [35], a variant of this problem may be found as early as 1907, in a paper of Schmidt [32]. The decision version of this problem was shown to be NP-Complete by Natarajan [28]. (See [25] for further discussion.)

The field of sparse approximation theory has become exceedingly popular in the last decade. For example, it was the subject of the Sparse and Low Rank Approximation workshop at the Banff International Research Station in 2011, as well as the Sparse Representation and Low-rank Approximation workshop at NIPS'11. The applications of sparse approximation theory include signal representation and recovery [10,29], amplitude optimization [33] and function approximation [28]. When the dictionary vectors are Fourier coefficients, this problem is a classic problem in Fourier analysis, with applications in data compression, feature extraction, locating approximate periods and similar data mining problems [6,16,17,43]. There is a host of results for exact dictionary representation (see for example [7]) though all are heuristics or approximations under some qualifying assumptions. In fact, Amaldi and Kann [1] showed that this problem (they called it RVLS—'relevant variables in the linear system') is as hard to approximate as min unsatisfy, though their result seems to have escaped the notice of the sparse approximation theory community.

Our contribution As a first step, we note that the matrix problems sparse null space and matrix sparsification are equivalent, and that the vector problems min unsatisfy and exact dictionary representation are equivalent as well. Note that although these equivalences are straightforward, they seem to have escaped researchers in this field. For example, [5] claimed that the sparse null space problem is computationally more difficult than matrix sparsification.

We then proceed to show that matrix sparsification is hard to approximate, via a reduction from min unsatisfy. We will thereby show that the two matrix problems are hard to approximate within a factor $2^{\log^{.5-o(1)} n}$ of optimal (assuming NP does not admit quasi-polynomial time deterministic algorithms).

This hardness result for matrix sparsification is important in its own right, but it further leads us to ask what *can* be done for this problem. Specifically, what restrictions or simplifying assumptions may be made upon the input matrix to make matrix sparsification problem tractable? In addressing this question, we provide the major contribution of this paper and show how to adapt the vast number of heuristics and algorithms for exact dictionary representation to solve matrix sparsification (and hence sparse null space as well). This allows us to conclude, for example, that matrix sparsification admits a randomized $\frac{m}{c \log m}$-approximation algorithm, and also to give limiting conditions under which a known ℓ_1 relaxation scheme for exact dictionary matching solves matrix sparsification exactly. Our results also carry over to relaxed version of these problems, where the input is extended by an error term δ which relaxes a constraint. These versions are defined in the "Appendix 2", although we omit

the proof of their equivalence. All of our results assume that the vector variables are over $\mathbb{Q}$.[1]

An outline of our paper follows: In Sect. 2 we review some linear algebra and introduce notation. In closing the preliminary Sect. 2.3, we prove equivalences between the two matrix problems and the two vector problems. In Sect. 3 we prove that matrix sparsification is hard to approximate, and in Sect. 4 we show how to adapt algorithms for exact dictionary representation to solve matrix sparsification.

2 Preliminaries

In this section we review some linear algebra, introduce notation and definitions, and formally state our four problems.

2.1 Linear Algebra and Notation

Matrix and vector properties Given a set V of n m-dimensional column vectors, an m-vector $v \notin V$ is *independent* of the vectors of V if there is no linear combination of vectors in V that equals v. A set of vectors is independent if each vector in the set is independent of the rest.

Now let the vectors of V be arranged as columns of an $m \times n$ matrix A; we refer to a column of A as a_i, and to a position in A as a_{ij}. We define #col(A) to be the number of columns of A. The column *span* of A (col(A)) is the (infinite) set of column vectors that can be produced by a linear combination of the columns of A. The *column rank* of A is the dimension of the column space of A (rank(A) = dim(col(A))); it is the size of the maximal independent subset in the columns of A. If the column rank of A is equal to n, then the columns of A are independent, and A is said to be *full rank*.

Other matrices may be produced from A using *elementary column operations*. These include multiplying columns by a nonzero factor, interchanging columns, and adding a multiple of one column to another. These operations produce a matrix A' which has the same column span as A; we say A and A' are *column equivalent*. It can be shown that A, A' are column equivalent iff $A' = AX$ for some invertible matrix X.

Let R be a set of rows of A, and C be a set of columns. $A(R, C)$ is the submatrix of A restricted to R and C. Let $A(:, C)$ ($A(R, :)$) be the submatrix of A restricted to all rows of A and to columns in C (restricted to the rows of R and all columns in A). A *square matrix* is an $m \times m$ matrix. A square matrix is *nonsingular* if it is invertible.

Null space The *null space* (or *kernel*) of A (null(A)) is the set of all nonzero n-length vectors b for which $Ab = 0$. The rank of A's null space is called the *corank* of A.

[1] More precisely, we assume that all input variables are rational and that their numerators and denominators can be stored in words of size polynomial in the length of the input vector or matrix. All reductions and algorithms presented here require only polynomial time and space over the field of rational numbers. For example, Householder triangularization [36] is a numerically stable process for determining the QR decomposition of a matrix, and can be used to compute a full null matrix as seen in the proof of Theorem 3. The space and time bounds for this process remain polynomial when applied to arbitrary-precision rational values.

The rank-nullity theorem states that for any matrix A, $\text{rank}(A) + \text{corank}(A) = n$. Let N be a matrix consisting of column vectors in the null space of A; we have that $AN = 0$. If the rank of N is equal to the corank of A then N is a *full null matrix* for A.

Given matrix A, a full null matrix for A can be constructed in polynomial time. Similarly, given a full rank matrix N, polynomial time is required to construct a matrix A for which N is a full null matrix [30].

Notation Throughout this paper, we will be interested in the number of zero and nonzero entries in a matrix A. Let $\mathsf{nnz}(A)$ denote the number of nonzero entries in A. For a vector x, let $||x||_0$ denote the number of nonzero entries in x. This notation refers to the quasi-norm ℓ_0, which is not a true norm since $\lambda||x||_0 \neq ||\lambda x||_0$, although it does obey the triangle inequality.

For vector x, let x_i be the value of the ith position in x. The *support* of x (supp(x)) is the set of indices in x which correspond to nonzero values, $i \in supp(x) \Leftrightarrow x_i \neq 0$.

The notation $A|B$ indicates that the rows of matrix B are concatenated to the rows of matrix A. The notation $\binom{A}{B}$ indicates that the columns of B are appended to the columns of A. $M = A \otimes B$ denotes the Kronecker product of two matrices, where M is formed by multiplying each individual entry in A by the entire matrix B. (If A is $m \times n$, B is $p \times q$, then M is $mp \times nq$.)

By *equivalent* problems, we mean that reductions between them preserve approximation factors. A formal definition of approximation equivalence is found in "Appendix 1".

2.2 Minimization Problems

In this section, we formally state the four major minimization problems discussed in this paper. The first two problems have vector solutions, and the second two problems have matrix solutions. Our results hold when the variables are over $\mathbb{Q}$, although these problems can be defined over $\mathbb{R}$. $\mathcal{I}_F$ is the set of input instances, $S_F(x)$ is the solution space for $x \in \mathcal{I}_F$, $m_F(x, y)$ is the objective metric for $x \in \mathcal{I}_F$ and $y \in S_F(x)$.

Problem 1 Exact Dictionary Representation (EDR)

$$\mathcal{I}_{\mathsf{EDR}} = \langle D, s\rangle, m \times n \text{ matrix } D, \text{ vector } s \text{ with } s \in \mathsf{col}(D)$$
$$S_{\mathsf{EDR}}(D, s) = \{v \in \mathbb{Q}^n : Dv = s\}$$
$$m_{\mathsf{EDR}}(\langle D, s\rangle, v) = ||v||_0$$

Problem 2 Min Unsatisfy (MU)

$$\mathcal{I}_{\mathsf{MU}} = \langle A, y\rangle, m \times n \text{ matrix } A, \text{ vector } y \in \mathbb{Q}^m$$
$$S_{\mathsf{MU}}(A, y) = \{x : x \in \mathbb{Q}^n\}$$
$$m_{\mathsf{MU}}(\langle A, y\rangle, x) = ||y - Ax||_0$$

Problem 3 Sparse Null Space (SNS)

$$\mathcal{I}_{\mathsf{SNS}} = \text{ matrix } A$$
$$S_{\mathsf{SNS}}(A) = \{N : N \text{ is a full null matrix for } A\}$$
$$m_{\mathsf{SNS}}(A, N) = \mathsf{nnz}(N)$$

Problem 4 Matrix Sparsification (MS)

$$\mathcal{I}_{\mathsf{MS}} = \text{ full rank } m \times n \text{ matrix } B$$
$$S_{\mathsf{MS}}(B) = \{\text{matrix } N : N = BX \text{ for some invertible matrix } X\}$$
$$m_{\mathsf{MS}}(B, N) = \mathsf{nnz}(N)$$

2.3 Equivalences

In closing the preliminary section, we show that min unsatisfy and exact dictionary representation are equivalent. We then show that sparse null space and matrix sparsification are equivalent. The type of equivalence is formally stated in Definition 2 in the "Appendix 1", and guarantees exact equality of approximation factors among polynomial-time algorithms (Corollary 1).

Here we show that EDR and min unsatisfy are equivalent.

We reduce EDR to min unsatisfy. Given input $\langle D, s\rangle$ to EDR, we seek a vector v with minimum $||v||_0$ that satisfies $Dv = s$. Let y be any vector that satisfies $Dy = s$, and A be a full null matrix for D. (These can be derived in polynomial time.) Let $x = MU(A, y)$ and $v = y - Ax$. We claim that v is a solution to EDR. First note that v satisfies $Dv = s$: $Dv = D(y - Ax) = Dy - DAx = s - 0 = s$. Now, the call to $MU(A, y)$ returned a vector x for which $||y - Ax||_0 = ||v||_0$ is the minimization measure; and, as x ranges over $\mathbb{R}^n$, the vector $v = y - Ax$ ranges over all vectors with $Dv = s$. Hence, the oracle for min unsatisfy directly minimizes $||v||_0$, and so v is a solution to EDR.

We now reduce min unsatisfy to EDR (see [7] for a similar result). Given input $\langle A, y\rangle$ to min unsatisfy, we seek a vector x which minimizes $||y - Ax||_0$. We may assume that A is full rank. (Otherwise, we can simply take any matrix $\tilde{A}$ whose columns form a basis of $\mathsf{col}(A)$, and it follows easily that $||MU(A, y)|| = ||MU(\tilde{A}, y)||$.) Find (in polynomial time) a matrix D such that A is a full null matrix for D (this can be achieved by finding D^T as a null matrix of A^T). Let $s = Dy$, and $v = \mathsf{EDR}(D, s)$. Since $Dv = s$ we have that $D(y - v) = Dy - Dv = 0$, from which we conclude that $y - v$ is in the null space of D, and therefore in the column space of A. It follows that we can find an x such that $Ax = y - v$. We claim that x solves the instance of min unsatisfy: It suffices to note that the call to $\mathsf{EDR}(D, s)$ minimizes $||v||_0 = ||y - Ax||_0$, and that as v ranges over $\{v : Dv = s\}$, the vector $Ax = y - v$ ranges over all of $\mathsf{col}(A)$. In conclusion,

Lemma 1 *The problems* exact dictionary representation *and* min unsatisfy *are equivalent.*

Now we'll demonstrate that sparse null space and matrix sparsification are equivalent. Recall that in the description of matrix sparsification on input matrix B, we required that B be full rank, $\#\mathsf{col}(B) = \mathsf{rank}(B)$. (We could in fact allow $\#\mathsf{col}(B) > \mathsf{rank}(B)$, but this would trivially result in $\#\mathsf{col}(B) - \mathsf{rank}(B)$ zero columns in the solution, and these columns are not interesting.) We will need the following lemma:

Lemma 2 *Let B be a full null matrix for $m \times n$ matrix A. The following statements are equivalent: (1) $N = BX$ for some invertible matrix X. (2) N is a full null matrix for A.*

Proof In both cases, N and B must have the same number of columns, the same rank, and the same span. This is all that is required to demonstrate either direction. □

We can now prove that sparse null space and matrix sparsification are equivalent. The problem sparse null space may be solved utilizing an oracle for matrix sparsification. Given input A to sparse null space, create (in polynomial time) a matrix B which is a full null matrix for A, and let $N = \mathsf{MS}(B)$. We claim that N is a solution to $\mathsf{SNS}(A)$. Since $N = BX$ for some invertible matrix X, by Lemma 2 N is a full null matrix for A. Therefore the call to $\mathsf{MS}(B)$ is equivalent to a call to $\mathsf{MS}(N)$, which solves sparse null space on A.

We show that matrix sparsification can be solved using an oracle for sparse null space. Given input B to matrix sparsification, create (in polynomial time) matrix A such that B is a full null matrix for A. Let $N = \mathsf{SNS}(A)$. We claim that N is a solution to $\mathsf{MS}(B)$. By the lemma, $N = BX$ for some invertible matrix X, so N can be derived from B via elementary column reductions. The call to $\mathsf{SNS}(A)$ finds an optimally sparse N, which is equivalent to solving matrix sparsification on B. In conclusion,

Lemma 3 *The problems* matrix sparsification *and* sparse null space *are equivalent.*

3 Hardness of Approximation for Matrix Problems

In this section, we prove the hardness of approximation of matrix sparsification (and therefore sparse null space). This motivates the search for heuristics or algorithms under simplifying assumptions for matrix sparsification, which we undertake in the next section. For the reduction, we will need a relatively dense matrix which we know cannot be further sparsified. We will prove the existence of such a matrix in the first subsection, and demonstrate how to construct it in the second subsection.

3.1 Unsparsifiable Matrices

Any $m \times n$ matrix A may be column reduced to contain at most $(m - r + 1)r$ nonzeros, where $r = \mathsf{rank}(A)$. For example, Gaussian elimination on the columns of the matrix will accomplish this sparsification. We will say that a rank r, $m \times n$ matrix A is

unsparsifiable if and only if, for any invertible matrix X, $\mathsf{nnz}(AX) \geq (m-r+1)r$. A matrix A is *optimally sparse* if, for any invertible X, $\mathsf{nnz}(AX) \geq \mathsf{nnz}(A)$. The main result of this section follows.

Theorem 1 *Let A be an $m \times n$ matrix with $m \geq n$. If every $n \times n$ submatrix of A is nonsingular, then A has rank n and is unsparsifiable. Moreover, if every square submatrix of A is nonsingular, then the matrix $\binom{I}{A}$ is optimally sparse, where I is the $n \times n$ identity matrix.*

Proof Suppose there is an invertible X such that $\mathsf{nnz}(AX) < (m-n+1)n$; that is, suppose that A is not unsparsifiable. Then there must be a column c_i of $C = AX$ such that $\mathsf{nnz}(c_i) \leq m-n$. Choose a row set $R \subset [m]$ so that $|R| = n$ and $c_{ri} = 0$ for all $r \in R$. Then $A(R,:)X$ has an all-zero column; since X is invertible, it must be the case that $A(R,:)$ is a singular $n \times n$ submatrix of A. This demonstrates the first part of the theorem.

To show the second part, it is sufficient to show that every $n \times n$ submatrix of $\binom{I}{A}$ is nonsingular, as this shows that $\binom{I}{A}$ itself is unsparsifiable and therefore has already achieved optimal sparsity.

Suppose an $n \times n$ submatrix M of $\binom{I}{A}$ is singular. It cannot be I itself; nor can it be entirely contained within A. Thus we can choose a smaller identity matrix I' and submatrices A_1, A_2 of A that allow us to rewrite M as $\begin{pmatrix} I' & 0 \\ A_1 & A_2 \end{pmatrix}$ after a possible column reordering. Since M is singular, there is a nonzero row matrix $(x \;\; y)$ so that

$$(x \;\; y) \begin{pmatrix} I' & 0 \\ A_1 & A_2 \end{pmatrix} = 0.$$

This means that $x + yA_1 = 0$ so that $y \neq 0$; we also know that $yA_2 = 0$, so that A_2 must be singular. Thus, for any matrix A without a singular submatrix, $\binom{I}{A}$ is optimally sparse. □

3.2 Efficiently Building an Unsparsifiable Matrix

The next lemma establishes that we can easily construct an unsparsifiable matrix with a given column, a useful fact for the reductions to follow.

Lemma 4 *If $n \times n$ matrix $M = (M_{ij})$ has entries $m_{ij} = i^{p_j}$ for distinct positive reals $p_1, p_2, \ldots, p_n$, then every subsquare of M is nonsingular.*

Proof Let f be a *signomial* (a polynomial allowed to have nonintegral exponents). We define $\mathsf{positive\_zeros}(f) := \{x : x > 0 \;\&\; f(x) = 0\}$ and $\#\mathsf{sign\_changes}(f) := \#\{i : \mu_i \mu_{i+1} < 0\}$, where $f = \sum_i \mu_i x^{p_i}$, and no $\mu_i = 0$. A slight generalization of Descartes' rule of signs [40] states that

$$\#\mathsf{positive\_zeros}(f) \leq \#\mathsf{sign\_changes}(f). \tag{1}$$

Consider any $k \times k$ subsquare $M(R, C)$ given by $R = \{r_1, \ldots, r_k\}$ and $C = \{c_1, \ldots, c_k\} \subset [n]$, and any nonzero vector $\mu \in \mathbb{R}^k$. Then $M(R, C) \cdot \mu$ matches the signomial $f(x) = \sum \mu_i x^{p_{c_i}}$ evaluated at $x = r_1, \ldots, r_k$. Using (1), $\#\{i : f(r_i) = 0\} \leq \#\mathsf{sign\_changes}(f) < k$, so that some $f(r_i) \neq 0$, and $M(R, C)\mu \neq 0$. Hence the subsquare has a trivial kernel, and is nonsingular. □

For simplicity, we will choose powers of p_j to be consecutive integers beginning at 0. This yields the Vandermonde matrix over $\mathbb{Q}$, which can clearly be stored in polynomial-sized words.

Remark Random matrices In the case of matrices over infinite fields, a random matrix is almost surely (i.e., with probability 1) unsparsifiable when its entries are chosen independently over a non-atomic distribution. (For any element of the field, such a distribution will choose that element with probability zero.)

We first prove that, for any random square matrix, every square submatrix is nonsingular with probability 1. We do so by induction on the size of the matrix. Consider a random $n \times n$ matrix A where all entries have been fixed except a_{11}, which will be chosen randomly. Then $\det(A) = \lambda a_{11} + \mu$, where $\lambda \neq 0$ with probability 1 by inductive assumption as λ is the determinant of a $(n-1) \times (n-1)$ submatrix of A. Since only one value for a_{11} would render A singular, and the distribution is non-atomic, the event $\det(A) = 0$ has probability 0. In addition, there are only finitely many smaller square submatrices, each of which is nonsingular with probability 1 by inductive assumption. The probability of all of them being simultaneously nonsingular is still 1. This completes the proof for square matrices.

A general rectangular random matrix is unsparsifiable if and only if all square submatrices are nonsingular. This is a finite collection of events with probability 1. Again, the probability of all these events happening simultaneously is still 1. This completes the general case.

To create an unsparsifiable matrix for use in a polynomial-time reduction, it suffices to use words of linear size, and repeating the above argument in this setting shows that a random matrix is unsparsifiable with high probability. Further, if the entries are chosen uniformly at random from a normal distribution, then the matrix is also well-conditioned [19]. Using a random matrix would result in the reduction of Sect. 3.3 below being randomized instead of deteministic; Lemma 4, however, avoids any probability and allows us to construct such a matrix as quickly as we can iterate over the entries.

Remark More examples The class of matrices with all nonsingular submatrices are exactly the Maximum Distance Separable matrices arising in coding theory [24], and also include the Reed-Solomon code matrices. Also, many simple Cauchy matrices (a matrix C defined as $c_{ij} = \frac{1}{x_i + y_j}$) satisfy this property: Since the determinant of the Cauchy matrix is known to be $\frac{\Pi_{1 \leq i < j \leq n}(x_j - x_i)(y_j - y_i)}{\Pi_{1 \leq i, j \leq n}(x_i + y_j)}$, we can ensure that all submatrices have nonzero determinants if we stipulate that all x_j's are distinct, all y_j's are distinct, and $x_i + y_j \neq 0 \, \forall \, i, j$; see [24]. The entries of these Cauchy matrices may be stored in words of logarithmic size, much smaller than in the Vandermonde matrix.

3.3 Reduction for Matrix Problems

After proving the existence of an unsparsifiable matrix in the last section, we can now prove the hardness of approximation of matrix sparsification. We reduce min unsatisfy to matrix sparsification. Given an instance $\langle A, y\rangle$ of min unsatisfy, we create a matrix M such that matrix sparsification on M solves the instance of min unsatisfy.

Before describing the reduction, we outline the intuition behind it. We wish to create a matrix M with many copies of y and some copies of A. The number of copies of y should greatly outnumber the number of copies of A. The desired approximation bounds will be achieved by guaranteeing that M is composed mostly of zero entries and of copies of y. It follows that minimizing the number of nonzero entries in the matrix (solving matrix sparsification) will reduce to minimizing the number of nonzero entries in the copies of y by finding a sparse linear combination of y with some other dictionary vectors (solving min unsatisfy).

The construction is as follows: Given an instance $\langle A, y\rangle$ of min unsatisfy (where A is an $m \times n$ matrix, $y \in \mathbb{Q}^m$, and $q \geq p$ are free parameters), take an optimally sparse $(p+q) \times p$ matrix $\binom{I_p}{X}$ as given by Lemma 4 and Theorem 1 (where I_p is a $p \times p$ identity matrix), and create matrix $M_l = \binom{I_p}{X} \otimes y = \binom{I_p \otimes y}{X \otimes y}$ (of size $(p+q)m \times p$). Further create matrix $I_q \otimes A$ (of size $qm \times qn$), and take matrix 0 (of size $pm \times qn$) and form matrix $M_r = \binom{0}{I_q \otimes A}$ (of size $(p+q)m \times qn$). Append M_r to the right of M_l to create matrix $M = M_l | M_r$ of size $(p+q)m \times (p+qn)$. We can summarize this construction as $M = \begin{pmatrix} I_p \otimes y & 0 \\ X \otimes y & I_q \otimes A \end{pmatrix}$.

M_l is composed of $p + pq$ m-length vectors, all corresponding to copies of y. M_r is composed of qn m-length vectors, all corresponding copies of vectors in A. By choosing $p = q = n^2$, we ensure that the term pq is larger than qn by a factor of n. Note that M now contains $O(n^3)$ columns.

It follows that the number of zeros in M depends mostly on the number of zeros induced by a linear combination of dictionary vectors that include y. Because M_l is unsparsifiable, vectors in the rows of M_l will not contribute to sparsifying other vectors in these rows; only vectors in M_r (which are copies of the vectors of A) may sparsify vectors in M_l (which are copies of the vectors in y). It follows that an approximation to matrix sparsification will yield a similar approximation—within a factor of $1 + n^{-\frac{1}{3}}$—to min unsatisfy, and that matrix sparsification is hard to approximate within a factor $2^{\log^{.5-o(1)} n^{1/3}} = 2^{\log^{.5-o(1)} n}$ of optimal (assuming NP does not admit quasi-polynomial time deterministic algorithms).

4 Solving Matrix Sparsification Through Min Unsatisfy

In the previous section we showed that matrix sparsification (MS) is hard to approximate. This motivates the search for heuristics and algorithms under simplifying assumptions for matrix sparsification. In this section we show how to extend algorithms and heuristics for min unsatisfy (MU) to apply to matrix sparsification—and

hence sparse null space— while preserving approximation guarantees. (Note that this result is distinct from the hardness result; neither one implies the other.)

We first present an algorithm for matrix sparsification which is in essence identical to the one given by Coleman and Pothen [11] for sparse null space. The algorithm assumes the existence of an oracle for a problem we will call the sparsest independent vector problem. The algorithm makes a polynomial number of queries to this oracle, and yields an optimal solution to matrix sparsification.

The sparsest independent vector problem takes full-rank input matrices A and B, where the columns of B are a contiguous set of right-most columns from A (informally, one could say that B is a suffix of A, in terms of columns). The output is the sparsest vector in the span of A but not in the span of B. We'll write $A \backslash B$ to denote the matrix composed of columns of A that aren't in B. For convenience, we add an extra output parameter—a column of $A \backslash B$ which can be replaced by the sparsest independent vector while preserving the span of A. More formally, sparsest independent vector is defined as follows. (See §1 for the definition of a problem instance.)

Problem 5 Sparsest Independent Vector (SIV)

$$\begin{aligned}
&\mathcal{I}_{\mathsf{SIV}} = \langle A, B\rangle;\ A \text{ is an } m \times n \text{ full rank matrix with}\\
&A = (C|B) \text{ for some non-empty matrix } C.\\
&S_{\mathsf{SIV}}(A, B) = \{a : a \in \mathsf{col}(A), a \notin \mathsf{col}(B)\}\\
&m_{\mathsf{SIV}}(\langle A, B\rangle, a) = ||a||_0
\end{aligned}$$

The following algorithm reduces matrix sparsification on an $m \times n$ input matrix A to making a linear number of queries to an oracle for sparsest independent vector. As shown above in Sect. 2.3, this is equivalent to making a linear number of queries to an oracle for exact dictionary representation (EDR), where each such query is preceded by the computation of a full null rank matrix.

Algorithm Matrix_Sparsification(A)
$B \leftarrow$ null
for $i = n$ to 1:
  $\langle b_i, a_j\rangle = \mathsf{SIV}(A, B)$
  $A \leftarrow (A \setminus \{a_j\}|b_i)$
  $B \leftarrow (b_i|B)$
return B

This greedy algorithm sparsifies the matrix A by generating a new matrix B one column at a time. The first-added column (b_n) is the sparsest possible, and each subsequent column is the next sparsest. It is decidedly non-obvious why such a greedy algorithm would actually succeed; we refer the reader to [11] where it is proven that greedy algorithms yield an optimal result on matroids such as the set of vectors in $\mathsf{col}(A)$. Our first contribution is in expanding the result of [11] as follows.

Lemma 5 *Let subroutine* SIV *in algorithm Matrix_Sparsification be a* λ*-approximation oracle for* sparse independent vector. *Then the algorithm yields a* λ*-approximation to* matrix sparsification.

Proof Given $m \times n$ matrix A, suppose $\tilde{C}$ exactly solves $\mathsf{MS}(A)$, and that the columns $\tilde{c}_1, \ldots, \tilde{c}_n$ of $\tilde{C}$ are sorted in decreasing order by number of nonzeros. Let $s_i = ||\tilde{c}_i||_0$; then $s_1 \geq s_2 \geq \ldots \geq s_n$. As already mentioned, given a true oracle to sparsest independent vector, algorithm Matrix_Sparsification would first discover a column with s_n nonzeros, then a column with s_{n-1} nonzeros, etc.

Now suppose algorithm Matrix_Sparsification made calls to a $\lambda-$approximation oracle for sparse independent vector. The first column generated by the algorithm, call it b_n, will have at most λs_n nonzeros, since the optimal solution has s_n nonzeros. The second column generated will have at most λs_{n-1} nonzeros, since the optimal solution to the call to SIV has no more than s_{n-1} nonzeros: even if b_n is suboptimal, it is true that at least one of $\tilde{c}_n$ or $\tilde{c}_{n-1}$ is an optimal solution to $\mathsf{SIV}(A, b_n)$.

More generally, the ith column found by the algorithm has no more then λs_i nonzeros, since at least one of $\{\tilde{c}_n, \ldots, \tilde{c}_i\}$ is an optimal solution to the ith query to SIV. Thus we have $\mathsf{nnz}(B) = \sum_i ||b_i||_0 \leq \sum \lambda ||\tilde{c}_i||_0 = \lambda\ \mathsf{nnz}(\tilde{C})$, and may conclude that the algorithm yields a $\lambda-$approximation to matrix sparsification. □

It follows that in order to utilize the aforementioned algorithm for matrix sparsification, we need some algorithm for sparsest independent vector. This is in itself problematic, as the sparsest independent vector problem is hard to approximate—in fact, we will demonstrate later that sparsest independent vector is as hard to approximate as min unsatisfy. Hence, although we have extended the algorithm of [11] to make use of an approximation oracle for sparsest independent vector, the benefit of this algorithm remains unclear.

To this end, we will show how to solve sparsest independent vector while making queries to an approximate oracle for min unsatisfy. This algorithm preserves the approximation ratio of the oracle. This implies that *all* algorithms for min unsatisfy immediately carry over to sparsest independent vector, and further that they carry over to matrix sparsification as well. This also implies a useful tool for applying heuristics for min unsatisfy to the other problems.

The problem sparsest independent vector on input $\langle A, B\rangle$ asks to find the sparsest vector in the span of A but not in the span of B. It is not difficult to see that min unsatisfy solves a similar problem: Given a matrix A and target vector y not in the span of A, find the sparsest vector in the span of $(A|y)$ but not in the span of A. Hence, if we query the oracle for min unsatisfy once for each vector $a_j \notin \mathsf{col}(B)$, one of these queries must return the solution for the sparsest independent vector problem. This discussion implies the following algorithm:

Algorithm Sparse_Independent_Vector(A, B)
$s \leftarrow m + 1$
for $j = 1$ to n:
  if $a_j \notin col(B)$:
    $A_j \leftarrow A \backslash \{a_j\}$
    $x \leftarrow \mathsf{MU}(A_j, a_j)$
    $c' \leftarrow A_j x - a_j$
    if $||c'||_0 < s$
      $c \leftarrow c'$; $s \leftarrow ||c'||_0$; $\alpha \leftarrow a_j$
return $\langle c, \alpha \rangle$

Note that when this algorithm is given a λ-approximate oracle for min unsatisfy, it yields a λ-approximate algorithm for sparsest independent vector. (In this case, the approximation algorithm is valid over the field for which the oracle is valid.)

We conclude this section by giving hardness results for sparsest independent vector by reduction from min unsatisfy; we show that any instance $\langle A, b\rangle$ of min unsatisfy may be modeled as an instance $\langle A', B'\rangle$ of sparsest independent vector: Let $A' = A|y$, and $B' = A$. This suffices to force the linear combination to include y. It follows that sparsest independent vector is as hard to approximate as min unsatisfy, and in fact that the two problems are approximation equivalent.

4.1 When ℓ_1-Minimization is Sufficient

We have presented a tool for extending algorithms and heuristics for exact dictionary representation to min unsatisfy and then directly to the matrix problems. When these algorithms make assumptions on the dictionary of EDR, it is useful to investigate how these assumptions carry over to the other problems.

To this end, we consider here one of the most popular heuristics for EDR, ℓ_1-minimization, and a case where it is guaranteed to provide the optimal result. The heuristic is to find a vector v that satisfies $Dv = s$, while minimizing $||v||_1$ instead of $||v||_0$. (See [13,38,39] for more details.) In [15], Fuchs shows that under the following relatively simple condition ℓ_1-minimization provides the optimal answer to EDR.

Given a matrix A, define $M(A)$ as $\max_{i\neq j} \frac{|a_i^T a_j|}{||a_i||\cdot||a_j||}$; here and throughout this section, the unsubscripted norm $||\cdot||$ indicates the ℓ_2 Euclidean norm.

Theorem 2 (Fuchs) *Suppose we have a matrix D with unit columns and a vector s. If there exists a v with $Dv = s$ and $||v||_0 < \frac{1}{2}(1 + 1/M)$, where $M = M(D)$, then v is the unique solution to $Dv = s$ which minimizes $||v||_0$, and is likewise the unique solution which minimizes $||v||_1$.*

This elegant condition provides an easily computable threshold value such that any solutions v with $||v||_0$ below the threshold are guaranteed to be found by ℓ_1-minimization. The value $M(A)$ can be thought of as the maximum $\cos(\theta_{ij})$ where θ_{ij} is the angle between a_i and a_j. In other words, $M(A)$ reflects the smallest angle (with the largest cosine) between any pair of columns of A. Informally, if two columns of A were separated by a small angle, it would be difficult for ℓ_1-minimization to "decide" between them; avoiding such small angles promotes concentration of the support of v when a sparse v is viable.

We extend the result of Fuchs to min unsatisfy in Theorem 3 and to matrix sparsification in Theorem 4.

It will be useful to define, for a matrix A with columns $||a_i|| < 1$, $\tilde{M}(A) = \max_{i\neq j} \frac{|a_i^T a_j|}{\sqrt{(1-||a_i||^2)(1-||a_j||^2)}}$. When we speak of ℓ_1-minimization for min unsatisfy, we mean a solution for min unsatisfy derived by an ℓ_1-minimization solver for EDR, as described above.

Theorem 3 *Given an instance* $\langle A, y\rangle$ *of* min unsatisfy*, where* A *has orthonormal columns, if there exists a solution vector* x *such that*

$$||y - Ax||_0 < \frac{1}{2}\big(1 + 1/\tilde{M}(A^T)\big),$$

then x *is the unique optimal solution of this instance, and* x *can be found by applying* ℓ_1*-minimization.*

Note that arbitrary inputs to min unsatisfy can be orthonormalized without affecting the solution, so the orthonormality assumption imposes no loss of generality.

Proof In this proof, we use the notation a^i for the ith row of A, and continue to use a_i for the ith column. We also use the notation $\langle x, y\rangle$ to denote the inner product of vectors x and y regardless of their being column or row vectors.

Recall that our reduction finds a matrix D with full null matrix A, and uses $s = Dy$; this D, s pair is the input to EDR. A solution v to $Dv = s$ for EDR corresponds to a solution $v = y - Ax$ of min unsatisfy since $D(y - Ax) = s$.

Complete the matrix A^T into a full unitary matrix $Q = \begin{pmatrix} A^T \\ D \end{pmatrix}$. Since $\langle q^i, q^j\rangle = 0$ for any $i \neq j$, we have $\langle d^i, a_j\rangle = 0$ for any row d^i of D and column a_j of A. In other words, $DA = 0$. Clearly $\mathsf{rank}(A) = \mathsf{corank}(D)$, and D will work for our reduction.

For $i \neq j$, $\langle q_i, q_j\rangle = 0$, so that $\langle a^i, a^j\rangle + \langle d_i, d_j\rangle = 0$, and $|\langle d_i, d_j\rangle| = |\langle a^i, a^j\rangle|$. Since $||a^i||^2 + ||d_i||^2 = ||q_i||^2 = 1$, we arrive at

$$\frac{|\langle d_i, d_j\rangle|}{||d_i|| \cdot ||d_j||} = \frac{|\langle a^i, a^j\rangle|}{\sqrt{(1 - ||a^i||^2)(1 - ||a^j||^2)}} \quad \text{for all } i \neq j.$$

This means $M(D) = \tilde{M}(A^T)$, which concludes this part of the proof. □

The next result addresses matrix sparsification (MS), and depends on a slightly modified version of our earlier-described reduction to min unsatisfy, which is itself reduced to EDR. We'll first describe the modified algorithm, then provide the ℓ_1-minimization guarantees.

Recall that our reduction starts with matrix $A^{(0)} = A$ and replaces one column with a sparser column (or at least an equally-sparse column) to arrive at $A^{(1)}$ with $\mathsf{col}(A^{(1)}) = \mathsf{col}(A^{(0)})$; this process repeats, updating one column at a time as we generate matrices $A^{(2)}$, $A^{(3)}$, etc. Our earlier reduction calls $\mathsf{MU}(A^{(s)} \setminus \{a_k\}, a_k)$ at step s. In order to use the results of Theorem 3, however, we need to send in a matrix with orthonormal columns.

Toward this end, we write $\mathsf{orth}(A)$ for the matrix $Q = AR^{-1}$ found by taking the (reduced) QR decomposition of A; thus, Q is the same size as A, with orthonormal columns, and with $\mathsf{col}(A[k]) = \mathsf{col}(Q[k])$ for any k, where $X[k]$ denotes the first k columns of X.

Let $B^{(s)}$ denote the set of columns of $A^{(s)}$ which have been sparsified. Our reduction remains valid at step s when we call $\mathsf{MU}(A'_k, a_k)$ with A'_k chosen so that

$\mathsf{col}(A'_k \cup \{a_k\}) = \mathsf{col}(A)$ and $\mathsf{col}(B^{(s)}) \subset \mathsf{col}(A'_k)$, ensuring that the result is linearly independent of $\mathsf{col}(B^{(s)})$. By ordering the columns of $A^{(s)}$ so that the sparsified columns are first, we know that the first s columns of $\mathsf{orth}(A^{(s)})$ have the same column span as $B^{(s)}$. So if matrix A'_k is $\mathsf{orth}(A^{(s)})$ with any non-sparsified column removed, then $\mathsf{col}(B^{(s)}) \subset \mathsf{col}(A'_k)$. Thus the sth step will work correctly by calling $\mathsf{MU}(\mathsf{orth}(A^{(s)})\backslash\{a_k\}, a_k)$ over all columns $a_k \notin B^{(s)}$.

The algorithm can be summarized as follows:

$A^{(0)} \leftarrow A\ ;\ B^{(0)} \leftarrow \{\};\ s \leftarrow 1$
for $s = 0$ to $\#\mathsf{cols}(A) - 1$:
  $t \leftarrow \#\mathsf{rows}(A)$
  for each column a_k in $A^{(s)}$ but not in $B^{(s)}$:
    $A'_k \leftarrow \mathsf{orth}(A^{(s)})\backslash\{a_k\}$
    $x \leftarrow \mathsf{MU}(A'_k, a_k)$
    $c' \leftarrow A'_k x - a_k$
    if $||c'||_0 < t$:
      $c \leftarrow c';\ t \leftarrow ||c'||_0;\ k' \leftarrow k$
  $A^{(s+1)} \leftarrow A^{(s)}$ with column $a_{k'}$ removed and c inserted as the first column
  $B^{(s+1)} \leftarrow B^{(s)} \cup \{c\}$
return $A^{(s)}$ with $s = \#\mathsf{cols}(A)$

Let $||A||_{\mathsf{max}}$ denote $\max_{i,j} |a_{ij}|$, and $||A||_{\mathsf{col}}$ denote $\max_i ||a_i||$.

Theorem 4 *Let matrix A be an instance of matrix sparsification, and suppose that $N = \mathsf{MS}(A)$ is an optimally sparse solution with columns n_s sorted by sparsity so that $||n_1||_0 \leq ||n_2||_0 \leq \ldots$. Let $A^{(s)}$ denote the matrices produced by the above algorithm, let $\tilde{A}^{(s)} = \mathsf{orth}(A^{(s)})^T$, and let*

$$M^{(s)} = \tilde{M}(\tilde{A}^{(s)}) + \frac{||\tilde{A}^{(s)}||^2_{\mathsf{max}}}{1 - ||\tilde{A}^{(s)}||^2_{\mathsf{col}}}.$$

If $||n_s||_0 < \frac{1}{2}(1 + 1/M^{(s)})$, then ℓ_1-minimization finds a column at least as sparse as n_s at the sth step.

Values of $\tilde{M}$ or $M^{(s)}$ are useful when they are $\ll 1$. Notice that $\tilde{M}(A)$ is small when all columns of A are pairwise uncorrelated, and that the $M^{(s)}$ values are useful when both $||\tilde{A}^{(s)}||_{\mathsf{max}}$ and $||\tilde{A}^{(s)}||_{\mathsf{col}}$ are small. Intuitively, this can happen when $\tilde{A}^{(s)}$ is a wide matrix composed of unit-length rows with evenly distributed values.

Proof Fix a step s of our matrix sparsification algorithm. We call $\mathsf{MU}(\mathsf{orth}(A^{(s)})\backslash\{a_k\}, a_k)$ and would like to find the M value corresponding to the underlying call to EDR. Since s is fixed, we write $\tilde{A}$ (with columns $\tilde{a}_i$) instead of $\tilde{A}^{(s)}$. Let $\tilde{A}(k) = \left(\mathsf{orth}(A^{(s)})\backslash\{a_k\}\right)^T$, and let $\tilde{a}_i(k)$ denote the ith column of $\tilde{A}(k)$. Find unitary $Q = \begin{pmatrix} \tilde{A}(k) \\ D \end{pmatrix}$. This D, with ith column d_i, is the matrix sent to EDR.

Let $\eta(i,j) = \frac{|\langle d_i, d_j\rangle|}{||d_i||\cdot||d_j||}$. Then

$$\begin{aligned}\eta(i,j) &= \frac{|\langle \tilde{a}_i(k), \tilde{a}_j(k)\rangle|}{\sqrt{(1-||\tilde{a}_i(k)||^2)(1-||\tilde{a}_j(k)||^2)}} \leq \frac{|\langle \tilde{a}_i(k), \tilde{a}_j(k)\rangle|}{\sqrt{(1-||\tilde{a}_i||^2)(1-||\tilde{a}_j||^2)}} \\ &\leq \frac{|\langle \tilde{a}_i, \tilde{a}_j\rangle| + |\tilde{a}_{ki}\tilde{a}_{kj}|}{\sqrt{(1-||\tilde{a}_i||^2)(1-||\tilde{a}_j||^2)}} \\ &\leq \frac{|\langle \tilde{a}_i, \tilde{a}_j\rangle|}{\sqrt{(1-||\tilde{a}_i||^2)(1-||\tilde{a}_j||^2)}} + \frac{||\tilde{A}||^2_{\mathsf{max}}}{1-||\tilde{A}||^2_{\mathsf{col}}}.\end{aligned}$$

The first equality above follows from the definition of Q and the proof of Theorem 3. The last inequality above uses the facts that $\forall i,j, |\tilde{a}_{ij}| \leq ||\tilde{A}||_{\mathsf{max}}$ and $\forall i, (1-||\tilde{a}_i||^2)^{-1/2} \leq (1-||\tilde{A}||^2_{\mathsf{col}})^{-1/2}$.

The result is that $M(D) = \max_{i\neq j}\eta(i,j) \leq \tilde{M}(\tilde{A}) + \frac{||\tilde{A}||^2_{\mathsf{max}}}{1-||\tilde{A}||^2_{\mathsf{col}}}$, which concludes the proof. □

Acknowledgments We thank Daniel Cohen for finding an error in an earlier version of this paper, Robi Krauthgamer for helpful discussions, and the anonymous reviewers for many helpful suggestions.

Appendix 1: Approximation Equivalence

Here we define approximation equivalence. Some of our notation and definitions are inspired by [1], which itself built upon [22].

Definition 1 An *optimization problem* is a four-tuple $F = \{\mathcal{I}_F, S_F, m_F, \mathrm{opt}_F\}$, where $\mathcal{I}_F$ is the set of input instances, $S_F(x)$ is the solution space for $x \in \mathcal{I}_F$, $m_F(x,y)$ is the objective metric for $x \in \mathcal{I}_F$ and $y \in S_F(x)$, and $\mathrm{opt}_F \in \{\min,\max\}$.

We will assume throughout the paper that $\mathrm{opt}_F = \min$.

For any optimization problem F and $x \in \mathcal{I}_F$, we define $F(x) = \mathrm{argmin}_{y\in S_F(x)} m_F(x,y)$ and $||F(x)|| = m_F(x, F(x))$. An approximation $\tilde{F}$ to F is any map on $\mathcal{I}_F$ with $\tilde{F}(x) \in S_F(x)$. We write $||\tilde{F}(x)||$ for $m_F(x, \tilde{F}(x))$. $\tilde{F}$ is an λ-*approximation* for F when, for all $x \in \mathcal{I}_F$, $\frac{||\tilde{F}(x)||}{||F(x)||} \leq \lambda(|x|)$.

Definition 2 Given optimization problems F and G, an *exact reduction* from F to G is a pair $\langle t_1, t_2\rangle$ that satisfies the following: (1) $t_1, t_2 \in P$. (2) $t_1 : \mathcal{I}_F \to \mathcal{I}_G$ and for all $x \in \mathcal{I}_F$, $y \in S_G(t_1(x))$, we have $t_2(x,y) \in S_F(x)$. (3) For all $x \in \mathcal{I}_F$, $y \in S_G(t_1(x))$, we have $m_F(x, t_2(x,y)) = m_G(t_1(x), y)$. (4) For all $x \in \mathcal{I}_F$, $||F(x)|| \geq ||G(t_1(x))||$.

We write $F \preceq G$. We write $F \sim G$ to denote that $F \preceq G$ and $G \preceq F$, and call these problems *equivalent*.

Theorem 5 *If $F \preceq G$ and G admits a λ-approximation, then so does F.*

Proof We are given that for G, there exists $\tilde{G}$ with $\frac{||\tilde{G}(x)||}{||G(x)||} \leq \lambda$ for all $x \in \mathcal{I}_G$. Let $\tilde{F}(x) = t_2(x, \tilde{G}(t_1(x)))$, where $\langle t_1, t_2 \rangle$ is the $F \preceq G$ exact reduction. It suffices to show that $\frac{||\tilde{F}(x)||}{||F(x)||} \leq \frac{||\tilde{G}(x')||}{||G(x')||}$, where $x' = t_1(x)$. By the fourth item of the definition, it suffices to demonstrate that $||\tilde{F}(x)|| \leq ||\tilde{G}(x')||$. By the third item, $||\tilde{F}(x)|| = m_F(x, t_2(x, \tilde{G}(x'))) = m_G(x', \tilde{G}(x')) = ||\tilde{G}(x')||$. □

In fact, it can be shown that $\frac{||\tilde{F}(x)||}{||F(x)||} = \frac{||\tilde{G}(x')||}{||G(x')||}$.

Corollary 1 *If $F \sim G$, then F admits a λ-approximation if and only if G admits a λ-approximation.*

Appendix 2: Relaxed Versions

The following problems are variations which work with more approximate solution spaces. This can be considered as allowing some noise in either the inputs or outputs. It is not difficult to extend the the equivalence proofs and algorithmic approximation results of the paper to the relaxed variants.

Problem 6 Relaxed Dictionary Representation (RDR)

$$\begin{aligned}
&\mathcal{I}_{\mathsf{RDR}} = \langle D, s, \delta \rangle, m \times n \text{ matrix } D, \text{ vector } s \text{ with } s \in \mathsf{col}(D), \delta \geq 0 \\
&S_{\mathsf{RDR}}(D, s, \delta) = \{\langle v, w \rangle \text{ each in } \mathbb{R}^n : D(v - w) = s, ||w|| \leq \delta\} \\
&m_{\mathsf{RDR}}(\langle D, s \rangle, \langle v, w \rangle) = ||v||_0
\end{aligned}$$

Problem 7 Relaxed MinUnsatisfy (RMU)

$$\begin{aligned}
&\mathcal{I}_{\mathsf{RMU}} = \langle A, y, \delta \rangle, m \times n \text{ matrix } A, \text{ vector } y \in \mathbb{R}^m, \delta \geq 0 \\
&S_{\mathsf{RMU}}(A, y, \delta) = \{\langle x \in \mathbb{R}^n, w \in \mathbb{R}^m \rangle : ||w|| \leq \delta\} \\
&m_{\mathsf{RMU}}(\langle A, y \rangle, \langle x, w \rangle) = ||y - Ax + w||_0
\end{aligned}$$

Problem 8 Relaxed Sparse Null Space (RSNS)

$$\begin{aligned}
&\mathcal{I}_{\mathsf{RSNS}} = \langle A, \delta \rangle, \text{ matrix } A, \text{ and } \delta \geq 0 \\
&S_{\mathsf{RSNS}}(A, \delta) = \{\langle M, N \rangle : N \text{ is a full null matrix for } A, ||M|| \leq \delta\} \\
&m_{\mathsf{RSNS}}(\langle A, \delta \rangle, \langle M, N \rangle) = \mathsf{nnz}(M + N)
\end{aligned}$$

Problem 9 Relaxed Matrix Sparsification (RMS)

$$\begin{aligned}
&\mathcal{I}_{\mathsf{RMS}} = \langle B, \delta \rangle, \text{ matrix } B, \text{ and } \delta \geq 0 \\
&S_{\mathsf{RMS}}(B, \delta) = \{\langle M, N \rangle : N = BX, X \text{ invertible}, ||M|| \leq \delta\} \\
&m_{\mathsf{RMS}}(\langle B, \delta \rangle, \langle M, N \rangle) = \mathsf{nnz}(M + N)
\end{aligned}$$

References

1. Amaldi, E., Kann, V.: The complexity and approximability of finding maximum feasible subsystems of linear relations. Theor. Comput. Sci. **147**(1–2), 181–210 (1995)
2. Arora, S., Babai, L., Stern, J., Sweedyk, Z.: The hardness of approximate optima in lattices, codes and linear equations. J. Comput. Syst. Sci. **54**(2), 317–331 (1997)
3. Berman, Piotr., Karpinski, Marek.: Approximating minimum unsatisfiability of linear equations. In: *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '02*, pp. 514–516. Society for Industrial and Applied Mathematics (2002)
4. Berry, M., Heath, M., Kaneko, I., Lawo, M., Plemmons, R., Ward, R.: An algorithm to compute a sparse basis of the null space. Numer. Math. **47**(4), 483–504 (1985)
5. Brualdi, R.A., Friedland, S., Pothen, A.: The sparse basis problem and multilinear algebra. SIAM J. Matrix Anal. Appl. **16**(1), 1–20 (1995)
6. Candès, E., Romberg, J., Tao, T.: Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. IEEE Trans. Inf. Theory **52**(2), 489–509 (2006)
7. Candès, E.J., Tao, T.: Decoding by linear programming. IEEE Trans. Inf. Theory **51**(12), 4203–4215 (2005)
8. Chang, S.Frank, McCormick, S.Thomas: A hierarchical algorithm for making sparse matrices sparser. Math. Program. **56**(1–3), 1–30 (1992)
9. Chartrand, R.: Nonconvex compressed sensing and error correction. In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '07*, volume 3, pages III–889–III–892, (April 2007)
10. Coifman, R., Wickerhauser, M.: Entropy-based algorithms for best basis selection. IEEE Trans. Inf. Theory **38**(2), 713–718 (1992)
11. Coleman, T.F., Pothen, A.: The null space problem I. Complexity. SIAM J. Algebraic Discrete Methods **7**(4), 527–537 (1986)
12. Dinur, Irit, Kindler, Guy, Raz, Ran, Safra, Shmuel: Approximating CVP to within almost-polynomial factors is NP-hard. Combinatorica **23**(2), 205–243 (2003)
13. Donoho, David L.: For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution. Commun. Pure Appl. Math. **59**(6), 797–829 (2006)
14. Duff, I.S., Erisman, A.M., Reid, J.K.: Direct Methods for Sparse Matrices. Oxford University Press, Oxford (1986)
15. Fuchs, J.J.: On sparse representations in arbitrary redundant bases. IEEE Trans. Inf. Theory **50**(6), 1341–1344 (2004)
16. Gilbert, A.C., Guha, S., Indyk, P., Muthukrishnan, S., Strauss, M.: Near-optimal sparse fourier representations via sampling. In: *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing, STOC '02*, pp. 152–161. ACM (2002)
17. Gilbert, A.C., Muthukrishnan, S., Strauss, M.: Improved time bounds for near-optimal sparse fourier representations. In: *Proceedings of SPIE Wavelets XI*, vol. 5914, (2005)
18. Gilbert, J.R., Heath, M.T.: Computing a sparse basis for the null space. SIAM J Algebraic Discrete Methods **8**(3), 446–459 (1987)
19. Higham, Nicholas J.: *Accuracy and Stability of Numerical Algorithms*. In: Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2002. Chapter 28: A Gallery of Test Matrices (2002)
20. Hoffman, A.J., McCormick, S.T.: A fast algorithm that makes matrices optimally sparse. In: *Progress in Combinatorial Optimization*. Academic Press (1984)
21. Johnson, D.S., Preparata, F.P.: The densent hemisphere problem. Theor. Comput. Sci. **6**(1), 93–107 (1978)
22. Kann, Viggo: Polynomially bounded minimization problems that are hard to approximate. Nordic J. Comput. **1**(3), 317–331 (1994)
23. Kavitha, T., Mehlhorn, K., Michail, D., Paluch, K.: A faster algorithm for minimum cycle basis of graphs. In: Diaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) Automata, Languages and Programming, vol. 3142 of Lecture Notes in Computer Science, pp. 846–857. Springer, Berlin (2004)
24. MacWilliams, F.J., Sloane, N.J.A.: The theory of error-correcting codes. Amsterdam, North-Holland (1983)
25. Matoušek, J., Gartner, B.: Understanding and Using Linear Programming. Springer, NewYork (2007)

26. McCormick, S.T.: *A combinatorial approach to some sparse matrix problems*. Ph.D. thesis, Stanford Univ., Stanford, California, 1983. Technical Report 83-5, Stanford Optimization Lab (1983)
27. Muthukrishnan, S.: Nonuniform sparse approximation with Haar wavelet basis. DIMACS TR:2004-42, (2004)
28. Natarajan, B.K.: Sparse approximate solutions to linear systems. SIAM J. Comput. **24**(2), 227–234 (1995)
29. Needell, D., Tropp, J.: Cosamp: iterative signal recovery from incomplete and inaccurate samples. Appl. Comput. Harmon. Anal. **26**(3), 301–322 (2009)
30. Neylon, T.: *Sparse Solutions for Linear Prediction Problems*. Ph.D. thesis, New York University, New York (2006)
31. Pothen, A.: *Sparse Null Bases and Marriage Theorems*. Ph.D. thesis, Cornell University, Ithica, New York (1984)
32. Schmidt, E.: Zur thoerie der linearen und nichtlinearen integralgleichungen, I. *Mathematische Annalen***64**(4), 433–476 (1906–1907)
33. Singhal, S.: Amplitude optimization and pitch prediction in multi-pulse coders. IEEE Trans. Acoust Speech Signal Process **37**(3), 317–327 (1989)
34. Smola, Alex J., Schökopf, Bernhard.: Sparse greedy matrix approximation for machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning, ICML '00*, pp. 911–918 (2000)
35. Temlyakov, V.: Nonlinear methods of approximation. Found. Comput. Math. **3**(1), 33–107 (2003)
36. Trefethen, L.N., Bau, D.: Numerical Linear Algebra. SIAM, Philadelphia, PA (1997)
37. Tropp, J.A.: Greed is good: algorithmic results for sparse approximation. IEEE Trans. Inf. Theory **50**(10), 2231–2242 (2004)
38. Tropp, J.A.: Just relax: convex programming methods for subset selection and sparse approximation. IEEE Trans. Inf. Theory **50**(3), 1030–1051 (2006)
39. Tropp, J.A.: Recovery of short, complex linear combinations via ℓ_1 minimization. IEEE Trans. Inf. Theory **51**(1), 188–209 (2006)
40. Wang, X.: A simple proof of Descartes' rule of signs. Am. Math. Mon. **111**(6), 525–526 (2004)
41. Wright, John, Ma, Yi: Dense error correction via l1-minimization. IEEE Trans. Inf. Theor. **56**(7), 3540–3560 (2010)
42. Zhao, X., Zhang, X., Neylon, T., Shasha, D.: Incremental methods for simple problems in time series: algorithms and experiments. In *Ninth International Database Engineering and Applications Symposium (IDEAS 2005)*, pages 3–14 (2005)
43. Zhou, J., Gilbert, A., Strauss, M., Daubechies, I.: Theoretical and experimental analysis of a randomized algorithm for sparse fourier transform analysis. J. Comput. Phys. **211**, 572–595 (2006)