SIAM J. DISCRETE MATH.
Vol. 32, No. 2, pp. 966–985

MATRIX RIGIDITY FROM THE VIEWPOINT OF PARAMETERIZED COMPLEXITY*

FEDOR V. FOMIN†, DANIEL LOKSHTANOV†, S. M. MEESUM‡, SAKET SAURABH§, AND MEIRAV ZEHAVI†

Abstract. For a target rank r, the *rigidity* of a matrix A over a field $\mathbb{F}$ is the minimum Hamming distance between A and a matrix of rank at most r. Rigidity is a classical concept in computational complexity theory: constructions of rigid matrices are known to imply lower bounds of significant importance relating to arithmetic circuits. Yet, from the viewpoint of parameterized complexity, the study of central properties of matrices in general, and of the rigidity of a matrix in particular, has been neglected. In this paper, we conduct a comprehensive study of different aspects of the computation of the rigidity of *general matrices* in the framework of parameterized complexity. Naturally, given parameters r and k, the MATRIX RIGIDITY problem asks whether the rigidity of A for the target rank r is at most k. We show that in the case $\mathbb{F} = \mathbb{R}$ or $\mathbb{F}$ is any finite field, this problem is fixed-parameter tractable with respect to $k + r$. To this end, we present a dimension reduction procedure, which may be a valuable primitive in future studies of problems of this nature. We also employ central tools in real algebraic geometry, which are not well known in parameterized complexity, as a black box. In particular, we view the output of our dimension reduction procedure as an algebraic variety. Our main results are complemented by a W[1]-hardness result and a subexponential-time parameterized algorithm for a special case of MATRIX RIGIDITY, highlighting the different flavors of this problem.

Key words. matrix rigidity, parameterized complexity, linear algebra

AMS subject classifications. 15A03, 68W40, 68Q25

DOI. 10.1137/17M112258X

1. Introduction.

The *rigidity of a matrix* is a classical concept in computational complexity theory, which was introduced by Grigoriev [7, 8] in 1976 and by Valiant [23] in 1977. Constructions of rigid matrices are known, for instance, to imply lower bounds of significant importance relating to arithmetic circuits. Yet, from the viewpoint of parameterized complexity, the study of central properties of matrices in general, and of the rigidity of a matrix in particular, has been neglected. The few papers that do consider such properties are restricted to the very special case of adjacency matrices, and therefore they are primarily studies in graph theory rather than matrix theory [16, 17]. In this paper, we conduct a comprehensive study of different aspects of the computation of the rigidity of *general matrices* in the framework of parameterized complexity.

Formally, given a matrix A over a field $\mathbb{F}$, the rigidity of A, denoted by $\mathcal{R}_A^{\mathbb{F}}(r)$, is defined as the minimum Hamming distance between A and a matrix of rank at most r. In other words, $\mathcal{R}_A^{\mathbb{F}}(r)$ is the minimum number of entries in A that need to be edited in order to obtain a matrix of rank at most r. Naturally, given parameters r

*Received by the editors March 27, 2017; accepted for publication (in revised form) December 22, 2017; published electronically May 1, 2018. A preliminary version of this article has appeared in the *Proceedings of STACS* 2017.

http://www.siam.org/journals/sidma/32-2/M112258.html

Funding: This work was funded by European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant 306992.

†Department of Informatics, University of Bergen, Bergen, Norway (fomin@ii.uib.no, daniello@ii.uib.no, meirav.zehavi@ii.uib.no).

‡Theoretical Computer Science, The Institute of Mathematical Sciences, HBNI, Chennai, Tamil Nadu 600113, India (meesum@imsc.res.in).

§Department of Informatics, University of Bergen, Bergen, Norway and Theoretical Computer Science, The Institute of Mathematical Sciences, HBNI, Chennai, Tamil Nadu 600113, India (saket@imsc.res.in).

and k, the MATRIX RIGIDITY problem asks whether $\mathcal{R}_A^{\mathbb{F}}(r) \leq k$. In the case when $\mathbb{F} = \mathbb{Q}$ or the edited entries must contain integers, it is not even known whether the problem is decidable [20]. We therefore focus on the cases where $\mathbb{F} = \mathbb{R}$ or $\mathbb{F} = \mathbb{F}_p$ is a finite field for some prime p. Formally, we study the following forms of MATRIX RIGIDITY. Here, FF MATRIX RIGIDITY is not restricted to a specific finite field $\mathbb{F}_p$ but includes $\mathbb{F}_p$ as part of the input.

REAL MATRIX RIGIDITY	PARAMETER: r, k
INPUT: A matrix A with each entry an integer, and two nonnegative integers r, k.	
QUESTION: Is $\mathcal{R}_A^{\mathbb{R}}(r) \leq k$?	

FF MATRIX RIGIDITY	PARAMETER: p, r, k
INPUT: A finite field $\mathbb{F}_p$ of order p, a matrix A over $\mathbb{F}_p$, and two nonnegative integers r, k.	
QUESTION: Is $\mathcal{R}_A^{\mathbb{F}_p}(r) \leq k$?	

Valiant [23] presented the notion of the rigidity of a matrix as a means to prove lower bounds for linear algebraic circuits. He showed that the existence of an $n \times n$ matrix A with $\mathcal{R}_A^{\mathbb{F}}(\epsilon n) \geq n^{1+\delta}$ would imply that the linear transformation defined by A cannot be computed by any arithmetic circuit having size $\mathcal{O}(n)$ and depth $\mathcal{O}(\log n)$ in which each gate of the circuit computes a linear combination of its inputs. Later, Razborov [18] (see [12]) established relations between lower bounds on rigidity of matrices over the reals or finite fields and strong separation results in communication complexity. Although many efforts have been made in this direction [6, 21, 13, 10] (this is not an exhaustive list), proofs of separation lower bounds (quadratic) for explicit families of matrices still remain elusive. For a recent survey on this topic we refer the reader to [14]. The formulation of MATRIX RIGIDITY as stated in this paper was first considered by Mahajan and Sarma [15], and it was shown to be NP-hard for any field by Deshpande [4]. In this paper, we study the concept of the rigidity of a matrix from a different perspective, given by the framework of parameterized complexity (see section 2).

We remark that Meesum, Misra, and Saurabh [16] and Meesum and Saurabh [17] studied the following problems, which are related to MATRIX RIGIDITY but are simpler as they are restricted to graphs. Given a graph $G = (V, E)$ and two nonnegative integers r, k, the problem r-RANK VERTEX DELETION (r-RANK EDGE DELETION) asks whether one can delete at most k vertices (resp., edges) from G so that the rank of its adjacency matrix would be at most r, while r-RANK EDGE EDITING asks whether one can edit k edges in G so that the rank of its adjacency matrix would be at most r.[1] For undirected graphs, Meesum, Misra, and Saurabh [16] proved that these problems are NP-hard even if r is fixed, but can be solved in time $\mathcal{O}^*(2^{\mathcal{O}(k \log r)})$. They also showed that r-RANK EDGE DELETION and r-RANK EDGE EDITING can be solved in time $\mathcal{O}^*(2^{\mathcal{O}(f(r)\sqrt{k} \log k)})$. Meesum and Saurabh [17] obtained similar results for directed graphs.

Our contribution. In this paper, we establish that both REAL MATRIX RIGIDITY and FF MATRIX RIGIDITY are FPT with respect to $r + k$. Specifically, we obtain the following results.

THEOREM 1.1. REAL MATRIX RIGIDITY *can be solved in* $\mathcal{O}^*(2^{\mathcal{O}(r \cdot k \cdot \log(r \cdot k))})$ *time.*

[1] Editing an edge $\{u, v\}$ means that if $\{u, v\} \in E$, then $\{u, v\}$ is deleted, and otherwise it is added.

THEOREM 1.2. FF MATRIX RIGIDITY *can be solved in* $\mathcal{O}^*(f(r,k))$ *time for a function* f *that depends only on* r *and* k.

Observe that the dependency of the running times on the dimension of the input matrix is polynomial, and in the case of FF MATRIX RIGIDITY, the dependency of the running time on p is also polynomial. In the case of REAL MATRIX RIGIDITY, the dependency of the running time on the maximum length (in binary) of any entry in *both* input and output matrices is polynomial. In this context, recall that in the case $\mathbb{F} = \mathbb{Q}$ or the edited entries must contain integers, it is not even known whether MATRIX RIGIDITY is decidable [20]. We also show the following.

THEOREM 1.3. FF MATRIX RIGIDITY *is solvable in time* $\mathcal{O}^*(2^{\mathcal{O}(f(r,p)\sqrt{k}\log k)})$ *for some function* f *that depends only on* r *and* p.

Here, the dependency of the running time on k is subexponential, but the dependency of the running time on p is unsatisfactory in the case p is not fixed. This algorithm adapts ideas from the papers [16, 17].

To obtain our main results, we first present a dimension reduction procedure, which we believe to be a valuable primitive in future studies of problems of this nature. Our procedure is simple to describe, and given an instance of MATRIX RIGIDITY, it outputs (in polynomial time) an equivalent instance where the matrix contains at most $\mathcal{O}((r \cdot k)^2)$ entries. Furthermore, the set of entries of the output matrix is a subset of the set of entries of the input matrix. We believe this procedure to be of interest independent of our main results as it establishes that FF MATRIX RIGIDITY admits a polynomial kernel with respect to $r + k + p$. The simplicity of our procedure also stems from its modularity—it handles rows and columns in separate phases. On a high-level, this procedure is defined as follows. For $k + 1$ steps, it repeatedly selects a set of maximum size consisting of rows that are linearly independent, where if the size of this set exceeds $r + 1$, it is replaced by a subset of size exactly $r + 1$. Each such set of rows is removed from the input matrix, and then it is inserted into the output matrix. At the end of this greedy process, rows that remain in the input matrix are simply discarded. The correctness of our procedure relies on two key insights: (i) if the input instance contains more than $k + 1$ pairwise-disjoint sets of rows that are linearly independent, and each of these sets is of size at least $r + 1$, then the input instance is a NO-instance; (ii) by the pigeonhole principle, any row discarded from the input matrix belongs to the span of at least one set of rows that cannot be edited. Having an intermediate matrix with a small number of rows, the procedure applies the exact same process to the input that is the transpose of this intermediate matrix, thus overall obtaining a matrix with a small number of entries.

Armed with our dimension reduction procedure, we tackle REAL MATRIX RIGIDITY and FF MATRIX RIGIDITY by employing central tools in algebraic geometry, which are not well known in parameterized complexity, as a black box. For this purpose, we first recall that the rank of a matrix is at most r if and only if the determinant of all of its $(r+1) \times (r+1)$ submatrices is 0. Since at this point we can assume that we have a matrix containing only $\mathcal{O}((r \cdot k)^2)$ entries at hand, we may "guess" *which* entries should be edited. Yet, it is not clear *how* these entries should be edited. However, with the above observation in mind, we are able to proceed by viewing our current problem in terms of an algebraic variety. (Such a formulation was also used in the context of complexity analysis in [20].) In particular, this viewpoint gives rise to the applicability of firmly established tools [19, 9] that determine the feasibility of a system of polynomials.

Our main results are complemented by a W[1]-hardness result and a subexponential-time parameterized algorithm for a special case of Matrix Rigidity, which overall present the different flavors of this problem and the techniques relevant to its study. We show that both Real Matrix Rigidity and FF Matrix Rigidity are W[1]-hard with respect to the parameter k. (The papers [16, 17] already imply that both of these problems are para-NP-hard with respect to the parameter r.) Our reduction is inspired by studies in parameterized complexity that involve the Odd Set problem [5], and consists of four reductions, one of which builds upon the recent work of Bonnet, Egri, and Marx [1].

The complexity of our reduction stems from the fact that unlike previous studies of this nature, we establish the W[1]-hardness of our problem of interest over *any* finite field and over the field of reals rather than only over a specific finite field. Thus, we first need to define a special case of Odd Set, which we call Partitioned Unique Intersection, and observe that its W[1]-hardness follows from the proof of the W[1]-hardness of Odd Set that is given in [5]. The correctness of our reductions crucially relies on the implications of the properties of this special case. Our first reduction translates Partitioned Unique Intersection to a problem involving matrices rather than sets, which we call Partitioned Unit Multiplication. Then, to be able to discuss any finite field as well as the field of reals, we introduce new variants of Partitioned Unit Multiplication and the Nearest Codeword problem, called $\mathbb{F}$-Unit Multiplication and $\mathbb{F}$-Nearest Codeword, respectively. The application of our second reduction results in an instance of $\mathbb{F}$-Unit Multiplication. Then, the application of our third reduction, which builds upon [1], results in an instance of $\mathbb{F}$-Nearest Codeword. Finally, we devise a reduction whose application results in an instance of Matrix Rigidity. Here, we make explicit use of the fact that the rank of the target matrix can be large. The overall structure of the reduction may be relevant to studies of other problems where the field is not fixed.

2. Preliminaries. The notation $[n]$ is used to denote the set of integers $\{1, \ldots, n\}$. The *Hamming distance* between two strings of equal length is the number of positions at which they differ.

Linear algebra. The symbols $\mathbb{R}$, $\mathbb{Q}$, and $\mathbb{F}_p$ are used to denote the field of real numbers, the field of rational numbers, and a finite field of order p, respectively. We also use the unsubscripted symbol $\mathbb{F}$ to denote a field, in which case its order is denoted by $|\mathbb{F}|$. A vector v of length n is an ordered tuple of n values from a field $\mathbb{F}$. A collection of vectors $\{v_1, v_2, \ldots, v_k\}$ is said to be *linearly dependent* if there exist values $a_1, a_2, \ldots, a_k \in \mathbb{F}$, not all equal 0, such that $\sum_{i=1}^{k} a_i v_i = 0$; otherwise the collection is said to be *linearly independent*.

A matrix A of dimension $m \times n$ is a sequence of values (a_{ij}) for $i \in [m]$ and $j \in [n]$. The ith row of A, denoted by A_i, is defined as the vector $(a_{i1}, a_{i2}, \ldots, a_{in})$, and the jth column of A, denoted by A^j, is defined as the vector $(a_{1j}, a_{2j}, \ldots, a_{mj})$. Given $I \subseteq [m]$ and $J \subseteq [n]$, we define $A[I, J] = \big(a_{ij} \ : \ i \in I, \ j \in J\big)$, i.e., $A[I, J]$ is the submatrix (or minor) of A with the row set I and the column set J. The *rank* of a matrix is the cardinality of a maximum-sized collection of linearly independent columns (or rows), and is denoted by $\mathsf{rank}(A)$.

We use S_n to denote the collection of all permutation functions of n elements. We call a matrix $\widetilde{A}$ a *jumbled matrix* of A if one can perform a series of row and column exchanges on $\widetilde{A}$ to obtain the matrix A. Equivalently, for an $m \times n$ matrix A and its *jumbled matrix* $\widetilde{A}$, there exist two permutations $\sigma_r \in S_m$ and $\sigma_c \in S_n$ such that $\widetilde{A}_i^j = A_{\sigma_r(i)}^{\sigma_c(j)}$ for all $i \in [m]$ and $j \in [n]$. Similarly, we call a matrix $\widetilde{A}$ a *jumbled*

submatrix of A if there exists a submatrix of A which is a *jumbled matrix* of $\widetilde{A}$. A *mixed matrix* is a matrix having either an indeterminate or a value at each entry. We will be dealing with mixed matrices where the values belong to a finite field or $\mathbb{Z}$. We use I_n to denote the *identity matrix* of size $n \times n$.

System of polynomial equations. Let $x_1, \ldots, x_n$ be variables. Then, a *monomial* is defined as a product $\prod_{i=1}^{n} x_i^{a_i}$ for nonnegative integers $a_1, \ldots, a_n$. The degree of a variable x_i in a monomial $\prod_{i=1}^{n} x_i^{a_i}$ is defined to be the number a_i for $i \in [n]$. The degree of a *monomial* is defined as the sum of degrees of each variable occurring in it. A polynomial over a field $\mathbb{F}$ consists of a sum of monomials with coefficients from the field $\mathbb{F}$. The *total degree* of a polynomial is the degree of a monomial having maximum degree. Given a system of polynomial equations $\mathcal{P} = \{P_1 = 0, P_2 = 0, \ldots, P_m = 0\}$ over a field $\mathbb{F}$, we say that $\mathcal{P}$ is *feasible* over $\mathbb{F}$ if there exists an assignment of values from the field $\mathbb{F}$ to the variables in $\mathcal{P}$ which satisfies every polynomial equation contained in $\mathcal{P}$.

Parameterized complexity. Each problem instance is associated with a parameter k, and we say that a problem is *fixed parameter tractable* (FPT) if any instance (I, k) of the problem can be solved in time $\tau(k)|I|^{\mathcal{O}(1)}$, where τ is an arbitrary function of k. Throughout this paper, we use the standard notation $\mathcal{O}^*$ to hide factors polynomial in $|I|$.[2] On the one hand, to prove that a problem is FPT, it is possible to give an explicit algorithm of the required form, called a *parameterized algorithm*, which solves it. On the other hand, to show that a problem is unlikely to be FPT, it is possible to use parameterized reductions analogous to those employed in classical complexity. Here, the concept of W[1]-hardness replaces the one of NP-hardness, and we need not only construct an equivalent instance in FPT time, but also ensure that the size of the parameter in the new instance depends only on the size of the parameter in the original instance. For our purposes, it is sufficient to note that if there exists such a reduction transforming a problem known to be W[1]-hard to another problem Π, then the problem Π is W[1]-hard as well.

A central notion in parameterized complexity is the one of *kernelization*. Formally, a parameterized problem Π is said to admit a *polynomial kernel* if there is a polynomial-time algorithm, called a *kernelization algorithm*, that given any instance of Π, outputs an equivalent instance of Π whose input size as well as the parameter is bounded by $\tau(k)$, where τ is a function polynomial in k and independent of $|I|$. We say that the reduced instance is a $\tau(k)$-*kernel* for Π. Roughly speaking, a kernelization algorithm can be viewed as an efficient preprocessing procedure that satisfies a well defined restriction with respect to the size of its output. For more information about parameterized complexity in general and kernelization in particular, we refer the reader to monographs such as [5, 2].

Bounded search trees. Informally, a *bounded search tree* or *branching* is used to represent the execution of an algorithm which solves a problem based on the solution of subproblems. It can be represented as a tree and the algorithm can be imagined to solve the subproblems one at a time by traversing this tree. The correctness of a branching algorithm can be justified by arguing that in the case of a YES-instance some sequence of decisions captured by the algorithm leads to a feasible solution. The running time of the algorithm is given by the size of the branching tree. For a parameterized instance, if the size of the branching tree is bounded by a function of the parameter and each step of the algorithm takes polynomial time, then such a branching algorithm leads to an FPT algorithm.

[2] That is, $\mathcal{O}^*(\tau(k)) = \tau(k) \cdot |I|^{\mathcal{O}(1)}$.

One method to bound the size of a branching tree employs the notion of *branching vectors*. To each node of the tree we associate a value using a function which depends on the instance to be solved at that node. This function, usually referred to as a *measure function*, is set up in such a way that it takes a smaller value for a subproblem. It should also satisfy the property that it is a bounded function. Now the size of the tree can be upper-bounded by looking at the drop in value of the measure function at each branch of a node. This drop is represented as a tuple of numbers and is referred to as a *branching vector*. Formally, suppose that the algorithm executes a rule which has ℓ branches (each corresponding to a recursive call), such that in the i^{th} branch, the current value of the measure decreases by b_i. Then, $(b_1, b_2, \ldots, b_\ell)$ is the branching vector of this rule. We say that α is the *branching number* of $(b_1, b_2, \ldots, b_\ell)$ if it is the (unique) positive real root of $x^{b^*} = x^{b^*-b_1} + x^{b^*-b_2} + \cdots + x^{b^*-b_\ell}$, where $b^* = \max\{b_1, b_2, \ldots, b_\ell\}$. If $r > 0$ is the initial value of the measure, and the algorithm returns a result when (or before) it becomes negative, the running time is bounded by $\mathcal{O}^*(\alpha^r)$. For more details we refer the reader to [2].

3. Dimension reduction procedure. In this section we show how to compress an input instance of Matrix Rigidity to an equivalent instance in which the matrix has at most $\mathcal{O}(r^2 \cdot k^2)$ entries. This is a crucial step in obtaining our FPT algorithms for Real Matrix Rigidity and FF Matrix Rigidity. In particular, this step will imply that FF Matrix Rigidity admits a polynomial kernel with respect to $r+k+p$.

Our algorithm is based on the following intuition. Suppose that A is a matrix of rank ℓ. If we could obtain a sequence $B_1, \ldots, B_k$ of pairwise disjoint sets of columns of A where each set forms a column basis of A, then the answer to the question "can we reduce the rank of A to a number $r < \ell$ by editing at most k entries in A" would have been completely determined by the answer to the same question where the editing operations are restricted to the submatrix of A formed by columns in the sets $B_1, \ldots, B_k$. The same conclusion is also true in the case where each B_i is not necessarily a basis, but simply a set of $r+1$ linearly independent columns. Keeping this intuition in mind, we turn to examine an approach where we greedily select and remove (one-by-one) $k+1$ pairwise disjoint sets of linearly independent columns. In each iteration, we attempt to select a set whose size is exactly $r+1$, where if it is not possible, we select a set of maximum size.

Now, let us move to the formal part of our arguments. Note that the relation "is a jumbled matrix of" as defined in section 2 is an equivalence relation. We need the following simple observations which follow from the definition of the rank of a matrix.

Observation 3.1. Let $A \in \mathbb{F}^{m \times n}$ be a matrix of rank equal to r. To make the rank of A at most $r-1$, one needs to change at least one entry in A.

Observation 3.2. For a matrix A, let $\widetilde{A}$ be a *jumbled matrix* of A. Then, the instances (A, r, k), (A^T, r, k), $(\widetilde{A}, r, k)$, and $(\widetilde{A}^T, r, k)$ are equivalent instances of Matrix Rigidity.

Observation 3.3. Let $\widetilde{A}$ be a *jumbled submatrix* of A. Then $\mathsf{rank}(\widetilde{A}) \leq \mathsf{rank}(A)$. If $\widetilde{A}$ is a *jumbled matrix* of A, then $\mathsf{rank}(\widetilde{A}) = \mathsf{rank}(A)$.

Using Observation 3.3, we have the following.

Observation 3.4. If $\widetilde{A}$ is a *jumbled submatrix* of A and $(\widetilde{A}, r, k)$ is a No-instance of Matrix Rigidity, then (A, r, k) is also a No-instance of Matrix Rigidity.

A solution S to an instance (A, r, k) of Matrix Rigidity is a set of size at most k consisting of tuples having three values. For an element $(i, j, e) \in S$ the value of

A_i^j is set to the value e in the edited matrix. We denote the matrix edited using the solution S by A_S.

LEMMA 3.5. *Let $\widetilde{A}$ be a* jumbled matrix *of an $m \times n$ matrix A. Let $\sigma_r \in S_m$ and $\sigma_c \in S_n$ be the permutations which generate the* jumbled matrix *$\widetilde{A}$. If S is a solution of the instance (A, r, k) of* MATRIX RIGIDITY*, then a solution of $(\widetilde{A}, r, k)$ is given by $\widetilde{S} = \{(\sigma_r(i), \sigma_c(j), e) : (i, j, e) \in S\}$.*

Proof. Using the definition of jumbled matrices and the set $\widetilde{S}$, we get that $\widetilde{A}_{\widetilde{S}}$ is a jumbled matrix of A_S. By Observation 3.3, we get that the $\mathsf{rank}(\widetilde{A}_{\widetilde{S}}) \leq r$. The size of $\widetilde{S}$ is equal to k; this proves that $\widetilde{S}$ is a solution of $(\widetilde{A}, r, k)$. ∎

As stated before, our procedure greedily selects a set of columns of A of appropriate dimension iteratively. A detailed description of the procedure, called COLUMN-REDUCTION, can be found in Figure 1. We will now explain the ideas necessary to understand this procedure, which is the heart of this section. The input to COLUMN-REDUCTION consists of a matrix A over any field, given along with nonnegative integers k and r. It outputs a matrix $\widetilde{A}$ whose number of columns is bounded by a function of k and r such that the instances (A, r, k) and $(\widetilde{A}, r, k)$ are equivalent instances of MATRIX RIGIDITY. The computation of a column basis and linearly independent vectors is done in the field $\mathbb{F}$ over which the matrix A is provided.

Algorithm: COLUMN-REDUCTION
INPUT: A matrix A over some field $\mathbb{F}$, and two nonnegative integers r, k.
OUTPUT: A matrix having $\mathcal{O}(r \cdot k)$ columns.

1. **if** $\mathsf{rank}(A) \leq r$ **then** return a YES-instance of appropriate size and exit.
2. Initialize $M_0 = A$ and $i = 0$.
3. **while** $\mathsf{rank}(M_i) \geq r + 1$:
   (a) Let L_i be a set of columns of M_i which is linearly independent in $\mathbb{F}$ and whose size is $r + 1$.
   (b) Let M_{i+1} be the matrix obtained by deleting the columns in L_i from M_i.
   (c) Increment i by 1.
4. **if** $i > k$ **then** return a NO-instance of appropriate size and exit.
   // The matrix A has more than k pairwise-disjoint blocks of the form L_j for $j \leq i$, each having $r + 1$ linearly independent columns. By Observation 3.1*, each block L_i requires at least 1 edit, hence, by Observation* 3.4*, (A, r, k) is a* NO*-instance of* MATRIX RIGIDITY.
5. Let $i_{\leq r} = i$ store the index where the rank of M_i falls below $r + 1$.
6. **while** $i \leq k$:
   (a) Let B_i be a column basis of M_i.
   (b) Obtain M_{i+1} by deleting the columns in B_i from M_i.
   (c) **if** M_{i+1} is empty (in other words, $B_i = M_i$) **then** return A.
   (d) Increment i by 1.
7. Let $\mathcal{L}$ be a matrix formed by the columns in each L_i for $i \in \{0, \ldots, i_{\leq r} - 1\}$.
8. Let $\mathcal{B}$ be a matrix formed by the columns in each B_i for $i \in \{i_{\leq r}, \ldots, k\}$.
9. Return the matrix formed by the columns in $\mathcal{L} \cup \mathcal{B}$.
   //Note that M_{k+1} is nonempty if output occurs here.

FIG. 1. *The column reduction procedure.*

Algorithm: MATRIX-REDUCTION
INPUT: A matrix A over some field $\mathbb{F}$, and two nonnegative integers r, k.
OUTPUT: A matrix having $\mathcal{O}(r \cdot k) \times \mathcal{O}(r \cdot k)$ entries.

1. Let $C_A = \text{COLUMN-REDUCTION}(A, r, k)$.
2. Let $R_A = \text{COLUMN-REDUCTION}(C_A^T, r, k)$.
3. Return R_A^T.

FIG. 2. *The dimension reduction procedure.*

The procedure employs several variables. The variable i is used as an index variable whose initial value is 0, and it is incremented by 1 at a time. For the case when the value of i exceeds k we will show that we are dealing with a NO-instance; otherwise the value depends on a particular input matrix A and is at most k. The variables $M_0, M_1, \ldots$ are submatrices of the input matrix A, satisfying the property that M_i is a submatrix of M_{i-1} with $M_0 = A$. In the first loop of COLUMN-REDUCTION (line 3), if the matrix M_i has rank at least $r+1$, then the variable L_i stores a set of $r+1$ linearly independent columns in the matrix M_i. Additionally, M_i can be obtained by appending the columns in L_i to the matrix M_{i+1}. The variable $i_{\leq r}$ is set to the value of i where the rank of M_i falls below $r+1$—after its initialization the value of $i_{\leq r}$ is not changed. In the second half of the procedure, similar to the set of variables L_i, we define a set of variables B_i which store a column basis of the matrix M_i (line 6). Recall that in this half of the procedure $i \geq i_{\leq r}$, and therefore each matrix M_i is of rank at most r. Additionally, M_i can be obtained by appending the columns in B_i to the matrix M_{i+1} for $i \geq i_{\leq r}$. Finally, the matrix $\mathcal{L}$ is constructed using all the columns in each matrix L_i, and the matrix $\mathcal{B}$ is constructed using all the columns in each matrix B_i for appropriate values of i. By Observation 3.1, we have to edit at least $i_{\leq r}$ entries of $\mathcal{L}$ to make its rank at most r.

In the procedure COLUMN-REDUCTION, a YES-instance of appropriate size can be obtained by taking the matrix $Z = [0]$ (of rank 0), which contains 0 as its only entry. Clearly, (Z, r, k) is a YES-instance of MATRIX RIGIDITY irrespective of the values of r and k. On the other hand, the instance (I_{r+k+1}, r, k) is a NO-instance of MATRIX RIGIDITY. Therefore, the matrix I_{r+k+1} can be used in place of a NO-instance of appropriate size. We need Z and I_{r+k+1} to satisfy the constraint that a kernel is an instance of the same problem as the input instance (even though, if the output is given by either line 1 or 4, we have actually solved the input instance (A, r, k) of MATRIX RIGIDITY in polynomial time). Using the procedure COLUMN-REDUCTION, it is straightforward to reduce the number of rows as well. The details of this procedure are given in Figure 2.

LEMMA 3.6. *Let A be a matrix over some field $\mathbb{F}$, and let r and k be two nonnegative integers. Given an instance (A, r, k), the procedure* MATRIX-REDUCTION *runs in time polynomial in input size and returns a matrix $\widetilde{A}$ satisfying the following properties:*

1. *$\widetilde{A}$ has $\mathcal{O}(r^2 \cdot k^2)$ entries.*
2. *If the output is produced by lines* 6c *and* 9 *of* COLUMN-REDUCTION *(when called by* MATRIX-REDUCTION*), then $\widetilde{A}$ is a* jumbled submatrix *of A.*
3. *(A, r, k) is a* YES*-instance of* MATRIX RIGIDITY *if and only if $(\widetilde{A}, r, k)$ is a* YES*-instance.*

Proof. The steps of procedure COLUMN-REDUCTION are all computable in polynomial time, and therefore MATRIX-REDUCTION runs in polynomial time. We now prove the desired properties one by one. Let the matrix $\widetilde{N}$ denote the output of COLUMN-REDUCTION on the input instance (N, r, k).

Proof of 1. We first bound the size of the output of COLUMN-REDUCTION. The output of this procedure can occur at lines 1, 4, 6c, and 9. If the output happens at line 1, it has 1 column by construction. Similarly, if the output happens at line 4, it has $r + k + 1 \leq (r + 1) \cdot (k + 1)$ columns by construction. If the output occurs at line 6c or line 9, then the number of columns in $\widetilde{N}$ is at most $(k + 1) \cdot (r + 1)$ as it is constructed using columns of at most $i \leq k$ matrices, $L_0, \ldots, L_{i_{\leq r}-1}, B_{i_{\leq r}}, \ldots, B_i$, each having at most $r + 1$ columns.

The procedure MATRIX-REDUCTION first obtains a matrix C_A with the aforementioned number of columns by running COLUMN-REDUCTION. Then, it runs COLUMN-REDUCTION again on the transpose of C_A to get its rows bounded. Thus, the dimensions of the output matrix are as claimed.

Proof of 2. The relation "is a jumbled submatrix of" is a transitive relation, and therefore it suffices to show that the procedure COLUMN-REDUCTION outputs a jumbled submatrix of A. If the output happens at lines 6c and 9, then the columns in the output matrix are a subset of the columns in the input matrix. Therefore, in the first line of procedure MATRIX-REDUCTION C_A is a jumbled submatrix of A. Similarly, R_A is a jumbled submatrix of C_A^T. Finally note that for matrices X and Y, X is a jumbled submatrix of Y if and only if X^T is a jumbled submatrix of Y^T. Hence, the output matrix R_A^T is a jumbled submatrix of A.

Proof of 3. We first show that the procedure COLUMN-REDUCTION produces an equivalent instance of MATRIX RIGIDITY. In the forward direction, suppose that (N, r, k) is a YES-instance of MATRIX RIGIDITY. If the output occurs at line 1, it is a YES-instance by construction. The output cannot occur at line 4 as (N, r, k) is a YES-instance. At lines 6c and 9, by property 2, COLUMN-REDUCTION outputs a jumbled submatrix $\widetilde{N}$ of the input matrix N. Let S denote a solution of the instance (N, r, k) of MATRIX RIGIDITY. By the definition of a jumbled submatrix, there exists a jumbled matrix N' of N such that $\widetilde{N}$ is a submatrix of N'. Construct a solution S' of N' from S using Lemma 3.5. Now construct a set $\widetilde{S}$ from S' by discarding the elements of S' with indices not occurring in the submatrix $\widetilde{N}$. Observe that $\widetilde{N}_{\widetilde{S}}$ is a submatrix of $N'_{S'}$, and therefore it is a jumbled submatrix of N_S. By Observation 3.3, $\mathsf{rank}(\widetilde{N}_{\widetilde{S}}) \leq \mathsf{rank}(N_S) \leq r$, and hence $(\widetilde{N}, r, k)$ is a YES-instance of MATRIX RIGIDITY.

In the backward direction, suppose $(\widetilde{N}, r, k)$ is a YES-instance of MATRIX RIGIDITY. If the output of $\widetilde{N}$ occurs at lines 1 or 4, then we actually know the solution to the instance (N, r, k) of MATRIX RIGIDITY as explained in the comment of the pseudocode. If the output occurs at line 6c, then the output $\widetilde{N}$ of COLUMN-REDUCTION is a jumbled matrix of N and the result holds by Observation 3.2. Now we are left with the case when the output occurs at line 9. Let $\widetilde{S}$ be any solution to the instance $(\widetilde{N}, r, k)$ of MATRIX RIGIDITY. The matrix edited using a solution $\widetilde{S}$ is denoted by $\widetilde{N}_{\widetilde{S}}$. Notice that the matrix $\widetilde{N}$ consists of two submatrices $\mathcal{L}$ and $\mathcal{B}$. As $\mathcal{L}$ consists of $i_{\leq r}$ blocks having rank $r + 1$, by Observation 3.1, we need to edit at least $i_{\leq r}$ entries in $\mathcal{L}$. So, we can afford to make at most $k - i_{\leq r}$ edits in the matrix $\mathcal{B}$. As $\mathcal{B}$ consists of $k + 1 - i_{\leq r}$ blocks, by the pigeonhole principle there exists at least one block in $\mathcal{B}$, say, B_t, which is not subject to any edit by the solution $\widetilde{S}$. Construct the matrix N' by concatenating the columns of M_{k+1} (the columns discarded by the procedure) at the end of the matrix $\widetilde{N}$. By construction N' is a jumbled matrix of N and $\widetilde{N}$ is its submatrix.

Moreover, the matrix $N'_{\widetilde{S}}$ has rank at most r due to the presence of the unedited block B_t in $\widetilde{N}_{\widetilde{S}}$ which spans the matrix M_{k+1}. As $\widetilde{S}$ is a solution of (N', r, k), use Lemma 3.5 to get a solution S of (N, r, k). Thus, $\mathsf{rank}(N_S) = \mathsf{rank}(N'_{\widetilde{S}}) = \mathsf{rank}(\widetilde{N}_{\widetilde{S}}) \leq r$, proving that the instance (N, r, k) is a YES-instance of MATRIX RIGIDITY.

To complete the proof, observe that in the procedure MATRIX-REDUCTION, the instances (A, r, k) and (C_A, r, k) are equivalent by the argument above. By Observation 3.2, (C_A, r, k) and (C_A^T, r, k) are equivalent. As R_A is the output of COLUMN-REDUCTION, (R_A, r, k) is equivalent to (C_A, r, k). Finally, by Observation 3.2, again (R_A, r, k) and (R_A^T, r, k) are equivalent. ∎

If the matrix A is over a fixed finite field $\mathbb{F}$, we obtain a kernel as well.

THEOREM 3.7. *Given an instance (A, r, k) of* FF MATRIX RIGIDITY *over the field $\mathbb{F}_p$, the procedure* MATRIX-REDUCTION *outputs an $\mathcal{O}(r^2 \cdot k^2 \cdot \log p)$-kernel.*

Proof. The number of entries in the output matrix of MATRIX-REDUCTION is bounded by $\mathcal{O}(r^2 \cdot k^2)$, and the bit length of each entry is at most $\lceil \log_2 p \rceil$. ∎

In case the field $\mathbb{F}$ is infinite—for example, if $\mathbb{F}$ is either $\mathbb{Q}$ or $\mathbb{R}$—the procedure is not guaranteed to produce a kernel as the bit lengths of matrix entries may not be bounded by a function of r and k.

4. Fixed-parameter tractability with respect to $k + r$. This section describes an algorithm for MATRIX RIGIDITY. The formulation it presents was also used in the context of complexity analysis in [20].

Using Lemma 3.6, we can reduce any instance (A, r, k) to an equivalent instance (A', r, k) such that the matrix A' is a jumbled submatrix of A and the number of entries in A' is $\mathcal{O}(r^2 \cdot k^2)$. Once we have such a matrix A', it is useful to examine an alternative definition of the rank of a matrix, which is given in terms of the determinant of its square submatrices. Specifically, we will rely on the following proposition.

PROPOSITION 4.1 (see [22, Chapter 7]). *A matrix A over $\mathbb{R}$ has rank at most r if and only if all the $(r+1) \times (r+1)$ submatrices of A have determinant 0.*

The correctness of our algorithm MATRIG-ALG for MATRIX RIGIDITY, which is described in Figure 3, follows in a straightforward fashion using Proposition 4.1. This algorithm for MATRIX RIGIDITY crucially relies on a procedure which can decide the feasibility of a system of polynomials over a given field. This procedure shall be the object of discussion in the rest of the section.

Observe that each polynomial in $\mathcal{P}$, as defined in the algorithm MATRIG-ALG, has at most k unknowns and its *total degree* is at most k. The size of $\mathcal{P}$ is of order $(r \cdot k)^{\mathcal{O}(r)}$. The bit sizes of the coefficients of polynomials in $\mathcal{P}$ are bounded using the following.

LEMMA 4.2. *Let A be a matrix over $\mathbb{R}$. If the longest length entry in A has bit length L, then the bit lengths of the coefficients of the polynomials in $\mathcal{P}$, as computed by the algorithm* MATRIG-ALG*, are of size $\mathcal{O}(r \cdot L + r \cdot \log r)$.*

Proof. The coefficients of polynomials in $\mathcal{P}$ are obtained by computing the determinant of matrices which have size at most $r \times r$. Moreover, the coefficient of a monomial is given by the determinant of a single matrix (as opposed to being the sum of many determinants) because the indeterminates occur only once in the *mixed matrix*. By *Hadamard's inequality* (for a proof see [11]), for a $r \times r$ matrix M, we have $\det(M) \leq \prod_{i \in [r]} \|M_i\|_2$. As the bit length of entries in A is at most L, the coefficients

Algorithm: MatRig-Alg
INPUT: A matrix A over a field $\mathbb{F}$, and two nonnegative numbers r, k.
OUTPUT: Can we edit at most k entries of A to obtain a matrix of rank at most r?

1. Let $A' =$Matrix-Reduction(A, r, k).
2. **for** each set E of k entries in A':
   (a) Replace each entry of A' indexed by an element in E by a distinct indeterminate to obtain a *mixed matrix* A'_E.
   (b) Let $\mathcal{P}$ be the set of equations obtained by setting the determinant of each $(r+1) \times (r+1)$ submatrix of A'_E to 0.
   (c) If $\mathcal{P}$ is feasible over $\mathbb{F}$, then return Yes and exit.
3. Return No and exit.

FIG. 3. *Description of the algorithm for* Matrix Rigidity.

of polynomials in $\mathcal{P}$ are at most $\prod_{i\in[r]} \sqrt{r \cdot 2^{L+1}} = (r \cdot 2^{L+1})^{\frac{r}{2}}$; taking its logarithm gives us the bit length. ∎

We use the following proposition to check the feasibility of the system of polynomials $\mathcal{P}$ when it is defined over $\mathbb{R}$.

PROPOSITION 4.3 (see [19, Proposition 4.2]). *Given a set $\mathcal{P}$ of ℓ polynomials of degree d in k variables with integer coefficients of bit length L, we can decide the feasibility of $\mathcal{P}$ with $L \log L \log\log L(\ell \cdot d)^{\mathcal{O}(k)}$ bit operations.*

Applying the proposition above on the system of equations $\mathcal{P}$, we get the following.

THEOREM 4.4. *Let A be a matrix over $\mathbb{R}$ such that the bit length of each of its entries is bounded by L, and let r and k be two nonnegative integers. Then, the instance (A, r, k) of* Real Matrix Rigidity *can be solved in time $\mathcal{O}^*(2^{\mathcal{O}(r\cdot k\cdot \log(r\cdot k))})$.*

Proof. The algorithm MatRig-Alg generates $\mathcal{O}((r \cdot k)^{2k})$ systems of equations. Each system of equations has $\ell = (r \cdot k)^{\mathcal{O}(r)}$ equations, where the degree $d = k$ and there are k variables. Using Proposition 4.3 along with Lemma 4.2, we get the required running time.

Notice that a system of equations $\mathcal{P}$ is feasible if and only if the chosen entries of the matrix can be edited to reduce the rank. Since we exhaustively try all possible entries that can be edited, the correctness of MatRig-Alg follows. ∎

In the case where the underlying field $\mathbb{F}_p$ is finite, the coefficients of the polynomials are elements of $\mathbb{F}_p$ and hence have bounded bit lengths. The feasibility of $\mathcal{P}$ over a finite field can be decided using the following known algorithm which also gives us an algorithm for FF Matrix Rigidity.

PROPOSITION 4.5 (Kayal [9]). *There is a deterministic algorithm which, given an input consisting of a finite field $\mathbb{F}_p$ and system of polynomials $f_1, \ldots, f_\ell \in \mathbb{F}_p[x_1, \ldots, x_k]$ of* total degree *bounded by d, decides its feasibility in time $d^{k^{\mathcal{O}(k)}} \cdot (\ell \cdot \log p)^{\mathcal{O}(1)}$.*

Similar to the proof of Theorem 4.4, we obtain the following.

THEOREM 4.6. *The problem* FF Matrix Rigidity, *where the input matrix A is an $m \times n$ matrix over a field $\mathbb{F}_p$, can be solved in time $f(r,k)(\log p + m + n)^{\mathcal{O}(1)}$ for some function f.*

This algorithm for FF MATRIX RIGIDITY has the advantage that it runs in time which is *polynomial in the logarithm of the order of the field*, even though the dependence on k is exponential.

5. An algorithm for FF MATRIX RIGIDITY with subexponential dependency on k. In this section, we will also rely on the classic technique of bounded search trees, which is presented in the preliminaries. Our objective is to prove the following theorem.

THEOREM 5.1. *For some function f, the* FF MATRIX RIGIDITY *problem is solvable in $\mathcal{O}^*(2^{\mathcal{O}(f(r,p)\sqrt{k}\log k)})$ time.*

Let $\mathbb{F}_p$ be a finite field. We will prove that MATRIX RIGIDITY over $\mathbb{F}_p$ is solvable in the desired time. Let $(A_{m\times n}, r, k)$ be an instance of this problem. Meesum and Saurabh [17] proved that any rank r skew-symmetric matrix A with entries from $\{-1, 0, 1\}$ has at most 3^r distinct columns [17, Theorem 2]. Their proof with a straightforward modification gives us the following corollary, which we state here without proof.

COROLLARY 5.2. *Any rank r symmetric matrix A with entries from $\mathbb{F}_p$ has at most p^r distinct rows and at most p^r distinct columns.*

Given a matrix M, let I be a maximum sized set of distinct rows of M, and let J be a maximum sized set of distinct columns of M. Then, we define $\mathsf{distinct}(M) = M[I, J]$. To be more precise, $\mathsf{distinct}(M)$ should be defined as an equivalence class of submatrices up to reordering of rows and columns, but here we slightly abuse notation and consider some specific submatrix $M[I, J]$ as $\mathsf{distinct}(M)$. Now, let $\mathcal{D}_r$ be the set of all rank r matrices with distinct rows and distinct columns. As each matrix in $\mathcal{D}_r$ has at most p^r rows as well as at most p^r columns, we get the following observation.

OBSERVATION 5.3. *The value of $|\mathcal{D}_r|$ is bounded by a function of r and p.*

To solve $(A_{m\times n}, r, k)$ in the desired time, for each $D \in \mathcal{D}_r$, we need to check in time $\mathcal{O}^*(2^{\mathcal{O}(f(r,p)\sqrt{k}\log k)})$, for some function f, whether it is possible to change at most k entries of A to obtain a matrix M such that $\mathsf{distinct}(M) = D$.

Next, we show how to interpret a given matrix as the adjacency matrix of a weighted undirected graph. Given an $m \times n$ matrix M, let

$$\mathsf{sym}(M) = \begin{bmatrix} \mathbf{0} & M \\ M^{\mathrm{T}} & \mathbf{0} \end{bmatrix},$$

where $\mathbf{0}$ is the matrix of appropriate dimension that contains zero at each of its entries. Now, suppose that we are given a matrix $D \in \mathcal{D}_r$. Observe that at most k entries in A can be changed to obtain a matrix M such that $\mathsf{distinct}(M) = D$ if and only if there are at most k pairs (i, j), $1 \le i \le m$ and $1 \le j \le n$, such that the entries $a_{i,j+m}$ and $a_{i+n,j}$ in $\mathsf{sym}(A)$ can be changed to obtain a matrix M such that $\mathsf{distinct}(M) = \mathsf{sym}(D)$. Now, we think of the matrices $\mathsf{sym}(A)$ and $\mathsf{sym}(D)$ as adjacency matrices of weighted undirected complete graphs where the weights belong to $\mathbb{F}_p$. More precisely, given a symmetric matrix $M_{t\times t}$ with zeros at its diagonal, the construction of $\mathsf{graph}(M)$ is performed as follows. For each index $i \in [t]$ we introduce a vertex v_i, and the weight of an edge $\{v_i, v_j\}$ is given at the entry m_{ij}. Given a weighted undirected complete graph $G = (V, E)$, let $(V_1, \ldots, V_\ell)$ be a partition of V minimizing ℓ such that for all $i \in [\ell]$, the weight of each edge between any two vertices in V_i is 0, and for all distinct $i, j \in [\ell]$, $v, v' \in V_i$, and $u \in V_j$, the weight of $\{v, u\}$ equals the weight of $\{v', u\}$. Observe that this partition is unique up to reordering the sets V_i. Informally, two

vertices of the graph are in the same partition if they correspond to entrywise equal columns. Now, we let $\mathsf{distinct}(G)$ be the weighted undirected complete graph having one vertex representing each set V_i, and let the edge between the vertex representing V_i and the vertex representing V_j have the same weight as any edge between a vertex in V_i and a vertex in V_j in G. Since changing an entry in a matrix M is equivalent to changing the weight of an edge in $\mathsf{graph}(\mathsf{sym}(M))$, we conclude that to prove Theorem 5.1, it is sufficient to prove the following lemma.

LEMMA 5.4. *Let G and H be weighted undirected complete graphs with weights from $\mathbb{F}_p$ such that $\mathsf{distinct}(H) = H$ and $|V(H)| = g(r,p)$ for some function g. Then, it is possible to determine in time $\mathcal{O}^*(2^{\mathcal{O}(f(r,p)\sqrt{k}\log k)})$, for some function f, whether the weights of at most k edges in G can be changed to obtain a graph G' such that $\mathsf{distinct}(G') = H$.*

Thus, in the rest of this section, it is sufficient to focus on the proof of Lemma 5.4, which is based on a proof given in the paper [17] (which, in turn, is inspired by the paper [3]). The proof idea relies on a branching algorithm in which the branching parameter is the number of edits allowed for a particular branch. The proof transforms a given graph G into another fixed graph H, satisfying $\mathsf{distinct}(H) = H$, whose rank is already known to be at most r, the target rank. To do so, the vertices of H are treated like bags which need to be filled in with vertices of G. The operation of placing a vertex inside a bag corresponds to creating a repeated row and a column, which does not change the rank. Finally, we need to keep track of number of edits required for a vertex to be placed in a particular bag of H. This procedure ensures that the graph G is step by step transformed into another graph G' such that $\mathsf{distinct}(G')$ is a subgraph of H, and thus the rank of G' is at most the rank of H. Next, we state a proposition about the branching vectors.

PROPOSITION 5.5 (see [3]). *For $t > 0$, let $(1, t, t, \ldots, t)$ be the branching vector in which t appears s times. If t is significantly larger than s ($t > 2^s$), then its branching number is bounded by $1 + \frac{\log_2 t}{t}$.*

Proof of Lemma 5.4. A vertex of H will be referred to as a bag and will be filled in with the vertices of G. Let $b = |V(H)|$ denote the number of bags in H and use $\mathcal{B} = \{B_i : i \in [b]\}$ to denote the set of bags. Assume that the vertices in H are $v_1, \ldots, v_b$ and B_i corresponds to the vertex v_i for all $i \in [b]$. The collection of vertex-sets of G corresponding to the vertices in $\mathsf{distinct}(G)$ is denoted by $\mathcal{M} = \{V_1, \ldots, V_t\}$. (Recall that $\mathcal{M}$ is a partition of $V(G)$ and each V_i denotes a set of vertices in G represented by the same vertex in $\mathsf{distinct}(G)$.) Observe that each change of a weight of an edge in G can decrease the number of vertices in $\mathsf{distinct}(G)$ by at most 2, and therefore if $t > 2k + |V(H)|$, the answer to (G, H, r, p, k) is NO. Therefore, we may next assume that $t \leq 2k + |V(H)|$. If at any point during the calculations below, the value of the parameter k drops below 0, we return the answer NO at the current node of the search tree; if at least one leaf of the search tree returns YES, we propagate the value YES—that is, if a node has several children (corresponding to different branches considered by a branching rule) and at least one of them returns YES, we return YES. In the preprocessing phase of the algorithm we add a "sufficient" number of vertices to each bag, which will allow us later to perform branching rules associated with branching vectors where the drop in the parameter at all but one branch is large.

Preprocessing phase. Each bag can be in one of two states: *closed* or *open*. At the start of the algorithm all the bags are empty and open. In every bag we create $s = \lfloor \sqrt{k} \rfloor$ empty slots. Overall there are $b \cdot s$ slots that can be filled. We say that

a bag is *free* if it is open and less than s slots have been filled in it. The state of a node in the bounded search tree is denoted using $(\mathcal{M}, \mathcal{B})$. As long as there is a free bag, we perform the following branching rule, consisting of $t+1$ branches. In its ith branch, for $i \in [t]$, pick an arbitrary vertex u from V_i, delete it from V_i, and add it to the lowest number free bag in $\mathcal{B}$. In the $(t+1)$th branch we mark the lowest number free bag as closed. The application of branching rules in the preprocessing phase is finished when no bag is free. At each leaf node $(\mathcal{M}, \mathcal{B})$ of the search tree, we construct a graph H' over the vertices in $\bigcup \mathcal{B}$. For every weighted edge $\{v_i, v_j\}$ in H we add all the edges $B_i \times B_j$ in the graph H' with the same weight as $\{v_i, v_j\}$. Edges between vertices of the same bag have weight 0. Next, for every edge $\{v, u\}$ in H', we check whether the weight of the edge in H' is the same as its weight in G—if this is not the case, we decrease k by 1. This operation is equivalent to changing the weight of the edge in G to its weight in H'.

At the end of the preprocessing phase the following two cases arise. For each vertex v_i of H either we know exactly which are the at most s vertices of G that should be equivalent to it in a solution graph G' (i.e., $\mathsf{distinct}(G') = H$), or we know exactly s vertices of G equivalent to v_i in G'. In the first case, the bag has been closed, and in the second case, it remains open. In the preprocessing phase, each branching rule consists of at most $t+1$ branches and the depth of the search tree is $b \cdot s$. This gives us the following.

OBSERVATION 5.6. *The preprocessing phase can be performed in time* $\mathcal{O}^*((t+1)^{b\cdot s}) = \mathcal{O}^*((b+2k+1)^{b\cdot s}) = \mathcal{O}^*(2^{\mathcal{O}(f'(r,p)\sqrt{k}\log k)})$ *for some function* f'.

Assigning bags to undecided vertices. This phase of the algorithm begins at a node $(\mathcal{M}, \mathcal{B})$ along with the graph H' and the reduced parameter k as provided by a leaf of the search tree procedure in the previous phase. The vertices in $V(G) \setminus (\bigcup \mathcal{B})$, which have not yet been added to any bag, are called *undecided vertices*. We note that so far the weight modifications have been done only within the bag vertices added in the preprocessing phase and that the (possibly modified) weights of edges between these bag vertices remain fixed for the rest of the algorithm. The branching rules stated in the next paragraph are applied exhaustively in the given order—if at any node of the search tree a rule is applied, then none of the previous rules is applicable. We first consider the case when all the bags are open and handle the closed bags later.

Before an undecided vertex is placed in a bag, as a first step we need to ensure that its edge weight with all the vertices of particular bag are the same. If there exists an undecided vertex u and a bag $B_i \in \mathcal{B}$ such that not all of the edges between u and the vertices in B_i have the same weight, then we apply the following rule. For each weight in $\mathbb{F}_p$, we have a separate branch. In the branch corresponding to some weight $w \in \mathbb{F}_p$, for each edge between u and a vertex in B_i whose weight is not w, we change the weight of the edge to w and decrease k by 1. For now, u is not yet added to any bag and remains an undecided vertex. Let us denote $\mathbb{F}_p = \{w_1, w_2, \ldots, w_p\}$. Then, for all $j \in [p]$, we let s_j denote the number of edges between u and B_i whose weight is not $w_j \in \mathbb{F}_p$. Let us denote $\ell = |B_i|$. Notice that $\sum_{j=1}^{p} s_j = \ell$. As B_i is an open bag (due to our current assumption), it has at least s slots filled, which means that $\ell \geq s$. Observe that the branching vector we obtain is precisely $(\ell - s_1, \ell - s_2, \ldots, \ell - s_p)$.

We now show that the branching number of $(\ell - s_1, \ell - s_2, \ldots, \ell - s_p)$ is upper bounded by $1 + \frac{\log_2(s/2)}{(s/2)}$. For this purpose, since $\ell \geq s$, it is sufficient to show that the branching number of $(\ell - s_1, \ell - s_2, \ldots, \ell - s_p)$ is upper bounded by $1 + \frac{\log_2(\ell/2)}{(\ell/2)}$. As $\sum_{j=1}^{p} s_j = \ell$, there can be at most one $j \in [p]$ such that $s_j > \ell/2$. Without loss

of generality (w.l.o.g.), suppose that this j equals 1. Then, $(\ell - s_1, \ell - s_2, \ldots, \ell - s_p)$ is at least as good as $(1, \ell - s_2, \ldots, \ell - s_p)$, where each s_j is at most $\ell/2$. In turn, this means that the latter branching vector is at least as good as $(1, \ell/2, \ell/2, \ldots, \ell/2)$, where $\ell/2$ occurs $(p-1)$ times. Then, by Proposition 5.5, we derive that the root of this branching vector is upper bounded by $1 + \frac{\log_2(\ell/2)}{(\ell/2)}$. Now, after applying this rule exhaustively, given any undecided vertex u, for each bag in $\mathcal{B}$ the weights of the edges between u and the vertices in this bag are the same.

We say that a vertex u fits a bag $B(u) \in \mathcal{B}$ if u has edges of the same weights as vertices in $B(u)$ in the graph induced on the bags. That is, the weight of each edge between u and a vertex in $B(u)$ is 0, and the weight of an edge between u and a vertex $v \in (\bigcup \mathcal{B}) \setminus B(u)$ is the same as the weight of an edge between any vertex in $B(u)$ and v. Since $\mathsf{distinct}(H) = H$, all the vertices in H have distinct weighted neighborhoods, and therefore every vertex u fits at most one bag. If there exists an undecided vertex u that fits no bag, we branch and decide on a bag for u, put u in this bag, and perform the necessary changes of weights of edges between u and vertices in $\bigcup \mathcal{B}$, updating k accordingly. Here, we have b branches, and at each branch the weights of all of the edges between u and at least one bag are changed, and therefore the parameter decreases by at least s. That is, we obtain a branching vector at least as good as $(s, \ldots, s)$, where s appears b times. Below we will obtain a worse branching vector.

After the last step, every undecided vertex u fits exactly one bag $B(u)$. If there are no two undecided vertices u and v such that if we put u in $B(u)$ and v in $B(v)$, no conflict is created (that is, the weight of the edge between u and v is the same as the weight of any edge between a vertex in $B(u)$ and a vertex in $B(v)$), we can simply return the answer YES. Indeed, in this case, since $k \geq 0$ (else recall that we would have already returned NO), we have used at most k changes to modify G to a graph G' such that $\mathsf{distinct}(G') = H$—each undecided vertex can be put in the only bag it fits and no changes are required. Therefore, we now suppose that there are two undecided vertices u and v that do create a conflict. We apply a branching rule that is an exhaustive search consisting of the following branches:

1. In the first branch, we address the conflict by changing the weight of the edge between u and v to the weight of any edge between a vertex in $B(u)$ and a vertex in $B(v)$, decrease k by 1, and then put u in $B(u)$ and v in $B(v)$.
2. Next, we consider $b-1$ branches that find a new bag for u. More precisely, in each of these branches, we put u in a different bag B_i in $\mathcal{B} \setminus \{B(u)\}$, and update the weights of the edges between u and other vertices in $\bigcup \mathcal{B}$ accordingly. (That is, if the weight of an edge between u and a vertex in B_i is not 0, we update it to 0, and if the weight of an edge between u and a vertex in a different bag B_j is not the same as the weight of the edge $\{v_i, v_j\}$ in H, we update it to this weight.) Moreover, in each of these branches, we decrease k by the number of the changes that were made. Observe that since u fits only $B(u)$, in each of these branches k is decreased by at least s.
3. Finally, we consider $b-1$ branches that find a new bag or v. These branches are symmetric to those considered in the previous item.

The branching vector we obtain is at least as good as $(1, s, \ldots, s)$, where s appears $2(b-1)$ times. By Proposition 5.5, the branching number is bounded by $1 + \frac{\log_2 s}{s}$.

Thus, if there are no closed bags the branching rules mentioned above will be sufficient to decide the fate of that branch. Next, we will show that closed bags can be ignored safely during the branching procedure and the branching rules can be applied on the graph induced on open bags.

Handling the closed bags. The closed bags do not guarantee branching vectors as good as those given previously. For example, when we change all of the weights of the edges between some undecided vertex u and the vertices of a closed bag, we are changing less than s values, and therefore the drop in k is smaller than s. However, the closed bags do not really pose a problem due to the following arguments, which rely on the fact that after a bag is marked closed in the preprocessing phase, no vertex will ever be added in it later. In what follows, we show that we can handle the closed bags by making greedy choices after the branching choices have been made according to the graph induced on the open bags.

Let U be the set of vertices of H' corresponding to the *open* bags. Note that $H'[U]$ may not remain a graph satisfying $\mathsf{distinct}(H'[U]) = H'[U]$. In that case we group together bags which are equivalent in $H'[U]$ and call each group a superbag, where two bags B_i an B_j are equivalent if the weight of each edge between them is 0, and for every bag B_ℓ, the weight of an edge between a vertex in B_i and a vertex in B_ℓ is the same as the weight any edge between a vertex in B_j and a vertex in B_ℓ. So each bag in $H'[U]$ is either a "normal" bag or a superbag. Observe that each bag in $H'[U]$ has at least s vertices in it as we are only merging open bags together, which had s vertices in them to begin with. Considering the graph $H[U]$ with the superbags, we perform exactly the same branching rules as above, where we assume that all the bags are open. We have fewer branches and the branching vectors do not get worse. After branching, at the point where we have previously returned YES, we have that the each undecided vertex fits one of the bags in $H'[U]$ and the weights of the edges between them are fixed without conflicts. Now, while actually adding a vertex u to a bag we also decide on the bag within a superbag S that will host u. Adding u to a bag does not change the weights of edges within the union of open bags and set of undecided vertices. Therefore we can take these decisions independently for each u, adding u to any bag in S that causes the minimum number of changes of weights of edges between u and vertices in the closed bags, and updating k accordingly.

Time complexity. Recall that the preprocessing phase is performed in the desired time. Thus, it remains to analyze the time necessary to perform the calculations following this phase. The worst branching number we obtained was bounded by $1 + \frac{\log_2(s/2)}{(s/2)}$, assuming that s is significantly larger than r and p. Therefore, since $s = \lfloor \sqrt{k} \rfloor$, we obtain that the running time of our algorithm is bounded by

$$\mathcal{O}^*\left(2^{f(r,p)} \cdot \left(1 + \frac{\log_2(s/2)}{(s/2)}\right)^k\right) = \mathcal{O}^*\left(2^{\mathcal{O}(f(r,p)\sqrt{k}\log k)}\right)$$

for some function f. ∎

6. W[1]-hardness with respect to k. In this section, we first reduce (in two steps) a special case of ODD SET to a problem that has a formulation easier to use in our context. The latter problem is reduced to a variant of NEAREST CODEWORD, which, in turn, is reduced to REAL MATRIX RIGIDITY and FF MATRIX RIGIDITY.

ODD SET	PARAMETER: k
INPUT: A family $\mathcal{F}$ of sets over a universe U and a nonnegative integer k.	
QUESTION: Does there exist a subset $S \subseteq U$ of size at most k such that the intersection of S with every set in $\mathcal{F}$ has odd size?	

NEAREST CODEWORD	PARAMETER: k
INPUT: An $m \times n$ matrix M and an m-dimensional vector b over $\mathbb{F}_2$, along with a nonnegative integer k.	
QUESTION: Is there an n-dimensional vector x over $\mathbb{F}_2$ such that the Hamming distance between Mx and b is at most k?	

THEOREM 6.1. REAL MATRIX RIGIDITY *and* FF MATRIX RIGIDITY *for any choice of a finite field* $\mathbb{F}_p$ *are* $\mathsf{W}[1]$*-hard with respect to* k.

Proof. Let $\mathbb{F}$ denote the field, which is either $\mathbb{R}$ or some finite field $\mathbb{F}_p$, over which we define MATRIX RIGIDITY. First, we observe that the reduction from MULTICOLORED CLIQUE given in the book [2, Theorem 13.31] to show that ODD SET is $\mathsf{W}[1]$-hard actually shows that the following special case of ODD SET is $\mathsf{W}[1]$-hard. That is, the constructed instances have the form specified in the special case.

PARTITIONED UNIQUE INTERSECTION	PARAMETER: k
INPUT: A family $\mathcal{F}$ of sets over a universe U, a nonnegative integer k, a partition $(U_1, \ldots, U_k)$ of U such that for every $i \in [k]$, $U_i \in \mathcal{F}$, and for every $F \in \mathcal{F}$, there exist $i, j \in [k]$ for which $F \subseteq U_i \cup U_j$.	
QUESTION: Is there a subset $S \subseteq U$ of size at most k such that the intersection of S with every set in $\mathcal{F}$ has size 1?	

The arguments below will crucially rely on the fact that we restrict ourselves to this special case. Given a vector v, we let $\mathrm{supp}(v)$ denote the indices of the entries of v that do not contain 0. Now, we reformulate PARTITIONED UNIQUE INTERSECTION in the language of matrices as follows.

PARTITIONED UNIT MULTIPLICATION	PARAMETER: k
INPUT: A $t \times r$ binary matrix L over $\mathbb{R}$, a nonnegative integer k, a partition $(U_1, \ldots, U_k)$ of $[r]$ such that for every $i \in [k]$, there exists $j \in [t]$ for which $U_i = \mathrm{supp}(L_j)$, and for every $i \in [t]$, there exist $j, \ell \in [k]$ for which $\mathrm{supp}(L_i) \subseteq U_j \cup U_\ell$.	
QUESTION: Is there an r-dimensional binary vector x such that $\lvert\mathrm{supp}(x)\rvert \leq k$ and $Lx = 1$?	

Given an instance $(\mathcal{F}, U, (U_1, \ldots, U_k), k)$ of PARTITIONED UNIQUE INTERSECTION, it is straightforward to obtain (in polynomial time) an equivalent instance $(L_{t\times r}, (U'_1, \ldots, U'_k), k')$ of PARTITIONED UNIT MULTIPLICATION as follows. First, we let $t = |\mathcal{F}|$ and $r = |U|$. We assume w.l.o.g. that $U = [r]$. Now, we associate a row L_i with each set $F \in \mathcal{F}$ by letting L_i contain 1 at each entry whose index belongs to F and 0 at each of the remaining entries. That is, $\mathrm{supp}(L_i) = F$. Finally, we let $(U'_1, \ldots, U'_k) = (U_1, \ldots, U_k)$ and $k' = k$. It is easy to see that $S \subseteq U$ is a solution to $(\mathcal{F}, U, (U_1, \ldots, U_k), k)$ if and only if the binary vector $x_{r\times 1}$ such that $\mathrm{supp}(x) = S$ is a solution to $(L_{t\times r}, (U'_1, \ldots, U'_k), k)$, and therefore the instances are equivalent.

We now incorporate the input field $\mathbb{F}$.

$\mathbb{F}$-UNIT MULTIPLICATION	PARAMETER: k
INPUT: A $t \times r$ binary matrix L over $\mathbb{F}$ and a nonnegative integer k.	
QUESTION: Is there an r-dimensional vector x over $\mathbb{F}$ such that $\lvert\mathrm{supp}(x)\rvert \leq k$ and $Lx = 1$?	

We reduce PARTITIONED UNIT MULTIPLICATION to $\mathbb{F}$-UNIT MULTIPLICATION as follows. Given an instance $(L_{t\times r}, (U_1, \ldots, U_k), k)$ of PARTITIONED UNIT MULTIPLICA-

TION, we simply output $(L_{t\times r}, k)$ as the equivalent instance of $\mathbb{F}$-UNIT MULTIPLICATION. In one direction, let x be a solution to $(L_{t\times r}, (U_1, \ldots, U_k), k)$. Recall that L is a binary matrix. Thus, since x is a binary vector satisfying $Lx = 1$ over $\mathbb{R}$, it must also satisfy $Lx = 1$ over $\mathbb{F}$. Since $|\mathrm{supp}(x)| \leq k$, we get that x is a solution to $(L_{t\times r}, k)$. In the second direction, let x be a solution to $(L_{t\times r}, k)$. Assume w.l.o.g. that for $s \in [k]$ we have $\mathrm{supp}(L_s) = U_s$. As the given k sets U_i form a partition and $|\mathrm{supp}(x)| \leq k$, for every $s \in [k]$ we have $|\mathrm{supp}(x) \cap U_s| = 1$. Since L is a binary matrix and for every $s \in [k]$ we have $L_s x = 1$ over $\mathbb{F}$, it implies that x is a binary vector. It remains to show that $Lx = 1$ over $\mathbb{R}$. For any index $i \in [t]$, there exist $j, \ell \in [k]$ such that $\mathrm{supp}(L_i) \subseteq U_j \cup U_\ell$. As L and x are both binary, over $\mathbb{R}$ we have $1 \leq L_i x \leq L_j x + L_\ell x \leq 2$. To complete the proof, we claim that $L_i x \neq 2$ over $\mathbb{R}$. Assume that $\mathbb{F} = \mathbb{F}_p$, i.e., the order of the field is some prime number p. On the contrary, assume that $L_i x = 2$ over $\mathbb{R}$. For $L_i x$ to equal 1 over $\mathbb{F}$, we must have $(2 \bmod p) = 1$, which is impossible as $p \geq 2$. This allows us to conclude that $L_i x = 1$ also over $\mathbb{R}$.

In what follows, calculations are performed over $\mathbb{F}$. Next, we reduce $\mathbb{F}$-UNIT MULTIPLICATION to the following variant of the NEAREST CODEWORD problem which is inspired by a reduction from NEAREST CODEWORD to ODD SET of Bonnet, Egri, and Marx [1].

$\mathbb{F}$-NEAREST CODEWORD	PARAMETER: k
INPUT: An $m \times n$ matrix M, an m-dimensional vector b over $\mathbb{F}$, and a nonnegative integer k.	
QUESTION: Is there an n-dimensional vector y over $\mathbb{F}$ such that the Hamming distance between My and b is at most k?	

Given an instance $(L_{t\times r}, k)$ of $\mathbb{F}$-UNIT MULTIPLICATION, construct an instance $(M_{m\times n}, b, k')$ of $\mathbb{F}$-NEAREST CODEWORD as follows. First, let $k' = k$. Now, let M be an $m \times n$ matrix, where $m = r$ and $n = r - \mathsf{rank}(L)$, such that the rows of L form a basis for the subspace orthogonal to the column space of M. Then, an r-dimensional vector v over $\mathbb{F}$ satisfies $Lv = 0$ if and only if v belongs to the column space of M (i.e., there is an n-dimensional vector y over $\mathbb{F}$ such that $My = v$). Finally, let b be an r-dimensional vector such that $Lb = -1$. If no such vector exists, then there is no r-dimensional vector over $\mathbb{F}$ such that $Lv = 1$, which in particular implies that $(L_{t\times r}, k)$ is a NO-instance, and thus we can return a trivial NO-instance of $\mathbb{F}$-NEAREST CODEWORD. Therefore, next assume that b exists. To prove that the reduction is correct, first let x be a solution to $(L_{t\times r}, k)$. Then, $Lx = 1$, and since $Lb = -1$, we have that $L(x + b) = Lx + Lb = Lx - 1 = 0$. Therefore, by the choice of M, there exists an n-dimensional vector y over $\mathbb{F}$ such that $My = (x + b)$. Since $|\mathrm{supp}(x)| \leq k$, we have that the Hamming distance between My and b is at most k, which implies that y is a solution to $(M_{m\times n}, b, k')$. In the other direction, let y be a solution to $(M_{m\times n}, b, k')$. Then, since the Hamming distance between My and b is at most k, there exists an m-dimensional vector x such that $|\mathrm{supp}(x)| \leq k$ and $My = x + b$. Therefore, by the choice of M, $L(x + b) = 0$. Since $Lb = -1$, we get that $Lx = 1$, which implies that x is a solution to $(L_{t\times r}, k)$.

Finally, we reduce $\mathbb{F}$-NEAREST CODEWORD to MATRIX RIGIDITY over $\mathbb{F}$. For this purpose, let $(M_{m\times n}, b, k)$ be an instance of $\mathbb{F}$-NEAREST CODEWORD. We can assume that the columns of M are linearly independent. To see this, let $n' = \mathsf{rank}(M)$ and let M' be the $m \times n'$ submatrix of M whose columns are a column basis of M. Notice that the span of columns of M and M' are exactly the same. For any choice of

vector y, the vector My lies in the column space of M'. Thus it easily follows that the instances (M, b) and (M', b) are equivalent instances of $\mathbb{F}$-Nearest Codeword. Therefore, for the rest of the proof we assume w.l.o.g. that the columns of M are linearly independent. We construct an equivalent instance $(A_{s\times t}, r, k)$ of Matrix Rigidity over $\mathbb{F}$ as follows. Let $s = m$, $r = n$ and $t = (k+1)n + 1$. The matrix A consists of $k+1$ repeated copies of M and b as the last column:

$$A = [\underbrace{M, \ldots, M}_{k+1 \text{ times}}, b].$$

On the one hand, let y be a solution to $(M_{m\times n}, b, k)$. Then, there are at most k entries that should be changed in b to obtain an m-dimensional vector b' over $\mathbb{F}$ such that $My = b'$. In the matrix A, replace the last column b by b'. Denote the resulting matrix by A'. Then, the last column of A' is a linear combination of its other columns, by the construction of A and since $My = b'$. Therefore, $\mathsf{rank}(A') = n$, which implies that $(A_{s\times t}, r, k)$ is a Yes-instance. In the other direction, suppose that $(A_{s\times t}, r, k)$ is a Yes-instance. Then, it is possible to change at most k entries in A and obtain a matrix A' such that $\mathsf{rank}(A') = n$. Since besides the last column of A, A consists of $k+1$ repeated copies of M (i.e., more times than the number of changes), it must be that one copy of M remains unedited in A'. Let b' be the last column of A', as the rank of A' is n, we get that there exists an n-dimensional vector y over $\mathbb{F}$ such that $My = b'$. Since the Hamming distance between b and b' is at most k, we have that y is a solution to $(M_{m\times n}, b, k)$. ∎

REFERENCES

[1] E. Bonnet, L. Egri, and D. Marx, *Fixed-parameter approximability of boolean MinCSPs*, in Proceedings of ESA, 2016, 18.

[2] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*, Springer, New York, 2015.

[3] P. Damaschke and O. Mogren, *Editing simple graphs*, in Proceedings of WALCOM, 2014, pp. 249–260.

[4] A. J. Deshpande, *Sampling-Based Algorithms for Dimension Reduction*, Ph.D. thesis, Massachusetts Institute of Technology, 2007.

[5] R. G. Downey and M. R. Fellows, *Fundamentals of Parameterized Complexity*, Texts Comput. Sci., Springer, New York, 2013.

[6] J. Friedman, *A note on matrix rigidity*, Combinatorica, 13 (1993), pp. 235–239.

[7] D. Grigoriev, *Using the Notions of Separability and Independence for Proving the Lower Bounds on the Circuit Complexity*, Notes to Leningrad Branch of the Steklov Mathematical Institute, Nauka, 1976 (in Russian).

[8] D. Grigoriev, *Using the notions of separability and independence for proving the lower bounds on the circuit complexity*, J. Soviet Math., 14 (1980), pp. 1450–1456.

[9] N. Kayal, *Solvability of a system of bivariate polynomial equations over a finite field*, in Proceedings of ICALP, 2005, pp. 551–562.

[10] A. Kumar, S. V. Lokam, V. M. Patankar, and M. N. J. Sarma, *Using elimination theory to construct rigid matrices*, Comput. Complexity, 23 (2013), pp. 531–563.

[11] K. Lange, *Hadamards determinant inequality*, Amer. Math. Monthly, 121 (2014), pp. 258–259.

[12] S. V. Lokam, *Spectral methods for matrix rigidity with applications to size-depth tradeoffs and communication complexity*, in Proceedings of FOCS, 1995, pp. 6–15.

[13] S. V. Lokam, *On the rigidity of Vandermonde matrices*, Theoret. Comput. Sci., 237 (2000), pp. 477–483.

[14] S. V. Lokam, *Complexity lower bounds using linear algebra*, Found. Trends Theor. Comput. Sci., 4 (2009), pp. 1–155.

[15] M. Mahajan and J. Sarma M.N., *On the complexity of matrix rank and rigidity*, in Proceedings of CSR, 2007, pp. 269–280.

[16] S. M. Meesum, P. Misra, and S. Saurabh, *Reducing rank of the adjacency matrix by graph modification*, in Proceedings of COCOON, 2015, pp. 361–373.

[17] S. M. Meesum and S. Saurabh, *Rank reduction of oriented graphs by vertex and edge deletions*, Algorithmica, 2017, pp. 1–20.
[18] A. A. Razborov, *On Rigid Matrices*, manuscript, 1989 (in Russian).
[19] J. Renegar, *On the computational complexity and geometry of the first-order theory of the reals. Part* I*: Introduction. Preliminaries. The geometry of semi-algebraic sets. The decision problem for the existential theory of the reals*, J. Symbolic Comput., 13 (1992), pp. 255–299.
[20] J. Sarma M.N., *Complexity Theoretic Aspects of Rank, Rigidity and Circuit Evaluation*, Ph.D. thesis, The Institute of Mathematical Sciences, CIT Campus, Taramani, Chennai, 2009.
[21] M. A. Shokrollahi, D. Spielman, and V. Stemann, *A remark on matrix rigidity*, Inform. Process. Lett., 64 (1997), pp. 283–285.
[22] L. Sigler, *Algebra*, Undergrad. Texts Math., Springer, New York, 1976.
[23] L. G. Valiant, *Graph-theoretic arguments in low-level complexity*, in Proceedings of MFCS, 1977, pp. 162–176.