Tight Cell Probe Bounds for Succinct Boolean Matrix-Vector Multiplication

Diptarka Chakraborty* Lior Kamma† Kasper Green Larsen‡

Abstract

The conjectured hardness of Boolean matrix-vector multiplication has been used with great success to prove conditional lower bounds for numerous important data structure problems, see Henzinger et al. [STOC'15]. In recent work, Larsen and Williams [SODA'17] attacked the problem from the upper bound side and gave a surprising cell probe data structure (that is, we only charge for memory accesses, while computation is free). Their cell probe data structure answers queries in $\tilde{O}(n^{7/4})$ time and is succinct in the sense that it stores the input matrix in read-only memory, plus an additional $\tilde{O}(n^{7/4})$ bits on the side. In this paper, we essentially settle the cell probe complexity of succinct Boolean matrix-vector multiplication. We present a new cell probe data structure with query time $\tilde{O}(n^{3/2})$ storing just $\tilde{O}(n^{3/2})$ bits on the side. We then complement our data structure with a lower bound showing that any data structure storing r bits on the side, with $n < r < n^2$ must have query time t satisfying $tr = \tilde{\Omega}(n^3)$. For $r \leq n$, any data structure must have $t = \tilde{\Omega}(n^2)$. Since lower bounds in the cell probe model also apply to classic word-RAM data structures, the lower bounds naturally carry over. We also prove similar lower bounds for matrix-vector multiplication over $\mathbb{F}_2$.

*Computer Science Institute of Charles University, Prague. diptarka@iuuk.mff.cuni.cz. Supported by the funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreement no. 616787.

†MADALGO. Aarhus University. lior.kamma@cs.au.dk. Supported by a Villum Young Investigator Grant.

‡MADALGO. Aarhus University. larsen@cs.au.dk. Supported by a Villum Young Investigator Grant and an AUFF Starting Grant.

1 Introduction

Matrix-vector multiplication is one of the most fundamental algorithmic primitives. In the data structure variant of the problem, we are given an $n \times n$ matrix M as input. The goal is to preprocess M into a data structure, such that upon receiving any n-dimensional query vector v, we can quickly compute Mv. Constructing a data structure instead of computing Mv directly may pay off as soon as we have to answer multiple matrix-vector multiplication queries on the same matrix M.

When defined over the Boolean semiring (with addition replaced by OR and multiplication replaced by AND) the above problem is a special case of the well-known *Online Matrix-Vector (OMV)* problem: Given a matrix $M \in \{0,1\}^{n \times n}$ and a stream of vectors $v^{(1)}, \cdots, v^{(n)} \in \{0,1\}^n$ the goal is to output the value of $Mv^{(i)}$ before seeing $v^{(j)}$ for any $j \in \{i+1, \cdots, n\}$. Henzinger *et al.* [HKNS15] conjectured that the OMV problem cannot be solved by any randomized algorithm with error probability at most $1/3$ within time $O(n^{3-\epsilon})$ for any constant $\epsilon > 0$ (i.e. amortized $O(n^{2-\epsilon})$ per vector $v^{(i)}$). This conjecture is known as the OMV conjecture and is one of the central conjectures in the "Hardness in P" area, along with the Strong Exponential Time Hypothesis (SETH see [IP01]), the 3SUM conjecture (see e.g. [GO95]) and the All Pairs Shortest Paths conjecture (APSP see e.g. [WW10]). The conjecture implies a whole range of near-tight conditional lower bounds for classic (fully/partially) dynamic data structure problems such as dynamic reachability. For the matrix-vector multiplication problem over the Boolean semiring, the OMV conjecture in particular implies that for any polynomial preprocessing time and space, the query time must be $n^{2-o(1)}$ [HKNS15].

The current best upper bound for the OMV problem is due to Larsen and Williams [LW17], who gave a randomized (word-RAM) data structure with a total running time of $n^3/2^{\Omega(\sqrt{\lg n})}$ over a sequence of n queries, i.e. amortized $n^2/2^{\Omega(\sqrt{\lg n})}$ per query. While this new upper bound is non-trivial, it does not violate the OMV conjecture.

The holy grail is, of course, to replace the OMV conjecture by an unconditional $n^{3-o(1)}$ lower bound. In contrast to SETH, 3SUM and APSP, the OMV problem is a data structure problem, rather than an algorithmic problem. Since we have been vastly more successful in proving unconditional lower bounds for data structures than for algorithms, it does not a priori seem completely hopeless to prove a tight unconditional lower bound for OMV in the foreseeable future. Data structure lower bounds are typically proved in the cell probe model of Yao [Yao81]. In this model, computation is free of cost, and the complexity of a data structure is solely the amount of memory it uses and the number of memory accesses it performs on answering a query. In particular, lower bounds proved in the cell probe model apply to data structures developed in the standard word-RAM model, regardless of which unit cost instructions are available. Quite surprisingly, Larsen and Williams [LW17] showed that the performance of their OMV data structure greatly improves if implemented in the cell probe model (i.e. if computation is free and we only charge for accessing memory). Their cell probe data structure for matrix-vector multiplication over the Boolean semiring has a query time of $O(n^{7/4}/\sqrt{w})$, where w denotes the word size (typically $w = \Theta(\lg n)$). Thus the OMV conjecture is *false* in the cell probe model!

But what is then the true complexity of matrix-vector multiplication in the cell probe model? Is it the best one can hope for, namely $O(n/w)$ time (the size of the output measured in words) with $O(n^2)$ bits of space? Or does one have to pay a polynomial factor in either space or time? While not matching the OMV conjecture, a polynomial lower bound ($n^{1+\Omega(1)}$ query time with, say, polynomial space) would still be immensely valuable as it would be the first polynomial lower bound for any data structure problem and would be a huge leap forward in proving unconditional

lower bounds. Furthermore, it would still imply non-trivial polynomial lower bounds for numerous important data structure problems via the reductions already given in previous papers.

Our main contribution is to prove (near-)tight polynomial cell probe lower bounds for the class of *succinct* matrix-vector multiplication data structures. Before formally presenting our results, we survey the current barriers for proving cell probe lower bounds as this will help understand the context of our results.

Cell Probe Lower Bound Barriers. Much effort has gone into developing techniques for proving data structure lower bounds in the cell probe model. For static data structures (like matrix-vector multiplication), the current strongest techniques [Lar12b] can prove lower bounds of $t = \Omega(\lg m / \lg \alpha)$ where t is the query time, m is the number of distinct possible queries in the problem and α is the space-overhead over linear. Thus the strongest previous lower bounds peak at $t = \Omega(\lg m)$ with linear space. For matrix-vector multiplication with an $n \times n$ matrix M (over $\{0,1\}$), there are 2^n possible queries v, thus the strongest possible lower bound current techniques would allow us to prove is $t = \Omega(n)$. This is unfortunately not much more than the trivial $t = \Omega(n/w)$ one would get for just writing the output.

For dynamic data structures, i.e. data structures where one receives both updates to the input data and queries, the current strongest techniques [Lar12a, WY16] give lower bounds of $t = \Omega(\lg m \lg n / \lg^2(uw))$ where u is the update time, w the word-size and n the input size/number of updates performed. This is about a $\lg n$ factor more than for static data structures, thus still leaves us quite far from proving lower bounds close to the conditional ones. If we restrict ourselves to lower bounds for decision problems, i.e. problems where the answer to a query is just one bit, the situation is worse, with the strongest lower bounds being of the form $t = \Omega(\lg m \sqrt{\lg n} / \lg^2(uw))$ [LWY17].

Restricted Data Structures. For many data structure problems, the lower bounds one can prove with previous techniques are in fact tight, see e.g. [FS89, PD06, Păt11, PT11, Lar12a, WY16]. However, as we can see from matrix-vector multiplication, there are also problems where the current techniques are quite far from proving what we believe should be the right lower bound. This has resulted in researchers proving a number of exciting lower bounds for special classes of data structures. For instance, Clifford *et al.* [CGL15] consider matrix-vector multiplication over a finite field $\mathbb{F}_p$ of exponential size $p = 2^{\Theta(n)}$. This results in more queries to the problem ($p^n = 2^{\Theta(n^2)}$) and thereby enabled proving a lower bound of $t = \Omega(n^2 / \lg \alpha)$. Their lower bounds hold when the word size w is big enough to store an element of the field, i.e. $w = \Theta(n)$ bits. An interesting interpretation of the lower bound is that, as long as we do not take advantage of the size of the field, any data structure is bound to use $\Omega(n^2 / \lg \alpha)$ query time. Another line of work has focused on non-adaptive dynamic data structures [BL15, BBK17, RR17]. These are data structures where the memory locations read upon answering a query depend *only* on the query and *not* on the contents of probed cells, i.e. the query algorithm may not branch based on what it reads.

Succinct Data Structures. The last class of data structures we consider are *succinct* data structures. Succinct data structures use space very close to the information theoretic minimum. More formally, we say that a data structure has *redundancy* of r bits if its space usage is $\Pi + r$ bits where Π is the information theoretic minimum for solving the problem and $r = o(\Pi)$. The space usage of succinct data structures is measured only in terms of its *redundancy*. Using succinct data structures may be crucial in applications where memory is scarce. We typically distinguish two types

of succinct data structures, namely *systematic* and *non-systematic* data structures. Systematic data structures are more restricted than non-systematic ones, in the sense that they always store the input in read-only memory and then build an r-bit data structure on the side. Non-systematic data structures just use at most $\Pi + r$ bits (and thus do not have to store the input in the format in which it is given). Succinct data structures have been studied extensively for decades with many fundamental and important upper and lower bounds, see e.g. [Jac88, GM07, Păt08, PV10]. The current strongest technique typically allows one to prove lower bounds of the form $tr = \Omega(\Pi)$, see e.g. [GM07, BL13].

The reason we take special interest in succinct data structures, is that the matrix-vector multiplication data structure by Larsen and Williams [LW17] is, in fact, a succinct data structure. In addition to answering queries in just $O(n^{7/4}/\sqrt{w})$ time, it is systematic and just stores the input matrix as read-only, plus an additional $r = O(n^{7/4}\sqrt{w})$ bits on the side. With the current techniques for proving lower bounds for succinct data structures, we actually have hopes of proving something stronger than the $t = \Omega(n)$ lower bounds we can hope for if we just consider general data structures. Since $\Pi = n^2$ for Boolean matrix-vector multiplication, it seems reasonable to hope for something of the form $tr = \Omega(n^2)$. Such lower bounds would shed interesting new light on this central data structure problem and would bring us closer to understanding the true complexity of matrix-vector multiplication.

1.1 Our Results

Our main results are near-matching upper and lower bounds for systematic succinct data structures solving Boolean matrix-vector multiplication. On the upper bound side, we improve on the results of Larsen and Williams and give a new randomized data structure with the following guarantees:

Theorem 1.1. *Given any matrix $M \in \{0,1\}^{n\times n}$ there exists a systematic succinct data structure $R(M)$ consisting of $r = O(n^{3/2}(\sqrt{w} + \frac{\lg n}{\sqrt{w}}))$ additional bits, and a query algorithm such that, given any $v \in \{0,1\}^n$ it computes Mv over the Boolean semiring with probability $\geq 1 - 1/n$ by probing at most $O(n^{3/2}\sqrt{w}\lg n)$ cells of $R(M)$ and M, where w is the word size.*

Our data structure thus reduces the redundancy from $O(n^{7/4}\sqrt{w})$ bits to $O(n^{3/2}(\sqrt{w} + \frac{\lg n}{\sqrt{w}}))$ bits. Moreover, our randomized query algorithm returns the correct answer with high probability, and improves the query time over the previously known deterministic algorithm from $O(n^{7/4}/\sqrt{w})$ to $O(n^{3/2}\sqrt{w}\lg n)$.

We complement our new upper bound by a near-matching lower bound:

Theorem 1.2. *Assume that for every matrix $M \in \{0,1\}^{n\times n}$ there exists a systematic succinct data structure $R = R(M)$ consisting of at most $r = r(n)$ bits and there is a randomized algorithm that given any $v \in \{0,1\}^n$ computes Mv over the Boolean semiring with probability $\geq 1 - 1/n$ by probing R and at most $t = t(n)$ entries from M, Then for $n \leq r \leq n^2/4$, $t \cdot r = \Omega(n^3)$; otherwise for $r < n$, $t = \Omega(n^2)$.*

Our lower bound comes within polylogarithmic factors of the upper bound and is in fact higher than what we could hope for with previous techniques (recall that previous techniques peak at $tr = \Omega(\Pi)$). The proof of our lower bound exploits the large number m of possible queries and we essentially manage to derive lower bounds of the form $tr = \Omega(\Pi \lg m) = \Omega(n^3)$. Also note that our lower bound allows the data structure to probe all of R, i.e. all the redundant bits, and still

it says that one has to read a lot from the matrix itself. Another exciting point is that our lower bound shows that $t = \Omega(n^2)$ for any $r < n$. Previous lower bounds of $tr = \Omega(\Pi)$ always degenerate linearly in r all the way down to $r = 1$. In contrast, our lower bounds say that one cannot do much better (up to a constant factor) than reading all n^2 entries of M if $r < n$.

Finally, we also study matrix-vector multiplication over $\mathbb{F}_2$. Here we prove lower bounds even for the vector-matrix-vector multiplication where one is given a pair of vectors $u, v \in \mathbb{F}_2^n$ as queries and must compute $u^\intercal Mv$. This problem has just one bit in the output, making it more difficult to prove lower bounds. Nonetheless, we prove the following lower bound:

Theorem 1.3. *Assume that for every matrix $M \in \mathbb{F}_2^{n\times n}$ there exists a data structure $R = R(M)$ consisting of at most $r = r(n)$ bits and there is an algorithm that given $u, v \in \mathbb{F}_2^n$ computes $u^\intercal Mv$ by probing R and at most $t = t(n)$ entries from M, then for $n \le r \le n^2/4$, $t \cdot r = \Omega(n^3/\lg n)$. Moreover, if $r < n$ then $t = \Omega(n^2/\lg n)$.*

We believe it is quite remarkable that we can get $t = \tilde{\Omega}(n^2)$ lower bounds for any $r < n$ for this 1-bit output problem. Since any $u^\intercal Mv$ query can be answered by just taking inner product between u and Mv, as a corollary of the above we also get the same trade-off for matrix-vector problem over $\mathbb{F}_2$. To the best of our knowledge, prior to this result there was no trade-off known for such a small sized field. Our proof is completely information theoretic and based on an encoding argument. It is worth noting that our proof technique can be generalized to give lower bound for the case when the query algorithm may err with probability at most $1/64$ (though any small constant probability will work) on average over the random choices of M, u, v.

Finally, we also consider vector-matrix-vector multiplication over the Boolean semiring. Since one can compute Mv by running the following sequence of n queries: $(e^{(1)})^\intercal Mv, \cdots, (e^{(n)})^\intercal Mv$ where $\{e^{(i)}\}_{i\in[n]}$ is the standard basis over $\{0,1\}^n$, from Theorem 1.2 we get a lower bound of $tr \ge \Omega(n^2)$ for the Boolean $u^\intercal Mv$ problem. Instead of this simple reduction, even if we use the much more elegant reduction in [HKNS15], we will not be able to derive any better lower bound from the above theorem. However in Section 4.2 we show how to extend the proof of Theorem 1.2 to get a $t = \Omega(n/\lg n)$ bound on the worst case number of probes into M for the Boolean vector-matrix-vector problem with $r \le n^2/4$.

2 Preliminaries

Notations. For $k \in \mathbb{N}$, let $[k]$ denote the set $\{1, 2, \ldots, k\}$. For every $v \in \{0,1\}^n$ and $i \in [n]$, let v_i denote the i-th entry of v. All the logarithms we consider are over base 2. We use the notation $x \in_R \mathcal{X}$ to denote that x is drawn uniformly at random from the domain $\mathcal{X}$.

Information Theory. Throughout this paper we use several basic definitions and notations from information theory. For further exposition readers may refer to any standard textbook on information theory (e.g. [CT06]).

Let X, Y be discrete random variables on a common probability space. Let $p(x), p(y), p(x,y)$ denote $\Pr[X = x], \Pr[Y = y], \Pr[X = x, Y = y]$ respectively. The *entropy* of X is defined as $H(X) := -\sum_x p(x) \lg p(x)$. The *joint entropy* of (X, Y) is defined as $H(X,Y) := -\sum_{(x,y)} p(x,y) \lg(p(x,y))$. The *mutual information* between X and Y is defined as $I(X;Y) := \sum_{(x,y)} p(x,y) \lg \frac{p(x,y)}{p(x)p(y)}$ and the *conditional entropy* of Y given X is defined as $H(Y \mid X) := H(Y) - I(X;Y)$.

Proposition 2.1 (Chain Rule of Entropy)**.** *Then* $H(X,Y) = H(X) + H(Y \mid X)$.

The seminal work of Shannon [Sha48] establishes a connection between the entropy and the expected length of an optimal code encoding a random variable.

Theorem 2.2 (Shannon's Source Coding Theorem [Sha48])**.** *Let* X *be a discrete random variable over domain* $\mathcal{X}$*. Then for every uniquely decodable code* $C : \mathcal{X} \to \{0,1\}^*$*,* $\mathbb{E}(|C(X)|) \geq H(X)$*. Moreover, there exists a uniquely decodable code* $C : \mathcal{X} \to \{0,1\}^*$ *such that* $\mathbb{E}(|C(X)|) \leq H(X) + 1$*.*

3 Upper Bound for Boolean Matrix-Vector Problem

In this section we prove Theorem 1.1 by introducing an efficient cell probe data structure for solving Boolean matrix-vector problem (with high probability). Let us first recall the theorem.

Theorem 1.1. *Given any matrix* $M \in \{0,1\}^{n \times n}$ *there exists a data structure* R *consisting of* $O(n^{3/2}(\sqrt{w} + \frac{\lg n}{\sqrt{w}}))$ *bits, and a query algorithm such that, given* $v \in \{0,1\}^n$ *it computes* Mv *with high probability by probing at most* $O(n^{3/2}\sqrt{w}\lg n)$ *cells of* R *and* M*, where* w *is the word size.*

Preprocessing. In what follows, we present an algorithm that, given a matrix $M \in \{0,1\}^{n \times n}$ constructs the data structure guaranteed in Theorem 1.1. Loosely speaking, the data structure is composed of a list $\mathcal{L}$ consisting of pairs (I, J), and an encoding $\mathcal{E}$ of all the entries $(i,j) \in \bigcup_{(I,J)\in\mathcal{L}}(I \times J)$ such that $M_{(i,j)} = 1$. The key step of the preprocessing algorithm is deciding which set-pairs to add to the list. Informally, going over all possible pairs (I, J), the algorithm adds a pair (I, J) to $\mathcal{L}$, if there exists a large subset of $I \times J$ of entries not "covered" by the pairs already added in the list, and such that the "uncovered" part of the submatrix $M_{I,J}$ contains "few" 1-entries per-row. The algorithm is formally described as Algorithm 1.

1: let $\mathcal{L} \leftarrow \emptyset$.
2: **for all** $(I, J) \in 2^{[n]} \times 2^{[n]}$ **do**
3: let $\mathcal{U} := \bigcup_{(I',J')\in\mathcal{L}}(I' \times J')$.
4: add (I, J) to $\mathcal{L}$ if all of the following conditions hold.

1. $|(I \times J) \setminus \mathcal{U}| \geq \frac{n^{3/2}}{\sqrt{w}}$; and
2. $\forall i \in I$, $\Pr_{j\in J:(i,j)\notin\mathcal{U}}[M_{(i,j)} = 1] \leq \frac{1}{\sqrt{nw}}$.

5: let $\mathcal{E}$ be the list of all $(i,j) \in \bigcup_{(I,J)\in\mathcal{L}}(I \times J)$ such that $M_{(i,j)} = 1$.
6: **return** $(\mathcal{L}, \mathcal{E})$

Algorithm 1: Preprocessing M

The following claim implies the first part of Theorem 1.1.

Claim 3.1. *The string* $(\mathcal{L}, \mathcal{E})$ *can be encoded using at most* $O(n^{3/2}(\sqrt{w} + \frac{\lg n}{\sqrt{w}}))$ *bits.*

Proof. Observe the conditions in line 4 of the algorithm. Due to condition 1, there are at most $n^{1/2}\sqrt{w}$ many different pairs in the list $\mathcal{L}$, and each pair can be encoded using only $2n$ bits (an indicator bit per row and column). Therefore $\mathcal{L}$ can be encoded using at most $O(n^{3/2}\sqrt{w})$ bits.

Condition 2 asserts that the density of 1-entries in the submatrix covered by all the subsets of rows and columns listed in $\mathcal{L}$ is at most $\frac{1}{n^{1/2}\sqrt{w}}$. Each such entry can be encoded using $2\lg n$ bits and hence $\mathcal{E}$ can be encoded using at most $O(n^{3/2}\lg n/\sqrt{w})$ bits. □

Answering queries. To prove the second part of the theorem, we give a query algorithm that receives $v \in \{0,1\}^n$ and gets access to M, as well as to $\mathcal{L}$ and $\mathcal{E}$, and computes Mv with high probability. Let $J = \{j \in [n] : v_j = 1\}$ be the set of columns of M which are relevant for computing Mv, and let $\mathcal{U} := \bigcup_{(I',J')\in\mathcal{L}}(I' \times J')$ be the set of all matrix indices that appear in $\mathcal{L}$. Starting with the set $I = [n]$ of all possible rows, the algorithm "prunes" I throughout the execution. Whenever an index i is removed from I, the algorithm fixes $u_i \in \{0,1\}$. During the first step, the algorithm goes over $\mathcal{E}$. If for some $i \in I$, there exists $j \in J$ such that (i,j) is encoded in $\mathcal{E}$, the algorithm sets $u_i = 1$ and removes i from I. During the second step, for every $i \in I$ the algorithm samples $2\sqrt{n}\lg n$ entries (i,j) from the set $(\{i\} \times J) \setminus \mathcal{U}$. If $M_{(i,j)} = 1$ for at least one of these entries, the algorithm sets $u_i = 1$ and removes i from I. During the third step, the algorithm examines the set $\mathcal{R} := \{(i,j) : i \in I \ , \ j \in J \ \textit{and} \ (i,j) \notin \mathcal{U}\}$ of remaining entries. If this set has more than $O(n^{3/2}\lg n/\sqrt{w})$ elements, the algorithm reports "failure". Otherwise for every $i \in I$, the algorithm probes all entries $(i,j) \in \mathcal{R}$. If $M_{(i,j)} = 1$ for at least one of these entries, the algorithm sets $u_i = 1$ and removes i from I. Otherwise, the algorithm sets $u_i = 0$ and removes i from I. The algorithm terminates either by reporting "failure" or by returning $u = (u_1, \ldots, u_n)$. It is formally described as Algorithm 2.

```
1: let I ← [n], J ← {j ∈ [n] : v_j = 1}.
2: let U ← ⋃_{(I',J')∈L}(I' × J').
3: for all (i,j) encoded in E do
4:    if i ∈ I and j ∈ J then
5:       set u_i = 1 and let I ← I \ {i}.
6: for all i ∈ I do
7:    sample uniformly (with repetition) 2√(nw) lg n entries from {j ∈ J : (i,j) ∉ U}.
      denote the set of sampled entries J''(i).
8:    if there exists j ∈ J''(i) such that M_(i,j) = 1 then
9:       set u_i ← 1 and let I ← I \ {i}.
10: let R ← (I × J) \ U.
11: if |R| ≥ n^{3/2}/√w then
12:    return "failure"
13: for all i ∈ I do
14:    if there exists j ∈ J such that (i,j) ∈ R and M_(i,j) = 1 then
15:       set u_i ← 1 and let I ← I \ {i}.
16:    else
17:       set u_i ← 0 and let I ← I \ {i}.
18: return u = (u_1, ..., u_n)
```

Algorithm 2: Querying Mv

We will first show that the algorithm probes "few" bits. Since $(\mathcal{L},\mathcal{E})$ can be encoded using at most $O(n^{3/2}(\sqrt{w}+\frac{\lg n}{\sqrt{w}}))$ bits, and since the algorithm samples at most $2\sqrt{nw}\lg n$ entries from each row of the matrix, we conclude the following.

Lemma 3.2. *Algorithm 2 probes at most $O(n^{3/2}\sqrt{w}\lg n)$ bits of $M, \mathcal{L}, \mathcal{E}$ throughout the execution.*

To finish the proof of Theorem 1.1 we show that with high probability, the algorithm returns the correct answer. To this end, fix $v \in \{0,1\}^n$, and consider an execution of Algorithm 2 on v. Let I_1, I_2 be the set I after the first step of the algorithm (lines 3-5), and the second step of the algorithm (lines 6-9) respectively. In these notations, $\mathcal{R} \leftarrow (I_2 \times J) \setminus \mathcal{U}$. Finally, let

$$I_1^* := \left\{ i \in I_1 : \Pr_{j \in J:(i,j) \notin \mathcal{U}}[M_{(i,j)} = 1] > \frac{1}{\sqrt{nw}} \right\} .$$

Lemma 3.3. *Algorithm 2 fails with probability at most $\frac{1}{n}$. Moreover, if the algorithm does not fail, then it returns $Mv = u$.*

Proof. Let $\mathcal{F}$ be the event $I_2 \subseteq I_1 \setminus I_1^*$. By definition of I_1^* we get that

$$\Pr[\mathcal{F}] \geq 1 - \sum_{i \in I_1^*} \Pr[\forall j \in J''(i).\ M_{(i,j)} = 0] \geq 1 - \sum_{i \in I_1^*} \left(1 - \frac{1}{\sqrt{nw}}\right)^{2\sqrt{nw}\lg n} \geq 1 - \frac{1}{n}$$

Conditioned on $\mathcal{F}$ occurring, for every $i \in I_2$, $\Pr_{j \in J:(i,j) \notin \mathcal{U}}[M_{(i,j)} = 1] \leq \frac{1}{\sqrt{nw}}$. Since $I_2 \times J \not\subseteq \mathcal{U}$, then $(I_2, J) \notin \mathcal{L}$. By the construction of $\mathcal{L}$ we therefore conclude that $|(I_2 \times J) \setminus \mathcal{U}| < \frac{n^{3/2}}{\sqrt{w}}$, and the algorithm does not fail. We will show next that if the algorithm does not fail, then for every $i \in [n]$, $u_i = [Mv]_i$. First note that if $i \notin I_2$, then the algorithm finds $j \in J$ such that $M_{(i,j)} = 1$, and therefore $u_i = 1 = [Mv]_i$. Otherwise, assume $i \in I_2$. Then for every $j \in J$, if $(i,j) \in \mathcal{U}$, then $M_{ij} = 0$, since otherwise (i,j) would be encoded in $\mathcal{E}$ and removed during the first step of the execution. Therefore, for every $j \in J$, if $M_{(i,j)} = 1$ then $(i,j) \in (I_2 \times J) \setminus \mathcal{U}$. Since the algorithm does not fail, it goes over all entries in $(I_2 \times J) \setminus \mathcal{U}$, and therefore $[Mv]_i = 1$ if and only if there exists $j \in J$ such that $M_{(i,j)} = 1$, which in turn implies $u_i = 1$. □

4 Matching Lower Bound on Boolean Matrix-Vector Problem

In this section we consider the Boolean Mv problem, where $Mv = (\vee_{j \in [n]}(M_{(i,j)} \wedge v_j))_{i \in [n]}$, and prove Theorem 1.2. The presented bound matches the upper bound shown in the last section up to some (small) polylogarithmic factor. Although Theorem 1.2 allows the query algorithm to be randomized that returns right answer with high probability, for the sake of simplicity we first focus on the deterministic regime. In Section 4.1 we refine the proof to hold for randomized query algorithms.

Theorem 4.1. *Assume that for every matrix $M \in \{0,1\}^{n \times n}$ there exists a data structure $R = R(M)$ consisting of at most $r = r(n)$ bits and there is an algorithm that given any $v \in \{0,1\}^n$ computes Mv by probing R and at most $t = t(n)$ entries from M. Then for $n \leq r \leq n^2/4$, $t \cdot r = \Omega(n^3)$; otherwise for $r < n$, $t = \Omega(n^2)$.*

To prove the theorem, we will define a family $\mathcal{M} \subseteq \{0,1\}^{n \times n}$ of matrices, and a set of queries $v^{(1)}, \ldots, v^{(4r/n)} \in \{0,1\}^n$ such that the following holds for every $M \in \mathcal{M}$. (1) The set of answers to the queries $Mv^{(1)}, \ldots, Mv^{(4r/n)}$ hold a large amount of information on M; and (2) one can succinctly "encode" the execution of the query algorithm on the respective sequence of queries.

That is, there exists a short (in terms of t, r) string $\mathcal{E}$ such that given $\mathcal{E}, R$, one can emulate the query algorithm over the sequence of queries, and return the answers $Mv^{(1)}, \ldots, Mv^{(4r/n)}$.

For the rest of the section, we additionally assume $n \leq r \leq \sqrt{n^3/4}$. The proof for the case $\sqrt{n^3/4} \leq r \leq n^2/4$ is similar, and the differences will be discussed towards the end of the proof.

A Family of Input Matrices. Let $\mathcal{M} \subseteq \{0,1\}^{n \times n}$ be the family of all matrices $M \in \{0,1\}^{n \times n}$ with the following property: If each row of M is divided into r/n contiguous *blocks* each containing n^2/r consecutive entries, then exactly one entry in each block is 1, and the rest are all 0s. Consider a matrix $M \in \mathcal{M}$. Each of the r blocks in M contains exactly one 1-entry out of n^2/r entries. The following claim is thus implied from the definition of entropy.

Claim 4.2. *Let $M \in_R \mathcal{M}$. Then $H(M) = r \lg \frac{n^2}{r}$.*

Encoding argument. Consider the following sequence of $4r/n$ vectors in $\{0,1\}^n$. For every $m \in [4r/n]$ define $v^{(m)} \in \{0,1\}^n$ such that $v^{(m)}_j = 1$ if and only if

$$j (\mathrm{mod}\, \tfrac{n^2}{r}) \in \{\frac{(m-1)n^3}{4r^2} + 1, \ldots, \frac{mn^3}{4r^2}\} .$$

Fix some $M \in \mathcal{M}$. When querying for $Mv^{(1)}, \ldots, Mv^{(4r/n)}$, the algorithm reads at most $4tr/n$ bits from M. Observe that one can trivially encode all the probed entries using $\frac{4tr}{n}(2 \lg n + 1)$ bits by specifying the indices and the entry values. In turn, this encoding implies $t \geq \Omega(n/\lg n)$. However, by employing a subtler argument inspired by [BL13], we provide a much better bound on t and r, which also implies the simpler one.

To this end, let b, k denote the total number of different entries and the number of different 1-entries read by the algorithm throughout this sequence of $4r/n$ many Mv queries respectively. Then $k \leq b \leq 4tr/n$. Let B be the sequence of b different entries probed by the query algorithm, when queried for $Mv^{(1)}, \ldots, Mv^{(4r/n)}$, in the order they are probed. That is, $B : [b] \hookrightarrow [n] \times [n]$ is injective. Let $K := \{j \in [b] : M_{B(j)} = 1\} \subseteq [b]$. Define $\mathcal{E}$ to be the bit-string composed of the three following sub-strings. The first $\lg(4tr/n)$ bits of $\mathcal{E}$ encode b. The next $\lg(4tr/n)$ bits encode k. The remaining bits encode K as a subset of $[b]$. Since $|K| = k$, the following is straightforward.

Claim 4.3. *$\mathcal{E}$ can be encoded using at most $\lg \binom{4tr/n}{k} + 2 \lg(4tr/n)$ bits.*

Next, we show that $\mathcal{E}$ and R hold a "large amount" of information of M.

Lemma 4.4. *The bit-string $\mathcal{E}$ encodes the subset of k 1-entries among all the entries probed. Furthermore, given access to $\mathcal{E}$ and R one can answer all the queries $Mv^{(1)}, \ldots, Mv^{(4r/n)}$.*

Proof. First note, that given $\mathcal{E}$, one can directly decode b, k and K. We next show an emulation algorithm that, given access to R, b, k, K finds the k 1-entries in question, and moreover, answers all the queries $Mv^{(1)}, \ldots, Mv^{(4r/n)}$. The algorithm emulates the query algorithm for $Mv^{(1)}, \ldots, Mv^{(4r/n)}$. Whenever the query algorithm probes a matrix entry, the emulation algorithm feeds it with an answer as follows. If the entry has been previously probed during the execution, the query algorithm is fed with the same answer. Otherwise, the answer is determined to be 1 or 0 by whether the index of the matrix entry in the sequence of probes is in K or not respectively, and the answer is stored for later probes. The algorithm is formally described as Algorithm 3.

```
1: let j ← 1.
2: let B^alg = ∅.
3: for i = 1, 2, ..., 4r/n do
4:    run the query algorithm for Mv^(i)
5:    whenever the query algorithm probes a matrix entry (p, q) ∈ [n] × [n]
6:    if there exists j_0 < j such that B^alg[j_0] = (p, q) then
7:       ℓ ← j_0
8:    else
9:       B^alg[j] ← (p, q), ℓ ← j and j ← j + 1.
10:   if ℓ ∈ K then
11:      feed the query algorithm with M_(p,q) = 1.
12:   else
13:      feed the query algorithm with M_(p,q) = 0.
```

Algorithm 3: Emulating a Sequence of Queries

To prove the lemma, it is enough to show that the emulation algorithm always feeds the query algorithm with the correct answer. Denote the sequence of entry probes (with repetitions) performed by the query algorithm by $\{(p_m, q_m)\}_{m=1}^{4tr/n}$ (we may assume for simplicity that the query algorithm performs exactly t entry probes for each query). The crux of the argument is that for all $j \in [b]$, $B^{alg}[j] = B(j)$. We prove the claim by induction on m. Clearly, during the first probe, $j = 1$ and therefore the condition in line 6 of Algorithm 3 is false. Therefore the algorithm sets $B^{alg}[1]$ to be (p_1, q_1), which equals $B(1)$. Moreover, the emulation algorithm answers 1 if and only if $1 \in K$, which occurs if and only if $M_{(p_1,q_1)} = 1$. Assuming correctness for all $m' < m$, we will prove correctness for m. If there exists $j_0 < j$ such that $B^{alg}[j_0] = (p_m, q_m)$, then this entry probe has already been answered correctly by the induction hypothesis. Since the emulation algorithm gives the same answer as before, the answer is the correct one. Otherwise, this entry has not been probed yet, and therefore $B^{-1}((p_m, q_m)) = j$. The emulation algorithm then answers 1 if and only if $j \in K$, which happens if and only if $M_{(p_m,q_m)} = 1$. Therefore the emulation algorithm always gives the correct answer, thus finding the correct set of k 1-entries and answering all the queries $Mv^{(1)}, \ldots, Mv^{(4r/n)}$. □

The next lemma states that, in addition to learning the k 1-entries of M, by answering all the queries $Mv^{(1)}, \ldots, Mv^{(4r/n)}$, we can learn a lot of information about the other blocks of M.

Lemma 4.5. *Let $M \in_R \mathcal{M}$. Then $H(M|R, \mathcal{E}) \leq (r - k) \lg \frac{n^2}{4r}$.*

Proof. By Lemma 4.4, given access to only $\mathcal{E}$ and R, we can answer $Mv^{(1)}, \ldots, Mv^{(4r/n)}$, thus finding k many 1-entries of M. Moreover, for each block in M, the algorithm finds at least $3n^2/4r$ many 0-entries.

By emulating the query algorithm for $Mv^{(1)}, \ldots, Mv^{(4r/n)}$, an algorithm can conclude from $\mathcal{E}$ and R the exact locations of the k many 1-entries. Next, let $i \in [n]$ and let $m \in [4r/n]$. Note that if $[Mv^{(m)}]_i = 0$, then for each block in the i-th row of M, all the entries corresponding to 1-entries in $v^{(m)}$ must be 0. Since there are exactly $n^3/4r^2$ such entries, and since $[Mv^{(m)}]_i = 1$ for at most r/n values of m, the algorithm learns at least $\frac{3r}{n} \cdot \frac{n^3}{4r^2} = \frac{3n^2}{4r}$ 0-entries in each block of M. Now the lemma follows from the Shannon's source coding theorem. □

We now turn to finish the proof of Theorem 4.1.

Proof of Theorem 4.1. First observe that $r + \lg\binom{4tr/n}{k} + 2\lg(4tr/n) \geq H(R,\mathcal{E}) \geq I(M;(R,\mathcal{E})) = H(M) - H(M|R,\mathcal{E})$. By Lemma 4.5, the right hand side is at least $r\lg\frac{n^2}{r} - (r-k)\lg\frac{n^2}{4r} = 2r + k\lg\frac{n^2}{4r}$. Rearranging we get that $\lg\binom{4tr/n}{k} + 2\lg(4tr/n) - k\lg\frac{n^2}{4r} \geq r$. Since $\lg\binom{4tr/n}{k} \leq k\lg\frac{4etr}{kn}$, we get that $k\lg\frac{16etr^2}{n^3k} + 2\lg(4tr/n) \geq r$, and as for every $x, \alpha > 0$, $x\lg\frac{\alpha}{x} \leq \alpha/2$ we have $\frac{8etr^2}{n^3} + 2\lg(4tr/n) \geq r$, or $tr \geq \Omega(n^3)$. This completes the proof for the case $n \leq r \leq \sqrt{n^3/4}$.

The proof for $\sqrt{n^3/4} \leq r \leq n^2/4$ is similar. The only difference is that in this case we choose $4r/n$ vectors in a slightly different way. For every $m \in [n^2/r]$ and $i \in [4r^2/n^3]$, we define a vector $v^{(m,i)}$ by setting $v^{(m,i)}_j = 1$ if and only if $j(\mathrm{mod}\ \frac{n^2}{r}) = m$ and $(i-1)n^4/4r^2 + 1 \leq j \leq in^4/4r^2$. The proof then follows in an analogous manner.

For the second part of the theorem, i.e. when $r < n$, we use a simple padding argument. If $r < n$, create a new data structure R' by appending some arbitrary bits to R such that R' contains exactly n bits. Now from the previous argument it follows that $t \geq \Omega(n^2)$. □

4.1 Lower Bound for Randomized Query Algorithms

In this section we will extend the lower bound results shown previously to the case of randomized query algorithms, thus completing the proof of Theorem 1.2. Assume that given a matrix $M \in \{0,1\}^{n\times n}$ there exists a data structure $R = R(M)$ consisting of at most $r = r(n)$ bits and there exists a *randomized* algorithm that, given $v \in \{0,1\}^n$ returns Mv with probability $\geq 1 - \frac{1}{n}$ by probing R and at most $t = t(n)$ entries from M.

Theorem 1.2. *If $n \leq r \leq n^2/4$, then $t \cdot r \geq \Omega(n^3)$; otherwise for $r < n$, $t \geq \Omega(n^2)$.*

The proof will employ similar arguments to the proof of Theorem 4.1. We will therefore focus only on the case $n \leq r \leq \sqrt{n^3/4}$. Consider the vectors $v^{(1)}, \ldots, v^{(4r/n)}$ from the proof of Theorem 4.1. Fix some $M \in \mathcal{M}$, and run the query algorithm on $Mv^{(1)}, \ldots, Mv^{(4r/n)}$. For every $j \in [4r/n]$, denote the string of random bits used by the algorithm when queried for $Mv^{(j)}$ by x_j, and let $X := \langle x_1, \ldots, x_{4r/n}\rangle$. Let F denote the indicator for the event that the query algorithm failed answering at least one of the queries. Applying union bound, $p := \Pr[F = 1] \leq 1/3$. Let B, b, K, k be as in the proof of Theorem 4.1. If $F = 0$, we encode the string $\mathcal{E}$ the same way as in the previous proof and append 0 as its first bit, whereas if $F = 1$ we let $\mathcal{E}$ be an encoding of M and append 1 as its first bit. In a similar manner to the one presented in Algorithm 3, given $R, \mathcal{E}$ and X, one can emulate the sequence of queries $Mv^{(1)}, \ldots, Mv^{(4r/n)}$ performed by the query algorithm, while using $x_1, \ldots, x_{4r/n}$ as random strings respectively. By computing $H(F, M \mid R, \mathcal{E}, X)$ in two different ways we get the following.

$$\begin{aligned} H(F, M \mid R, \mathcal{E}, X) &= H(M \mid R, \mathcal{E}, X) + H(F \mid M, R, \mathcal{E}, X) \\ &= H(F \mid R, \mathcal{E}, X) + H(M \mid F, R, \mathcal{E}, X) \end{aligned}$$

Note that $H(F \mid R, \mathcal{E}, X) \leq H(F) < 1$ and $H(F \mid M, R, \mathcal{E}, X) = 0$, since given $\mathcal{E}$ one learns also F. Rearranging we get that

$$H(M \mid R, \mathcal{E}, X) \leq 1 + H(M \mid F, R, \mathcal{E}, X)\ .$$

$$
\begin{aligned}
H(M \mid R, \mathcal{E}, X) &\leq 1 + H(M \mid F, R, \mathcal{E}, X) \\
&= 1 + p \cdot H(M \mid F = 1, R, \mathcal{E}, X) + (1-p) \cdot H(M \mid F = 0, R, \mathcal{E}, X) \qquad (1) \\
&\leq 1 + 0 + (1-p)(r-k) \lg \tfrac{n^2}{4r}
\end{aligned}
$$

where the last inequality follows from the same arguments as in Lemma 4.5.

Proof of Theorem 1.2. First observe that from the Shannon's source coding theorem,

$$
H(\mathcal{E}) \leq (1-p)\left(\lg \binom{4tr/n}{k} + 2 \lg \frac{4tr}{n}\right) + pr \lg \frac{n^2}{r}
$$

and hence

$$
\begin{aligned}
r + (1-p)\left(\lg \binom{4tr/n}{k} + 2 \lg \frac{4tr}{n}\right) + pr \lg \frac{n^2}{r} &= H(R) + H(\mathcal{E}) \\
&\geq H(R, \mathcal{E}) \\
&\geq I((M,X);(R,\mathcal{E})) = H(M,X) - H(M,X|R,\mathcal{E}) \; .
\end{aligned}
$$

Since M, X are independent, we get that the r.h.s is at least $H(M) - H(M \mid R, \mathcal{E}, X)$. From (1) we get that this is at least

$$
r \lg \tfrac{n^2}{r} - \left(1 + (1-p)(r-k) \lg \tfrac{n^2}{4r}\right) = pr \lg \tfrac{n^2}{r} + 2(1-p)r + (1-p)k \lg \tfrac{n^2}{4r} - 1 \; .
$$

Rearranging we get that

$$
\lg \binom{4tr/n}{k} + 2 \lg(4tr/n) - k \lg \tfrac{n^2}{4r} \geq 2r - r/(1-p) - 1/(1-p) \geq r/3
$$

By the same arguments as in the proof of Theorem 4.1, we get that $\frac{8etr^2}{n^3} + 2 \lg(4tr/n) \geq r/3$, or $tr \geq \Omega(n^3)$. □

4.2 Lower Bounding Boolean Vector-Matrix-Vector Problem

In this section we extend the technique from Section 4 to get a lower bound on Boolean $u^\intercal M v$ problem, albeit a weaker one. Assume that given a matrix $M \in \{0,1\}^{n \times n}$, there exists a data structure $R = R(M)$ containing at most $r = r(n)$ bits, and there exists an algorithm that, given $u, v \in \{0,1\}^n$ returns $u^\intercal M v = (\vee_{i,j \in [n]} (M_{(i,j)} \wedge u_i \wedge v_j))$ while probing only R and at most $t = t(n)$ bits from M.

Let us first observe an easy corollary of Theorem 1.2. Since answer of each Mv query can be derived from the following sequence of n queries: $(e^{(1)})^\intercal Mv, \cdots, (e^{(n)})^\intercal Mv$ where $\{e^{(i)}\}_{i \in [n]}$ is the standard basis over $\{0,1\}^n$, we get an lower bound of $tr \geq \Omega(n^2)$ for the Boolean $u^\intercal M v$ problem for $n \leq r \leq n^2/4$. In this section we will prove a better lower bound for Boolean $u^\intercal M v$ problem as stated in the following theorem.

Theorem 4.6. *If $r \leq n^2/4$ then $t \geq \Omega(n/\lg n)$.*

We prove the above theorem for $n \leq r \leq \sqrt{n^3/4}$. The proof extends for all $r < n^2/4$ by arguments similar to those used in the proof of Theorem 4.1. First note, that when comparing with the matrix-vector problem, the main caveat of the vector-matrix-vector problem is that each query results in exactly one bit, rather than n bits. Loosely speaking, we have observed that by querying $Mv^{(1)}, \ldots, Mv^{(4r/n)}$, where $v^{(1)}, \ldots, v^{(4r/n)}$ are defined as before, we gain a lot of information about M. This approach seems less beneficial when concerning vector-matrix-vector queries. More specifically, it seems that we need n times more queries to get the same amount of information. By using the trivial argument demonstrated right before Theorem 4.6 we can not get our claimed bound. A more subtle observation into the proof of Lemma 4.4 shows that we can get a lot of information when the answer to the query is 0. Particularly, assume $u^\intercal Mv = 0$. Then we know that whenever $u_i = v_j = 1$, $M_{(i,j)} = 0$. In what follows, we will need the following notation.

Notation 1. *Let $x \in \{0,1\}^n$. Suppose $\overline{x} \in \{0,1\}^n$ denotes the* complement *of x. That is, $\overline{x}_j = 1$ if and only if $x_j = 0$ for every $j \in [n]$.*

Clearly, $\overline{x}^\intercal x = 0$, and moreover, $\overline{x}$ is the unique heaviest vector (in terms of Hamming weight) satisfying this property.

Lemma 4.7. *There exists a bit-string $\mathcal{E} = \mathcal{E}(M)$, such that $\mathcal{E}$ can be encoded using at most $\frac{4tr}{n}(2\lg n + 1)$ bits, and moreover, given access to $\mathcal{E}$ and R one can answer of $Mv^{(1)}, \ldots, Mv^{(4r/n)}$.*

Proof. For every $j \in [4r/n]$, let $u^{(j)} := \overline{Mv^{(j)}}$. Take $\mathcal{E}$ to be the encoding of the list of all entries of M probed throughout the sequence of queries $(u^{(1)})^\intercal Mv^{(1)}, \cdots, (u^{(4r/n)})^\intercal Mv^{(4r/n)}$, in the order they are probed. For simplicity, we may assume that the query algorithm probes exactly t matrix entries for each query. For each entry we encode it's location in the matrix (using $2\lg n$ bits), and its value (one more bit). Then $\mathcal{E}$ is composed of $4r/n$ "segments" of t entries each. Clearly $\mathcal{E}$ can be encoded using at most $\frac{4tr}{n}(2\lg n + 1)$ bits. Let $j \in [4r/n]$. Now we provide an emulation algorithm that, given access to $\mathcal{E}$ and R, finds $u^{(j)}$, and thus finds $Mv^{(j)} = \overline{u^{(j)}}$. The emulation algorithm starts by setting $u^{(*)}$ to be all 0 vector and goes over all $u \in \{0,1\}^n$ and emulates the query algorithm for $u^\intercal Mv^{(j)}$. Whenever the query algorithm probes an entry of M, the emulation algorithm looks for the encoding of this entry in $\mathcal{E}$. If it finds this entry, it feeds it to the query algorithm. Otherwise, it breaks and continues to the next $u \in \{0,1\}^n$. If the query algorithm terminates, and the answer is 0, then the algorithm compares the Hamming weight of u (denoted as $wt(u)$) with that of $u^{(*)}$. If $wt(u) > wt(u^{(*)})$ the algorithm replaces $u^{(*)}$ with u. The algorithm is formally given as Algorithm 4.

It is straightforward that $u^{(*)}$ is the unique $u \in \{0,1\}^n$ that satisfies the following.

1. Every entry probed by the query algorithm when querying $u^\intercal Mv^{(j)}$ is encoded in $\mathcal{E}$ (thus given access to $\mathcal{E}, R$, one can answer $u^\intercal Mv^{(j)}$);
2. $u^\intercal Mv^{(j)} = 0$; and
3. u is of maximal Hamming weight.

Therefore, $u^{(*)} = u^{(j)}$. □

We can now prove Theorem 4.6 using similar arguments to those in the proof of Theorem 4.1. Suppose while probing entries of the matrix M we read total k 1-entries. Then by following the

let $u^{(*)} \leftarrow 0^n$.
for all $u \in \{0,1\}^n$ **do**
  run the query algorithm for $u^\intercal M v^{(j)}$
  whenever the query algorithm probes a matrix entry $(p,q) \in [n] \times [n]$
  if (p,q) is encoded in $\mathcal{E}$ **then**
    feed the query algorithm with the corresponding bit in $\mathcal{E}$ as $M_{(p,q)}$.
  else
    stop the query algorithm.
  if the query algorithm outputs $u^\intercal M v^{(j)} = 0$ and $wt(u) > wt(u^{(*)})$ **then**
    $u^{(*)} \leftarrow u$.
return $u^{(*)}$

Algorithm 4: Finding $u^{(j)}$

argument of the proof of Theorem 4.1, we get that

$$\begin{aligned} r + \tfrac{4tr}{n}(2\lg n + 1) &\geq H(R,\mathcal{E}) \geq H(M) - H(M|R,\mathcal{E}) \\ &\geq 2r + k \lg \tfrac{n^2}{4r} \geq 2r. \end{aligned}$$

Now by rearranging the terms, we get that $t \geq \Omega(n/\lg n)$. □

Using the argument similar to that in Section 4.1 we can also extend Theorem 4.6 to randomized query algorithms.

Theorem 4.8. *Assume that for every matrix $M \in \{0,1\}^{n \times n}$ there exists a data structure R consisting of at most $r = r(n)$ bits and a query algorithm that can answer any $u^\intercal M v$ query with error probability at most $1/n$ by probing R and at most $t = t(n)$ entries of M. Then for $r \leq n^2/4$, $t \geq \Omega(n/\lg n)$.*

Note: Though in both Theorem 1.2 and Theorem 4.8 we consider the error probability to be at most $1/n$, one can easily generalize the results for error probability to be any $1/n < \epsilon < 1$. However we will lose extra $\lg n$ factor in the lower bound. More specifically, given any randomized algorithm $\mathcal{A}$ with probability of error ϵ, we can boost the success probability to $(1 - 1/n)$ by repeating $\mathcal{A}$ $O(\lg n/\lg(\frac{1}{\epsilon}))$ times and taking the majority vote. Now let us denote the new algorithm to be $\mathcal{B}$. Observe that $\mathcal{B}$ probes at most $O(t \lg n/\lg(\frac{1}{\epsilon}))$ cells of the matrix and thus we will lose $O(\lg n/\lg(\frac{1}{\epsilon}))$ factor in all the bounds given in Theorem 1.2 and Theorem 4.8.

5 Lower Bound on Vector-Matrix-Vector Problem over $\mathbb{F}_2$

This section is devoted to the proof of Theorem 1.3. To this end, assume that given a matrix $M \in \mathbb{F}_2^{n \times n}$, there exists a data structure $R = R(M)$ consisting of at most $r = r(n)$ bits, and there exists an algorithm that, given $u, v \in \mathbb{F}_2^n$ returns $u^\intercal M v$ while probing only R and at most $t = t(n)$ bits from M. Under these assumptions, Theorem 1.3 states the following.

Theorem 1.3. *If $n \leq r \leq \frac{n^2}{64}$ then $t \cdot r = \Omega(n^3/\lg n)$; otherwise for $r < n$, $t = \Omega(n^2/\lg n)$.*

To prove the theorem we will show that for most matrices $M \in \mathbb{F}_2^n$, one can succinctly (in terms of r, t) encode M. More precisely, by fixing some parameter B and dividing M into segments

of B consecutive rows, we will show that there is a single segment that contains a large amount of information, and moreover, there exists a short (in terms of r, t) bit-string that encodes this segment. It is worth noting that our proof technique can be generalized to give lower bound for the case when the query algorithm may err with probability at most $1/64$ on average over the choices of M, u, v. However for the sake of simplicity we first focus only on the query algorithm that never errs and we defer the comment on the generalization to the end of this section.

We start with introducing some notations. Given $u \in \mathbb{F}_2^n$ and a subset $I \subseteq [n]$, let u_I be the projection of u onto $\mathbb{F}_2^{|I|}$. Similarly denote $M_{I,J}$ for any $M \in \mathbb{F}_2^{n \times n}$ and subsets $I, J \subseteq [n]$.

Let B be some parameter, the value of which will be fixed later. For every $i \in [n/B]$, let $I_i := \{(i-1)B + 1, \ldots, iB\}$, and let $M_i := M_{I_i,[n]}$ be the i-th segment of M composed of all the rows in I_i, and $M_{-i} := M_{([n]\setminus I_i),[n]}$. Finally, for every $u, v \in \mathbb{F}_2^n$, $M \in \mathbb{F}_2^{n \times n}$ and $i \in [n/B]$, let $t_i(u, M, v)$ be the number of cell probes performed by the algorithm in M_i when queried for $u^\intercal M v$.

Our first claim shows that there exists an $i^* \in [n/B]$ such that for many vectors $u \in \mathbb{F}_2^n$, the expected number (over random M, v) of probes performed by the algorithm on M_{i^*} is not too large.

Lemma 5.1. *There exists $i^* \in [n/B]$ such that $\Pr_u[\mathbb{E}_{M,v}[t_{i^*}(u, M, v)] \leq \frac{4tB}{n}] \geq \frac{3}{4}$.*

Proof. First note that $\mathbb{E}_i[\mathbb{E}_{u,M,v}[t_i(u, M, v)]] = \mathbb{E}_{u,M,v}[\mathbb{E}_i[t_i(u, M, v)]] \leq t$. Therefore there exists $i^* \in [n/B]$ such that $\mathbb{E}_{u,M,v}[t_{i^*}(u, M, v)] \leq \frac{tB}{n}$. The claim now follows from Markov's inequality. □

For every $u \in \mathbb{F}_2^n$ and $i \in [n/B]$, denote $M_{i|u} := u_{I_i}^\intercal M_i$. The following lemma shows that for most vectors $u \in \mathbb{F}_2^n$, $M_{i^*|u}$ contains a large amount of information. One may note that the lemma is true for all $i \in [n/B]$, though for our purpose it suffices to consider i^* only.

Lemma 5.2. *Suppose $M \in_R \mathbb{F}_2^{n \times n}$. Then $\Pr_u[H(M_{i^*|u} \mid R, M_{-i^*}) \geq n - \frac{8r}{B}] \geq \frac{3}{4}$.*

Proof. Let $U = \{u \in \mathbb{F}_2^n : H(M_{i^*|u} \mid R, M_{-i^*}) < n - \frac{8r}{B}\}$, and let $u^{(1)}, \ldots, u^{(\ell)} \in U$ be a sequence in U such that $u^{(1)}_{I_{i^*}}, \ldots, u^{(\ell)}_{I_{i^*}}$ are linearly independent over $\mathbb{F}_2^{|I_{i^*}|}$. Then the random variables $M_{i^*|u^{(1)}}, \ldots, M_{i^*|u^{(\ell)}}, M_{-i^*}$ are independent. To see this, first note that by the definition ,M_{-i^*} is independent of M_{i^*} and hence of $M_{i^*|u^{(1)}}, \ldots, M_{i^*|u^{(\ell)}}$. Next observe that for any $k \in [\ell]$ and $b \in \mathbb{F}_2^n$, $Pr_M[M_{i^*|u^{(k)}} = b] = 1/2^n$. Now since the vectors $u^{(1)}, \ldots, u^{(\ell)}$ are linearly independent, for any $b^{(1)}, \ldots, b^{(\ell)} \in \mathbb{F}_2^n$, $Pr_M[\text{for all } k,\ M_{i^*|u^{(k)}} = b^{(k)}] = (1/2^n)^\ell$. Therefore

$$
\begin{aligned}
r \geq H(R) &\geq I(R; M_{i^*|u^{(1)}}, \ldots, M_{i^*|u^{(\ell)}} \mid M_{-i^*}) \\
&= H(M_{i^*|u^{(1)}}, \ldots, M_{i^*|u^{(\ell)}} \mid M_{-i^*}) - H(M_{i^*|u^{(1)}}, \ldots, M_{i^*|u^{(\ell)}} \mid R, M_{-i^*}) \\
&\geq \sum_{j=1}^{\ell} \Big(H(M_{i^*|u^{(j)}} \mid M_{-i^*}) - H(M_{i^*|u^{(j)}} \mid R, M_{-i^*}) \Big) \\
&\geq \sum_{j=1}^{\ell} \left(n - \left(n - \frac{8r}{B} \right) \right) = \frac{8r\ell}{B} \,,
\end{aligned}
$$

thus $\ell \leq B/8$ implying $|\{u_{I_{i^*}} : u \in U\}| \leq 2^{B/8}$ and we get that $|U| \leq 2^{n-2}$. □

Now the following is a simple application of union bound.

Corollary 5.3. *There exists $u^* \in \mathbb{F}_2^n$ such that*

$$H(M_{i^*|u} \mid R, M_{-i^*}) \geq n - 8r/B \quad \textit{and} \quad \mathbb{E}_{M,v}[t_{i^*}(u, M, v)] \leq 4tB/n \,.$$

Let us define $\mathcal{M} := \{M \in \mathbb{F}_2^{n\times n} : \mathbb{E}_v[t_{i^*}(u^*, M, v)] \leq \frac{8tB}{n}\}$ and then Markov's inequality implies the following.

Claim 5.4. $\Pr_M[M \in \mathcal{M}] \geq \frac{1}{2}$.

The next Lemma shows that whenever $M \in \mathcal{M}$, $M_{i^*|u^*}$ can be encoded using a few bits.

Lemma 5.5. *If $M \in \mathcal{M}$ then $M_{i^*|u^*}$ can be encoded using $\frac{64tB}{n}\lg n$ extra bits (in addition to R and M_{-i^*}).*

Proof. Fix some $M \in \mathcal{M}$. Then by Markov's inequality, $\Pr_v[t_{i^*}(u^*, M, v) \leq \frac{16tB}{n}] \geq \frac{1}{2}$. Therefore there exists a set $\mathcal{S}$ of $\frac{16tB}{n}$ entries in M_{i^*} (note that the submatrix M_{i^*} is of size nB) such that by probing only entries from M_{-i^*}, R and $\mathcal{S}$ the algorithm can answer at least

$$\frac{2^{n-1}}{\binom{nB}{16tB/n}} \geq \frac{2^{n-1}}{\left(\frac{enB}{16tB/n}\right)^{\frac{16tB}{n}}} = 2^{n-1-\frac{16tB}{n}\lg\frac{en^2}{16t}} \geq 2^{n-\frac{32tB}{n}\lg n}$$

queries of the form $(u^*)^\intercal Mv$. The set $\mathcal{S}$ can be encoded using $\frac{16tB}{n}\lg(n^2)$ bits. Let $V \subseteq \mathbb{F}_2^n$ denote the set of vectors such that for any $v \in V$ the query $(u^*)^\intercal Mv$ can be answered by probing entries only from $M_{-i^*}, R, \mathcal{S}$. Observe that V is a linear subspace of $\mathbb{F}_2^n$, and $\mathsf{dim}(V) \geq n - \frac{32tB}{n}\lg n$.

Next, fix an ordering $v^{(1)}, \ldots, v^{(2^n)}$ of $\mathbb{F}_2^n$, and consider the string $\mathcal{E}$ of bits constructed as follows. Starting with an empty string $\mathcal{E}$, for every $k \in [2^n]$, if $v^{(k)} \notin \mathsf{span}\left(V \cup \{v^{(1)}, \ldots, v^{(k-1)}\}\right)$, append $(u^*)^\intercal Mv^{(k)}$ to $\mathcal{E}$. Since $\mathsf{dim}(V) \geq n - \frac{32tB}{n}\lg n$, it follows that $\mathcal{E}$ can be encoded using at most $\frac{32tB}{n}\lg n$ bits.

Now by probing only $M_{-i^*}, R, \mathcal{S}, \mathcal{E}$ we can answer $(u^*)^\intercal Mv$ for all $v \in \mathbb{F}_2^n$, which in terms suffices to retrieve the string $M_{i^*|u^*}$. □

Now we are ready to prove the main result of this section.

Proof of Theorem 1.3. We prove the theorem by showing that $n - \frac{8r}{B} \leq \frac{64tB}{n}\lg n + \frac{n}{2} + 1$. Setting $B = \lfloor\frac{32r}{n}\rfloor$ then implies the theorem.

To this end, let $\mathbb{1}_{M\in\mathcal{M}}$ denote the indicator random variable for the event $M \in \mathcal{M}$. By definition of u^* we have

$$n - \frac{8r}{B} \leq H(M_{i^*|u^*} \mid R, M_{-i^*}) \leq H(M_{i^*|u^*}, \mathbb{1}_{M\in\mathcal{M}} \mid R, M_{-i^*})\,. \tag{2}$$

Applying the chain rule of entropy we get that

$$\begin{aligned} H(\mathbb{1}_{M\in\mathcal{M}}, M_{i^*|u^*} \mid R, M_{-i^*}) &= H(\mathbb{1}_{M\in\mathcal{M}} \mid R, M_{-i^*}) + H(M_{i^*|u^*} \mid R, M_{-i^*}, \mathbb{1}_{M\in\mathcal{M}}) \\ &\leq H(\mathbb{1}_{M\in\mathcal{M}}) + H(M_{i^*|u^*} \mid R, M_{-i^*}, \mathbb{1}_{M\in\mathcal{M}})\,. \end{aligned} \tag{3}$$

Clearly, $H(\mathbb{1}_{M\in\mathcal{M}}) \leq 1$. Next we bound $H(M_{i^*|u^*} \mid R, M_{-i^*}, \mathbb{1}_{M\in\mathcal{M}})$ as follows.

$$\begin{aligned} &H(M_{i^*|u^*} \mid R, M_{-i^*}, \mathbb{1}_{M\in\mathcal{M}}) = \\ &= H(M_{i^*|u^*} \mid R, M_{-i^*}, \mathbb{1}_{M\in\mathcal{M}} = 1) \cdot \Pr[\mathbb{1}_{M\in\mathcal{M}} = 1] + H(M_{i^*|u^*} \mid R, M_{-i^*}, \mathbb{1}_{M\in\mathcal{M}} = 0) \cdot \Pr[\mathbb{1}_{M\in\mathcal{M}} = 0]. \end{aligned} \tag{4}$$

Conditioned on $\mathbb{1}_{M\in\mathcal{M}} = 1$, Lemma 5.5 guarantees that we can encode $M_{i^*|u^*}$ using at most $\frac{64tB}{n}\lg n$ bits in addition to R, M_{-i^*}. Therefore by the Shannon's source coding theorem

$$H(M_{i^*|u^*} \mid R, M_{-i^*}, \mathbb{1}_{M\in\mathcal{M}} = 1) \cdot \Pr[\mathbb{1}_{M\in\mathcal{M}} = 1] \leq \frac{64tB}{n}\lg n \cdot 1 \ .$$

Claim 5.4 implies that

$$H(M_{i^*|u^*} \mid R, M_{-i^*}, \mathbb{1}_{M\in\mathcal{M}} = 0) \cdot \Pr[\mathbb{1}_{M\in\mathcal{M}} = 0] \leq n \cdot \frac{1}{2} \ .$$

Plugging the last two inequalities into (4) we get that $H(M_{i^*|u^*} \mid R, M_{-i^*}, \mathbb{1}_{M\in\mathcal{M}}) \leq \frac{64tB}{n}\lg n + \frac{n}{2}$. Plugging this into (2), (3) we get that $n - \frac{8r}{B} \leq \frac{64tB}{n}\lg n + \frac{n}{2} + 1$. Now for $r \geq n$ by substituting $B = \lfloor \frac{32r}{n} \rfloor$, we conclude that $\frac{2048tr}{n^2}\lg n \geq \frac{n}{4}$, and thus $tr \geq \Omega(n^3/\lg n)$.

For $r < n$ we use the following simple padding argument. Append R with some arbitrary bits so that the size (no. of bits) of the new data structure R' becomes n. Now from the previous argument it follows that $t \geq \Omega(n^2/\lg n)$, thus completing the proof. □

Comment on query algorithms with error. In Theorem 1.3 we consider query algorithms those always output the correct answer and provide lower bound. It is worth noting that our proof technique can be generalized to give lower bound for the case when the query algorithm may err with probability at most 1/64 (though any small constant probability will work) on average over the choices of M, u, v. We need to modify the proof a bit by considering the event that the algorithm (say $\mathcal{A}$) errs, i.e., $\mathcal{A}(u, M, v) \neq u^\intercal Mv$. From $Pr_{u,M,v}[\mathcal{A}(u, M, v) \neq u^\intercal Mv] \leq 1/64$, using Markov's inequality we can deduce that $Pr_u[Pr_{M,v}[\mathcal{A}(u, M, v) \neq u^\intercal Mv] \geq 1/16] \leq 1/4$. Now we choose u^* that satisfies Corollary 5.3 and $Pr_{M,v}[\mathcal{A}(u^*, M, v) \neq (u^*)^\intercal Mv] \leq 1/16$. The existence of such a u^* follows from simple union bound. Similarly $Pr_M[Pr_v[\mathcal{A}(u^*, M, v) \neq (u^*)^\intercal Mv] \geq 1/4] \leq 1/4$. Now define $\mathcal{M} := \{M \in \mathbb{F}_2^{n\times n} : \mathbb{E}_v[t_{i^*}(u^*, M, v)] \leq \frac{16tB}{n}\}$ and hence $Pr_M[M \in \mathcal{M}$ and $Pr_v[\mathcal{A}(u^*, M, v) \neq (u^*)^\intercal Mv] \leq 1/4] \geq 1/2$. Next we modify Lemma 5.5 by saying that for any $M \in \mathcal{M}$,

$$\Pr_v[\mathcal{A}(u^*, M, v) = (u^*)^\intercal Mv \text{ and } t_{i^*}(u^*, M, v) \leq \frac{64tB}{n}] \geq \frac{1}{2}.$$

The remaining argument will be the same and we will get similar lower bound. One can further extend this lower bound result to randomized query algorithms that given u, v output correct answer with high probability, by using the technique described in Section 4.1.

References

[BBK17] J. Boninger, J. Brody, and O. Kephart. Non-adaptive data structure bounds for dynamic predecessor search. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:50, 2017.

[BL13] K. Bringmann and K. G. Larsen. Succinct sampling from discrete distributions. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13. ACM, 2013.

[BL15] J. Brody and K. G. Larsen. Adapt or die: Polynomial lower bounds for non-adaptive dynamic data structures. *Theory of Computing*, 11:471–489, 2015.

[CGL15] R. Clifford, A. Grønlund, and K. G. Larsen. New unconditional hardness results for dynamic and online problems. In *56th Annual Symposium on Foundations of Computer Science, 2015*, pages 1089–1107, 2015.

[CT06] T. M. Cover and J. A. Thomas. *Elements of information theory (2. ed.)*. Wiley, 2006.

[FS89] M. L. Fredman and M. E. Saks. The cell probe complexity of dynamic data structures. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 345–354, 1989.

[GM07] A. Gál and P. B. Miltersen. The cell probe complexity of succinct data structures. *Theoretical Computer Science*, 379:405–417, July 2007.

[GO95] A. Gajentaan and M. H. Overmars. On a class of $O(N^2)$ problems in computational geometry. *Comput. Geom. Theory Appl.*, 5(3):165–185, October 1995.

[HKNS15] M. Henzinger, S. Krinninger, D. Nanongkai, and T. Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, 2015*, pages 21–30, 2015.

[IP01] R. Impagliazzo and R. Paturi. On the complexity of k-sat. *J. Computer and System Sciences*, 62(2):367–375, March 2001.

[Jac88] G. J. Jacobson. *Succinct Static Data Structures*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1988.

[Lar12a] K. G. Larsen. The cell probe complexity of dynamic range counting. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012*, pages 85–94, 2012.

[Lar12b] K. G. Larsen. Higher cell probe lower bounds for evaluating polynomials. In *53rd Annual IEEE Symposium on Foundations of Computer Science*, pages 293–301, 2012.

[LW17] K. G. Larsen and R. R. Williams. Faster online matrix-vector multiplication. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, 2017*, pages 2182–2189, 2017.

[LWY17] K. G. Larsen, O. Weinstein, and H. Yu. Crossing the logarithmic barrier for dynamic boolean data structure lower bounds. *CoRR*, abs/1703.03575, 2017. Available from: `http://arxiv.org/abs/1703.03575`.

[Păt08] M. Pătraşcu. Succincter. In *Proc. 49th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 305–313, 2008.

[Păt11] M. Pătraşcu. Unifying the landscape of cell-probe lower bounds. *SIAM J. Comput.*, 40(3):827–847, 2011.

[PD06] M. Pătraşcu and E. D. Demaine. Logarithmic lower bounds in the cell-probe model. *SIAM J. Comput.*, 35(4):932–963, 2006.

[PT11] M. Pătraşcu and M. Thorup. Don't rush into a union: take time to find your roots. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011*, pages 559–568, 2011.

[PV10] M. Pătraşcu and E. Viola. Cell-probe lower bounds for succinct partial sums. In *Proc. 21st ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 117–122, 2010.

[RR17] S. N. Ramamoorthy and A. Rao. Non-adaptive data structure lower bounds for median and predecessor search from sunflowers. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:40, 2017.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27, 1948.

[WW10] V. V. Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. In *51st Annual IEEE Symposium on Foundations of Computer Science*, pages 645–654, 2010.

[WY16] O. Weinstein and H. Yu. Amortized dynamic cell-probe lower bounds from four-party communication. In *57th Annual IEEE Symposium on Foundations of Computer Science*, pages 305–314, 2016.

[Yao81] A. C. Yao. Should tables be sorted? *J. ACM*, 28(3):615–628, 1981.