Electronic Colloquium on Computational Complexity, Report No. 103 (2016)

The zero-error randomized query complexity of the pointer function

Jaikumar Radhakrishnan Swagato Sanyal

Tata Institute of Fundamental Research, Mumbai
{jaikumar, swagato.sanyal}@tifr.res.in

Abstract

The pointer function of Göös, Pitassi and Watson [5] and its variants have recently been used to prove separation results among various measures of complexity such as deterministic, randomized and quantum query complexities, exact and approximate polynomial degrees, etc. In particular, the widest possible (quadratic) separations between deterministic and zero-error randomized query complexity, as well as between bounded-error and zero-error randomized query complexity, have been obtained by considering *variants* [2] of this pointer function.

However, as was pointed out in [2], the precise zero-error complexity of the original pointer function was not known. We show a lower bound of $\widetilde{\Omega}(n^{3/4})$ on the zero-error randomized query complexity of the pointer function on $\Theta(n \log n)$ bits; since an $\widetilde{O}(n^{3/4})$ upper bound is already known [7], our lower bound is optimal up to a factor of $\operatorname{polylog} n$.

1 Introduction

Understanding the relative power of various models of computation is a central goal in complexity theory. In this paper, we focus on one of the simplest models for computing boolean functions—the query model or the decision tree model. In this model, the algorithm is required to determine the value of a boolean function by querying individual bits of the input, possibly adaptively. The computational resource we seek to minimize is the number of queries for the worst-case input. That is, the algorithm is charged each time it queries an input bit, but not for its internal computation.

There are several variants of the query model, depending on whether ~~or not~~ randomization is allowed, and on whether error is acceptable. Let $D(f)$ denote the deterministic query complexity of f, that is, the maximum number of queries made by the algorithm for the worst-case input; let $R(f)$ denote the maximum number of queries made by the best randomized algorithm that errs with probability at most $1/3$ (say) on the worst-case input. Let $R_0(f)$ be the zero-error randomized query complexity of f, that is, the expected number of queries made for the worst-case input by the best randomized algorithm for f that answers correctly on every input.

The relationships between these query complexity measures have been extensively studied in the literature. That randomization can lead to significant savings has been known for a long time. Snir [10] showed a $O(n^{\log_4 3})$ randomized linear query algorithm (a more powerful model than what we discussed) for complete binary NAND tree function for which the deterministic linear query complexity is $\Omega(n)$. Later on Saks and Wigderson [9] determined the zero-error randomized query complexity of the complete binary NAND tree function to be $\Theta(n^{0.7536\cdots})$. They also presented a result of Ravi Boppana which states that the uniform rooted ternary majority tree function has randomized zero-error query complexity $O(n^{0.893\cdots})$ and deterministic query complexity n. All these example showed that randomized query complexity can be substantially lower than its deterministic counterpart. On the other hand, Nisan showed that the $R(f) = \Omega(D(f)^{1/3})$ [8]. Blum and Impagliazzo [3], Tardos [11], Hartmanis and Hemachandra [6] independently showed that $R_0(f) = \Omega(D(f)^{1/2})$. Thus, the question of the largest separation between deterministic and randomized complexity remained open. Indeed, Saks and Wigderson conjectured that the complete binary NAND tree function exhibits the widest separation possible between these two measures of complexity.

ISSN 1433-8092

Conjecture 1 ([9])**.** *For any boolean function f on n variables, $R_0(f) = \Omega(D(f)^{0.753\cdots})$.*

This conjecture was recently refuted independently by Ambainis *et al.* [2] and Mukhopadhyay and Sanyal [7]. Both works based their result on the pointer function introduced by Göös, Pitassi and Watson [5], who used this function to show a separation between deterministic decision tree complexity and unambiguous non-deterministic decision tree complexity. In Section 2, we present the formal definition of the function $\mathsf{GPW}^{r\times s}$, which is a Boolean function on $\widetilde{\Theta}(rs)$ bits.

Mukhopadhyay and Sanyal [7] used $\mathsf{GPW}^{s\times s}$ to obtain the following refutation of Conjecture 1: $R_0(\mathsf{GPW}^{s\times s}) = \widetilde{O}(s^{1.5})$ while $D(\mathsf{GPW}^{s\times s}) = \Omega(s^2)$. While this shows that $\mathsf{GPW}^{s\times s}$ witnesses a wider separation between deterministic and zero-error randomized query complexities than conjectured, the separation shown is not the widest possible for a Boolean function. Independently, Ambainis *et al.* modified $\mathsf{GPW}^{s\times s}$ in subtle ways, to establish the widest possible (near-quadratic) separation between deterministic and zero-error randomized query complexity, and between zero-error randomized and bounded-error randomized query complexities.

Ambainis et al. [2] pointed out, however, that the precise zero-error randomized query complexity (i.e. $R_0(\mathsf{GPW}^{s\times s})$) was not known. One could ask if the optimal separation demonstrated by Ambainis *et al.* is also witnessed by $\mathsf{GPW}^{s\times s}$ itself. In this work, we prove a near-optimal lower bound on the zero-error randomized query complexity of $\mathsf{GPW}^{r\times s}$, which is slightly more general than the $\mathsf{GPW}^{s\times s}$ considered in earlier works.

Theorem 1 (Main theorem)**.** $R_0(\mathsf{GPW}^{r\times s}) = \widetilde{\Omega}(r + \sqrt{r}s)$.

Such a result essentially claims that randomized algorithms cannot efficiently locate certificates for the function. This would be true, for example, if the function could be shown to require large certificates, since the certificate complexity of a function is clearly a lower bound on its zero-error randomized complexity. This straightforward approach does not yield our lower bound, as the certificate complexity of $\mathsf{GPW}^{r\times s}$ is $\widetilde{O}(r + s)$. In our proof, we set up a special distribution on inputs, and by analyzing the expansion properties of the pointers, show that a certificate will evade a randomized algorithm that makes only a small number of queries. In fact, the distribution we devise is almost entirely supported on inputs X for which $\mathsf{GPW}^{r\times s}(X) = 0$. This is not an accident: a randomized algorithm can quickly find a certificate for inputs X if $\mathsf{GPW}^{r\times s}(X) = 1$ (see Theorem 3 below).

It follows from Theorem 1 that the algorithm of Mukhopadhyay and Sanyal [7] is optimal up to polylog factors.

Corollary 1. $R_0(\mathsf{GPW}^{s\times s}) = \widetilde{\Omega}(s^{1.5})$.

In addition to nearly determining the zero-error complexity of the original $\mathsf{GPW}^{s\times s}$ function, our result has two interesting consequences.

(a) The above mentioned result of Mukhopadhyay and Sanyal [7] showed that $R_0(\mathsf{GPW}^{s\times s}) = \widetilde{\Omega}(D(\mathsf{GPW}^{s\times s})^{0.75})$. Our main theorem shows that $\mathsf{GPW}^{s\times s}$ cannot be used to show a significantly better separation between the deterministic and randomized zero-error complexities (ignoring polylog factors). However, the function $\mathsf{GPW}^{s^2\times s}$ allows us to derive a better separation[1]: $R_0(\mathsf{GPW}^{s^2\times s}) = O(D(\mathsf{GPW}^{s^2\times s})^{2/3})$. Our main theorem shows that this is essentially the best separation that can be derived from $\mathsf{GPW}^{r\times s}$ by varying r relative to s, so this method cannot match the near-quadratic separation between these measures shown by Ambainis *et al.* [2] by considering a variant of the $\mathsf{GPW}^{s\times s}$ function.

(b) $\mathsf{GPW}^{s\times s}$ exposes a non-trivial polynomial separation between the zero-error and bounded-error randomized query complexities: $R(\mathsf{GPW}^{s\times s}) = \widetilde{O}(R_0(\mathsf{GPW}^{s\times s})^{2/3})$. This falls short of the near-quadratic separation shown by Ambainis *et al.* [2], but note that before that result no separation between these measures was known.

2 The GPW function

The input X to the *pointer function*, $\mathsf{GPW}^{r\times s}$, is arranged in an array with r rows and s columns. The cell $X[i,j]$ of the array contains two pieces of data, a bit $b_{ij} \in \{0,1\}$ and a pointer $\mathsf{ptr}_{ij} \in ([r]\times[s]) \cup \{\perp\}$.

[1] In [1], a similar separation between $R(\mathsf{GPW}^{s^2\times s})$ and $D(\mathsf{GPW}^{s^2\times s})$ is mentioned.

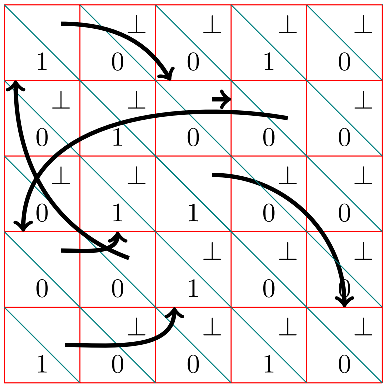

Figure 1: Input to $\mathsf{GPW}^{r \times s}$ for $r = 5, s = 5$.

Let $\mathcal{A}$ denote the set of all such arrays. The function $\mathsf{GPW}^{r\times s} : \mathcal{A} \to \{0,1\}$ is defined as follows: $\mathsf{GPW}^{r\times s}(X) = 1$ if and only if the following three conditions are satisfied.

1. There is a unique column j^* such that for all rows $i \in [r]$, we have $b_{ij^*} = 1$.
2. In this column j^*, there is a unique row i^* such that $\mathsf{ptr}_{i^*j^*} \neq \perp$.
3. Now, consider the sequence of locations $(p_k : k = 0, 1, \ldots, s-1)$, defined as follows: let $p_0 = (i^*, j^*)$, and for $k = 0, 1, \ldots, s-2$, let $p_{k+1} = \mathsf{ptr}_{p_k}$. Then, $p_0, p_1, \ldots, p_{s-1}$ lie in distinct columns of X, and $b_{p_k} = 0$ for $k = 1, 2, \ldots, s-1$. In other words, there is a *chain of pointers*, which starts from the unique location in column j^* with a non-null pointer, visits all other columns in exactly $s-1$ steps, and finds a 0 in each location it visits (except the first).

Note that $\mathsf{GPW}^{r\times s}$ can be thought of as a Boolean function on $\Theta(rs \log rs)$ bits.

Upper Bound The pointer function $\mathsf{GPW}^{r\times s}$, as defined above, is parameterized by two parameters, r and s. Göös, Pitassi and Watson [5] focus on the special case where $r = s$. Mukhopadhyay and Sanyal [7] also state their zero-error randomized algorithm with $\widetilde{O}(s^{1.5})$ queries for this special case; however, it is straightforward to extend their algorithm so that it applies to the function $\mathsf{GPW}^{r\times s}$.

Theorem 2. $R_0(\mathsf{GPW}^{r\times s}) = \widetilde{O}(r + \sqrt{r}s)$.

Mukhopadhyay and Sanyal also gave a one-sided error randomized query algorithm that makes $\widetilde{O}(s)$ queries on average but never errs on inputs X, where $\mathsf{GPW}^{s\times s}(X) = 1$. Again a straightforward extension yields the following.

Theorem 3. *There is a randomized query algorithm that makes $\widetilde{O}(r+s)$ queries on each input, computes $\mathsf{GPW}^{r\times s}$ on each input with probability at least $1/3$, and in addition never errs on inputs X where $\mathsf{GPW}^{r\times s}(X) = 1$.*

Theorem 1, thus, completely determines the deterministic and all randomized query complexities of a more general function $\mathsf{GPW}^{r\times s}$.

2.1 The distribution

To show our lower bound, we will set up a distribution on inputs in $\mathcal{A}$. Let V be the locations in the first $s/2$ columns, i.e., $V = [r] \times [s/2]$; let W be the locations in the last $s/2$ columns, i.e., $W = [r] \times ([s] \setminus [s/2])$. In order to describe the random input X, we will need the following definitions.

Pointer chain: For an input in $\mathcal{A}$, we say that a sequence of locations $\mathbf{p} = \langle \ell_0, \ell_1, \ell_2, \ldots, \ell_k \rangle$ is a pointer chain, if for $i = 0, 1, \ldots, k-1$, $\mathsf{ptr}_{\ell_i} = \ell_{i+1}$; the location ℓ_0 is the *head* of the $\mathbf{p}$ and is denoted by $\mathsf{head}(\mathbf{p})$; similarly, ℓ_k is the *tail* of $\mathbf{p}$ and is denoted by $\mathsf{tail}(\mathbf{p})$. Note that $\mathsf{ptr}(\ell_k)$ is not specified as part of the definition of pointer chain $\mathbf{p}$; in particular, it is allowed to be $\perp$.

Random pointer chain: To build our random input X, we will assign the pointer values of the various cells of X randomly so that they form appropriate pointer chains. For a set of locations S we build a *random pointer chain on* S as follows. First, we uniformly pick a permutation of S, say $\langle \ell_0, \ell_2, \dots, \ell_k \rangle$. Then, we set $\mathsf{ptr}_{\ell_i} = \ell_{i+1}$ (for $i = 0, 1, \dots, k-1$). We will make such random assignments for sets S consisting of consecutive locations in some row of W. We call the special (deterministic) chain that starts at the first (leftmost) location of S, visits the next, and so on, until the last (rightmost) location, a *path.* Given two pointer chains $\mathbf{p}_1$ and $\mathbf{p}_2$ on disjoint sets of locations S_1 and S_2, we may set $\mathsf{ptr}_{\mathsf{tail}(\mathbf{p}_1)} = \mathsf{head}(\mathbf{p}_2)$, and obtain a single pointer chain on $S_a \cup S_b$, whose head is $\mathsf{head}(\mathbf{p}_1)$ and tail is $\mathsf{tail}(\mathbf{p}_2)$. We will refer to this operation as the *concatenation* of $\mathbf{p}_1$ and $\mathbf{p}_2$.

We are now ready to define the random input X. First, consider W. For all $\ell \in W$, we set $b_\ell = 0$. To describe the pointers corresponding to W, we partition the columns of W into $K := \log s - 3 \log\log s$ *blocks*, $W_1, \dots, W_K$, where W_1 consists of the first $s/(2K)$ columns of W, W_2 consists of the next $s/(2K)$ columns, and so on.

$$\left[\begin{array}{c|cccc} V & W_1 & W_2 & \dots & W_K \end{array}\right]$$

The block W_j, will be further divided into *bands;* however, the number of bands in different W_j will be different. There will be $20 \cdot 2^j$ bands in W_j, each consisting of $w_j := s/(20 \cdot 2^j \cdot 2K)$ contiguously chosen columns. See Figure 2.

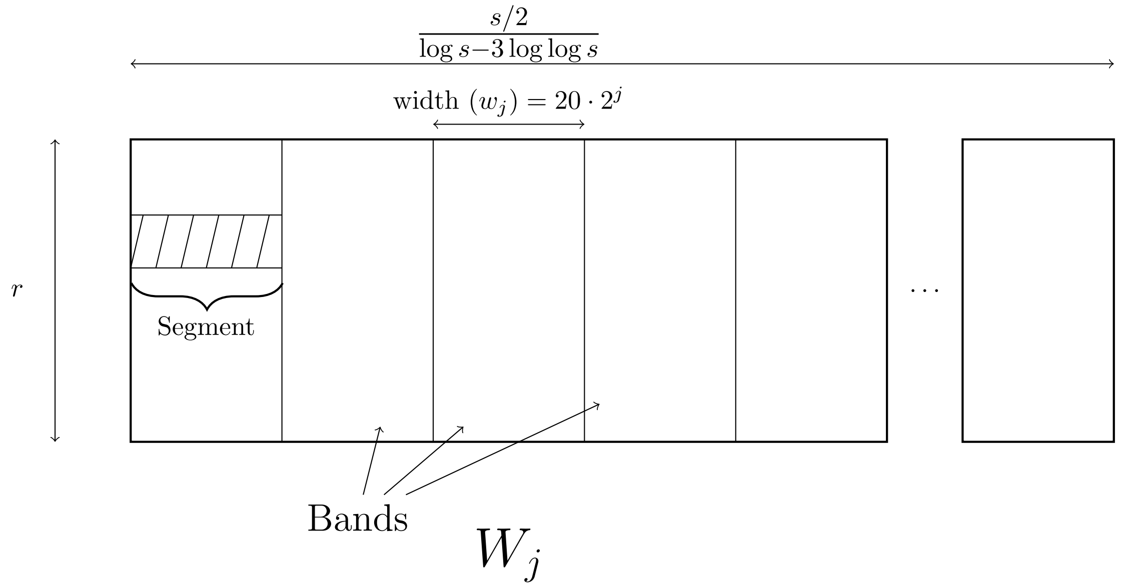

Figure 2: Bands and segments inside block W_j.

Each such band will have r rows; the locations in a single row of a band will be called a *segment*; we will divide each segment into two equal parts, left and right, each with $w_j/2$ columns. (See Figure 3.)

We are now ready to specify the pointers in each segment of W_i. In the first half of each segment we place a random (uniformly chosen) pointer chain; in the right half we place a path starting at its leftmost cell and leading to its rightmost cell. Once all pointer chains in all the segments in a given row are in place, we concatenate them from left to right. All pointers in the last column of W are set to $\perp$. In the resulting input, each row of W is a single pointer chain with head in the leftmost segment of W_1 and tail in the last column of W. This completes the description of X for the locations in W.

Next, we consider locations in V. Let $q := 500 \log s/\sqrt{r}$. Independently, for each location $\ell \in V$:

- with probability q, set $b_\ell = 0$ and ptr_ℓ to be a random location that is in the *left half* of some segment in W (that is, among all locations that fall in the left half of some segment, pick one at random and set ptr_ℓ to that location);

- with probability $1-q$, set $b_\ell = 1$ and $\mathsf{ptr}_\ell = \perp$.

This completes the description of the random input X.

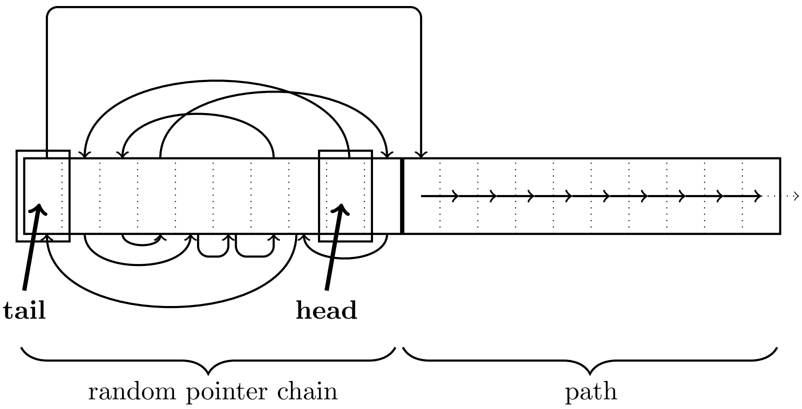

Figure 3: A segment consists of a random pointer chain concatenated with a path.

3 The lower bound for $\mathsf{GPW}^{r\times s}$

We will consider algorithms that are given query access to the input bits of $\mathsf{GPW}^{r\times s}$. A location $\ell \in [r]\times[s]$ of an input $X \in \mathcal{A}$ is said to be queried if either b_ℓ is queried, or some bit in the encoding of ptr_ℓ is queried. By *number of queries*, we will always mean the number of locations queried. A lower bound on the number of locations queried is clearly a lower bound on the number of bits queried.

It can be shown that the certificate complexity of $\mathsf{GPW}^{r\times s}$ is $\Omega(r+s)$; hence $R_0(\mathsf{GPW}^{r\times s}) = \Omega(r+s)$. It remains to show that any zero-error randomized query algorithm for $\mathsf{GPW}^{r\times s}$ must make $\Omega(\sqrt{r}s/\mathrm{polylog}(s))$ queries in expectation. We will assume that there is a significantly more efficient algorithm and derive a contradiction.

Assumption 4. *There is a zero-error randomized algorithm that makes at most $\sqrt{r}s/(\log s)^5$ queries in expectation (taken over the algorithm's coin tosses) on every input X.*

If $r < (\log s)^3$ (say), then this assumption immediately leads to a contradiction because $R_0(\mathsf{GPW}^{r\times s}) = \Omega(s)$. So, we will assume that $r \geq (\log s)^3$.

Consider inputs X drawn according to the distribution described in the previous section. Since with probability $1-o(1)$ every column of X has at least one zero (see Lemma 3 (a)), $\mathsf{GPW}^{r\times s}(X) = 0$ with probability $1-o(1)$; thus, the algorithm returns the answer 0 with probability $1-o(1)$. Taking expectation over inputs X and the algorithm's coin tosses, the expected number of queries made by the algorithm is at most $\sqrt{r}s/(\log s)^5$. Using Markov's inequality, with probability $1-o(1)$, the algorithm stops after making at most $\sqrt{r}s/(\log s)^4$ queries. By truncating the long runs and fixing the random coin tosses of the algorithm, we obtain a deterministic algorithm. Hence we have the following.

Proposition 5. *If Assumption 4 holds, then there is a deterministic algorithm that (i) queries at most $\sqrt{r}s/(\log s)^4$ locations, (ii) never returns a wrong answer (it might give no answer on some inputs), and (iii) returns the answer* 0 *with probability* $1-o(1)$ *for the random input X.*

Fix such a deterministic query algorithm $\mathcal{Q}$. We will show that with high probability the locations of X that are left unqueried by $\mathcal{Q}$ can be modified to yield an input X' such that $\mathsf{GPW}^{r\times s}(X') = 1$. Thus, with high probability, $\mathcal{Q}(X') = \mathcal{Q}(X) = 0$. This contradicts Proposition 5 (ii). In fact, in the next section, we formally establish the following.

Lemma 1 (Stitching lemma)**.** *With probability $1-o(1)$ over the choices of X, there is an input $X' \in \mathcal{A}$ that differs from X only in locations not probed by $\mathcal{Q}$ such that $\mathsf{GPW}^{r\times s}(X') = 1$.*

By the discussion above, this immediately implies Theorem 1.

4 The approach

In this section, we will work with the algorithm $\mathcal{Q}$ that is guaranteed to exist by Proposition 5. For an input $X \in \mathcal{A}$ to $\mathsf{GPW}^{r\times s}$, let $G_X = (V', W', E)$ be a bipartite graph, where V' is the set of columns of V and W' is the set of all bands in all blocks of of W. The edge set $E(G_X)$ is obtained as follows. Recall that pointers from V lead to segments in W. Each such segment contains a pointer chain. For a location ℓ in such a chain, let $\mathsf{pred}(\ell)$ denote the location ℓ' that precedes ℓ in the chain (if ℓ is the head, then $\mathsf{pred}(\ell)$ is undefined); thus, $\mathsf{ptr}_{\ell'} = \ell$. We include the edge (j, β) (connecting column $j \in V'$ to band $\beta \in W'$) in $E(G_X)$ if the following holds:

There is a location v in column j and a segment p in some row of band β such that

(c1) $\mathsf{ptr}_v \in p$, *that is,* ptr_v *is non-null and points to a location in the left half of segment p;*

(c2) $\mathsf{pred}(\mathsf{ptr}_v)$ *is well defined and is not probed by $\mathcal{Q}$;*

(c3) $\mathcal{Q}$ makes fewer than $|p|/4$ probes in segment p. (Note that this implies that there is a location in the right half of p that is left unprobed by $\mathcal{Q}$.)

In the next section, we will show the following.

Lemma 2 (Matching lemma)**.** *With probability $1 - o(1)$ over the choice of X, for every subset $R \subseteq V'$ of at most $s/(\sqrt{r}(\log s)^4)$ columns, there is a matching in G_X that saturates R.*

In this section, we will show how Lemma 2 enables us to modify the input X to obtain an input X' for which $\mathsf{GPW}^{r\times s}(X') = 1$, thereby establishing Lemma 1.

Lemma 3. *(a) With probability $1 - o(1)$, each column j of the input X has a location ℓ such that $b_\ell = 0$.*

(b) With probability $1 - o(1)$, there is a column $j \in [s/2]$ such that $\mathcal{Q}$ does not read any location ℓ in column j with $b_\ell = 0$.

Proof. (a) All the bits in the columns in $[s] \setminus [s/2]$ are 0. We show that with high probability, each column in V' has a 0. The probability that a particular column in V' does not have any 0 is $(1 - 500 \log s/\sqrt{r})^r \leq s^{-\Omega(\sqrt{r})}$. Thus the probability that there is a column $j \in V'$ which does not have any 0 is at most $(s/2) \cdot s^{-\Omega(\sqrt{r})} = o(1)$.

(b) Suppose $\mathcal{Q}$ makes $t \leq s\sqrt{r}/(\log s)^4$ queries. For $i = 1, 2, \ldots, t$, let R_i be the indicator variable for the the event that in the i-th query, $\mathcal{Q}$ reads a 0 from V. Then, the expected number of 0's read by $\mathcal{Q}$ in V is (we assume that $\mathcal{Q}$ does not read the same location twice)

$$\sum_{i=1}^{q} \mathbb{E}[R_i] \leq t \cdot 500 \log s/\sqrt{r} \leq 500s/(\log s)^3.$$

By Markov's inequality, with probability $1 - o(1)$, the number number of 0's read by $\mathcal{Q}$ is less than $s/2$. It follows, that there is a column in V in which $\mathcal{Q}$ has read no 0.

□

Proof of Lemma 1. Assume that the high probability events of Lemmas 2 and 3 hold. This happens with probability $1 - o(1)$. We will now describe a sequence of modifications to the input X at locations not queried by $\mathcal{Q}$ to transform it into a input X' such that $\mathsf{GPW}^{r\times s}(X') = 1$. Let $j^* \in V'$ be the column in V guaranteed by Lemma 3 (b). Define $A_0 = \{\mathsf{col}_1, \ldots, \mathsf{col}_N\} \subseteq V' \setminus \{j^*\}$ to be the set of columns in $V' \setminus \{j^*\}$ that are not completely read by $\mathcal{Q}$ (i.e. each column in A_0 has a location unread by $\mathcal{Q}$). Let ℓ_i be a location in the column col_i that is unread by $\mathcal{Q}$. We first make the following changes to X, with the aim of starting a pointer chain at column j^* that passes through $\mathsf{col}_1, \mathsf{col}_2, \ldots, \mathsf{col}_N$.

(i) For each unread location ℓ in the column j^*, set b_ℓ to 1.

(ii) Let ℓ^* be the first unread location of j^*. Set ptr_{ℓ^*} to ℓ_1.

(iii) For each location $\ell \neq \ell^*$ in column j^*, set ptr_ℓ to $\perp$.

(iv) For $i = 1, \ldots, N-1$, set b_{ℓ_i} to 0 and ptr_{ℓ_i} to ℓ_{i+1}.

(v) Set b_{ℓ_N} to 0.

Clearly, the locations modified are not probed by $\mathcal{Q}$. Notice that the current input has the pointer chain $\mathbf{p}_0 = (\ell^*, \ell_1, \ldots, \ell_N)$ and the head ℓ^* of the chain lies in the all-ones column j^*. Furthermore, all locations on the chain except ℓ^* have 0 as their bit. We now show how to further modify our input and extend $\mathbf{p}$ and visit the remaining columns through locations with 0's. The columns in W are already neatly arranged in pointer chains. The difficulty is in ensuring that we also visit the set of columns in V' that are completely read by $\mathcal{Q}$, for we are not allowed to make any modifications there. Let A_1 denote these completely read columns in V'. Since $\mathcal{Q}$ makes at most $\sqrt{r}s/(\log s)^4$ queries, we have that $|A_1| \leq s/(\sqrt{r}(\log s)^4)$. Lemma 2 implies that there exists a matching $\mathcal{M}$ in G_X that saturates A_1. Order the elements of A_1 as $d_1, \ldots, d_L$ in such a way for all $i = 1, \ldots, L-1$, $\mathcal{M}(d_i) < \mathcal{M}(d_{i+1})$ (where we order the bands in W from left to right), that is, the band that is matched with d_i lies to the left of the band that is matched to d_{i+1}.

We will now proceed as follows. For $i = 1, \ldots, L$, we modify the input (at locations not read by $\mathcal{Q}$) appropriately to induce a pointer chain $\mathbf{p}_i$. This pointer chain in addition to visiting a contiguous set of columns in W, will visit column d_i. By concatenating these pointer chains in order with the initial pointer chain $\mathbf{p}_0$ we obtain the promised input X' for which $\mathsf{GPW}^{r\times s}(X') = 1$.

To implement this strategy, recall that there is an edge in G_X between the column d_i and the band $\mathcal{M}(d_i)$. From the definition of G_X, it follows that there is a location q_i in d_i and a segment S_i in band $\mathcal{M}(d_i)$ such that

(s1) ptr_{q_i} leads to the left half of S_i;

(s2) $\mathsf{pred}(\mathsf{ptr}_{q_i})$ is not probed by $\mathcal{Q}$;

(s3) $\mathcal{Q}$ makes fewer than $|S_i|/4$ queries in segment S_i.

First, let us describe how $\mathbf{p}_1$ is constructed. Let $a_1 = \mathsf{ptr}_{q_1}$ and $b_1 = \mathsf{pred}(a_1)$ (by (s2) b_1 is not probed by $\mathcal{Q}$); let c_1 be the first location in the second half of S_1 that is not probed by $\mathcal{Q}$ (by (s3) there is such a location). Now, we modify the input X by setting $\mathsf{ptr}_{b_1} = q_1$. Then, $\mathbf{p}_1$ is the pointer chain that starts at the head of the leftmost segment of W_1 in the same row as S_1 and continues until location c_1. That is, starting from its head, it follows the pointers of the input until b_1. Then it follows the pointer leading out of b_1 into q_1, thereby visiting column d_1. After that, it follows the pointer out of q_1 and comes to a_1, and keeps following the pointers until c_1.

In general, suppose $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{i-1}$ have been constructed. Suppose $\mathsf{tail}(\mathbf{p}_{i-1})$ appears in column k_{i-1}. Then, $\mathbf{p}_i$ is obtained as follows. Let $a_i = \mathsf{ptr}_{q_i}$ and $b_i = \mathsf{pred}(a_i)$; let c_i be the first location in the second half of S_i that is not probed by $\mathcal{Q}$. We modify the input by setting $\mathsf{ptr}_{b_i} = q_i$. Then $\mathbf{p}_i$ is the pointer chain with its head in the same row as a_i and in column $k_{i-1}+1$; this pointer chain terminates in location c_i. See Figure 4. Note that $\mathbf{p}_i$ entirely keeps to one row (the row of S_i), except for the diversion from b_i to q_i, when it visits column d_i and returns to a_i. When $i = L$, we let the pointer chain continue until the last column of W.

In obtaining the pointer chains $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_L$, we modified X at location $b_1, b_2, \ldots, b_L$. Finally, we concatenate the pointer chains $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_L$; this requires us to modify X at locations $\ell_N = \mathsf{tail}(\mathbf{p}_0), c_1, c_2, \ldots, c_{L-1}$, which were left unprobed by $\mathcal{Q}$. The resulting input after these modifications is X'.

The pointer chain obtained by this concatenation visits each column other than j^* exactly once, and the bit at every location on it, other than its head, is 0. Hence, $\mathsf{GPW}^{r\times s}(X') = 1$. □

5 Proof of the matching lemma

We will show that every subset $R \subseteq V'$ of at most $s/(\sqrt{r}(\log s)^4)$ columns has at least $|R|$ neighbors in W'. Then, the claim will follow from Hall's theorem.

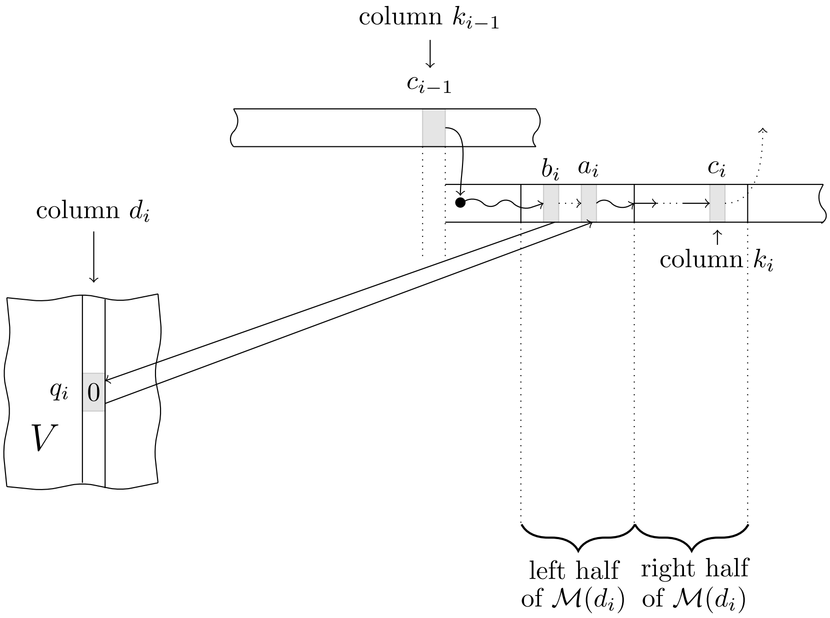

Figure 4: Construction of pointer chain $\mathbf{p}_i$

Observe that with high probability every column in V' has $\Omega(\sqrt{r}\log s)$ pointers leaving it. We expect these pointers to be uniformly distributed among the at most $\log s$ blocks in W; in particular, we should expect that every column in V' sends $\Omega(\sqrt{r})$ pointers into each block. We now formally establish this.

Claim 6. *Let V_j be the j-th column of V' and $W_{j'}$ the j'-th block of W; then,*

$$\Pr[\forall j, j' : |\mathsf{ptr}(V_j) \cap W_{j'}| \leq 400\sqrt{r}] = o(1).$$

Proof. Fix a location in $\ell \in V_j$. Let χ_ℓ be the indicator variable for the event $\mathsf{ptr}_\ell \in W_{j'}$. Then, the number of pointers from V_j into $W_{j'}$ is precisely $\sum_{\ell \in V_j} \chi_\ell$. Since

$$\Pr[\chi_\ell = 1] \geq \frac{500\log s}{\sqrt{r}} \times \frac{1}{\log s} = \frac{500}{\sqrt{r}},$$

the expected number of pointers from column V_j into $W_{j'}$ is at least $500\sqrt{r}$. Our claim follows from the Chernoff bound and the union bound (over choices of j and j' since $r = \Omega((\log s)^3)$). Here, we use the following version of the Chernoff bound (see Dubhashi and Panconesi [4], page 6): for the sum of r independent 0-1 random variables Z_ℓ, each taking the value 1 with probability at least α,

$$\Pr[\sum_\ell X_\ell \leq (1-\varepsilon)\alpha r] \leq \exp(-\frac{\varepsilon^2}{2}\alpha r).$$

Note that in our application $\alpha r \gg \sqrt{r} \geq \log s$. $\square$

Suppose j is such that $2^j \leq |R| < 2^{j+1}$. Then, we will show that R has the required number of neighbors among the bands of the block W_j.

Claim 7. *For a set $R \subseteq V'$ and a block W_j, consider the set of bands of W_j into which at least $2\sqrt{r}$ pointers from R fall, that is,*

$$B_j(R) := \{b \in W_j : |\mathsf{ptr}(R) \cap b| \geq 2\sqrt{r}\},$$

Then, for $j = 1, \ldots, K$ and for all all R such that $2^j \leq |R| < 2^{j+1}$, we have

$$\Pr[|B_j(R)| \leq 2|R|] = o(1).$$

Proof. We will use the union bound over the choices of j and R. Fix the set R. We may, using Claim 6, condition on the event that there are at least $400\sqrt{r}|R|$ pointers from R to W_j. Fix $400\sqrt{r}|R|$ of these pointers. Now, the number of pointers that fall outside $B_j(R)$ is at most $20 \cdot 2^j \cdot 2\sqrt{r} \leq 100\sqrt{r}|R|$. That is,if $|B_j(R)| < 2|R|$, then there is a set T of $2|R|$ bands into which more than $400\sqrt{r}|R| - 100\sqrt{r}|R| = 300\sqrt{r}|R|$ pointers from R fall. We will show that it is unlikely for such a set T to exist. For a fixed T, the probability of this event is at most

$$\binom{400|R|\sqrt{r}}{300|R|\sqrt{r}} \left(\frac{2|R|}{20 \cdot 2^j}\right)^{300\sqrt{r}|R|} \leq 2^{-100\sqrt{r}|R|}.$$

Using the union bound to account for all choices of R and the $\binom{20 \cdot 2^j}{2|R|}$ choices of T, and using the fact that $\sqrt{r} \gg \log s$, we conclude that the probability that $B_j(R)$ fails to be large enough is at most

$$\sum_{j=0}^{\log s - 3\log\log s} \sum_{m=2^j}^{2^{j+1}-1} \binom{s/2}{m}\binom{20 \cdot 2^j}{2m} 2^{-100\sqrt{r}m} = o(1).$$

□

In order to show that with high probability the set R has the required number of neighbors, we will condition on the high probability event of Claim 7, that is, $|B_j(R)| > 2|R|$. Let $\mathcal{B}$ be the set of such bands b that receive at least $2\sqrt{r}$ pointers. For each $b \in \mathcal{B}$, let $P(b)$ be a set of $2\sqrt{r}$ locations in the columns in R whose pointers land in b. If in at least $|R|$ of the $2|R|$ such bands b, there is a pointer from $P(b)$ satisfying the conditions (c1)–(c3), then we will have obtained the required expansion. Fix a pointer out of $P(b)$ (which by definition of $P(b)$ lands in band b), and consider the following events.

$\mathcal{E}_1$: The pointer leads to the same segment as a previous pointer (assume the locations in $P(b)$ are totally ordered in some way).

$\mathcal{E}_2$: The pointer leads to the first entry of the pointer chain in its segment (so, that location has no predecessor).

$\mathcal{E}_3$: At least $w_j/8$ entries of the segment that the pointer lands in, are probed by $\mathcal{Q}$.

$\mathcal{E}_4$: The predecessor of the location where the pointer lands is probed by $\mathcal{Q}$.

Consider the pointers that emanate from $P(b)$ and land in some band $b \in \mathcal{B}$. Let n_1 be the number of those pointers for whom $\mathcal{E}_1$ holds; let n_2 be the number of those pointers for whom $\mathcal{E}_2$ holds; let n_3 be the number of those pointers for whom $\mathcal{E}_3$ holds but $\mathcal{E}_1$ does not hold; let n_4 be the number of those pointers for whom $\mathcal{E}_4$ holds but $\mathcal{E}_1, \mathcal{E}_2$ and $\mathcal{E}_3$ do not hold.

If the claim of our lemma does not hold, then it must be that in at least $|R|$ of the $2|R|$ bands of $\mathcal{B}$, all pointers that fall there fail to satisfy at least one of the conditions (c1)–(c3); that is, one of $\mathcal{E}_1, \ldots, \mathcal{E}_4$ holds for all $2\sqrt{r}$ of them. This implies that

$$n_1 + n_2 + n_3 + n_4 \geq 2\sqrt{r}|R|. \tag{1}$$

To prove our claim, we will show that with high probability each quantity on the left is less than $\sqrt{r}|R|/2$. In the following, we fix a set R and separately estimate the probability that one of the quantities on the left is large. To establish the claim for all R, we will use the union bound over R. In the proof, we use the following version of the Chernoff-Hoeffding bound, which can be found in Dubhashi and Panconesi ([4], page 7).

Lemma 4 (Chernoff-Hoeffding bound)**.** *Let* $X := \sum_{i \in [n]} X_i$ *where* $X_i, i \in [n]$ *are independently distributed in* $[0, 1]$*. Let* $t > 2e\mathbb{E}[X]$*. Then*

$$\mathbb{P}[X > t] \leq 2^{-t}.$$

Claim 8. $\Pr[n_1 \geq \sqrt{r}|R|/2] \leq 2^{-r|R|/2}$.

Proof. The probability that a pointer from $P(b)$ falls on a segment of a previous pointer is at most $2\sqrt{r}/r$. Thus, the expected value of n_1 is at most $8|R|$. We may invoke lemma 4 and conclude that

$$\Pr[n_1 \geq \sqrt{r}|R|/2] \leq 2^{-\sqrt{r}|R|/2}.$$

□

Claim 9. $\Pr[n_2 \geq \sqrt{r}|R|/2] \leq 2^{-\sqrt{r}|R|/2}$.

Proof. A pointer falls on head of random pointer chain in a segment with probability at most $2/w_j$. Thus,

$$\mathbb{E}[n_2] \leq \left(\frac{2}{w_j}\right) 4\sqrt{r}|R| \leq \frac{160|R|}{(\log s)^3}.$$

Again, our claim follows by a routine application of Lemma 4. □

Claim 10. $\Pr[n_3 \geq \sqrt{r}|R|/2] = 0$.

Proof. If $n_3 \geq \sqrt{r}|R|/2$, then the total number of locations read by $\mathcal{Q}$ is at least

$$\begin{aligned} n_3 \frac{w_j}{8} &\geq \left(\frac{\sqrt{r}|R|}{2}\right) \cdot \frac{w_j}{8} \\ &\geq \left(\frac{\sqrt{r}2^j}{2}\right)\left(\frac{s}{8 \cdot 20 \cdot 2^j \log s}\right) \\ &\gg \frac{\sqrt{r}s}{320 \log s}. \end{aligned}$$

This contradicts our assumption that $\mathcal{Q}$ makes at most $\sqrt{r}s/(\log s)^4$ queries. □

Claim 11. $\Pr[n_4 \geq \sqrt{r}|R|/2] \leq 2^{-r|R|/2}$.

Proof. Let us first sketch informally why we do not expect n_4 to be large. Recall that in our random input we place a random pointer chain in the left half of each segment. Once a pointer has landed at a location in this segment, its predecessor is equally likely to be any of the other locations in the segment. So the first probe into that segment has probability about one in $w_j/2 - 1$ of landing on the predecessor, the second probe has probability about one in $w_j/2 - 2$ of landing on the predecessor, and so on. Since we assume $\mathcal{E}_3$ does not hold, there are at least $w_j/2 - w_j/8 - 1$ possibilities for the location of the predecessor. This implies that in order for n_4 to be at least $\sqrt{r}|R|/2$ the query algorithm $\mathcal{Q}$ must make $\Omega(w_j\sqrt{r}|R|/2)$ queries; but this number exceeds the number of probes $\mathcal{Q}$ is permitted.

In order to formalize this intuition, fix (condition on) a choice of pointers from V. Let us assume that the algorithm makes t probes. For $i = 1, 2, \ldots, t$, define indicator random variables χ_i as follows: $\chi_i = 1$ iff the following conditions hold.

- Suppose the i-th probe is made to a segment p in band $b \in \mathcal{B}$. Let ℓ be the location where the first pointer (among the pointers from $P(b)$ to p) lands. Then, the i-th probe of $\mathcal{Q}$ is made to the predecessor of ℓ in the random pointer chain in b.
- Fewer than $w_j/8$ of the previous probes were made to this segment.

Observe that if more than one pointer land on p, then except for the first amongst them (according to the ordering on the locations in $P(b)$), event $\mathcal{E}_2$ does not hold for the remaining pointers, and hence by definition event $\mathcal{E}_4$ does not hold either.

Define $Z = \sum_{i=1}^{t} \chi_i$.. Note that Z is an upper bound on n_4, and we wish to estimate the probability that $Z \geq \sqrt{r}|R|/2$. The key observation is that for every choice σ of $\chi_1, \chi_2, \ldots, \chi_{i-1}$, we have

$$\Pr[\chi_i = 1 \mid \chi_1, \chi_2, \ldots, \chi_{i-1} = \sigma] \leq \frac{1}{3w_j/8 - 1} \leq \frac{4}{w_j}. \tag{2}$$

Thus,

$$\mathbb{E}[Z] \leq \left(\frac{4}{w_j}\right) t \leq \left(\frac{4}{w_j}\right) (\log s)^{-4}\sqrt{r}s \leq (\log s)^{-2}\sqrt{r}|R|.$$

The variables χ_i are not independent, but it follows from (2) that Lemma 4 is still applicable in this setting. We conclude that

$$\Pr[Z \geq \sqrt{r}|R|/2] \leq 2^{-\sqrt{r}|R|/2}.$$

Since, the above bound holds for each choice of pointers from V, it holds in general. □

Finally, to establish the required expansion for all sets R, we use the union bound over all R. The probability that some set R has fewer than $|R|$ neighbors is at most

$$
\begin{aligned}
&4 \sum_{k=1}^{s/(\sqrt{r}(\log s)^4)} \binom{s/2}{k} 2^{-\sqrt{r}k/2} \\
&\leq \sum_{k\geq 1} s^k 2^{-\sqrt{r}k/2} \\
&\leq \sum_{k\geq 1} s^{-k} = o(1),
\end{aligned}
$$

where we used our assumption that $r \gg (\log s)^2$. This completes the proof of the matching lemma.

Acknowledgment: We thank Sagnik Mukhopadhyay for useful discussions.

References

[1] Scott Aaronson. A query complexity breakthrough. shtetl-optimized.

[2] Andris Ambainis, Kaspars Balodis, Aleksandrs Belovs, Troy Lee, Miklos Santha, and Juris Smotrovs. Separations in query complexity based on pointer functions. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 800–813, 2016.

[3] Manuel Blum and Russell Impagliazzo. Generic oracles and oracle classes (extended abstract). In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 118–126, 1987.

[4] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[5] Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:50, 2015.

[6] Juris Hartmanis and Lane A. Hemachandra. One-way functions, robustness, and the non-isomorphism of np-complete sets. In *Proceedings of the Second Annual Conference on Structure in Complexity Theory, Cornell University, Ithaca, New York, USA, June 16-19, 1987*, 1987.

[7] Sagnik Mukhopadhyay and Swagato Sanyal. Towards better separation between deterministic and randomized query complexity. In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, pages 206–220, 2015.

[8] Noam Nisan. CREW prams and decision trees. *SIAM J. Comput.*, 20(6):999–1007, 1991.

[9] Michael E. Saks and Avi Wigderson. Probabilistic boolean decision trees and the complexity of evaluating game trees. In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 29–38.

[10] Marc Snir. Lower bounds on probabilistic linear decision trees. *Theor. Comput. Sci.*, 38:69–82, 1985.

[11] Gábor Tardos. Query complexity, or why is it difficult to seperate NP $^{\mathrm{a}}$ cap co NP$^{\mathrm{a}}$ from P$^{\mathrm{a}}$ by random oracles a? *Combinatorica*, 9(4):385–392, 1989.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de