Amortized Dynamic Cell-Probe Lower Bounds from Four-Party Communication

Omri Weinstein[*] Huacheng Yu[†]

Abstract

This paper develops a new technique for proving amortized, randomized cell-probe lower bounds on dynamic data structure problems. We introduce a new randomized nondeterministic four-party communication model that enables "accelerated", error-preserving simulations of dynamic data structures.

We use this technique to prove an $\Omega(n\,(\log n/\log\log n)^2)$ cell-probe lower bound for the dynamic 2D weighted orthogonal range counting problem (2D-ORC) with $n/\mathrm{poly}\log n$ updates and n queries, that holds even for data structures with $\exp(-\tilde{\Omega}(n))$ success probability. This result not only proves the highest amortized lower bound to date, but is also tight in the strongest possible sense, as a matching upper bound can be obtained by a deterministic data structure with worst-case operational time. This is the first demonstration of a "sharp threshold" phenomenon for dynamic data structures.

Our broader motivation is that cell-probe lower bounds for exponentially small success facilitate *reductions from dynamic to static* data structures. As a proof-of-concept, we show that a slightly strengthened version of our lower bound would imply an $\Omega((\log n/\log\log n)^2)$ lower bound for the *static* 3D-ORC problem with $O(n\log^{O(1)} n)$ space. Such result would give a near quadratic improvement over the highest known static cell-probe lower bound, and break the long standing $\Omega(\log n)$ barrier for static data structures.

[*]Department of Computer Science, New York University. Supported by a Simons Society Junior Fellowship.
[†]Department of Computer Science, Stanford University. Supported by NSF CCF-1212372.

1 Introduction

Understanding the limitations of data structures in the cell-probe model [Yao81] is one of the holy grails of theoretical computer science, primarily since this model imposes very weak implementation constraints and hence captures essentially any imaginable data structure. Unfortunately, this abstraction makes it notoriously difficult to obtain lower bounds on the operational time of data structures, in spite of nearly four decades of active research. For dynamic data structures, where a sequence of n database operations (interleaved updates and queries) is to be correctly maintained, the highest *amortized* cell-probe lower bound to date is $\Omega(\log n)$ per operation, i.e., $\Omega(n \log n)$ for a sequence of $\Theta(n)$ operations (Pătraşcu and Demaine [PD06][1]). The breakthrough work of Larsen [Lar12] brought a near-quadratic improvement for *worst-case* number of probes per operation. Larsen gave an $\Omega((\log n/\log\log n)^2)$ query time lower bound for the dynamic *weighted orthogonal range counting* problem in two-dimensional space (2D-ORC), which holds for any data structure with at most polylogarithmic update time. In this fundamental problem, the data structure needs to maintain a set of weighted points in the two-dimensional plane, and support the following operations:

- `update(`r`, `c`, `w`)`: insert a point at (r, c) with weight w,
- `query(`r`, `c`)`: the sum of weights of points dominated by (r, c),[2]

where $r, c, w \in [n]$.[3] Larsen's aforementioned bound is tight when $O(\log^{2+\epsilon} n)$ update time is allowed, as there is a deterministic data structure that solves the problem using $O(\delta \log^2 n)$ probes per update and $O((\log_\delta n)^2)$ probes per query in the worst-case for any $\delta > 1$. However, it is often the case that amortization can reduce the average cell-probe complexity (a notable example is the Union Find problem [Tar75, Blu85]), especially when randomization is permitted and the data structure is allowed to err with constant probability *per query*.

Indeed, one particular shortcoming of all known dynamic data structure lower bounds is that they are not robust to error: All previous lower bounds only apply to deterministic, Las-Vegas or at most constant-error randomized data structures (e.g., [FS89, PD06, PT11, Yu15]). A more robust question, which we motivate below, is to study the rate of decay of success probability in answering all (or most) of the queries, as a function of the allocated resources (in our context, the total number of probes). The distinction above is similar in spirit to the difference between "direct sum" theorems in complexity theory (e.g., [FKNN95, KKN95, PT06, BBCR10]) which assert a lower bound on the number of resources required for solving multiple instances of a given problem with *constant overall success*, and "direct product" theorems such as the celebrated parallel repetition theorem [Raz98] and Yao's XOR lemma [Yao82], which further asserts an exponential decay in the success probability if insufficient resources are provided. Beyond unravelling the nature of "parallel computation", one of the primary motivations of direct product theorems is black-box hardness amplification (see e.g., [DS14] and references therein). In the context of the cell-probe model, we argue that such theorems open a new path for proving both dynamic and static data structure lower bounds, via *reductions* (more on this below).

Despite the long history of direct product theorems in complexity theory (see e.g. [JPY12] and references therein), we are not aware of any such result in the cell-probe model.[4] Indeed, a crucial assumption

[1]Notably, this bound holds only for high-probability data structures which succeed on solving all queries with probability $1 - n^{-\Omega(1)}$.

[2]A point (r', c') is dominated by (r, c) if $r' \le r$ and $c' \le c$.

[3]$[n]$ stands for the set of integers $\{1, 2, \ldots, n\}$.

[4]It is noteworthy that, unlike direct product theorems in other computational models such as two-prover games [Raz98], circuit complexity [Yao82] and interactive models and proof systems [JPY12, BRWY13], the dynamic cell-probe model is closer to the setting of *sequential repetition*, since the model is online: The data structure needs to provide answers to one query before it receives the next. This feature potentially makes the problem harder than "parallel repetition" (where all problem instances appear in a "batch").

which direct sum and product theorems rely on is the premise that all copies of the problem are *independent* of each other. Alas, in the dynamic data structure model, all q queries $Q_1, Q_2, \ldots, Q_q$ are essentially with respect to the *same* (or slightly modified) database X! Due to this (asymmetric) correlation, one should not expect generic ("black-box") direct product theorems for arbitrary dynamic data structure problems, and such a surprising result may only be true due to the specific structure of the underlying problem. The main result of this paper asserts that the 2D-ORC problem exhibits such interesting structure, leading to the following strong amortized lower bound:

Theorem 1 (Amortized Lower Bound for 2D-ORC)**.** *For any integer n, $1 \leq c < o(\log n/\log\log n)$, and any (randomized) data structure D in the cell-probe model with word-size $\Theta(\log n)$, there is a sequence of $n/\log^c n$ updates and $n - n/\log^c n$ queries for the* 2D-ORC *problem, for which the probability (over the randomness of D) that*

- *D probes $o(n\,(\log n/c\log\log n)^2)$ cells in total, and*
- *D is correct on all $n - n/\log^c n$ queries*

is at most $2^{-n/\log^{c+O(1)} n}$.

Theorem 1 not only provides a near quadratic improvement over the previous highest amortized cell-probe lower bound, but it is also tight in the strongest possible sense, as it exhibits a "sharp threshold" phenomenon for 2D-ORC: while $O(n\,(\log n/c\log\log n)^2)$ probes are sufficient to solve the problem *deterministically*, Theorem 1 asserts that any dynamic data structure that spends $\ll n\,(\log n/c\log\log n)^2$ probes will have success probability which is hardly any better than the trivial success probability of *randomly guessing the answers to all queries*! To best of our knowledge, this is the first result of its kind in the cell-probe model.

We note that it is possible to modify our proof of Theorem 1 so that the lower bound holds even if the second condition is relaxed to "D is correct on 99% of the $n - n/\log^c n$ queries". In many realistic dynamic scenarios, where the data structure is executed as a sub-procedure that supports a long sequence of (possibly multi-user) applications (e.g., routing, navigation and other network computations), this relaxed error criteria is more suitable and much less restrictive than requiring the data structure to succeed on all queries with an overall probability of 99%. Nevertheless, this "Chernoff-type" variant of Theorem 1 rules out efficient dynamic data structures for 2D-ORC even under this substantially more modest and realistic requirement.

The broader agenda we suggest and promote in this paper is that proving dynamic cell-probe lower bounds for data structures with exponentially small success probability facilitates *reductions from dynamic to static* data structure problems. The general outline of such reduction is as follows: suppose we can show that any (randomized) dynamic data structure for a problem $\mathcal{P}$ that has at least $\exp(-u)$ success probability in answering a sequence of queries with u updates, must probe at least t cells. We would like to argue that a *static* data structure D with a too-good query time for some static problem related to $\mathcal{P}$, must use a lot of space. Indeed, D can be used to solve the *dynamic* problem $\mathcal{P}$ with $> \exp(-u)$ probability, simply by *guessing* all u updates, preprocessing them and storing in the memory in advance, which in turn would imply that D must use a at least $\Omega(t)$ memory cells. Since in the dynamic problem $\mathcal{P}$, updates and queries are *interleaved*, answering the ith query Q_i requires knowing precisely those updates *preceding* Q_i in the sequence. This means that D must guess (and store) not only the updates themselves, but also the *time* (i.e., order) at which they occurred. One way to incorporate this extra information is to add an extra "time coordinate" to each query and update of the problem $\mathcal{P}$, which results in a slightly augmented (static) problem $\mathcal{P}^+$. In general, $\mathcal{P}^+$ might not correspond to any natural data structure problem, however, when $\mathcal{P}$ = 2D-ORC, this extra "time coordinate" can be embedded as a *third* dimension of the (weighted, two-dimensional) points, in which case the augmented static problem $\mathcal{P}^+$ corresponds to nothing else but the *three-dimensional* weighted orthogonal range counting problem (3D-ORC). As a proof-of-concept of the approach above, we show that if the bound in Theorem 1 can be slightly strengthened so that it holds for even smaller success probability (by a polylogarithmic factor in the exponent), then the following breakthrough result would follow for static 3D-ORC:

Proposition 1 (From dynamic 2D-ORC to static 3D-ORC)**.** *Suppose the probability in Theorem 1 can be further reduced to* $n^{-3n/\log^c n} = 2^{-3n/\log^{c-1} n}$*. Then any (zero-error) static data structure for* 3D-ORC *that uses* $n \log^{O(1)} n$ *space, requires* $\Omega\left((\log n/\log\log n)^2\right)$ *query time.*

In contrast, the best static lower bound to date for orthogonal range counting (in any dimension) is only $\Omega(\log n/\log\log n)$, even for *linear*-space data structures. In fact, no $\omega(\log m)$ lower bound is known for any static data structure problem, where m is the range of the queries (e.g., $m = n^2$ for 2D-ORC and $m = n^3$ for 3D-ORC). So while the slightly stronger premise of Proposition 1 appears to be non-trivial to prove (see the discussion in Appendix A.1), if this approach can be realized, it would yield a near-quadratic improvement in static cell-probe lower bounds. The formal proof of Proposition 1 can be found in Appendix A.

We remark that the aforementioned reduction is merely an example, while other reductions (e.g., between different dynamic problems) may be possible via similar outline. More generally, if one can show that, conditioned on some (low probability) event $\mathcal{W}$, a solution to problem A produces a solution to problem B, then ruling out efficient data structures for problem B with $\approx p(\mathcal{W})$ success, would yield a cell-probe lower bound for A as well.

In the remaining subsections of this introduction, we provide a brief outline of the new techniques we develop en-route to proving Theorem 1, and how they overcome limitations of previous techniques used in the dynamic cell-probe model.

1.1 Related work and previous techniques

Several techniques have been developed along the years for proving lower bounds on the cell-probe complexity of dynamic data structure problems. This line of work was aimed not just at proving lower bounds for a broader class of problems, but also at facilitating higher lower bounds for stronger and more realistic data structures (e.g., randomized, amortized). In most problems and applications, it is natural to assume that the description of an operation can fit in $O(1)$ words, and the most natural assumption on the word-size of the cell-probe model is $w = \Theta(\log n)$. In this regime, Fredman and Saks [FS89] first introduced the *chronogram* method, and used it to prove an $\Omega(\log n/\log\log n)$ lower bound for the 0-1 partial sum problem. This lower bound stood as a record for 15 years, until Pătraşcu and Demaine [PD04] introduced the *information transfer tree* technique which led to a tight $\Omega(\log n)$ lower bound for general partial sum, improving the highest lower bound by a factor of $\log\log n$. About a decade later, Larsen [Lar12] showed how to combine the chronogram method with the *cell-sampling* technique (which was used for proving *static* data structure lower bounds [PTW10]), and proved an $\Omega((\log n/\log\log n)^2)$ worst-case lower bound for 2D-ORC. In the natural regime, this is also the highest lower bound proved for *any* explicit problem hitherto. In the remainder of this subsection, we outline Larsen's approach and the challenges in extending his techniques to the type of dynamic lower bounds we seek.

We remark that other lower bounds have been proved for the regime where $w = \omega(\log n)$. In particular, Pătraşcu [Pat07] proved a matching $\Omega((\log n/\log\log n)^2)$ lower bound for the 2D-ORC problem, but only when both the weights of points and word-size are $\log^{2+\epsilon} n$ bits long (we elaborate on the connection between this result and our techniques in Section 1.2).

Larsen's approach. To prove the aforementioned $\Omega((\log n/\log\log n)^2)$ lower bound for 2D-ORC, one considers a sequence of n random updates. The idea is to show that after these n updates have been performed, a random query must probe many cells. More specifically, the n updates are partitioned into

$\Theta(\log n/\log\log n)$ *epochs*: $\ldots, \mathbf{U}_i, \ldots, \mathbf{U}_2, \mathbf{U}_1$, where the i-th epoch $\mathbf{U}_i$ consists of β^i updates for $\beta = \operatorname{poly}\log n$. The goal is to show that in expectation, a random query must read $\Omega(\log n/\log\log n)$ memory cells that are written during epoch i, but never overwritten later. Let us restrict the attention to epoch i and assume that all updates in other epochs are fixed arbitrarily (i.e., only $\mathbf{U}_i$ is random). Let S_i denote the set of cells whose last update occurred in epoch i. Indeed, any cell that is written *before* epoch i cannot contain any information about $\mathbf{U}_i$, while the construction guarantees that there are few cells written *after* epoch i, due to the exponential decay in the lengths of epochs. Thus, one concludes that "most" of the information the data structure learns about $\mathbf{U}_i$ comes from cell-probes to S_i. Then the basic idea is to sample a subset $C_i \subseteq S_i$ of a *fixed* size. Then for each query, the fewer cells in S_i the data structure probes when answering it, the more likely that all of them will belong to the random subset C_i. Thus, if a random query probes too few cells in S_i (in expectation), there will be too many queries that can be answered without probing any cell in $S_i \setminus C_i$. One then argues that the answers to these queries reveal too much information about $\mathbf{U}_i$, even more than they should: all cells in C_i can contain at most $|C_i| \cdot w$ bits of information. This yields a lower bound on the number of cells a random query must probe in S_i, and implies a query time lower bound.

The above approach relies on the fact that update time has a *worst-case* upper bound. Indeed, the statement that "very few cells are probed after $\mathbf{U}_i$" may no longer hold when we only have an amortized guarantee on the update time, because the data structure could spend a long time on epoch $\mathbf{U}_1$ (say). In fact, if we allow amortization for updates, the above sequence of operations is no longer hard, since the data structure can simply record each update until the last one, and then spend $O(n \log n)$ time to construct a static 2D-ORC data structure that operates in $O(\log n)$ query time. Over the n updates, it only spends $O(\log n)$ time per update "on average". Obviously, this is not a good dynamic data structure in general, because it is not even in a ready-to-query state until the very end.

To prove an amortized lower bound, it is therefore necessary to *interleave* queries and updates as in [PD04, PD06, PT11, Yu15]. We observe that a variation of Larsen's approach can be adapted to prove a zero-error-data-structure version of Lemma 10. Combining this version of the lemma with our proof of Theorem 1 would yield an alternate proof of our amortized lower bound for zero-error data structures. However, it seems highly non-trivial to generalize this proof so that it applies to data structures with exponentially small success probability. Roughly speaking, on the one hand, the cell-sampling technique appears to be inapplicable for simultaneous analysis of multiple queries as it only applies to a fixed memory state, whereas in our setup different queries are performed on different memory states. On the other hand, the "direct product" lower bound we seek requires analyzing the *conditional* success probability (and performance) of a given query, conditioned on success in previous queries. Conditioning on this event may leak a lot of information about previous updates, making the proof much more subtle and hard to analyze (note that this was not an issue for zero-error data structures!).

1.1.1 Communication-based techniques for dynamic lower bounds

One of the successful approaches for proving dynamic data structure lower bounds relies on reductions from the *communication complexity* model, where the general idea is to partition the operation sequence between Alice and Bob and the communication task is to answer all queries in Bob's operation interval. To prove a (meaningful) lower bound on the number of probes required by any data structure operating over the operation sequence, one needs to show that Alice and Bob can efficiently simulate any data structure for the dynamic problem, so that a data structure with too few probes induces a too-good-to-be-true protocol for the communication game (the problem then boils down to proving a communication lower bound for the communication problem, which is often easier to analyze). For the simulation to be fast, Bob needs to be able to efficiently obtain the (memory contents of) "relevant" cells probed in his interval that were updated during Alice's operation interval.

Indeed, the choice of the communication model is crucial for the simulation argument: If the communication model is too weak, the simulation would be "too slow" for proving a strong (or even meaningful) cell-probe lower bound; On the other hand, if the communication model is too strong, proving a high lower bound on the amount of communication may be extremely difficult or even impossible (as we elaborate below). Pătraşcu [Pat07] used the standard two-party randomized communication model to prove an $\Omega((\log n/\log\log n)^2)$ cell-probe lower bound on 2D-ORC, but only for (somewhat unnatural) weight-size and word-size $w = \Theta(\log^{2+\epsilon} n)$. This caveat stems from his simulation being "too slow": Pătraşcu's simulation argument requires Alice to send a very long message (a *Bloom Filter* of length $\approx \log^2 n$ bits per operation) in order for Bob to figure out the aforementioned set of "relevant" cells, hence for the simulation to produce a non-trivial communication lower bound, Alice's input had better dominate the latter communication step, which is precisely why points are chosen to have $(\log^{2+\epsilon} n)$-bit weights.

Pătraşcu and Thorup [PT11] somewhat remedied this by introducing simulation in the two-party *nondeterministic* communication model, in which a know-all "prover" (Merlin) can help the players reduce their communication by providing some (untrusted) *advice* which requires verification. While this technique can be used to speed up the simulation process (leading to new dynamic lower bound for several data structure problems), it turns out to be still too slow for the type of lower bounds we seek (in particular for range-counting problems). But even more importantly, the nondeterministic reduction of [PT11] does not readily extend beyond *zero-error* (Las-Vegas) data structures. Indeed, when the data structure and hence the simulating protocol are allowed to err, the simulation above naturally leads to *randomized nondeterministic* communication models such as $\mathsf{MA}^{cc} \cap \mathsf{coMA}^{cc}$. Proving strong lower bounds on such powerful models is a notoriously hard open problem (see e.g., [Kla11]), and in our case may even be impossible (indeed, the 2D-ORC problem is related to computation of inner-products over finite fields, which in turn admits a surprisingly efficient ($\tilde{O}(\sqrt{n})$ bit) MA-protocol [AW08]).

1.2 Our techniques and the 4ANC communication model

We introduce a new randomized nondeterministic communication model (which we hence term 4ANC) that solves both problems above, namely, it enables faster (error-preserving[5]) simulations of randomized data structures, yet in some aspect is much weaker than MA^{cc}, and hence amenable to substantial lower bounds. To enable a faster simulation (than [PT11, Yu15]), our model includes *two* provers (hence four parties in total) who are communicating only with Bob: The first prover (Merlin) is *trusted* but has limited "communication budget", while the second prover (Megan) is *untrusted*, yet has *unlimited* "communication budget". More precisely, the model requires Alice and Bob's computation to be correct *only when Merlin is "honest"*, but charges for each bit sent by Merlin (hence the model is only meaningful for computing two-party functions with large range, as Merlin can always send the final answer and the players would be done). In contrast, the model doesn't charge for Megan's message length, but requires Bob to *verify* that her message is correct (with probability 1!). Intuitively, the model allows Bob to receive some short "seed" of his choice (sent by Merlin), in such way that this "seed" can be used to extract much more information (a longer message sent by Megan) in a verifiable (i.e., consistent) way.

We show that this model can indeed help the players "speed up" their simulation: Merlin can send a succinct message (the "seed", which in the data structure simulation would correspond to some "succinct encoding" of memory addresses of the relevant (intersecting) cells probed by the data structure in both Alice and Bob's operation intervals), after which Megan can afford to send a *significantly longer* message (which is supposed to be the actual memory addresses of the aforementioned cells). Alice can then send Bob all the relevant *content* of these cells (using communication proportional to the *number of "relevant" cells probed* by the data structure (times w)). If both Merlin and Megan are honest, Bob has all the necessary information

[5]When seeking lower bound for data structures with tiny (exponentially small) success, such reductions must not introduce any (non-negligible) error, or else the soundness of the reduction is doomed to fail.

to answer his queries. Otherwise, if Megan is cheating (by sending inconsistent addresses with Merlin's seed), we argue that Bob can detect this during his simulation given all the information he received from Merlin and Alice, yielding a fast and admissible 4ANC protocol.

For our setting of the parameters, this simulation saves a $\mathrm{poly}\log(n)$ factor of communication for Alice, and $\approx \log n$ factor of communication for Bob, compared to the standard nondeterministic simulations of [PT11, Yu15]. We stress that this speed-up is essential to prove the cell-probe lower bound we seek on 2D-ORC, and is most likely to be important in future applications.

To solve the second problem, namely, to limit the power of the model (so that it is amenable to substantial lower bounds), we impose two important constraints on the non-deterministic advice of Merlin and Megan: Firstly, the provers can only talk to Bob, hence the model is *asymmetric* ; Secondly and most importantly, we require that the provers' advice are *unambiguous*, i.e., Merlin's (honest) message is uniquely determined by some pre-specified function of the player's inputs, and similarly, for each message sent by Merlin (whether he tells the truth or not), Megan's (honest) message is uniquely specified by the players' inputs and Merlin's message. These restrictions are tailored for data structure simulations, since for any (deterministic) data structure D, the aforementioned set of "relevant" memory cells probed by D is indeed a deterministic function of the operation sequence.

We show that these two features imply a generic structural fact about 4ANC protocols, namely, that low-communication protocols with *any nontrivial success* probability in this model induce large biased (i.e., "semi-monochromatic") rectangles in the underlying communication matrix (see Lemma 2). Intuitively, this follows from the uniqueness property of the model, which in turn implies that for any fixed messages sent by Merlin, the resulting protocol induces a partition of the input matrix into *disjoint* biased rectangles. In contrast, we remark that rectangles induced by $\mathsf{MA}^{\mathsf{cc}}$ protocols may *overlap* (as there may be multiple transcripts that correspond to the same input (x, y)), which is part of why proving strong lower bounds on $\mathsf{MA}^{\mathsf{cc}}$ is so difficult.

Therefore, ruling out efficient randomized communication protocols (and hence a too-good-to-be-true data structure) for a given communication problem boils down to ruling out large biased rectangles of the corresponding communication matrix. Since we wish to prove a lower bound for protocols with tiny (exponentially small) success for 2D-ORC, we must rule out rectangles with exponentially small "bias". This "direct-product" type result for 2D-ORC in the 4ANC model (Lemma 5) is one of the main steps of the proof of Theorem 1.

1.3 Organization

We begin by formally defining the 4ANC model in Section 3. We then prove that 4ANC protocols can efficiently simulate dynamic data structures (Section 3.1), and on the other hand, that efficient 4ANC protocols induce large biased rectangles of the underlying communication matrix (Section 3.2). In Section 4 we prove our main technical lemma which rules out such rectangles (even with exponentially small bias) for 2D-ORC (Lemma 5), and finally tie the pieces together to conclude the proof of Theorem 1.

2 Preliminaries

2.1 The Cell-Probe Model

A dynamic data structure in the cell-probe model consists of an array of memory cells, each of which can store w bits. Each memory cell is identified by a w-bit address, so the set of possible addresses is $[2^w]$. It is natural to assume that each cell has enough space to address (index) all update operations performed on it, hence we assume that $w = \Omega(\log n)$ when analyzing a sequence of n operations.

Upon an update operation, the data structure can perform read and write operations to its memory so as to reflect the update, by *probing* a subset of memory cells. This subset may be an arbitrary function of the update and the content of the memory cells previously probed during this process. The update time of a data structure is the number of probes made when processing an update (this complexity measure can be measured in worst-case or in an amortized sense). Similarly, upon a query operation, the data structure can perform a sequence of probes to read a subset of the memory cells in order to answer the query. Once again, this subset may by an arbitrary (adaptive) function of the query and previous cells probed during the processing of the query. The query time of a data structure is the number of probes made when processing a query.

2.2 Communication Complexity

In the classical two-party communication complexity model [Yao79], two players (Alice and Bob) receive inputs $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ respectively (possibly from some joint distribution $(x, y) \sim \mu$), and need to collaborate to solve some joint function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ of their inputs. To do so, they engage in an interactive communication *protocol* π. In round i, one player (which must be specified by the protocol) sends the other player a message m_i. In a *deterministic* protocol, m_i is a function of the previous transcript $m_{<i}$ and the input of the player (x if Alice is the speaker in round i and y if Bob is the speaker). In a (public-coin) *randomized* protocol, messages may further depend on a public random string r which is observed by both players (when the protocol or inputs are randomized, we sometimes use Π to denote the (random variable) corresponding to the transcript of π). The *communication cost* of π is the (worst-case) number of bits transmitted in the protocol in any execution of π (over x, y, r).

The *distributional communication complexity* of f with respect to input distribution μ and success δ is the minimum communication cost of a (deterministic) protocol which correctly solves $f(x, y)$ with probability $\geq \delta$ over μ. The *randomized communication complexity* of f is the minimum communication cost of a protocol which correctly solves $f(x, y)$ *for all inputs* x, y, with probability $\geq \delta$ over the public randomness r of the protocol. The two measures are related via Yao's minimax theorem [Yao77, Yao79]. The following basic definitions and properties of communication protocols are well known (see [KN97] for a more thorough exposition).

Definition 1 (Communication matrix)**.** *The communication matrix $M(f)$ of a two-party function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$, is the matrix indexed by rows and columns $x \in \mathcal{X}, y \in \mathcal{Y}$, whose entries are $M(f)_{x,y} = f(x, y)$.*

Definition 2 (Combinatorial rectangles and monochromatic rectangles)**.** *A* combinatorial rectangle *(or simply, a rectangle) of $M(f)$ is a subset $R = X \times Y$ of inputs, such that $X \subseteq \mathcal{X}$, $Y \subseteq \mathcal{Y}$. A rectangle R is said to be* monochromatic *if f's value is fixed on all inputs $(x, y) \in R$.*

The following weaker definition will be more suitable for measuring error in the asymmetric randomized 4ANC model we define in this paper:

Definition 3 (α-column-monochromatic rectangles)**.** *Let μ be a joint distribution over inputs (x, y). A rectangle $R = X \times Y$ of $M(f)$ is said to be* α-column-monochromatic *with respect to μ, if for every $y \in Y$, at least an α-fraction of the entries in column y of R have the same value in $M(f)$, i.e., there is some function value $v_y \in \mathcal{Z}$ such that $\mu((X \times \{y\}) \cap \{f^{-1}(v_y)\}) \geq \alpha\mu(X \times \{y\})$.*

A basic fact in communication complexity is that a c-bit communication protocols that computes $f(x, y)$ in the *determinisitc* model, induces a partition of $M(f)$ into at most 2^c *monochromatic rectangles*. Each rectangle corresponds to a transcript of π (i.e., the set of inputs for which this transcript will occur forms a rectangle). A similar characterization holds in the *randomized* or *distributional* models, where rectangles are "nearly" monochromatic. We will show that protocols in the 4ANC model also induce a similar structure on

$M(f)$, in terms of biased column-monochromatic rectangles (see Lemma 2). Hence ruling out large biased column-monochromatic rectangles in $M(f)$ can be used to prove communication lower bounds on f in the 4ANC model.

3 4ANC: A New Four-Party Nondeterministic Communication Model

We now formally define the 4ANC model, which is a randomized, asymmetric, non-deterministic communication model involving four players: Alice, Bob, Merlin and Megan. Let μ be a distribution over input pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ to Alice and Bob (respectively). A 4ANC protocol P proceeds as follows: In the first stage, Alice and Bob use shared randomness to sample a public random string r (of infinite length), which is visible to all four players (Merlin, Megan, Alice and Bob). Merlin and Megan observe (x, y, r), and can each send, in turn, a message ("advice") *to Bob* before the communication proceeds in a standard fashion between Alice and Bob. As part of the protocol, P specifies, for each input pair and public string r, a unique message $M_{\text{mer}}(x, y, r)$ that Merlin is supposed to send given that input pair and random string (Merlin may not be honest, but we will only require the computation to be correct when he sends the correct message $M_{\text{mer}}(x, y, r)$). After Merlin sends Bob his message m_{mer} (which may or may not be the "correct" message $M_{\text{mer}}(x, y, r)$), it is Megan's turn to send Bob a message. Once again, P specifies (at most) one message[6] $M_{\text{meg}}(x, y, m_{\text{mer}}, r)$ that Megan is supposed to send to Bob, given x, y, r and Merlin's message m_{mer} (as we shall see, the difference between Merlin and Megan's role is that, unlike the case with Merlin's message, the players are *responsible to verify* that Megan's message is indeed correct, i.e., that $m_{\text{meg}} = M_{\text{meg}}(x, y, m_{\text{mer}}, r)$, no matter whether $m_{\text{mer}} = M_{\text{mer}}(x, y, r)$ or not!). In the next stage, Alice and Bob communicate in the standard public-coin communication model, after which Bob decides to "proceed" or "reject" (this is the verification step of Megan's message). Finally, if Bob chooses to proceed, he outputs a value v for $f(x, y)$. These stages are formally described in Figure 1.

Honest Protocol $\widetilde{P}$. Throughout the paper, we denote by $\widetilde{P}$ the *honest* execution of a 4ANC protocol P. More formally, we define $\widetilde{p}(x, y, r, \tau)$ to be the joint probability distribution of x, y, r and P's transcript τ, when Merlin and Megan send the honest messages (i.e., when $m_{\text{mer}} = M_{\text{mer}}(x, y, r)$ and $m_{\text{meg}} = M_{\text{meg}}(x, y, m_{\text{mer}}, r)$). Note that $\widetilde{p}$ induces a well defined distribution on transcripts τ, since the transcript of P is completely determined by (x, y, r) in this case.

Definition 4 (Valid protocols)**.** *A* 4ANC *protocol P is said to be* valid *if*

- *Bob* proceeds *if and only if $m_{meg} = M_{meg}(x, y, m_{mer}, r)$ (with probability 1).*

Definition 5 (Computation and notation in the 4ANC model)**.** *We say that a* 4ANC *protocol P δ-solves a two-party function $f : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathcal{Z}$ with communication cost (c_A, c_B, c_M) under input distribution μ if the following conditions hold.*

1. *(Perfect verification of Megan) P is a valid protocol.*
2. *(Communication and correctness) With probability at least δ (over the "honest" distribution $\widetilde{p}$ and input distribution μ), the honest protocol $\widetilde{P}$ satisfies that Alice sends no more than c_A bits, Bob sends no more than c_B bits, Merlin sends no more than c_M bits, and Bob outputs the correct value ($v = f(x, y)$).*

[6]Instead of *unique*, M_{meg} can be undefined for obviously wrong m_{mer}. But $M_{\text{meg}}(x, y, M_{\text{mer}}(x, y, r), r)$ is always defined.

A 4-party communication protocol P
0. Alice and Bob generate a public random string r, visible to all four players. 1. Merlin sends a message m_{mer} to *Bob* (m_{mer} is visible to Megan). 2. Megan sends a message m_{meg} to *Bob*. 3. Alice and Bob communicate based on their own inputs and m_{mer} and m_{meg} as if they were in the classic communication setting with public randomness. 4. Bob decides to *proceed* or *reject*. 5. If Bob chooses to proceed, he outputs a value v.

Figure 1: A communication protocol P in the 4ANC model.

For a two-party function $f : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathcal{Z}$ and parameters c_M, c_A, c_B, we denote by

$$\mathsf{Suc}^f_\mu(c_A, c_B, c_M)$$

the largest probability δ for which there is a 4ANC *protocol that δ-solves f under μ with communication cost (c_A, c_B, c_M).*

Remark. *A few remarks about the model are in order :*

1. *Any function $f : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathcal{Z}$ admits the following three trivial* 4ANC *protocols:*
   - *Alice sends Bob x: costs* $(\log|\mathcal{X}|, 0, 0)$.
   - *Bob sends Alice y: costs* $(\log|\mathcal{Z}|, \log|\mathcal{Y}|, 0)$.
   - *Merlin sends $f(x, y)$: costs* $(0, 0, \log|\mathcal{Z}|)$.
2. *The players always trust Merlin (since the* 4ANC *model requires the protocol to be correct only when he is honest). Nevertheless, Merlin is still allowed to cheat ($m_{mer} \neq M_{mer}$). Even in this case, there is always at most one "correct" message Megan should send. This property will be crucial for the characterization of* 4ANC *protocols in terms of monochromatic rectangles (see Subsection 3.2).*
3. *It is important that Bob is able to verify Megan's message with probability 1, but could be wrong on outputting the function value. This corresponds to the requirement that the players need to simulate the data structure perfectly, while the data structure itself might succeed with very small probability.*
4. *Megan is only useful when her advice (m_{meg}) is significantly longer than Merlin's advice (m_{mer}), as otherwise Merlin might as well send Megan's message (and she can remain silent). The benefit here is that the model doesn't charge the protocol for Megan's message length (only for verifying it is correct), so when Merlin is honest, Megan helps the players "speed up" the protocol.*

3.1 Data structure simulation in the 4ANC model

In this subsection, we show that it is possible to efficiently *simulate* any dynamic data structure on any sequence of operations in the 4ANC model with no additional error. To this end, consider a (deterministic) data structure D for some problem $\mathcal{P}$, and fix a sequence $\mathcal{O}$ of operations. Let I_A and I_B be two *consecutive* intervals of operations in $\mathcal{O}$ such that I_A occurs right before I_B. Let $P_D(I_A)$ and $P_D(I_B)$ be the set of cells probed by D during I_A and I_B respectively (when D is clear from context, we shall simply write $P(I_A)$ and $P(I_B)$). Alice is given all operations except for the ones in I_B, Bob is given all operations except for the ones in I_A. We now describe an 4ANC protocol that simulates D on $\mathcal{O}$ and has the same output as D on all queries in I_B.

The naive approach for this simulation is to let Bob simulate the data structure upto the beginning of I_A, then skip I_A and continue the simulation in I_B. To collect the "relevant" information on what happened in I_A, each time D probes a cell that has not been probed in I_B before, Bob asks Alice whether this cell was previously probed in I_A, and if it was, he asks Alice to send the new content of that cell. Unfortunately, this approach requires Bob to send $|P(I_B)| \cdot w$ bits and Alice to send $|P(I_B)| + |P(I_A) \cap P(I_B)| \cdot w$ bits. However, the players can do much better with Merlin's and Megan's help: Merlin reports Bob, upfront, which cells in $P(I_B)$ are probed in I_A in some succinct encoding. Given Merlin's succinct message, Megan can send Bob the actual memory addresses of these cells, and the players will be able to easily verify these addresses are consistent with the "seed" sent by Merlin, as the model requires. With this information in hand, Bob only needs to ask Alice for the contents of *relevant cells* in his simulation, instead of every cell in $P(I_B)$. Moreover, it allows Bob to send this set of cells *in batch*, further reducing his message length. We turn to describe the formal simulation.

Protocol SIM$_D$ for Simulating D:

1. (Protocol specification of M_{mer}.) Merlin simulates D upto the end of I_B, and generates $P(I_A), P(I_B)$. He sends Bob the sizes $|P(I_A)|$, $|P(I_B)|$ and $|P(I_A) \cap P(I_B)|$.[7] Then he writes downs the sequence of cells probed during I_B in the chronological order (if a cell is probed more than once, he keeps only the first occurrence). Each cell in the sequence is associated with a bit, indicating whether this cell is also in $P(I_A)$. By definition, this sequence has length $|P(I_B)|$, in which $|P(I_A) \cap P(I_B)|$ cells are associated with a "1". Merlin sends Bob the set of indices in the sequence associated with a "1". In total, Merlin sends
$$\begin{aligned} &O(\log|P(I_A)| + \log|P(I_B)|) + \log\binom{|P(I_B)|}{|P(I_A) \cap P(I_B)|} \\ &\le |P(I_A) \cap P(I_B)| \cdot \log\frac{e|P(I_B)|}{|P(I_A) \cap P(I_B)|} + O(\log n) \end{aligned}$$
bits. Note that Merlin's message encodes for each i, whether the i-th time (during I_B) that D probes a new cell, it was previously probed during I_A.

2. (Protocol specification of M_{meg}.) Megan simulates D upto the beginning of I_A, saves a copy of the memory M_A, and continues the simulation upto the beginning of I_B, saves a copy of the memory M_B. Then she continues to simulate D on I_B *from* $\mathbf{M_A}$ with the advice from Merlin. That is, whenever she needs to probe a cell that has not been probed in I_B before, if this is the i-th time that this happens and Merlin's message has i encoded in the set, Megan copies the content of the cell from M_B to the current memory, writes down the address of the cell and continues the simulation. Basically Megan simulates D assuming Merlin's claim about which cells probed in I_B are probed in I_A is correct. If there is anything inconsistent during the simulation, M_{meg} is undefined, e.g., $|P(I_B)|$ or $|P(I_A) \cap P(I_B)|$ is different from what Merlin claims, or D breaks during the simulation due to the wrong contents of the memory, etc. As long as Merlin's message is consistent with Megan's simulation, she sends

[7]Note that although Bob knows all the operations in I_B, he still does not know $P(I_B)$, since the operations in I_A are unknown to him, and the data structure can be adaptive.

the set of actual memory *addresses* of cells she has written down during the simulation (i.e., the set $P(I_A) \cap P(I_B)$ from Merlin's advice) using $|P(I_A) \cap P(I_B)| \cdot w$ bits.

3. (Bob asks the contents of $P(I_A) \cap P(I_B)$.) Denote the set of addresses received from Megan by S. If $|S| \neq |P(I_A) \cap P(I_B)|$, Bob rejects. Alice and Bob use public randomness to sample a random (hash) function $h : [2^w] \to [|P(I_A)|]$. Bob sends Alice the set of hash-values $h(S)$ using
$$\log \binom{|P(I_A)|}{|P(I_A) \cap P(I_B)|} \leq |P(I_A) \cap P(I_B)| \cdot \log \frac{e|P(I_A)|}{|P(I_A) \cap P(I_B)|}$$
bits.

4. (Alice replies with the contents.) Alice simulates D and obtains the set $P(I_A)$. For each hash-value $b \in h(S)$, Alice sends Bob both addresses and contents of *all* cells in $P(I_A)$ that are mapped to this value (i.e., of $h^{-1}(b) \cap P(I_A)$). Alice sends $4|P(I_A) \cap P(I_B)| \cdot w$ bits in expectation (over the randomness of the hash function).

5. (Bob simulates D and verifies Megan.) Bob checks whether Alice sends the information about all cells in S. If not, he rejects. Otherwise, he simulates the data structure up to the beginning of I_A, and then updates all cells in S to the new values. Bob continues the simulation on I_B from this memory state. At last, Bob checks whether the simulation matches Merlin's claim and whether S is exactly the set $P(I_A) \cap P(I_B)$ according to the simulation. If either check fails, he rejects. Otherwise, he proceeds, and generates the output of D on all queries in I_B.

Lemma 1. *Let $\mathcal{O}$ be an operation sequence, $I_A, I_B \subseteq \mathcal{O}$ be any consecutive operation intervals. Then for any deterministic data structure D operating over $\mathcal{O}$, $\text{SIM}_D(I_A, I_B)$ is a valid* 4ANC *protocol. Moreover, the honest protocol $\widetilde{\text{SIM}_D}$ has precisely the same output as D on all queries in I_B, with Alice sending*
$$4|P(I_A) \cap P(I_B)| \cdot w$$
bits in expectation, Bob sending at most
$$|P(I_A) \cap P(I_B)| \cdot \log \frac{e|P(I_A)|}{|P(I_A) \cap P(I_B)|}$$
bits, and Merlin sending at most
$$|P(I_A) \cap P(I_B)| \cdot \log \frac{e|P(I_B)|}{|P(I_A) \cap P(I_B)|} + O(\log n)$$
bits.

Proof. The claimed communication cost of the protocol can be directly verified from steps 1,3 and 4 respectively, so we only need to argue about the correctness and validity of the protocol. By construction, when both Merlin and Megan are honest, Bob has all the up-to-date information (i.e., latest memory state) of the cells $P(I_A) \cap P(I_B)$ probed by D during his operation interval, hence by definition of step 5, $\widetilde{\text{SIM}_D}$ has the same output as D on queries in I_B. It therefore remains to show that SIM_D is a valid 4ANC protocol.

To this end, recall that we need to show that for any message m_{mer} sent by Merlin, Bob proceeds iff $M_{\text{meg}}(\mathcal{O}, m_{\text{mer}}) = m_{\text{meg}}$. When Megan is honest (follows the protocol) and sends the set S, Bob's simulation of D in step 5 will be exactly the same as Megan's in step 2. By definition, S is the exact set of cells that Megan uses the contents from M_B instead of M_A. By copying the contents of S from Alice's memory state (M_B) to Bob's memory state (M_A), he recovers Megan's simulation, which is consistent with Merlin's message. Thus, Bob will proceed.

When $M_{\text{meg}}(\mathcal{O}, m_{\text{mer}})$ is undefined and Megan follows the protocol and sends the set S she generates in step 2 (but finds inconsistency), by the same argument as above, Bob recovers Megan's simulation, thus will find the same inconsistency as Megan does and reject.

The only case left is when Megan chooses to send a different set S' than S (no matter whether $M_{\text{meg}}(\mathcal{O}, m_{\text{mer}})$ is defined). Let $P_{m_{\text{mer}}}(I_B)$ be the set of cells probed during I_B as specified in step 2, given Merlin's advice m_{mer}. By definition, $S \subseteq P_{m_{\text{mer}}}(I_B)$. If $S' \cap P_{m_{\text{mer}}}(I_B) = S$ and $S' \neq S$, then by the same argument again, Bob recovers the simulation specified in step 2. In the end, he will find that not every cell in S' is probed and reject. Otherwise, consider the symmetric difference $(S' \cap P_{m_{\text{mer}}}(I_B)) \triangle S$. Let C be the first cell in the symmetric difference in the chronological order of probing cells in $P_{m_{\text{mer}}}(I_B)$, which is the j-th new cell probed. Thus, Bob will successfully recover the simulation until he is about to probe cell C. By definition, $C \notin S'$, if and only if $C \in S$, if and only if j is encoded in m_{mer}. Thus, on cell C, Bob will find the simulation does not match Merlin's claim, and thus reject. □

3.2 Efficient 4ANC protocols induce large biased rectangles

Let P be a four-party communication protocol computing f over a *product* input distribution $\mu = \mu_x \times \mu_y$ in the 4ANC communication model, with cost (c_A, c_B, c_M) and success probability δ. In this section, we are going to prove that if P is efficient (has low communication) and has any "non-trivial" accuracy in computing the underlying function f, then there must be a large biased-column-monochromatic rectangle in the communication matrix of f (see Definition 3 for the formal definition). We note that a variant of this lemma can be proved for general (non-product) distributions.

Lemma 2 (4ANC protocols imply large biased rectangles for product distributions)**.** *$M(f)$ has a rectangle $R = X \times Y$ such that*

1. *R is $\frac{\delta}{2} \cdot 2^{-c_M}$-column-monochromatic;*
2. *$\mu_x(X) \geq \frac{\delta}{4} \cdot 2^{-(c_M + c_A + c_B)}$;*
3. *$\mu_y(Y) \geq \frac{\delta}{4} \cdot 2^{-(c_M + c_B)}$.*

Proof. Recall that $\widetilde{P}$ denotes the honest execution of the protocol P, and by definition of the 4ANC model, the probability that $\widetilde{P}$ correctly computes $f(x, y)$ *and* communicates at most (c_A, c_B, c_M) bits respectively, is at least δ. Since we are working over a fixed input distribution μ, we may fix the public randomness of P to some fixed value ($r = r^*$) so that these conditions continue to hold for the *deterministic* protocol P_{r^*} (over the input distribution μ). Define S to be the set of *good* input pairs (x, y) for which $\widetilde{P}_{r^*}$ correctly computes $f(x, y)$ and the the protocol communicates (c_A, c_B, c_M) bits respectively. By definition,

$$\mu(S) \geq \delta.$$

For the remainder of the proof, we assume P is deterministic (i.e., we implicitly consider the protocol $P = P_{r^*}$). Now consider a transcript $\tau = (\pi, m_{\text{meg}})$ of the deterministic protocol P, where m_{meg} denotes Megan's message to Bob, and $\pi = (m_{\text{mer}}, \pi_A, \pi_B)$ denotes the message from Merlin and the transcript between Alice and Bob. For a given message m_{mer} sent by Merlin, the set of input pairs that will generate the transcript τ (and for which Bob "proceeds") form a combinatorial rectangle $R_\tau = X_\tau \times Y_\tau$.[8] Assuming Bob "proceeds" in τ, he is supposed to output a value after the communication, which may depend on the

[8]Note that not necessarily every pair in the R_τ has $M_{\text{mer}} = m_{\text{mer}}$, as P is not required to verify whether Merlin sends the correct message!

transcript and his input $y \in Y_\tau$. That is, conditioned on Merlin sending m_{mer} (which, once again, may not be the "honest" message), for each column y of the rectangle R_τ Bob will output the same value. Now, recall that for each input (x, y) and m_{mer} there is at most one message $m_{\text{meg}} = M_{\text{meg}}(x, y, m_{\text{mer}})$ that will make Bob accept (i.e., (x, y, m_{mer}) uniquely determine m_{meg}, and hence the entire transcript τ). Since P is deterministic, this fact implies that if we fix m_{mer}, all rectangles $\{R_\tau\}$ are *disjoint* from each other.

Furthermore, since Alice does not observe Megan's message (only Bob does), the set X_τ does not depend on m_{meg}. This means that if we fix π and vary over all m_{meg} consistent with π, all rectangles corresponding to resulting transcripts τ will have *the same* X_τ. Let $R_\pi = \bigcup_{\tau=(\pi,m_{\text{meg}})} R_\tau$ be the (disjoint) union of these rectangles, which is a rectangle itself, and let $X_\pi \times Y_\pi = R_\pi$. In this notation, for any $\tau = (\pi, m_{\text{meg}})$ we have that $X_\pi = X_\tau$, and $Y_\pi = \bigcup_{\tau=(\pi,m_{\text{meg}})} Y_\tau$. The following claim asserts that every column y of R_π will have the same output:

Claim 1. *For each transcript $\pi = (m_{mer}, \pi_A, \pi_B)$ and $y \in Y_\pi$, $M_{meg}(x, y, m_{mer})$ is fixed across $x \in X_\pi$.*

Proof. Suppose towards contradiction that there is some $y_0 \in Y_\pi$ and $x_1, x_2 \in X_\pi$, such that

$$M_{\text{meg}}(x_1, y_0, m_{\text{mer}}) \neq M_{\text{meg}}(x_2, y_0, m_{\text{mer}}).$$

Now, given m_{meg}, Bob's decision whether to "proceed" or not only depends on his input y_0 and the transcript π which, by definition, is the same for both inout pairs $(x_1, y_0), (x_2, y_0)$. This means that for at least one of the input pairs, say (x_1, y_0), Bob will "proceed" even when $m_{\text{meg}} = M_{\text{meg}}(x_2, y_0, m_{\text{mer}})$, contradicting the definition of the 4ANC model (proposition (1) in Definition 5). □

Indeed, the above claim asserts that Bob's output is only a function of (π, y), so let us henceforth denote by $v(\pi, y)$ the output of column y of R_π. Once again, note that for a fixed value of m_{mer}, the rectangles $\{R_\pi\}$ are all disjoint. In particular, this fact implies

$$\sum_{\substack{\pi=(m_{\text{mer}},\pi_A,\pi_B):\\ |m_{\text{mer}}|\le c_M, |\pi_A|\le c_A, |\pi_B|\le c_B}} \mu(R_\pi) \le 2^{c_M}. \tag{1}$$

Now, by definition, every *good* input pair $(x, y) \in S$ is contained in some rectangle R_π, where $m_{\text{mer}} = M_{\text{mer}}(x, y)$, $|M_{\text{mer}}(x, y)| \le c_M$, $|\pi_A| \le c_A$, $|\pi_B| \le c_B$ and $v(\pi, y) = f(x, y)$ (by definition of S). Therefore, we have

$$\sum_{\substack{\pi=(m_{\text{mer}},\pi_A,\pi_B):\\ |m_{\text{mer}}|\le c_M, |\pi_A|\le c_A, |\pi_B|\le c_B}} \mu(R_\pi \cap S \cap M_{\text{mer}}^{-1}(m_{\text{mer}})) = \mu(S) \ge \delta. \tag{2}$$

By Equation (1) and (2), we expect that "on average", each rectangle has roughly $\delta \cdot 2^{-c_M}$ fraction of the pairs that are good and match the m_{mer} value of the rectangle. By Markov's inequality, we can indeed show that many rectangles have many *columns* with at least this fraction (up to a constant factor). More formally, for each Merlin's message, define

$$\tilde{Y}_\pi = \left\{ y \in Y_\pi : \mu\left((X_\pi \times \{y\}) \cap S \cap M_{\text{mer}}^{-1}(m_{\text{mer}})\right) \ge \frac{\delta}{2} \cdot 2^{-c_M} \cdot \mu(X_\pi \times \{y\}) \right\}$$

to be the set of *columns* in R_π with many good input pairs and matching m_{mer},

$$\tilde{R}_\pi = X_\pi \times \tilde{Y}_\pi$$

to be the union of these columns,

$$\mathcal{R}_{m_{\text{mer}}} = \left\{ \tilde{R}_\pi : |\pi_A| \le c_A, |\pi_B| \le c_B \right\},$$

and let

$$\mathcal{R} = \bigcup_{m_{\text{mer}} \,:\, |m_{\text{mer}}| \le c_M} \mathcal{R}_{m_{\text{mer}}}$$

be the set of rectangles with "sufficiently many" good input pairs and matching m_{mer} in *every* column.

Again, by definition, for $(x, y) \in S$, when $m_{\text{mer}} = M_{\text{mer}}(x, y)$, the protocol outputs the correct function value $f(x, y)$. Thus, every column of each rectangle $\tilde{R}_\pi$ has at least $\frac{\delta}{2} \cdot 2^{-c_M}$ fraction of the inputs having the same function value, i.e., each $\tilde{R}_\pi$ is $\frac{\delta}{2} \cdot 2^{-c_M}$-column-monochromatic.

It remains to show that at least one of these rectangles is large (satisfying propositions 2 and 3 of the lemma). Indeed, by Equation (1) and (2), we have

$$\begin{aligned}
\sum_{\tilde{R}_\pi \in \mathcal{R}} \mu(\tilde{R}_\pi) &\ge \sum_{\tilde{R}_\pi \in \mathcal{R}} \mu(\tilde{R}_\pi \cap S \cap M_{\text{mer}}^{-1}(m_{\text{mer}})) \\
&= \sum_{\substack{\pi=(m_{\text{mer}},\pi_A,\pi_B): \\ |m_{\text{mer}}| \le c_M, |\pi_A| \le c_A, |\pi_B| \le c_B}} \mu(\tilde{R}_\pi \cap S \cap M_{\text{mer}}^{-1}(m_{\text{mer}})) \\
&\ge \delta - \sum_{\substack{\pi=(m_{\text{mer}},\pi_A,\pi_B): \\ |m_{\text{mer}}| \le c_M, |\pi_A| \le c_A, |\pi_B| \le c_B}} \mu((R_\pi \setminus \tilde{R}_\pi) \cap S \cap M_{\text{mer}}^{-1}(m_{\text{mer}})) \quad \text{(by (2))} \\
&\ge \delta - \frac{\delta}{2} \cdot 2^{-c_M} \sum_{\substack{\pi=(m_{\text{mer}},\pi_A,\pi_B): \\ |m_{\text{mer}}| \le c_M, |\pi_A| \le c_A, |\pi_B| \le c_B}} \mu(R_\pi \setminus \tilde{R}_\pi) \quad \text{(by definition of } \tilde{R}_\pi) \\
&\ge \delta - \delta/2 = \delta/2. \quad \text{(by (1))}
\end{aligned}$$

In particular, there is one $\hat{m}_{\text{mer}}$ such that

$$\sum_{\tilde{R}_\pi \in \mathcal{R}_{\hat{m}_{\text{mer}}}} \mu(\tilde{R}_\pi) \ge (\delta/2) \cdot 2^{-c_M}. \tag{3}$$

From now on, let us fix Merlin's message to be $\hat{m}_{\text{mer}}$, and focus on $\mathcal{R}_{\hat{m}_{\text{mer}}}$. Recall that, by definition, for each $\tilde{R}_\pi \in \mathcal{R}_{\hat{m}_{\text{mer}}}$ with $\pi = (\hat{m}_{\text{mer}}, \pi_A, \pi_B)$, we have $|\pi_A| \le c_A$ and $|\pi_B| \le c_B$. For every $x \in \mathcal{X}$, let

$$S_x^{\hat{m}_{\text{mer}}} := \{\pi : \tilde{R}_\pi \in \mathcal{R}_{\hat{m}_{\text{mer}}}, x \in X_\pi\}$$

be the set of all possible transcripts $\pi \in \mathcal{R}_{\hat{m}_{\text{mer}}}$ that can be generated by $x, \hat{m}_{\text{mer}}$ (and any y). Intuitively, since Bob sends at most c_B bits in π, $S_x^{\hat{m}_{\text{mer}}}$ can be of size at most 2^{c_B}. This is the content of the following simple claim:

Claim 2. *For every $x \in \mathcal{X}$, $|S_x^{\hat{m}_{mer}}| \le 2^{c_B}$.*

Proof. Let $(\Pi|\mathcal{R}_{\hat{m}_{\text{mer}}})$ denote a *uniformly* random transcript $\pi \in \mathcal{R}_{\hat{m}_{\text{mer}}}$. Since P is deterministic, the random variable $(\Pi|x, \mathcal{R}_{\hat{m}_{\text{mer}}})$ is uniformly distributed over $S_x^{\hat{m}_{\text{mer}}}$. Thus, $|S_x^{\hat{m}_{\text{mer}}}| = 2^{H(\Pi|x,\mathcal{R}_{\hat{m}_{\text{mer}}})}$. Let Π_i denote the i'th message (not necessarily bit) sent in π (assuming messages are prefix-free). Then we may assume, without loss of generality, that Alice speaks in odd rounds of π and Bob speaks in even rounds. By the chain rule for entropy, we have

$$\begin{aligned} H(\Pi|x, \mathcal{R}_{\hat{m}_{\text{mer}}}) &= \sum_{\text{round } i} H(\Pi_i|\Pi_{<i}, x, \mathcal{R}_{\hat{m}_{\text{mer}}}) \\ &= \sum_{\text{even } i} H(\Pi_i|\Pi_{<i}, x, \mathcal{R}_{\hat{m}_{\text{mer}}}) \leq \sum_{\text{even } i} |(\Pi_i|\mathcal{R}_{\hat{m}_{\text{mer}}})| \leq c_B, \end{aligned}$$

where in the second transition we used the fact that for messages Π_i sent by Alice, we have $H(\Pi_i|\Pi_{<i}, x, \mathcal{R}_{\hat{m}_{\text{mer}}}) = 0$ since Π is deterministic, and the last transition follows from the assumption that $|\pi_B| \leq c_B$ for every $\pi \in \mathcal{R}_{\hat{m}_{\text{mer}}}$. This completes the proof. $\square$

With this claim in hand, we can now bound the fraction of rectangles in $\mathcal{R}_{\hat{m}_{\text{mer}}}$ with a "small Bob side" ($\tilde{Y}_\pi$):

$$\begin{aligned} \sum_{\substack{\tilde{R}_\pi \in \mathcal{R}_{\hat{m}_{\text{mer}}}: \\ \mu_y(\tilde{Y}_\pi) < (\delta/4)\cdot 2^{-c_M - c_B}}} \mu(\tilde{R}_\pi) &= \sum_{\substack{\tilde{R}_\pi \in \mathcal{R}_{\hat{m}_{\text{mer}}}: \\ \mu_y(\tilde{Y}_\pi) < (\delta/4)\cdot 2^{-c_M - c_B}}} \mu_x(X_\pi) \cdot \mu_y(\tilde{Y}_\pi) \qquad (4) \\ &< \sum_{x \in \mathcal{X}} \sum_{\substack{\tilde{R}_\pi \in \mathcal{R}_{\hat{m}_{\text{mer}}}: \\ \mu_y(\tilde{Y}_\pi) < (\delta/4)\cdot 2^{-c_M - c_B}}} \mu_x(x) \cdot \mathbf{1}_{X_\pi}(x) \cdot (\delta/4) \cdot 2^{-c_M - c_B} \\ &\leq \sum_{x \in \mathcal{X}} \sum_{\pi \in S_x^{\hat{m}_{\text{mer}}}} \mu_x(x) \cdot (\delta/4) \cdot 2^{-c_M - c_B} \\ &\leq \sum_{x \in \mathcal{X}} \mu_x(x) \cdot 2^{c_B} \cdot (\delta/4) \cdot 2^{-c_M - c_B} \qquad \text{(by Claim 2)} \\ &= (\delta/4) \cdot 2^{-c_M}. \qquad (5) \end{aligned}$$

We now bound the fraction of rectangles in $\mathcal{R}_{\hat{m}_{\text{mer}}}$ with a "small Alice side" (X_π). To this end, recall that $|\mathcal{R}_{\hat{m}_{\text{mer}}}| \leq 2^{c_A + c_B}$, and therefore

$$\begin{aligned} &\sum_{\substack{\tilde{R}_\pi \in \mathcal{R}_{\hat{m}_{\text{mer}}}: \\ \mu_x(X_\pi) < (\delta/4)\cdot 2^{-c_M - c_A - c_B}}} \mu(\tilde{R}_\pi) \\ \leq &\sum_{\substack{\tilde{R}_\pi \in \mathcal{R}_{\hat{m}_{\text{mer}}}: \\ \mu_x(X_\pi) < (\delta/4)\cdot 2^{-c_M - c_A - c_B}}} \mu_x(X_\pi) < (\delta/4) \cdot 2^{-c_M}. \qquad (6) \end{aligned}$$

But by (3), we know that $\sum_{\tilde{R}_\pi \in \mathcal{R}_{\hat{m}_{\text{mer}}}} \mu(\tilde{R}_\pi) \geq (\delta/2) \cdot 2^{-c_M}$, hence there exists some $\tilde{R}_\pi \in \mathcal{R}_{\hat{m}_{\text{mer}}}$ such that both $\mu_y(\tilde{Y}_\pi) \geq (\delta/4) \cdot 2^{-c_M - c_B}$ and $\mu_x(X_\pi) \geq (\delta/4) \cdot 2^{-c_M - c_A - c_B}$. $\square$

Lemma 3. *Let P be a* 4ANC *protocol that δ-solves $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ under a product distribution $\mu = \mu_x \times \mu_y$. Let $\mathcal{G}_X \subseteq \mathcal{X}$ and $\mathcal{G}_Y \subseteq \mathcal{Y}$ be subsets of inputs such that* $\Pr_\mu[X \in \mathcal{G}_X \ \wedge \ Y \in \mathcal{G}_Y] \geq 1 - \epsilon$*. Then $M(f)$ has a rectangle $R_\pi = X_\pi \times Y_\pi$ such that*

1. *$X_\pi \subseteq \mathcal{G}_X$ and $Y_\pi \subseteq \mathcal{G}_Y$;*
2. *R_π is $((\delta - \epsilon)/2) \cdot 2^{-c_M}$-column-monochromatic;*

3. $\mu_x(X_\pi) \geq (1-\epsilon)((\delta-\epsilon)/4) \cdot 2^{-(c_M+c_A+c_B)}$;

4. $\mu_y(Y_\pi) \geq (1-\epsilon)((\delta-\epsilon)/4) \cdot 2^{-(c_M+c_B)}$.

Proof. The claim follows directly from Lemma 2 by considering the distribution $\mu' := (\mu_x|\mathcal{G}_X) \times (\mu_y|\mathcal{G}_Y)$. Note that μ' is still a *product* distribution, and that P must succeed in solving f under μ' with probability at least $(\delta-\epsilon)$ (or else it will have success $< \delta$ under μ), so we may indeed apply Lemma 2 with μ' and $\delta' := \delta - \epsilon$ to obtain a rectangle $R = X \times Y \subseteq \mathcal{G}_X \times \mathcal{G}_Y$ with $\mu'_x(X) \geq ((\delta-\epsilon)/4) \cdot 2^{-(c_M+c_A+c_B)}$ and $\mu'_y(Y) \geq ((\delta-\epsilon)/4) \cdot 2^{-(c_M+c_B)}$. Finally, since $\mu'_x(X) \leq \mu_x(X)/\mu_x(\mathcal{G}_X) \leq \mu_x(X)/(1-\epsilon)$, it follows that $\mu_x(X) \geq (1-\epsilon)\mu'_x(X) \geq (1-\epsilon)((\delta-\epsilon)/4) \cdot 2^{-(c_M+c_A+c_B)}$. The same argument applied to $\mu'_y(Y)$ completes the proof. □

Remark (General (non-product) distributions)**.** *The only step in the proof of Lemma 2 that uses the independence of x and y (i.e., the product assumption on μ), is the transition in equation* (4). *It is not hard to see that, following a similar calculation to that of Equation* (6), *it is possible to obtain a similar (yet weaker) lower bound on the measure of an induced rectangle $R_\pi = X_\pi \times Y_\pi$ under* arbitrary *(general) distributions μ, namely, that $\mu(R) \gtrsim \delta \cdot 2^{-c_M-c_A-c_B}$. Note that such bound does not distinguish between the measure of "Alice's side" (X_π) and "Bob's side" (Y_π), so it may be less useful to "lopsided" communication problems that typically arise from data structure reductions. Nevertheless, we stress that the lemma above is more general than stated.*

4 The Amortized Dynamic Cell-Probe Complexity of 2D-ORC

In this section, we prove our main theorem, an amortized lower bound for 2-dimensional weighted orthogonal range counting (2D-ORC) problem.

Theorem 1 (restate). *For any integer n, $1 \leq c < o(\log n/\log\log n)$, and any (randomized) data structure D in the cell-probe model with word-size $\Theta(\log n)$, there is a sequence of $n/\log^c n$ updates and $n - n/\log^c n$ queries for the* 2D-ORC *problem, for which the probability (over the randomness of D) that*

- *D probes $o(n\,(\log n/c\log\log n)^2)$ cells in total, and*
- *D is correct on all $n - n/\log^c n$ queries*

is at most $2^{-n/\log^{c+O(1)} n}$.

Remark. *In particular, the theorem implies the following: If D probes $o(n\,(\log n/c\log\log n)^2)$ cells in expectation on any sequence of $O(n/\log^c n)$ updates and $O(n)$ queries, then there is some operation sequence such that the probability D is correct on all queries is at most $2^{-n/\log^{c+O(1)} n}$.*

Plan. To prove the theorem, we first define a hard distribution $\mathcal{D}$ on the operation sequence for 2D-ORC, and fix a data structure D. By Yao's Minimax Principle [Yao77], we can always fix the random bits used by D, so that the probability that D is correct on all queries and makes too few probes is preserved. We may assume D is deterministic from now on. Consider the execution of D on a random sequence of operations. We shall decompose this sequence into many communication games in the 4ANC model, in a way that guarantees that if D is fast and has decent success probability, then most of the games can be solved with low communication cost and non-trivial success probability. On the other hand, we prove that non of these induced games can be solved both efficiently and with non-trivial accuracy. Combining these two facts together, we conclude that no data structure can be fast and have decent success probability simultaneously.

In the following, we first define the hard distribution $\mathcal{D}$, and its corresponding communication game $G_{\text{2D-ORC}}$. In Section 4.1, we propose a protocol for $G_{\text{2D-ORC}}$ given data structure D. In Section 4.2, we prove a lower bound for $G_{\text{2D-ORC}}$. In Section 4.3, we combine the results and prove Theorem 1.

Hard distribution $\mathcal{D}$. The sequence always has $n/\log^c n$ updates and $n - n/\log^c n$ queries such that there are (about) $\log^c n$ queries between two consecutive updates. Every update inserts a point at a uniformly random location in the $[n] \times [n]$ grid with a random weight uniformly chosen from $[n]$. Each query is a uniformly random point in the $[n] \times [n]$ grid. The random sequence is independent across the updates and the queries.

More formally, let $\mathcal{D}_U$ be the uniform distribution over all possible n^3 updates, $\mathcal{D}_Q$ be the uniform distribution over all possible n^2 queries. Let $\mathcal{D}_i$ be the distribution for i-th operation, i.e., $\mathcal{D}_i = \mathcal{D}_U$ if i is multiple of $\log^c n$, and $\mathcal{D}_i = \mathcal{D}_Q$ otherwise. Let $\mathcal{D} = \mathcal{D}_1 \times \mathcal{D}_2 \times \cdots \times \mathcal{D}_n$ be our hard distribution over sequences of n operations. We will focus on $\mathcal{D}$ in the following.

The Distributional Communication Game $G_{\textsf{2D-ORC}}(k, q, n)$. Let $\mathcal{X}$ be the set of k-tuples of *weighted* points in $[n] \times [n]$ with weights from $[n]$, $\mathcal{Y}$ be the set of q-tuples of *unweighted* points in $[n] \times [n]$. Let the input distribution $\mu = \mu_x \times \mu_y$ be the uniform distribution over $\mathcal{X} \times \mathcal{Y}$. Then, $x = ((x_1, w_1), \ldots, (x_k, w_k))$ is a k-tuple of weighted points and $y = (y_1, \ldots, y_q)$ is a q-tuple of unweighted points. Let $\textsf{2D-ORC}(x, y) : ([n]^2 \times [n])^k \times ([n]^2)^q \to [kn]^q$ denote the function whose output is a q-tuple of numbers from $[kn]$, whose i-th coordinate is the sum of w_j's for which $x_j \leq y_i$,[9] i.e.,

$$\textsf{2D-ORC}(x, y)_i := \sum_{j : x_j \leq y_i} w_j.$$

4.1 Efficient data-structure simulation in the 4ANC model

Consider the communication game $G_{\textsf{2D-ORC}}(k, q, n)$ in the 4ANC model. Let I_A and I_B be two consecutive intervals in a random operation sequence sampled from $\mathcal{D}$, such that the number of updates in I_A equals to k and the number of queries in I_B equals to q, i.e., $k \sim |I_A| \cdot \log^{-c} n$ and $q \sim |I_B|$.[10] We shall embed the game into a sequence of operations in the dynamic cell-probe model, so that the answers to all queries in the sequence produces a solution to communication game. In particular, if there is an efficient data structure D for the corresponding operation sequence, Alice and Bob can simulate D using the protocol SIM$_D$ from Section 3.1, which in turn would yield a too-good-to-be-true 4ANC protocol for $G_{\textsf{2D-ORC}}$.

To this end, fix a deterministic data structure D, and all operations before I_A. Both D and these operations are publicly known to the players and therefore can be hard-wired to the protocol. Let $P(I_A)$ and $P(I_B)$ be the set of cells probed by D during I_A and I_B respectively. We now provide a 4ANC protocol that simulates D and solves $G_{\textsf{2D-ORC}}(k, q, n)$.

The simulation protocol P_D for $G_{\textsf{2D-ORC}}(k, q, n)$:

1. (Generate the operations) The number of updates in I_A equals to k. The number of queries in I_B equals to q. Alice sets i-th update in I_A in chronological order to be `update(`x_i`, `w_i`)`. Bob sets j-th query in I_B to be `query(`y_j`)`. They use public randomness to sample queries in I_A and updates in I_B uniformly and independently.

2. (Simulate D on $\mathcal{O}$) Let $\mathcal{O}$ be the sequence of operations obtained by concatenating the hard-wired operations before I_A, and the operations in I_A and in I_B generated in the first step. Run SIM$_D$ from Section 3.1 on $\mathcal{O}$.

3. (Bob recovers the answer to $G_{\textsf{2D-ORC}}(k, q, n)$) For each query in I_B, the answer to the query from the simulation is the sum of weights of points updated and dominated by the query. This includes the

[9] $x_j \leq y_i$ means both coordinates of x_j are no larger than the corresponding coordinates of y_i.

[10] Throughout the paper, $f \sim g$ stands for $f = g + o(g)$ when n goes to infinity.

points updated before I_A, in I_A and in I_B but before the query. Bob knows exactly the updates before I_A and in I_B. By subtracting the sum of weights of points in those time periods and dominated by the query, Bob gets the sum of weights of points updated in I_A and dominated by the query. This sum is precisely the answer to the communication game.

By Lemma 1, we have the following conclusion on protocol P_D.

Lemma 4. *For protocol P_D, we have that*

1. *it is a valid protocol in* 4ANC*;*
2. *Alice sends at most $4|P(I_A) \cap P(I_B)| \cdot w$ bits in expectation;*
3. *Bob sends at most $|P(I_A) \cap P(I_B)| \cdot \log \frac{e|P(I_A)|}{|P(I_A) \cap P(I_B)|}$ bits;*
4. *Merlin sends at most $|P(I_A) \cap P(I_B)| \cdot \log \frac{e|P(I_B)|}{|P(I_A) \cap P(I_B)|} + O(\log n)$ bits;*
5. *when the input pair (x, y) are sampled from μ, operations generated in I_A and I_B will follow distribution $\mathcal{D}$;*
6. *the protocol outputs a correct answer for $G_{\text{2D-ORC}}$ if and only if D is correct on all queries in I_B in $\mathcal{O}$.*

4.2 4ANC communication complexity of orthogonal range counting

In this section we prove Lemma 5, asserting that the probability of any 4ANC protocol with communication $(o(\sqrt{kq}), o(q \log n), o(q \log n))$ in solving all q queries of $G_{\text{2D-ORC}}$ correctly, is hardly any better than the trivial probability obtained by randomly guessing the answers.

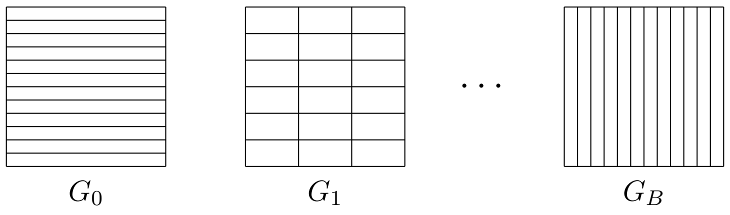
Lemma 5 ("Direct Product" for 2D-ORC in the 4ANC model)**.** *For n large enough, $k \geq \sqrt{n}$ and $k/q \sim \log^{1000} n$,*

$$\mathsf{Suc}_{\mu}^{G_{\text{2D-ORC}}}\left(0.5\sqrt{kq}, 0.005q\log n, 0.0005q\log n\right) \leq 2^{-0.2q\log\log n}.$$

The obvious strategy for proving this lemma is to use the argument in Section 3.2, which asserts that an efficient 4ANC protocol implies a large column-monochromatic rectangle. Therefore, ruling out the existence of a large column-monochromatic rectangle in $M(G_{\text{2D-ORC}})$ would give us a communication lower bound on $G_{\text{2D-ORC}}$.

Unfortunately, $G_{\text{2D-ORC}}(k, q, n)$ does in fact contain large column-monochromatic rectangles. For example, when all of Bob's q points have r-coordinate smaller than $n/\log^{\Theta(1)} n$, Alice does not have to tell Bob any information about her points with r-coordinate greater than that quantity. Thus, in expectation, Alice only needs to speak $k/\log^{\Theta(1)} n$ bits, and this case happens with $2^{-\Theta(q\log\log n)}$ probability over a random Bob's input. In the other word, there is a column-monochromatic rectangle of size $2^{-k/\log^{\Theta(1)} n} \times 2^{-\Theta(q\log\log n)}$. We cannot hope to prove a communication lower bound higher than $(k/\log^{\Theta(1)} n, \Theta(q\log\log n), 0)$ using this approach alone.

To circumvent such inputs from breaking the argument, and for other technical reasons, we only consider Alice's *evenly-spreading* inputs and Bob's *well-separated* inputs, which we will define in the following. Consider the following $B+1$ ways of partitioning $[n]^2$ into blocks of area A: for each $0 \leq i \leq B$, G_i partitions $[n]^2$ into blocks of size $\sim (A\alpha^i/n) \times (n/\alpha^i)$.



We set the parameters/notations in the following way:

- let the coordinate of a point on $[n] \times [n]$ be (r, c);
- let the ratio of number of Alice's points to Bob's points be $\beta = k/q \sim \log^{1000} n$.
- let the area $A \sim (4n^2 \log n)/k$;
- let the ratio $\alpha \sim \beta / \log^3 n$, i.e., $A\alpha \sim 4n^2/q \log^2 n$;
- let $B = \log(n^2/A)/\log\alpha \sim \log k / 997 \log\log n$, so that each block in G_B has size exactly $n \times A/n$.

We ensure that n^2/A is an integer and B is an integer.

Definition 6 (Evenly-spreading tuples). *We say that a k-tuple S of points in $[n] \times [n]$ is* evenly-spreading, *if in every G_i, all but $\leq \sqrt{kq}$ of the blocks have some points in it.*

The following lemma ensures that a uniformly random S is evenly-spreading with extremely high probability.

Lemma 6. *A uniformly random k-tuple S of points is evenly-spreading with probability $\geq 1 - 2^{-q \log^{500} n}$.*

Proof. For each G_i, the probability that it has $\geq \sqrt{kq}$ empty blocks is at most:

$$\begin{aligned} & \binom{n^2/A}{\sqrt{kq}} \cdot \left(1 - \frac{\sqrt{kq}}{n^2/A}\right)^k \\ \leq & \left(\frac{ek}{4\sqrt{kq}\log n}\right)^{\sqrt{kq}} \cdot e^{-4\sqrt{kq}\log n} \\ \leq & \, 2^{\sqrt{kq}(500 \log\log n - 4\log n)} \\ \leq & \, 2^{-q\log^{501} n}. \end{aligned}$$

Hence a union bound implies that the probability S is not evenly-spreading is at most:

$$B 2^{-q \log^{501} n} \leq 2^{-q\log^{500} n}.$$

□

Definition 7 (Far points). *Two points in $[n] \times [n]$ are* far *from each other, if they are not in any axis-parallel rectangle of area $A\alpha$, i.e., the product of differences in two coordinates is at least $A\alpha$.*

Definition 8 (Isolated tuples). *A tuple $\widetilde{Q}$ of points in $[n] \times [n]$ is* isolated, *if every pair of points in $\widetilde{Q}$ are far from each other.*

Definition 9 (Well-separated tuples). *A q-tuple Q of points on $[n] \times [n]$ is* well-separated, *if it contains an isolated subtuple $\widetilde{Q}$ with $|\widetilde{Q}| \geq q/2$.*

The following lemma ensures that a uniformly random q-tuple points is well-separated with extremely high probability.

Lemma 7. *A uniformly random q-tuple Q of points is well-separated with probability* $\geq 1 - 2^{-0.4q \log\log n}$.

Proof. The area of region that is not far from a given point p_0 can be bounded as follows:

$$\begin{aligned} 4\sum_{i=0}^{n} \min\{n, A\alpha/i\} &\leq 4(n + A\alpha \sum_{i=1}^{n} 1/i) \\ &\leq 4(n + 4n^2(1+\log n)/q \log^2 n) \\ &\leq 17n^2/q \log n. \end{aligned}$$

Pick all points that do not land in the above region of any point *before* it in the tuple. For each point, the probability that it is picked is at least $1 - 17\log^{-1} n$. The probability that no more than $q/2$ points are picked is at most:

$$2^q \left(17\log^{-1} n\right)^{q/2} \left(1 - 17\log^{-1} n\right)^q \leq 2^{-0.4q \log\log n}.$$

□

Lemma 8. *Let $\widetilde{Q}$ be a tuple of* isolated *points. Then*

- *the r-coordinates of all points,*
- *the* set *of blocks containing each point in each G_i (for $0 \leq i \leq B$), and*
- *the c-offset of each point* within *the block in G_B*

together uniquely determine the locations of all points in $\widetilde{Q}$.

Proof. Given the r-coordinate of a point p, there are α blocks of G_1 that could contain it without providing any extra information. Since $\widetilde{Q}$ is isolated, and these α blocks together form a rectangle of area $A\alpha$, there can be no other points from $\widetilde{Q}$ in any of them. Therefore, given the set of blocks in G_1 containing $\widetilde{Q}$, exactly one of the α blocks will belong to this set, i.e., the block containing p. This reduces the range of c-coordinate that p could be in by a factor of α. Given this information, there are α blocks of G_2 that could contain p. For the same exact reason, knowing the set of blocks in G_2 containing $\widetilde{Q}$ reduces the range by a factor of α further.

Given all sets, we get to know for each point, the r-coordinate and the block in G_B containing it. Therefore, knowing the offsets within G_B uniquely determines the locations of all points. □

We are now ready to prove our main lemma for the $G_{\textsf{2D-ORC}}$ communication game, asserting that $M(G_{\textsf{2D-ORC}})$ does not contain any large rectangle with even a slightly better bias than the trivial one (n^{-q}):

Lemma 9. *There is no rectangle $R = X \times Y$ in $M(G_{\textsf{2D-ORC}})$ satisfying all of the following conditions:*

1. *every $x \in X$ is evenly-spreading, every $y \in Y$ is well-separated;*
2. *R is $n^{-0.001q}$-column-monochromatic;*
3. *$\mu_x(X) \geq 2^{-\sqrt{kq}}$;*
4. *$\mu_y(Y) \geq 2^{-0.01q \log n}$.*

Let us first think about the case where R is 1-column-monochromatic and μ_x is the uniform distribution over k points with fixed locations (only the weights vary). In this case, all entries in every column $y \in Y$ have exactly the same function value in R. Given y and its function value, it can be seen as imposing q linear constraints to the k weights, i.e., each query with the answer tells us that the sum of some points should equal to some number. Intuitively, if $|Y|$ is too large, it will be inevitable that the union of all queries appeared in Y "hit everywhere" in $[n] \times [n]$. Even if different q-tuples $y \in Y$ may have overlaps in queries, it is still impossible to pack too many y's in a small area. However, if the union of all queries in Y hit too many places in $[n] \times [n]$, they will impose many independent linear constraints on the k weights, and thus $|X|$ must be small.

In general, we are going to focus on how many different regions of $[n] \times [n]$ Y "hits". If it hits very few, we show that $|Y|$ must be small, by encoding each $y \in Y$ using very few bits. This encoding scheme is a variation of the encoding argument by Larsen [Lar12]. On the other hand, if Y does hit many different regions, we show $|X|$ must be small. To show it, we will describe a new encoding scheme which encodes $x \in X$ using very few bits. The main idea is to use public randomness (which can be seen by both the encoder and the decoder, and is independent of x) to sample a few regions in $[n] \times [n]$, then with decent probability, Y "hits" all these regions, and the function value is correct for some y hitting some of the regions, due to the slight bias of the rectangle. They view the public randomness as infinite such samples, and the encoder just writes down the ID of the first sample that has the above property. Then the decoder will be able to "learn" q linear equations on x from it. The point here is that if the probability that a random sample has the above property is not too low, the ID of the first success sample will have less than $q \log n$ bits. Thus we will be able to use less bits compared to the naive encoding. At last, we show that if we do this multiple times, each time the q linear equations the decoder learns will be independent from the previous ones with high probability. This ensures us that it is possible to apply the sampling many times to save even more bits, which would allow us to prove the lemma.

Proof. Consider a well-separated y, and its lexicographically first[11] isolated $q/2$-subtuple $\widetilde{Q}(y)$ (we might use $\widetilde{Q}$ instead of $\widetilde{Q}(y)$ in the following, when there is no ambiguity). In every G_i, the $q/2$ points in $\widetilde{Q}$ appear in $q/2$ different blocks due to its isolation. Let $\mathcal{H}_i(y)$ denote the set of $q/2$ blocks in G_i containing a point from $\widetilde{Q}$. Call $\mathcal{H}_i(y)$ the *hitting pattern* of y on G_i. Let

$$\mathcal{H}(y) := \bigcup_{i=0}^{B} \mathcal{H}_i(y)$$

be the *hitting pattern* of y. We shall see that this combinatorial object ($\mathcal{H}(y)$) captures the delicate structure of a set of q queries.

Let us fix a rectangle $R = X \times Y$ that is $n^{-0.001q}$-column-monochromatic. We are going to show that either $|X|$ is small or $|Y|$ is small, as the lemma predicts. To this end, consider the set

$$T = \{\mathcal{H}(y) : y \in Y\}$$

[11] "Lexicographically first" is only used for the unambiguity of the definition.

of hitting patterns for all $y \in Y$. A trivial upper bound on $|T|$ is

$$\begin{aligned}\binom{n^2/A}{q/2}^{B+1} &= 2^{(B+1)\log\binom{n^2/A}{q/2}} \\ &= 2^{(B+1)q/2\cdot\log\Theta(n^2/Aq)} \\ &= 2^{(1+o(1))\log(n^2/A)/\log\alpha\cdot q/2\cdot\log\beta} \\ &= 2^{(1+o(1))\frac{1000}{997}|\widetilde{Q}|\log(n^2/A)} \\ &= 2^{(1+o(1))\frac{500}{997}q\log k},\end{aligned}$$

simply because there can only be that many different hitting patterns in total. However, we are going to show that if

$$|T| \le \binom{n^2/A}{q/2}^{0.95B},$$

polynomially fewer than the potential number of patterns, then $|Y|$ must be small. Otherwise, if $|T|$ is large, then $|X|$ must be small.

Case 1 : $|T|$ smaller than the threshold. This case intuitively says that if Y can only generate a small number of hitting-patterns, then Y itself cannot be too large (since $\mathcal{H}_i(y)$ can be used to determine y by Lemma 8). To formalize this intuition, we will show how to encode each $y \in Y$ using no more than $1.99q\log n$ bits, which implies $|Y| \le 2^{1.99q\log n}$ and $\mu_y(Y) \le 2^{-0.01q\log n}$. Given a $y \in Y$, which is a q-tuple of points, we apply the following encoding scheme:

1. Write down the r-coordinates of all points.
2. Write down one bit for each point in y, indicating whether it belongs to the set $\widetilde{Q}$ or not. For each point not in $\widetilde{Q}$, also write down its c-coordinate.
3. Write down the hitting pattern $\mathcal{H}(y)$.
4. For each point in $\widetilde{Q}$, write down its c-offset within the block in G_B.

Decoding: The scheme above writes down the coordinates of all points not in $\widetilde{Q}$, so they can clearly be decoded correctly. For points in $\widetilde{Q}$, it writes down the r-coordinates, the set of blocks containing them in each G_i, and the c-offsets within the block in G_B. By Lemma 8, it uniquely determines the locations of all well-separated points, and hence determines y.

Analysis: Now let us estimate the number of bits it uses. Let us analyze the number of bits used in each step:

1. It takes $q\log n$ bits to write down all the r-coordinates.
2. It takes 1 bit for each point, and extra $\log n$ bits for each point not in $\widetilde{Q}$. This step takes $q + (q - |\widetilde{Q}|)\log n$ bits in total.
3. Since we know $y \in Y$, and $\mathcal{H}(y) \in T$, this step takes $\log|T|$ bits.
4. This step takes $|\widetilde{Q}|\log(A/n)$ bits.

When $|T|$ is smaller than the threshold, step 3 and 4 take

$$\begin{aligned}\log|T| + |\widetilde{Q}|\log(A/n) &= 0.95(1+o(1))\frac{1000}{997}|\widetilde{Q}|\log(n^2/A) + |\widetilde{Q}|\log(A/n)\\ &\le 0.955|\widetilde{Q}|\log(n^2/A) + |\widetilde{Q}|\log(A/n)\\ &= |\widetilde{Q}|\log n - 0.045|\widetilde{Q}|\log(n^2/A)\end{aligned}$$

bits. Thus, the encoding takes

$$\begin{aligned}&q\log n + (q + (q - |\widetilde{Q}|)\log n) + (|\widetilde{Q}|\log n - 0.045|\widetilde{Q}|\log(n^2/A))\\ \le\ &2q\log n + q - 0.045|\widetilde{Q}|\log(n^2/A)\\ \le\ &2q\log n - 0.01q\log n = 1.99q\log n\end{aligned}$$

bits in total. We have $\mu_y(Y) = |Y|\cdot 2^{-2q\log n} \le 2^{-0.01q\log n}$ as desired.

Case 2: $|T|$ greater than the threshold. The main idea for this case is to efficiently encode an x assuming there is a shared random tape between the encoder and the decoder. They use the randomness to sample random hitting patterns, and hope there is one that happens to be the hitting pattern of some $y \in Y$, and (x, y) is correct in R. Specifying one such hitting pattern is a way to identify a correct entry in row x of the rectangle, which reveals some information about x to the decoder. The nature of hitting patterns guarantees that independent samples with the above property reveal "different" information about x with high probability. This allows us to encode x using very few bits by repeating the above procedure multiple times, and obtain an upper bound on $|X|$. To this end, first consider the following probabilistic argument:

1. sample a hitting pattern over all $\binom{n^2/A}{q/2}^{B+1}$ possibilities uniformly at random, i.e., for each $0 \le i \le B$, sample a set S_i of $q/2$ blocks in G_i uniformly and independently,
2. independent of step 1, sample a uniformly random $x \in X$.

Then the probability p that

1. $\exists y \in Y$, such that $S_i = \mathcal{H}_i(y)$ for all i, and
2. the lexicographically first such y has $\mathsf{2D\text{-}ORC}(x, y)$-value matching the color of column y of R,

is at least

$$\begin{aligned}p &\ge \frac{|T|}{\binom{n^2/A}{q/2}^{B+1}}\cdot n^{-0.001q}\\ &\ge \binom{n^2/A}{q/2}^{-0.05B-1}\cdot 2^{-0.001q\log n}\\ &= 2^{-0.05(1+o(1))\frac{500}{997}q\log k - 0.001q\log n}\\ &\ge 2^{-0.026q\log k - 0.001q\log n}\\ &\ge 2^{-0.027q\log n},\end{aligned}$$

where the first transition is by the assumption that R is $n^{-0.001q}$-column-monochromatic, and the second transition is by the assumption on $|T|$. By Markov's inequality, there are at least $p/2$-fraction of the x's in X that with probability at least $p/2$ (over the randomn choice of hitting pattern $\{S_i\}$), both conditions hold.

Let this set of x's be $\overline{X}$. We are going exhibit a randomized encoding scheme for each $x \in \overline{X}$ that uses very few bits in expectation (over the public randomness of the scheme). This will imply an upper bound on $|\overline{X}|$, which in turn will imply an upper bound on $|X|$. Given an $x \in \overline{X}$, which is a k-tuple of weighted points, we apply the following encoding scheme:

1. Write down the locations of each point in x.
2. View the infinitely long public random string as infinite samples of $\{S_i\}_{0 \le i \le B}$. Write down the index of the first sample such that $\{S_i\} \in T$, and the lexicographically first y with $\mathcal{H}(y) = \{S_i\}$ satisfies that 2D-ORC(x, y) matches the color of column y of R. This implicitly encodes the answers to q queries in y on x for some y.
3. Repeat Step 2 for $\sqrt{k/q}$ times (using fresh randomness each time).
4. Find *all* the possible $x \in \overline{X}$ that are consistent with all answers encoded in Step 2, sort them in lexicographical order. Write down the index of the x that we are trying to encode in this sorted list.

Deocoding: Assuming the above encoding scheme terminates and outputs an encoding, it is easy to see that we can recover each $x \in \overline{X}$. It is because in Step 4, we find all x's that are still possible, and explicitly specify which one it is.

Analysis: To show the encoding scheme uses very few bits in expectation, it will be useful to view each x as a vector $v_x \in \mathbb{R}^k$, where i-th coordinate encodes the weight of i-th point in x. Since the locations of all points are written down explicitly in Step 1, each query point y_j in y can also be viewed as a 0-1 vector $u_{y_j} \in \mathbb{R}^k$, where i-th coordinate indicates whether the i-th point is *dominated* by this point.[12] Call v_x the *weight vector* of x, u_{y_j} the *dominance vector* of y_i. In this notation, the sum of weights of dominated points in x is just the inner product $\langle v_x, u_{y_j} \rangle$ of these two vectors. In this sense, every execution of Step 2 implicitly encodes q linear constraints on x. Let L denote the number of *linearly independent* constraints encoded in total (note that L is a random variable). We have the following cost for each step:

1. It takes $2k \log n$ bits to write down all the locations.
2. Each time we run this step, it takes $\log 2/p \le 0.027 q \log n$ bits in expectation.
3. In total, all executions of Step 2 take $0.027\sqrt{kq} \log n$ bits in expectation.
4. Since there are $\le n^{k-L}$ different x's satisfying all linear constraints, this step takes $(k - L) \log n$ bits.

In order to conclude that the scheme uses a small number of bits, it remains show that L will be large in expectation. This is the content of the following technical claim.

Claim 3. *$\mathbb{E}[L] \ge 0.06\sqrt{kq}$.*

Assuming Claim 3, the expected total cost of the encoding scheme is at most

$$2k \log n + 0.027\sqrt{kq} \log n + (k - \mathbb{E}[L]) \log n \le 3k \log n - 0.033\sqrt{kq} \log n.$$

Thus, we have $|\overline{X}| \le 2^{3k \log n - 0.033\sqrt{kq} \log n}$, $\mu_x(\overline{X}) = |\overline{X}| \cdot 2^{-3k \log n} \le 2^{-0.033\sqrt{kq} \log n}$. Therefore, we have

$$\mu_x(X) \le 2^{-0.033\sqrt{kq} \log n + 0.01 q \log n} \ll 2^{\sqrt{kq}},$$

which proves the lemma.

[12]The vector u_{y_j} depends on not only on y_j but also on the locations of all points in x.

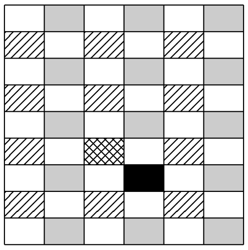


Figure 2: G_i

The only part left is to prove Claim 3. We show that each time we run Step 2, it creates many new constraints linearly independent from the previous ones with high probability. To this end, let us fix a G_i and consider a block in it, e.g., the ■ block in Figure 2. It is not hard to see that different query points in the block may have different dominance vectors (the 0-1 vector in $\mathbb{R}^k$). However, if we only focus on the coordinates corresponding to points from x in the ▨ blocks,[13] all query points in ■ block have the same values in these coordinates. In general, G_i can be viewed as a meta-grid with $n^2/A\alpha^i$ rows and α^i columns. For each block in G_i, all dominance vectors of points in the block have the same values, in the coordinates corresponding to points in blocks with opposite row and column parities in the meta-grid (the ▨ coordinates). Define the dominance vector of a block to be the dominance vector of some query point in it, restricted to all those coordinates. By the above argument, the dominance vector of a block is well-defined. Note that, different blocks in G_i may have dominance vectors defined to take values in different codomains. For example, the dominance vector of a ░ or ■ block only has ▨ coordinates, and vice versa. But all blocks with the same row and column parities in the meta-grid have the same codomain for their dominance vectors.

Now let us focus on all ($\sim n^2/4A$) dominance vectors of ░ (and ■) blocks. Except a few of them, all others are linearly independent. More specifically, we first remove all dominance vectors of blocks ($\sim \alpha^i/2$) in the first row of the meta-grid. Then by evenly-spreading of the input x, there are at most $\sqrt{kq}$ blocks in G_i having no points from x. For every such ▨ block, we remove the dominance vector of its *immediate lower-right* block, e.g., if the ▩ block has no points from x, we remove the dominance vector of the ■ block. All the remaining vectors are linearly independent. This is because if we sort all these vectors in the upper-lower left-right order of their blocks, every dominance vector has a non-zero value in some coordinate corresponding to points in its immediate upper-left block, while all previous vectors have zeros in these coordinates, i.e., every vector is independent from the vectors before it in the sorted list. In general, for all four possibilities of row and column parities in the meta-grid, among the $\sim n^2/4A$ dominance vectors, we can remove at most $\sim \sqrt{kq}+\alpha^i/2+n^2/2A\alpha^i$ of them, so that the remaining vectors are all linearly independent.

Using the above connection between the geometry of the points and linear independence, we will be able to show that each execution of step 2 creates many new linearly independent constraints with high probability. Intuitively, among the $q/2$ blocks in S_i, at least $q/8$ of them have the row and column parities, say they are all ░ blocks. Then we restrict all $< \sqrt{kq}$ dominance vectors from previous executions of Step 2 to the ▨ coordinates. There are many independent dominance vectors among those of the ░ blocks. Thus, except with exponentially small probability, at least $q/16$ of dominance vectors of the sampled blocks will be independent from the previous ones. If there is a y that $\mathcal{H}(y) = \{S_i\}$, then y must have one query point in each of the $q/8$ sampled ░ blocks. The dominance vector of a query point takes same values as that of the ░ block it is in in the ▨ coordinates. At least $q/16$ of the query points in y will create new

[13]We may use "▨ coordinates" to indicate these coordinates in the following.

independent linear constraints, as their dominance vectors are independent from the previous ones, even restricted to ▨ coordinates.

More formally, in Step 2 of the encoding scheme, imagine that instead of sampling S_i directly, we first randomly generate the numbers of blocks in S_i with different row and column parities in the meta-grid, then sample S_i conditioned on these four numbers. There must be one row and column parity that has at least $q/8$ blocks in S_i. Without loss of generality, we assume that at least $q/8$ blocks sampled will be in odd rows and odd columns. From now on, let us focus on sampling these $\geq q/8$ blocks.

Imagine that we sample the $\geq q/8$ blocks in S_i one by one independently. Before sampling each block, consider all dominance vectors of query points created by previous executions of Step 2, restricted to all coordinates corresponding to points in blocks with opposite parities (blocks in even rows and even columns), together with all dominance vectors of blocks just sampled. There are no more than $\sqrt{kq}$ such dominance vectors in total, i.e., among all $\geq n^2/4A - \sqrt{kq} - \alpha^i/2 - n^2/2A\alpha^i$ independent dominance vectors, at least $\geq n^2/4A - 2\sqrt{kq} - \alpha^i/2 - n^2/2A\alpha^i$ of them are linearly independent from the previous ones. Thus, the dominance vector of the new sampled block is independent from the previous ones with probability at least $1 - (2\sqrt{kq} + \alpha^i/2 + n^2/2A\alpha^i)/(n^2/4A)$. Among the $q/8$ blocks in S_i, if there are at least $q/16$ of them whose dominance vectors are independent from the previous ones, and $\mathcal{H}(y) = \{S_i\}$ for some y, then y creates at least $q/16$ new independent linear constraints. Since there is one query point in each block in S_i, including the $q/16$ of them with dominance vector independent from the previous ones, these $q/16$ points create one independent constraint each.

Thus, if $\mathcal{H}(y) = \{S_i\}$ and y imposes at most $q/16$ new independent constraint, then it must be the case that in every S_i, the dominance vectors of no more than $q/16$ of the blocks are independent from the previous ones. Since all S_i's are sampled independently and when $0.001B \leq i \leq 0.999B$, $2\sqrt{kq}+\alpha^i/2+n^2/2A\alpha^i \leq 2\sqrt{kq} + k^{1-\Omega(1)} \leq 3\sqrt{kq}.$, the probability of the latter is at most

$$\begin{aligned}
\left(\left(\frac{3\sqrt{kq}}{n^2/4A}\right)^{q/16} 2^{q/8}\right)^{0.998B} &\leq \left(\left(\frac{200\sqrt{kq}}{k/\log n}\right)^{q/16}\right)^{0.998B} \\
&\leq \left((200\beta^{-1/2}\log n)^{q/16}\right)^{0.998B} \\
&\leq 2^{(0.998qB/16)(8-0.5\log\beta+\log\log n)} \\
&\leq 2^{-\frac{(1+o(1))0.998q\log k\cdot 499\log\log n}{16\cdot 997\log\log n}} \\
&\leq 2^{-0.031q\log k}.
\end{aligned}$$

The probability that a sample $\{S_i\}$ succeeds is at least $p/2 \geq 2^{-0.026q\log k - 0.001q\log n} \gg 2^{-0.031q\log k}$. Thus, each time we run step 2, with $1-o(1)$ probability, y gives us at least $q/16$ new constraints. Therefore, the expected value of L is at least $\sqrt{k/q}\cdot(1-o(1))q/16 \geq 0.06\sqrt{kq}$, as claimed. $\square$

Proof of Lemma 5. Assume there is a 4ANC protocol $2^{-0.2q\log\log n}$-solves $G_{\text{2D-ORC}}(k,q,n)$ with communication cost $(0.5\sqrt{kq}, 0.005q\log n, 0.0005q\log n)$. By Lemma 6 and Lemma 7, the probability that in a random input pair (x,y) sampled from μ, x is not evenly-spreading or y is not well-separated is at most $1-2^{-0.4q\log\log n}-2^{-q\log^{500}n}$. By Lemma 3, for large enough n, $M(G_{\text{2D-ORC}})$ has a rectangle $R = X\times Y$ such that

1. every $x\in X$ is evenly-spreading and every $y\in Y$ is well-separated;

2. R is $2^{-0.0006q\log n}$-column-monochromatic;

3. $\mu_x(X) \geq 2^{-0.6\sqrt{kq}}$;

4. $\mu_y(Y) \geq 2^{-0.006q\log n}$.

However, by Lemma 9, such rectangle cannot exist in $M(G_{\textsf{2D-ORC}})$. We have a contradiction. $\square$

4.3 Proof of Theorem 1

Combining the communication lower bound for $G_{\textsf{2D-ORC}}(k,q,n)$ (Lemma 5) with Lemma 4 (the simulation argument in Section 4.1) gives us a lower bound on the efficiency and accuracy of D in I_A and I_B.

Lemma 10. *Let $\mathcal{O}$ be a random sequence of operations sampled from $\mathcal{D}$, let I_A and I_B be two consecutive intervals in $\mathcal{O}$, such that $|I_B| \geq \sqrt{n}$ and $|I_A| \sim |I_B|\log^{c+1000} n$, and denote by $\mathcal{O}_{<I_A}$ the sequence of operations preceding I_A. Then* conditioned on *$\mathcal{O}_{<I_A}$, the probability that all of the following events occur simultaneously*

1. *$|P(I_A)| \leq |I_A|\log^2 n$,*
2. *$|P(I_B)| \leq |I_B|\log^2 n$,*
3. *$|P(I_A) \cap P(I_B)| \leq |I_B|\frac{\log n}{200(c+1002)\log\log n}$, and*
4. *D answers all queries in I_B correctly*

is at most $2 \cdot 2^{-0.2|I_B|\log\log n}$.

Proof. Let p denote the probability that all four events above occur. Consider the protocol P_D on intervals I_A and I_B, solving $G_{\textsf{2D-ORC}}(k,q,n)$ for $k \sim |I_A| \cdot \log^{-c} n$ and $q \sim |I_B|$. Note that when the input pairs for $G_{\textsf{2D-ORC}}(k,q,n)$ are sampled uniformly at random (i.e., according to μ), the corresponding operations I_A and I_B in the simulation P_D are distributed according to $\mathcal{D}$. By Lemma 4, the first three conditions in the statement imply that the protocol P_D satisfies the following conditions:

1. Alice sends at most $\frac{4q\log^2 n}{200(c+1002)\log\log n} \ll 0.1\sqrt{kq}$ bits in expectation,
2. Bob sends
$$|P(I_A) \cap P(I_B)| \cdot \log\frac{e|P(I_A)|}{|P(I_A) \cap P(I_B)|} \leq 0.005q\log n$$
bits (since the function $f(u,v) = u \cdot \log\frac{v}{u}$ is increasing in both u and v when $v > eu$),
3. Merlin sends
$$|P(I_A) \cap P(I_B)| \cdot \log\frac{e|P(I_B)|}{|P(I_A) \cap P(I_B)|} + O(\log n) < 0.0005q\log n$$
bits.

Conditioned on the the three events above, with probability at least $1/2$, Alice sends no more than $0.2\sqrt{kq}$ bits. Thus, by Lemma 5, $p/2 \leq 2^{-0.2q\log\log n}$. This proves the lemma. $\square$

Using the above lemma, we are finally ready to prove our data structure lower bound for 2D-ORC (Theorem 1). Intuitively, if the data structure D correctly answers *all* queries under $\mathcal{D}$, then Lemma 10 is essentially saying either $|P(I_A)|$ or $|P(I_B)|$ is large, or during I_B, D reads at least $\Omega(|I_B| \cdot \log n / c\log\log n)$ cells that are also probed in I_A. If the former happens too often, it is not hard to see that D makes too many probes in total. Otherwise, "on average" for every operation in I_B, D must read at least $\Omega(\log n / c\log\log n)$ cells whose last probe was in I_A. This argument holds as long as $|I_A| \sim |I_B| \cdot \log^{c+O(1)} n$ and $|I_B| \geq \sqrt{n}$.

Thus, during an operation, "on average" D has to read $\Omega(\log n/c \log\log n)$ cells whose last probe was anywhere between $\sqrt{n}$ and $\sqrt{n} \cdot \log^{c+O(1)} n$ operations ago, $\Omega(\log n/c \log\log n)$ cells whose last probe was between $\sqrt{n} \cdot \log^{c+O(1)} n$ and $\sqrt{n} \cdot \log^{2(c+O(1))} n$ operations ago, and so on. All these sets of cells are disjoint, and there are $\Omega(\log n/c \log\log n)$ such sets. This gives us that "on average", each operation has to probe $\Omega((\log n/c \log\log n)^2)$ cells in total. While Lemma 10 does not account for the number of cells probed during any particular operation (but only the total number of probes), summing up the lower bounds for relevant interval pairs, the above argument gives an *amortized* lower bound assuming D correctly answers all queries.

When D is allowed to err, a natural approach is to partition the sequence into disjoint intervals $\{I\}$, and interpret the overall success probability as the product of conditional success probabilities for queries in I conditioned on the event that D succeeds on all queries preceding I. When the overall success probability is "non-trivial", there will be a constant fraction of I's with "non-trivial" success probability conditioned on succeeding on all previous intervals. As Lemma 10 also holds for D that is correct with exponentially in $|I_B| \log\log n$ small probability, the argument outlined in the last paragraph still goes through. A more careful argument proves the theorem. We now turn to formalize the intuition above, by showing how to combine the arguments in the last two paragraphs.

Proof of Theorem 1. We shall decompose the operation sequence into many (possibly overlapping) consecutive intervals, and then apply Lemma 10 to obtain a lower bound on the operational time (in terms of probes) the data structure spends on each interval. Summing the lower bounds together will yield the desired lower bound on the total number of probes. To this end, for $\gamma = \log^{c+1000} n$, consider the following decomposition:

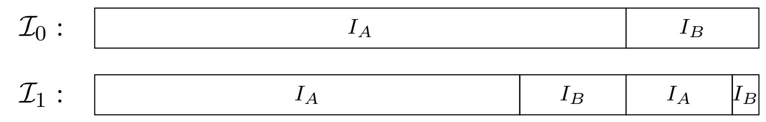
- For any (consecutive) interval of operations I, define $\mathrm{DEC}(I) := (I_A, I_B)$, where I_A is the first $|I| - \lceil |I|/\gamma \rceil$ operations in I and I_B is the last $\lceil |I|/\gamma \rceil$ operations in I. We have $I_A \cap I_B = \emptyset$, $I_A \cup I_B = I$, and $|I_A| \sim |I_B| \cdot \gamma$ for any large enough $|I|$.

- Let $\mathcal{I}_0 = \{\mathrm{DEC}(\mathcal{O})\}$ be the singleton set (I_A, I_B), where I_A is the first $n - \lceil n/\gamma \rceil$ operations and I_B is the last $\lceil n/\gamma \rceil$ operations in a random sequence $\mathcal{O}$ sampled from $\mathcal{D}$.

- For $i > 0$, let us recursively define

$$\mathcal{I}_i = \{\mathrm{DEC}(I) = (I_A, I_B) : |I_B| \geq \sqrt{n} \quad \text{s.t } \exists I', (I, I') \in \mathcal{I}_{i-1} \vee (I', I) \in \mathcal{I}_{i-1}\}$$

  to be the decomposition into (disjoint) intervals obtained by "refining" the decomposition $\mathcal{I}_{i-1}$ (as in the following illustration).



$$\vdots$$

The following claim, whose proof is deferred to Appendix B, states that for small enough i's, the total number of operations in "Bob's intervals" (I_B) when summing up over all interval pairs in the set $\mathcal{I}_i$, is large:

Claim 4. *For $i \leq 0.1\gamma \log n/\log\gamma$, we have*

$$\sum_{(I_A, I_B) \in \mathcal{I}_i} |I_B| \geq \frac{n}{2\gamma}.$$

Now, let D be any dynamic data structure for 2D-ORC, and define $\mathcal{E}(I_A, I_B)$ to be the event that all four conditions of Lemma 10 occur with respect to a specific interval pair I_A, I_B and the data structure D, namely:

1. $|P(I_A)| \le |I_A| \log^2 n$,
2. $|P(I_B)| \le |I_B| \log^2 n$,
3. $|P(I_A) \cap P(I_B)| \le |I_B| \frac{\log n}{200(c+1002)\log\log n}$, and
4. D answers all queries in I_B correctly.

Consider the event $\mathcal{E}_i$ that all queries are answered correctly and "most" of $(I_A, I_B) \in \mathcal{I}_i$ are efficient:

1. $\sum_{(I_A,I_B)\in\mathcal{I}_i} |P(I_A)| \le \frac{n\log^2 n}{8}$,
2. $\sum_{(I_A,I_B)\in\mathcal{I}_i} |P(I_B)| \le \frac{n\log^2 n}{8\gamma}$,
3. $\sum_{(I_A,I_B)\in\mathcal{I}_i} |P(I_A) \cap P(I_B)| \le \frac{n\log n}{1600(c+1002)\gamma\log\log n}$, and
4. D answers all queries in the sequence correctly.

By Markov's inequality, $\mathcal{E}_i$ implies that

$$\sum_{\substack{(I_A,I_B)\in\mathcal{I}_i:\\ |P(I_A)|>|I_A|\log^2 n}} |I_B| \le \frac{1}{\gamma\log^2 n} \sum_{\substack{(I_A,I_B)\in\mathcal{I}_i:\\ |P(I_A)|>|I_A|\log^2 n}} |P(I_A)| \le \frac{1}{\gamma\log^2 n}\cdot\frac{n\log^2 n}{8} = \frac{n}{8\gamma}.$$

Similarly, we have

$$\sum_{\substack{(I_A,I_B)\in\mathcal{I}_i:\\ |P(I_B)|>|I_B|\log^2 n}} |I_B| \le \frac{n}{8\gamma} \qquad \text{and} \qquad \sum_{\substack{(I_A,I_B)\in\mathcal{I}_i:\\ |P(I_A)\cap P(I_B)|>|I_B|\frac{\log n}{200(c+1002)\log\log n}}} |I_B| \le \frac{n}{8\gamma}.$$

Therefore, by Claim 4 and a union bound, the event $\mathcal{E}_i$ implies the event "$\sum_{(I_A,I_B)\in\mathcal{I}_i:\mathcal{E}(I_A,I_B)} |I_B| \ge \frac{n}{8\gamma}$" whenever $i \le 0.1\gamma\log n/\log\gamma$.

We now want to use this fact together with Lemma 10 to conclude that $\mathcal{E}_i$ cannot occur too often. Indeed, Lemma 10 asserts that the event $\mathcal{E}(I_A, I_B)$ occurs with extremely low probability, even conditioned on all operations before I_A. Since all the intervals in $\mathcal{I}_i$ are disjoint by construction, this gives us an upper bound on the probability of $\mathcal{E}_i$:

$$\begin{aligned}
\Pr[\mathcal{E}_i] &\le \Pr\left[\sum_{(I_A,I_B)\in\mathcal{I}_i:\mathcal{E}(I_A,I_B)} |I_B| \ge \frac{n}{8\gamma}\right] \\
&= \Pr\left[\exists S\subseteq \mathcal{I}_i, \sum_{(I_A,I_B)\in S} |I_B| \ge \frac{n}{8\gamma}, \forall (I_A,I_B)\in S, \mathcal{E}(I_A,I_B)\right] \\
&\le \sum_{\substack{S\subseteq\mathcal{I}_i,\\ \sum_{(I_A,I_B)\in S}|I_B|\ge\frac{n}{8\gamma}}} \Pr\left[\bigwedge_{(I_A,I_B)\in S}\mathcal{E}(I_A,I_B)\right] && \text{(by union bound)} \\
&= \sum_{\substack{S\subseteq\mathcal{I}_i,\\ \sum_{(I_A,I_B)\in S}|I_B|\ge\frac{n}{8\gamma}}} \prod_{(I_A,I_B)\in S} \Pr\left[\mathcal{E}(I_A,I_B) \,\middle|\, \bigwedge_{\substack{(I'_A,I'_B)\in S\\ (I'_A,I'_B)\text{ before }(I_A,I_B)}} \mathcal{E}(I'_A,I'_B)\right] \\
&\le \sum_{\substack{S\subseteq\mathcal{I}_i,\\ \sum_{(I_A,I_B)\in S}|I_B|\ge\frac{n}{8\gamma}}} \prod_{(I_A,I_B)\in S} \left(2\cdot 2^{-0.2|I_B|\log\log n}\right) && \text{(by Lemma 10)} \\
&\le \sum_{\substack{S\subseteq\mathcal{I}_i,\\ \sum_{(I_A,I_B)\in S}|I_B|\ge\frac{n}{8\gamma}}} 2^{|S|}2^{-0.025\frac{n}{\gamma}\log\log n} \\
&\le 4^{|\mathcal{I}_i|}2^{-0.025\frac{n}{\gamma}\log\log n}.
\end{aligned}$$

Since all I_B's in $\mathcal{I}_i$ are disjoint and have length at least $\sqrt{n}$, $|\mathcal{I}_i| \le \sqrt{n}$. Thus, for n large enough, $\Pr[\mathcal{E}_i] \le 2^{-0.02\frac{n}{\gamma}\log\log n}$.

Finally, recall that $\mathcal{E}_i$ is the event that all queries are answered correctly and most interval pairs in $\mathcal{I}_i$ are efficient. To finish the proof, we claim that the efficiency of interval pairs in all $\mathcal{I}_i$ characterizes the overall efficiency:

Claim 5. *If D probes $o(n(\log n/c\log\log n)^2)$ cells and is correct on all queries, then for some $i \le 0.1\gamma\log n/\log\gamma$, $\mathcal{E}_i$ occurs.*

Indeed, consider the contrapositive statement that non of $\mathcal{E}_i$ occurs. Then at least one of the following events must occur :

1. Some query is not answered correctly.

2. For some i, $\sum_{(I_A,I_B)\in\mathcal{I}_i} |P(I_A)| > \frac{n\log^2 n}{8}$. Since all I_A's are disjoint, this already implies that D probes at too many cells.

3. There are least $0.05\gamma\log n/\log\gamma$ different i's for which $\sum_{(I_A,I_B)\in\mathcal{I}_i} |P(I_B)| > \frac{n\log^2 n}{8\gamma}$. Since each operation can only appear in $\le \log n/\log\gamma$ different I_B's, the total number of probes is at least

$$\frac{\log\gamma}{\log n}\cdot(0.05\gamma\log n/\log\gamma)\cdot\frac{n\log^2 n}{8\gamma} \ge \Omega(n\log^2 n).$$

4. There are least $0.05\gamma\log n/\log\gamma$ different i's for which

$$\sum_{(I_A,I_B)\in\mathcal{I}_i} |P(I_A)\cap P(I_B)| > \frac{n\log n}{200(c+1002)\gamma\log\log n}.$$

By construction of the $\mathcal{I}_i$'s, each probe will appear only once across all interval pairs. The total number of probes is at least

$$0.05\gamma \log n/\log\gamma \cdot \frac{n\log n}{200(c+1002)\gamma\log\log n} \geq \Omega\left(n\left(\frac{\log n}{c\log\log n}\right)^2\right).$$

The above asserts that either D is wrong on some query or D probes $\Omega(n(\log n/c\log\log n)^2)$ cells, which finishes the proof of Claim 5.

By a union bound over all $\mathcal{E}_i$'s, we conclude that the probability that D probes $o(n(\log n/c\log\log n)^2)$ cells and is correct on all queries is at most

$$\sum_{i<0.1\gamma\log n/\log\gamma} \Pr[\mathcal{E}_i] < 2^{-n/\log^{c+O(1)} n},$$

which completes the proof of the entire theorem. □

References

[AW08] Scott Aaronson and Avi Wigderson. Algebrization: a new barrier in complexity theory. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 731–740, 2008.

[BBCR10] Boaz Barak, Mark Braverman, Xi Chen, and Anup Rao. How to compress interactive communication. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010*, pages 67–76, 2010.

[Blu85] Norbert Blum. On the single-operation worst-case time complexity on the disjoint set union problem. In *STACS 85, 2nd Symposium of Theoretical Aspects of Computer Science*, pages 32–38, 1985.

[BRWY13] Mark Braverman, Anup Rao, Omri Weinstein, and Amir Yehudayoff. Direct products in communication complexity. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*, pages 746–755, 2013.

[DS14] Irit Dinur and David Steurer. Direct product testing. In *IEEE 29th Conference on Computational Complexity, CCC 2014*, pages 188–196, 2014.

[FKNN95] Tomàs Feder, Eyal Kushilevitz, Moni Naor, and Noam Nisan. Amortized communication complexity. *SIAM Journal on Computing*, 24(4):736–750, 1995. Prelim version by Feder, Kushilevitz, Naor FOCS 1991.

[FS89] Michael L. Fredman and Michael 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.

[JPY12] Rahul Jain, Attila Pereszlényi, and Penghui Yao. A direct product theorem for the two-party bounded-round public-coin communication complexity. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012*, pages 167–176, 2012.

[KKN95] Mauricio Karchmer, Eyal Kushilevitz, and Noam Nisan. Fractional covers and communication complexity. *SIAM J. Discrete Math.*, 8(1):76–92, 1995.

[Kla11] Hartmut Klauck. On Arthur Merlin games in communication complexity. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011*, pages 189–199, 2011.

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[Lar12] Kasper Green 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.

[Pat07] Mihai Patrascu. Lower bounds for 2-dimensional range counting. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, STOC 2007*, pages 40–46, 2007.

[PD04] Mihai Pătraşcu and Erik D. Demaine. Tight bounds for the partial-sums problem. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004*, pages 20–29, 2004.

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

[PT06] Mihai Patrascu and Mikkel Thorup. Higher lower bounds for near-neighbor and further rich problems. In *47th Annual IEEE Symposium on Foundations of Computer Science FOCS 2006*, pages 646–654, 2006.

[PT11] Mihai Pătraşcu and Mikkel 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.

[PTW10] Rina Panigrahy, Kunal Talwar, and Udi Wieder. Lower bounds on near neighbor search via metric expansion. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010*, pages 805–814, 2010.

[Raz98] Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, June 1998. Prelim version in STOC '95.

[Tar75] Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225, 1975.

[Yao77] Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science*, pages 222–227, 1977.

[Yao79] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing. In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing*, pages 209–213, 1979.

[Yao81] Andrew Chi-Chih Yao. Should tables be sorted? *J. ACM*, 28(3):615–628, 1981.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, FOCS 1982*, pages 80–91, 1982.

[Yu15] Huacheng Yu. Cell-probe lower bounds for dynamic problems via a new communication model. *CoRR*, abs/1512.01293, 2015.

A Proof of Proposition 1

Let D_{stat} be a (zero-error) data structure for the *static* 3D-ORC problem, that uses $s(m)$ memory cells and $t_{\text{stat}}(m)$ probes to answer any query on m points. To solve the dynamic 2D-ORC problem on a sequence of $n/\log^c n$ updates and $n - n/\log^c n$ queries, we *guess* all updates before the first operation. Let the i-th update we guessed be `update(`r_i`, `c_i`, `w_i`)`. Then we use D_{stat} to build a static data structure on points (i, r_i, c_i) with weight w_i, and write to the memory. This step costs $s(n/\log^c n)$ probes. Note that the cell-probe model does not charge for the actual construction time of the data structure, but only for the number of probes to the memory. In addition, we also maintain a counter q in the memory, recording the number of updates performed so far, which is initialized to be 0.

To do an update, we simply increment the counter q by one. To answer a query `query(`r`, `c`)`, we first read the value of q from the memory, then query D_{stat} the sum of weights of points dominated by (q, r, c). This corresponds to asking D_{stat} what is the sum of weights of points appeared in the first q updates (we guessed) that are dominated by (r, c), which costs $t_{\text{stat}}(n/\log^c n)$ probes. In total, we spend $s(n/\log^c n) + n/\log^c n + n \cdot t_{\text{stat}}(n/\log^c n)$ probes.

If we happen to guess all $n/\log^c n$ update correctly, then all $n - n/\log^c n$ queries will be answered correctly, which happens with probability $n^{-3n/\log^c n}$. This is an upper bound on the probability asserted by the strengthened version of Theorem 1, that all queries are answered correctly and $o(n(\log n/c \log\log n)^2)$ cells are probed. In particular, we then must have

$$s(n/\log^c n) + n \cdot t_{\text{stat}}(n/\log^c n) \geq \Omega(n(\log n/c \log\log n)^2).$$

Thus, if $s(m) \leq O(m \log^c m)$, we must have $t_{\text{stat}}(m) \geq \Omega((\log m/c \log\log m)^2)$.

A.1 Challenges in improving Theorem 1

Although only a slightly strengthened version of Theorem 1 is required by Proposition 1, making the improvement still seems non-trivial, as it forces us to *break* Yao's Minimax Principle in the cell-probe model. In the cell-probe model, one direction of the principle is still true: A lower bound for deterministic data structures on a fixed hard input distribution is always a lower bound for any randomized data structure on its worst-case input. This is the direction that we (and also many previous works) use in the proof. But the other direction may no longer hold. One way to see it is that when we try to fix the random bits used by a randomized data structure, its running time may drop significantly. If the random bits are fixed, they can be hard-wired to the data structure, and can be accessed during the operations for free. On the other hand, if they are generated on the fly during previous operations, the data structure has to probe memory cells to recover them, because it does not remember anything across the operations by definition. A randomized data structure is a convex combination of deterministic data structures. Thus, in the cell-probe model, the function that maps a data structure to its performance on a fixed input may not be linear, i.e., the performance of a convex combination of deterministic data structures can be strictly larger than the same convex combination of the performances of deterministic data structures on a fixed input. In contrast, the proof of Yao's Minimax Principle relies on the linearity of this function.

Indeed, for the dynamic 2D-ORC problem and any distribution over operation sequences with $n/\log^c n$ updates and $n - n/\log^c n$ queries, we can solve it trivially if only $n^{-3n/\log^c n}$ correct probability is required: hard-wire the most-likely sequence of $n/\log^c n$ updates, and answer all queries based on it. Thus, each update and query can be done in constant time. When the most-likely sequence of updates occurs (with probability $\geq n^{-3n/\log^c n}$), all queries will be answered correctly. Thus, to improve Theorem 1, one would have to design a "data-structure-dependent" hard distribution, adversarially tailored to each data structure we are analyzing, and carrying out such argument seems to require new ideas.

B Proof of Claim 4

Proof of Claim 4. The procedure of generating the sets $\mathcal{I}_i$ can be modelled as a binary tree. The root of the tree is $(I_A, I_B) = \mathrm{DEC}(\mathcal{O})$. Its left child is $\mathrm{DEC}(I_A)$, and its right child is $\mathrm{DEC}(I_B)$. In general, for each node (I, I'), its left child is $\mathrm{DEC}(I)$ and its right child is $\mathrm{DEC}(I')$. We keep expanding until either $|I'| < \sqrt{n}$ or the node is in depth i. It is easy to verify that

- the sum of $|I_B|$ over all leaves (I_A, I_B) is $\sim n/\gamma$,
- $\sum_{(I_A,I_B)\in\mathcal{I}_i} |I_B|$ is just the sum of $|I_B|$ over all leaves with $|I_B| \geq \sqrt{n}$.

Thus, it is sufficient to bound the number of leaves (I_A, I_B) with $|I_B| < \sqrt{n}$. Fix a leaf, consider the path from root to it. Every time the path follows a left-child-edge, $|I_B|$ shrinks by a factor of $1 - 1/\gamma$. Every time it follows a right-child-edge, $|I_B|$ shrinks by a factor of $1/\gamma$. Since we stop expanding the tree as soon as $|I_B| < \sqrt{n}$, the path can follow a right-child-edge at most $\lceil \log(\sqrt{n}/\gamma)/\log\gamma \rceil$ times (and stops as soon as it follows the $\lceil \log(\sqrt{n}/\gamma)/\log\gamma \rceil$ one). Thus, there are at most

$$\begin{aligned}\binom{\leq 0.1\gamma\log n/\log\gamma}{\leq \lceil \log(\sqrt{n}/\gamma)/\log\gamma \rceil - 1} &\leq \binom{\leq 0.1\gamma\log n/\log\gamma}{\leq \log(\sqrt{n}/\gamma)/\log\gamma} \\ &\leq (0.2e\gamma)^{\log(\sqrt{n}/\gamma)/\log\gamma} \\ &< 0.6^{\log(\sqrt{n}/\gamma)/\log\gamma} \cdot \frac{\sqrt{n}}{\gamma} \ll \frac{\sqrt{n}}{2\gamma}\end{aligned}$$

leaves in total. By above observation, $\sum_{(I_A,I_B)\in\mathcal{I}_i} |I_B| \geq n/\gamma - \sqrt{n}\cdot\sqrt{n}/2\gamma \geq n/2\gamma$. $\square$