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

Omri Weinstein
Department of Computer Science
New York University

Huacheng Yu
Department of Computer Science
Stanford University

***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.

I. 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], [Yu16]). 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*

[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 $\{1,2,\dots,n\}$.

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 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 $Q_1, Q_2, \dots, Q_q$ 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*! While a similar-in-spirit phenomenon was previously shown by [CJL15] for *static* data structures,[5] to best of our knowledge, this is the first result of its kind for dynamic problems 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 subprocedure 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

[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").

[5]In their setting, each query requires linear number of bits to describe, which makes the exponentially low success probability more expected.

problem, however, when $\mathcal{P} = \text{2D-ORC}$, 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 Section IV for discussion), 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 the full version.

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 $\approx \Pr(\mathcal{W})$ 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.

A. 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 I-B).

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], [Yu16]. We observe that a variation of Larsen's approach can be adapted to prove a zero-error-data-structure version of Lemma 3. 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) 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 protocol, Alice's input size has to be larger than the number of bits transmitted in the communication step (as Alice can always send her input and the parties would be done). This 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 $\mathsf{MA}^{\mathsf{cc}} \cap \mathsf{coMA}^{\mathsf{cc}}$ [BFS86]. 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]).

B. *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[6]) simulations of randomized data structures, yet in some aspect is much weaker than $\mathsf{MA}^{\mathsf{cc}}$, and hence amenable to substantial lower bounds. To enable a faster simulation (than [PT11], [Yu16]), 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.[7]

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 to answer his queries. Otherwise, if Megan is cheating (by sending addresses inconsistent 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 $\operatorname{poly}\log(n)$ factor in communication for Alice, and a $\approx \log n$ factor in communication for Bob, compared to the standard nondeterministic simulations of [PT11], [Yu16]. We stress that this speed-up is essential to prove the cell-probe lower bound we seek on 2D-ORC, and is 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 1). 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-

[6]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.

[7]The main difference between ANC and the model defined in [Yu16] is the trusted prover.

product" type result for 2D-ORC in the 4ANC model (Lemma 2) is one of the main steps of the proof of Theorem 1.

C. Organization

Due to space constraints, the next 5 pages of this abstract only contain a high-level proof of our results, where most proofs can be found in the full version. We begin by formally defining the 4ANC model in Section II. We then prove that 4ANC protocols can efficiently simulate dynamic data structures (Section II-A), and on the other hand, that efficient 4ANC protocols induce large biased rectangles of the underlying communication matrix (Section II-B). In Section III we state our main technical lemma which rules out such rectangles (even with exponentially small bias) for 2D-ORC (Lemma 2), and finally tie the pieces together to conclude the proof of Theorem 1. In Section IV, we conclude with some discussion.

II. 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: A public random string r (of infinite length) 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[8] $M_{\text{meg}}(x, y, m_{\text{mer}}, r)$ 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.

A 4-party communication protocol P

0) Alice and Bob use public randomness to sample a 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.

Definition 1 (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 2 (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 $(v = f(x, y))$.

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

[8]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.

protocol that δ-solves f under μ with communication cost (c_A, c_B, c_M).

A. 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.[9]

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 $|P(I_A) \cap P(I_B)|$.[10] 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". 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, continues the simulation up to the beginning of I_B, and saves a copy of the memory M_B. Then she continues to simulate D on I_B *based on the contents of* $\mathbf{M_A}$ 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 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).
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 $h(S)$.
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 $h^{-1}(b) \cap P(I_A)$).
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

[9]We remark that a similar idea of encoding the cells by their update times was recently used by Clifford, Jalsenius and Sach [CJS14].

[10]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 $P(I_A) \cap P(I_B)$ 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.

The analysis of the protocol can be found in the full version.

B. *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. The following lemma asserts 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 the full version for the formal definition). We note that a variant of this lemma can be proved for general (non-product) distributions. Due to space restriction, the proof will be in the full version.

Lemma 1 (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) $\mu_y(Y) \geq \frac{\delta}{4} \cdot 2^{-(c_M+c_B)}$.*

III. 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. 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.

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 $G_{\text{2D-ORC}}(k, q, n)$: 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 $\text{2D-ORC}(x, y) : ([n]^2 \times [n])^k \times ([n]^2)^q \to [kn]^q$ 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 $x_j \leq y_i$,[11] i.e.,

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

The input of the communication game $G_{\text{2D-ORC}}(k, q, n)$ can be embedded into a sequence of operations for the 2D-ORC data structure problem, such that Alice's input corresponds to k updates, Bob's input corresponds to q queries, and output for the communication game corresponds to the answers to the q queries. Assuming there is a good data structure D, we can solve $G_{\text{2D-ORC}}(k, q, n)$ efficiently using the simulation protocol in Section II-A. See full version for more details. On the other hand, the following lemma asserts 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. An intuitive overview and the formal proof of this lemma can be found in the full version.

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

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

$\log^{1000} n$,

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

Combining the communication lower bound for $G_{\text{2D-ORC}}(k,q,n)$ with a simulation argument gives us a lower bound on the efficiency and accuracy of D in I_A and I_B. More details can be found the full version.

Lemma 3. *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 is at most $2 \cdot 2^{-0.2|I_B|\log\log n}$:*

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

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 3 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 3 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 3 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. The formal proof can be found in the full version.

IV. Final Remarks on Data-structure-dependent Hard Distribution

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: Even if for every possible input distribution, we managed to design an efficient deterministic data structure, it is still possible that no randomized data structure has good worst-case-input guarantees.

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.

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.

[BFS86] László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory (preliminary version). In *27th Annual Symposium on Foundations of Computer Science*, pages 337–347, 1986.

[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.

[CJL15] Raphaël Clifford, Allan Grønlund Jørgensen, and Kasper Green Larsen. New unconditional hardness results for dynamic and online problems. *CoRR*, abs/1504.01836, 2015.

[CJS14] Raphaël Clifford, Markus Jalsenius, and Benjamin Sach. Cell-probe bounds for online edit distance and other pattern matching problems. *CoRR*, abs/1407.6559, 2014.

[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.

[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.

[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.

[Yu16] Huacheng Yu. Cell-probe lower bounds for dynamic problems via a new communication model. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 362–374, 2016.