# Cell-Probe Lower Bounds for Dynamic Problems via a New Communication Model

Huacheng Yu
Stanford University
353 Serra Mall
Stanford, CA, USA
yuhch123@gmail.com

## ABSTRACT

In this paper, we develop a new communication model to prove a data structure lower bound for the dynamic interval union problem. The problem is to maintain a multiset of intervals $\mathcal{I}$ over $[0, n]$ with integer coordinates, supporting the following operations:

- `insert(a, b)`: add an interval $[a, b]$ to $\mathcal{I}$, provided that $a$ and $b$ are integers in $[0, n]$;
- `delete(a, b)`: delete an (existing) interval $[a, b]$ from $\mathcal{I}$;
- `query()`: return the total length of the union of all intervals in $\mathcal{I}$.

It is related to the two-dimensional case of Klee's measure problem. We prove that there is a distribution over sequences of operations with $O(n)$ insertions and deletions, and $O(n^{0.01})$ queries, for which any data structure with any constant error probability requires $\Omega(n \log n)$ time in expectation. Interestingly, we use the sparse set disjointness protocol of Håstad and Wigderson [ToC'07] to speed up a reduction from a new kind of nondeterministic communication games, for which we prove lower bounds.

For applications, we prove lower bounds for several dynamic graph problems by reducing them from dynamic interval union.

## Categories and Subject Descriptors

E.1 [**Data Structures**]

## General Terms

Algorithms, Theory

## Keywords

Klee's measure problem, lower bound, cell-probe model

## 1. INTRODUCTION

In computational geometry, Klee's measure problem [14, 4, 5] is the following: given $N$ rectangular ranges (axis-parallel hyper-rectangles) in $d$-dimensional space, compute the volume of their union.

A classic sweep-line algorithm by Bentley [4] solves the $d = 2$ case in $O(N \log N)$ time: consider the line $x = x_0$ with $x_0$ continuously increasing from $-\infty$ to $\infty$; the length of the intersection of this line and the union may change only when it reaches the left or right border of a rectangle. Bentley's algorithm uses a segment tree to dynamically maintain the length of the intersection efficiently. Surprisingly, this is the best known algorithm even for an intriguing special case: all coordinates are integers within a polynomially bounded range $[0, n]$. In this case, the segment tree in Bentley's algorithm is essentially used to solve the following dynamic problem, which we call the *dynamic interval union problem*:

Maintain a multiset $\mathcal{I}$ of intervals with integer coordinates in $[0, n]$, supporting the following operations:

- `insert(a, b)`: add an interval $[a, b]$ to $\mathcal{I}$, provided that $a$ and $b$ are integers in $[0, n]$;
- `delete(a, b)`: delete an (existing) interval $[a, b]$ from $\mathcal{I}$;
- `query()`: return the total length of the union of all intervals in $\mathcal{I}$.

The segment tree data structure solves this dynamic problem with $O(\log n)$ insertion and deletion time, and $O(1)$ query time. For the application to 2D Klee's measure problem, there are $N$ insertions, $N$ deletions and $N$ queries to the data structure. A natural question to ask here is whether we can improve the insertion and deletion time. However, there is a very simple reduction from the partial sum problem showing that the slowest operation among insertion, deletion and query needs to take $\Omega(\log n)$ time (see Appendix A). Moreover, Pătraşcu and Demaine [21, 23] showed an optimal trade-off between update and query time for partial sum, which can be carried over via the reduction to show that if both insertion and deletion need to be done in $O(\epsilon \log n)$ time, then query has to take $\Omega(2^{1/\epsilon} \log n)$ time.

This seems to be the end of the story. However, in fact, there is no $o(N \log N)$ time algorithm known even for $n = N^{0.51}$.[1] When we apply the above dynamic problem to this case, there will be $N$ insertions, $N$ deletions, and only $n = N^{0.51}$ queries! There will be far fewer queries than insertions and deletions. The argument above does not rule out the possibility of having a dynamic interval union data structure with $o(\log n)$ update time, and $O(n^{0.9})$ query time. It is even possible to have a data structure with $O(1)$ insertion

[1]There is a simple linear time algorithm for the $n \leq \sqrt{N}$ case.

and deletion time, and $O(n^{0.1})$ query time. Having such a data structure would give a linear time algorithm for the above special case of Klee's measure problem, making a breakthrough on this 40-year-old problem.

Unfortunately, we show that such data structure does not exist, even if we allow randomization, amortization and constant error probability.

THEOREM 1. *For any $\epsilon > 0$ and integer $n \geq 1$, there is a distribution over operation sequences to the dynamic interval union problem over $[0, n]$, with $O(n^{1-\epsilon})$ insertions and deletions, and $O(n^{\epsilon})$ queries, for which any data structure that is correct on* all *queries with probability at least $95\%$ must spend $\Omega(\epsilon^2 n^{1-\epsilon} \log n)$ probes in expectation, in the cell-probe model with word size $w = \Theta(\log n)$.*

We define the cell-probe model in Section 1.1.

It is an easy exercise to show that we can use the hard distribution from the theorem to obtain a new hard distribution with $\Theta(n^{1/0.51}) = \Theta(N)$ insertions and deletions and $n$ queries, such that any data structure requires $\Omega(N \log N)$ time on it. This lower bound rules out the possibility of using the plain-vanilla sweep-line algorithm with a sophisticated data structure to solve 2D Klee's measure problem faster than the classic algorithm. As a corollary, the theorem also implies that $o(\log n)$ insertion and deletion time leads to an almost linear lower bound on query time.

The type of running time considered by Theorem 1 is more general than the amortized time. Having amortized time $t(n)$ usually means that the first $k$ operations take at most $k \cdot t(n)$ time for every $k$, but here, even if we fix the number of operations in advance, and the data structure is allowed to use heavy preprocessing in order to optimize the total running time, there is still no way to break the lower bound. This notion of running time is usually what we care about, when applying a data structure to solve some computational problem. The only catch is that the data structure is online: it must output an answer before seeing the next operation, which makes it different from an offline computational problem. Moreover, using similar techniques, we can prove the same lower bound for data structures that correctly answer a constant fraction of the queries in expectation. The proof is omitted in this version.

THEOREM 1′. *For any $\epsilon > 0$ and integer $n \geq 1$, there is a distribution over operation sequences to the dynamic interval union problem over $[0, n]$, with $O(n^{1-\epsilon})$ insertions and deletions, and $O(n^{\epsilon})$ queries, for which any data structure with expected fraction of correct answers at least $\delta$ must spend $\Omega_\delta(\epsilon^2 n^{1-\epsilon} \log n)$ probes in expectation for any $\delta \in (0, 1]$, in the cell-probe model with word size $\Theta(\log n)$.*

We prove Theorem 1 via a reduction from a more accessible intermediate problem called *batch partial sum*, for which we prove a lower bound directly. The batch partial sum problem asks to maintain $K$ sequences $(A_{i,j})_{i\in[K],j\in[B]}$ of length $B$ over a finite field $\mathbb{F}_p$, supporting the following operations to the sequences:[2]

- `update(`$\mathbf{j}$`,` $\mathbf{v}$`)`: for all $i \in [K]$, set $A_{i,j_i}$ to value $v_i$;
- `query(`$\mathbf{j}$`)`: return $\sum_{i\in[K]} \sum_{l\leq j_i} A_{i,l}$,

provided that $\mathbf{j}$ and $\mathbf{v}$ are vectors of length $K$, and $j_i \in [B]$, $v_i \in \mathbb{F}_p$. Basically, we need to maintain $K$ independent copies of the partial sum problem, except that when answering queries, instead of returning $K$ individual prefix sums, we only need to return the sum of these $K$ numbers.

[2]In this paper, $[K]$ stands for the set $\{1, 2, \ldots, K\}$.

THEOREM 2. *For large enough integers $K, B, p$ with $p \geq B$, there is a distribution over operation sequences to the batch partial sum problem with $O(B)$ updates and $O(B)$ queries, for which any data structure that is correct on* all *queries with probability $\geq 95\%$ must spend $\Omega(KB \log^2 B/(w + \log B))$ probes in expectation, in the cell-probe model with word size $w$.*

Note that when $K, B, p$ are polynomials in $n$ and $w = \Theta(\log n)$ for some $n$, the lower bound becomes $\Omega(K \log n)$ per operation. Therefore, in this case, the best thing to do is just to use $K$ partial sum data structures to maintain the $K$ sequences independently. However, the techniques we use in the proof are very different from the proof of the lower bound for partial sum by Pătraşcu and Demaine [22]. See Section 1.3 for an overview.

Moreover, we also apply our main theorem to prove lower bounds for three dynamic graph problems: dynamic number of strongly connected components (or dynamic #SCC), dynamic weighted s-t shortest path, and dynamic planar s-t min-cost flow (see Appendix B for formal definitions). We prove that for these problems, under constant error probability, if we spend $o(\log n)$ update time, then queries must take $n^{1-o(1)}$ time. Note that a previous result of Pătraşcu and Thorup [24] also implies the same trade-off for the first two problems under zero error.

COROLLARY 2. *For the following three dynamic graph problems:*

*(a) dynamic number of strongly connected components,*[3]

*(b) dynamic planar s-t min-cost flow,*[4]

*(c) dynamic weight s-t shortest path,*[5]

*any data structure with amortized expected update time $o(\log n)$, and error probability $\leq 5\%$ under polynomially many operations must have amortized expected query time $n^{1-o(1)}$.*

## 1.1 Cell-Probe Model

The cell-probe model of Yao [30] is a strong non-uniform computational model for data structures. A data structure in the cell-probe model has access to a set of memory cells. Each cell can store $w$ bits. The set of cells is indexed by $w$-bit integers, i.e., the address is in $[2^w]$. $w$ is usually set to be $\Omega(\log n)$, where $n$ is the amount of information the data structure needs to handle. [6]

To access the memory, the data structure can *probe* a cell, which means that it can look at the content of the cell, and then optionally overwrite it with a new value. During an operation, the data structure based on the parameters of the operation decides which cell to probe the first, then based on the parameters and the information from the first probe, decides the cell to probe next, etc. Each cell-probe (including both the address and the new value) the data structure performs may be an arbitrary function of the parameters of the operation and the contents in the cells previously probed during this operation. If the operation is a query, in the end, the data structure returns an answer based on the parameters and the contents in all cells probed during this operation. The update (query

[3]It is to dynamically maintain the number of strongly connected components of a directed graph, under edge insertions and deletions.

[4]It is to maintain s-t min-cost flow on a planar graph with fixed planar embedding, under capacity updates.

[5]It is to maintain s-t shortest path on a weighted directed graph, under edge insertions and deletions.

[6]$n$ is usually a polynomial in the number of operations.

resp.) time is defined to be the number of cells probed during a(n) update (query resp.) operation.

The cell-probe model only counts the number of memory accesses during each operation, making itself a strong model, e.g., it subsumes the word-RAM model. Thus, data structure lower bounds proved in this model will hold in various other settings as well.

## 1.2 Previous Data Structure Lower Bounds

In 1989, Fredman and Saks introduced the *chronogram* method to prove an $\Omega(\log n/\log\log n)$ lower bound for partial sum in their seminal paper [10]. The lower bound is tight for maintaining a sequence of $\{0,1\}$s. $\Omega(\log n/\log\log n)$ was also the highest lower bound proved for any explicit data structure problem for a long time.

In 2004, Pătraşcu and Demaine [22] broke this $\log n/\log\log n$ barrier using a new approach: the *information transfer tree* technique. They proved an $\Omega(\log n)$ lower bound for the partial sum problem with numbers in $[n]$. Moreover, using this new technique, they proved an update-query time trade-off of $t_q \log \frac{t_u}{t_q} = \Omega(\log n)$, where $t_u$ is the update time and $t_q$ is the query time, while earlier approaches can only prove $t_q \log t_u = \Omega(\log n)$. Later on, the information transfer tree technique has been used to prove several other data structure lower bounds [21, 9, 7, 8].

In 2012, there was a breakthrough by Larsen [16] on dynamic data structure lower bounds. Larsen combined the chronogram method with the *cell-sampling* technique of Panigraphy, Talwar and Wieder [18], and proved an $\Omega((\log n/\log\log n)^2)$ lower bound for the 2D orthogonal range counting problem. This lower bound is also the highest lower bound proved for any explicit dynamic data structure problem so far. Similar approaches were also applied later [17, 6].

All above techniques can only be used to prove a relatively smooth trade-off between update and query time. However, Pătraşcu and Thorup [24] used a new idea to prove a sharp trade-off for the dynamic connectivity problem in undirected graphs. They proved that if one insists on $o(\log n)$ insertion and deletion time, query has to take $n^{1-o(1)}$ time. Besides the sharp trade-off, they also introduced the *simulation by communication games* of the data structure. They first decomposed the entire execution of the data structure on a sequence of operations into several communication games. For each communication game, they showed how to turn a "fast" data structure into an efficient communication protocol. Then they proved a communication lower bound for each game. Summing all these lower bounds up establishes a lower bound on the total number of probes in the entire execution.

Interestingly, Clifford, Grønlund and Larsen [6] used a completely different approach to prove a sharp trade-off in the opposite direction. Their result implies that for several dynamic graph problems (e.g., dynamic directed connectivity and dynamic undirected shortest path), if a data structure takes $o(\log n/\log\log n)$ query time and uses nearly linear space, it must take $n^{1-o(1)}$ update time.

Kaplan, Zamir and Zwick [13] also proved a sharp trade-off for data structures supporting priority queue operations in the *comparison* model: if both `insert` and `delete` can be done in amortized $o(\log n)$ time, then `find-min` must take $n^{1-o(1)}$ time. In general, a dynamic interval union data structure can be used to implement a priority queue over $[n]$: present an element $x$ by interval $[x,n]$, the length of the union tells us $n$ minus the minimum value. Thus, a similar result in the cell-probe model (and allowing constant error probability) would imply our Theorem 1. However, it seems nontrivial to generalize their proof to the cell-probe model, as proving lower bounds in different models requires different techniques.

## 1.3 Technical Contributions

Although the batch partial sum problem is a natural generalization of the partial sum problem, it seems hopeless to apply the information transfer technique directly to solve our problem. It is due to a critical difference between the two problems: in partial sum, the lower bound proved roughly equals to the number of bits in the answer to a query ($\log n$); while in batch partial sum, the lower bound we aim at is much larger than the size of an answer. The proof in [22] heavily relies on the fact that in partial sum problem, after fixing the values in a lot of entries in the sequence, as long as there is still one summand in the prefix sum left uniformly at random, the sum will also be uniformly at random. Therefore, the data structure needs to learn a certain amount of information from the memory to figure out the answers, even if we fix many updates. If we apply the same technique to batch partial sum, as an answer still contains only $\log n$ bits of information, we will again get a lower bound of $\Omega(\log n)$ per operation, while we aim at $\Omega(K\log n)$. The cell-sampling technique faces a similar problem. It can only be applied when the number of bits used to describe a query is comparable with the number of bits used in an answer.

The main idea of our proof is to use the simulation by communication games technique mentioned in Section 1.2. After decomposing into communication games, there are two things to prove: a "fast" data structure implies an efficient communication protocol, and no efficient communication protocol exists. The choice of communication model for the game is crucial. If we use a too weak communication model, it would be hard to take advantage of the model to design an efficient protocol given a fast data structure. If the communication model we use is too strong, it might be difficult or even impossible to prove a communication lower bound, especially when small chance of error is allowed. Pătraşcu and Thorup gave two different simulations (transforming a data structure into a communication protocol): one in the deterministic setting, the other in the nondeterministic setting. The deterministic simulation itself is not efficient enough to achieve our lower bound. The nondeterministic model, in our case of allowing error, would correspond to the $\mathsf{MA}^{\mathsf{cc}}$ model [3] (or $\mathsf{MA}^{\mathsf{cc}} \cap \mathsf{coMA}^{\mathsf{cc}}$ to be more precise). Proving a lower bound in this model is usually very difficult. In our application, the communication problem we want to prove a lower bound for is closely related to the inner product problem. Namely, Alice and Bob get $n$-dimensional binary vectors $x$ and $y$ respectively and the goal is to compute the inner product $\langle x,y\rangle$ over $\mathbb{F}_2$. There is a clever $\mathsf{MA}^{\mathsf{cc}} \cap \mathsf{coMA}^{\mathsf{cc}}$ protocol by Aaronson and Wigderson [1] which solves the inner product problem with only $\tilde{O}(\sqrt{n})$ bits of communication. It costs much less than one would expect, which suggests that it might even be impossible to prove a desired (linear) lower bound for our problem in this strong model.

To overcome this obstacle, we define a new communication model (see Section 3 for the definition and a comparison with other communication models), which is weaker than $\mathsf{MA}^{\mathsf{cc}} \cap \mathsf{coMA}^{\mathsf{cc}}$, so that we are capable of proving a desired communication lower bound. Moreover, we will be able to achieve the same performance of transforming data structure into protocol as in the nondeterministic model. Interestingly, in order to have less requirement on the power of communication model, we use an elegant protocol for computing sparse set disjointness by Håstad and Wigderson [12] as a subroutine:

THEOREM 3 (HÅSTAD AND WIGDERSON). *In the model of common randomness,* $\mathsf{R}_0(\mathsf{DISJ}_k^n) = O(k)$ *for instances of disjoint sets and* $\mathsf{R}_0(\mathsf{DISJ}_k^n) = O(k+\log n)$ *for non-disjoint sets.*

$\mathsf{DISJ}_k^n$ is the following problem: Alice and Bob get sets $X$ and $Y$ of size $k$ over a universe $[n]$ respectively, their goal is to com-

pute whether the two sets are disjoint. $\mathsf{R}_0(\mathsf{DISJ}_k^n)$ stands for the minimum expected communication cost by any zero-error protocol which computes $\mathsf{DISJ}_k^n$.

As we will see later, this new communication model has the power of nondeterminism. It is also restricted enough so that we can apply the classic techniques for proving randomized communication lower bounds. Using this model, we prove the first sharp update-query trade-off under constant error probability.

## 1.4 Overview

The remainder of this paper is organized as follows. In Section 3, we present the reduction from batch partial sum to dynamic interval union, and define the new communication model and the new simulation. In Section 4, we prove a communication lower bound in this new model, which completes the proof of our main result. In Section 5, we apply the main theorem to several dynamic graph problems. Finally, we conclude with some remarks in Section 6.

# 2. PRELIMINARIES

In the deterministic communication complexity setting [29], two players Alice and Bob receive inputs $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ respectively. Their goal is to collaboratively evaluate a function $f$ on their joint input $(x, y)$. The players send bits to each other according to some predefined protocol. At each step, the protocol must specify which player sends the next bit based on the transcript (the bits sent so far). The sender decides to send a bit 0 or 1 based on the transcript and his/her input. It the end, the answer $f(x, y)$ can only depend on the entire transcript. In the setting with public randomness, the players have access to a common random binary string of infinite length. Besides the sender's input and the transcript, each message may also depend on these random bits. The players have infinite computational power. The cost of a protocol is the number of bits communicated, i.e., the length of the transcript.

DEFINITION 1. *For function $f$ with domain $\mathcal{X} \times \mathcal{Y}$, the matrix $M(f)$ is a $|\mathcal{X}| \times |\mathcal{Y}|$ matrix, with rows indexed by $\mathcal{X}$ and columns indexed by $\mathcal{Y}$. The entry in row $x$ and column $y$ is the function value $f(x, y)$.*

DEFINITION 2. *A* combinatorial rectangle *or simply a* rectangle *in $M(f)$ is a set $X \times Y$ for $X \subseteq \mathcal{X}$ and $Y \subseteq \mathcal{Y}$.*

DEFINITION 3. *A* monochromatic rectangle *in $M(f)$ is a combinatorial rectangle in which the function value does not vary.*

DEFINITION 4. *Let $\mu$ be a distribution over $\mathcal{X} \times \mathcal{Y}$. A* $\alpha$-monochromatic rectangle *under $\mu$ is a combinatorial rectangle $X \times Y$ such that there is a function value $v$, $\alpha\mu(X \times Y) \leq \mu((X \times Y) \cap f^{-1}(v))$, i.e., a combinatorial rectangle with at least $\alpha$-fraction of the input pairs having the same function value.*

A classic result [15] in communication complexity is that every protocol in the deterministic setting with worst-case communication cost $C$ induces a partitioning of $M(f)$ into $2^C$ monochromatic rectangles. Each rectangle corresponds to one possible transcript, i.e., when the players are given an input pair in this rectangle, the corresponding transcript will be transmitted. A similar result shows that every randomized protocol with low error probability induces a partitioning into rectangles, such that *most* of the rectangles are *nearly monochromatic* ($\alpha$-monochromatic with $\alpha$ close to 1). Proving there is no large monochromatic rectangle or nearly monochromatic rectangle in $M(f)$ would imply communication lower bounds in deterministic or randomized setting respectively.

# 3. LOWER BOUND FOR DYNAMIC INTERVAL UNION

In this session, we will prove our main result, a lower bound for the dynamic interval union problem, via a reduction from the batch partial sum problem. Then we present the framework of proving a lower bound for batch partial sum, and will defer some of its details to Section 4.

THEOREM 1 (RESTATE). *For any $\epsilon > 0$ and integer $n \geq 1$, there is a distribution over operation sequences to the dynamic interval union problem over $[0, n]$, with $O(n^{1-\epsilon})$ insertions and deletions, and $O(n^{\epsilon})$ queries, for which any data structure that is correct on* all *queries with probability at least $95\%$ must spend $\Omega(\epsilon^2 n^{1-\epsilon} \log n)$ probes in expectation, in the cell-probe model with word size $w = \Theta(\log n)$.*

THEOREM 2 (RESTATE). *For large enough integers $K, B, p$ with $p \geq B$, there is a distribution over operation sequences to the batch partial sum problem with $O(B)$ updates and $O(B)$ queries, for which any data structure that is correct on* all *queries with probability $\geq 95\%$ must spend $\Omega(KB \log^2 B/(w + \log B))$ probes in expectation, in the cell-probe model with word size $w$.*

Assuming Theorem 2, we can prove Theorem 1 via the following reduction. The idea is similar to the proof of Proposition 1 in Appendix A.

PROOF OF THEOREM 1. Take prime $p = \Theta(n^{\epsilon})$, $B = \Theta(n^{\epsilon})$ with $B \leq p$, and $K = n/Bp = \Theta(n^{1-2\epsilon})$. We are going to show that we can solve the batch partial sum problem with this setting of the parameters given a dynamic interval union data structure over $[0, n]$. We first concatenate the $K$ sequences into one long sequence of length $KB$, such that $A_{i,j}$ will be $((i-1) \cdot B + j)$-th number in the long sequence, and try to maintain the whole sequence using one dynamic interval union data structure. Then we associate each number in the long sequence with a segment of length $p$ in $[0, n]$ such that the $k$-th number in the long sequence is associated with $[(k-1) \cdot p, k \cdot p]$. We use the length of interval in the associated segment to indicate the value of the number. That is, we always maintain the invariant that for $k$-th number in the long sequence with non-zero value $v$, we have exactly one interval $[(k-1) \cdot p, (k-1) \cdot p + v]$ intersecting its associated segment.

To set $k$-th number to a new value $v'$, if before the operation it had value $v \neq 0$, we first call $\texttt{delete}((k-1)p, (k-1)p + v)$ to reset the number. Then if $v' \neq 0$, we call $\texttt{insert}((k-1)p, (k-1)p + v')$ to update its new value to $v'$. Therefore, as an `update` of the batch partial sum problem is just setting $K$ numbers to new values, it can be implemented using $O(K)$ insertions and deletions of the dynamic interval union problem, with $O(K)$ extra probes to determine what the old value was and to record the new value.

To answer $\texttt{query}(\mathbf{j})$, we first insert intervals that correspond to associated segments of the $(j_i + 1)$-th number to the last number in sequence $i$ for $i \in [K]$, to set everything we are not querying to be "in the union", no matter how much they were covered by intervals before. That is, we insert $[((i-1)B + j_i) \cdot p, iB \cdot p]$ for each sequence $i$. Then we do one query, which will return the sum of all numbers as if each number not in the query was set to $p$ (or 0 modulo $p$). This number modulo $p$ is exactly the answer we want. At last, we do $K$ deletions to remove the temporary intervals we inserted earlier for this query, and return the answer. Therefore, every `query` of batch partial sum can be implemented using $O(K)$ insertions and deletions, and one query of the dynamic interval union.

Thus, any sequence of $O(B)$ updates and $O(B)$ queries of batch partial sum can be implemented using $O(KB) = O(n^{1-\epsilon})$ insertions and deletions, $O(B) = O(n^{\epsilon})$ queries of dynamic interval union, and extra $O(K) = O(n^{1-\epsilon})$ probes. However, by Theorem 2, there is a hard distribution consisting of $O(B)$ updates and $O(B)$ queries, which requires $\Omega(KB\log^2 B/(w+\log B)) = \Omega(\epsilon^2 n^{1-\epsilon}\log n)$ probes in expectation in the cell-probe model with word size $w = \Theta(\log n)$. By the above reduction, this hard distribution also induces a distribution over operation sequences for dynamic interval union with desired number of updates, queries and lower bound on the number of probes. This proves the theorem. □

By setting $\epsilon = \sqrt{t_u/\log n}$ in Theorem 1, we get the following corollary.

COROLLARY 1. *Any dynamic interval union data structure that answers all queries correctly in a sequence of $O(n)$ operations with probability $\geq 95\%$ with expected amortized insertion and deletion time $t_u$ and query time $t_q$ must have*

$$t_q \geq t_u n^{1-\sqrt{t_u/\log n}}.$$

*In particular, $t_u = o(\log n)$ implies that $t_q = n^{1-o(1)}$.*

In the following, we are going to prove a lower bound for the batch partial sum problem. We will first specify a hard distribution over operation sequences. Then by Yao's Minimax Principle [28], it suffices to show that any *deterministic* data structure that answers all queries correctly with high probability under this input distribution must be inefficient. To show this, we will consider a collection of communication games corresponding to different parts of the operation sequence (different time periods). For each communication game, if the data structure is *fast* within the corresponding time period under certain measure of efficiency, then the game can be solved with *low communication cost*. On the other hand, we will prove communication lower bounds for all these games. Summing these lower bounds up, we will be able to prove that the entire execution cannot be efficient.

*Hard distribution $\mathcal{D}$.* Without loss of generality, assume $B$ is a power of 2, and $B = 2^b$. The operation sequence will always have $B$ updates and $B$ queries occurring alternatively: $U_0$, $Q_0$, ..., $U_{B-1}$, $Q_{B-1}$, where the $U_t$'s are updates, and the $Q_t$'s are queries. The operations are indexed by integers between 0 and $B-1$, or they can be viewed as being indexed by $b$-bit binary strings (which corresponds to the binary representation of the integer). We may use either of these two views in the rest of the paper without further mention. Let $t$ be a binary string, $\mathrm{rev}(t)$ be the string with $t$'s bits reversed. For each $U_t$, we set it to `update(j, v)` with $j_i = \mathrm{rev}(t)$ and $v_i$ independently uniformly chosen from $\mathbb{F}_p$ for every $i$. For each $Q_t$, we set it to `query(j)` with $j_i$ independently and uniformly chosen from $[B]$. Different operations are sampled independently. Indicate this distribution by $\mathcal{D}$.

*Communication game.* A random sequence of operations sampled from $\mathcal{D}$ always has $2B$ operations with interleaving updates and queries. Let us fix two *consecutive* intervals $I_A, I_B$ of operations (with $I_A$ before $I_B$) in the sequence, e.g., let $I_A$ be 10th to 23rd operations, and $I_B$ be 24th to 32nd operations. Define the communication game $G(I_A, I_B)$ between two players Alice and Bob to be the following: sample a sequence from $\mathcal{D}$, Alice is given all operations except for those in $I_B$, Bob is given all operations except for those in $I_A$, their goal is to cooperatively compute the answers to all queries in $I_B$ by sending messages to each other alternatively.

Given a deterministic batch partial sum data structure, a natural way to solve this game is to let Bob first simulate the data structure up to the beginning of $I_A$, then skip all the operations in $I_A$ and try to continue simulating on $I_B$. Every time Bob needs to probe a cell, if it was probed in $I_A$, he sends a message to Alice asking for the new value, otherwise he knows exactly what is in the cell from his own simulation. The challenge for Bob is to figure out which cells were probed. Our main idea is to introduce a prover Merlin, who knows both Alice and Bob's inputs. Merlin will tell them this information in a *unique* and *succinct* way. Moreover, the players will be able to *verify* whether the message from Merlin is exactly what they expect, and will be able to solve the task efficiently if it is. This motivates the following definition of a new communication model.

*Communication model $\mathcal{M}$.* Draw an input pair $(x, y)$ from a known distribution. Alice is given $x$, Bob is given $y$ and Merlin is given both $x$ and $y$. Their goal is to compute some function $f(x, y)$. As part of the communication protocol, the players must specify a *unique* string $Z(x, y)$ for every possible input pair, which is the message Merlin is supposed to send. A communication procedure shall consist of the following four stages:

1. Merlin sends a message $z$ to Alice and Bob based on his input pair $(x, y)$;
2. Alice and Bob communicate based on $(x, z)$ and $(y, z)$ as in the classic communication setting with public randomness. Merlin does not see the random bits when sending the message $z$;
3. Alice and Bob decide to accept or reject;
4. if the players accept in Stage 3, they return a value $v$.

In this model, we say a protocol computes function $f$ with error $\varepsilon$ and communication cost $C$, if

1. Alice and Bob accept in Stage 3 if and only if Merlin sends what he is supposed to send, i.e., $z = Z(x, y)$ (with probability 1),
2. when $z = Z(x, y)$, the value $v$ they return equals to $f(x, y)$ with probability $\geq 1-\varepsilon$ over the input distribution and public randomness,
3. when $z = Z(x, y)$, the expected number of bits communicated between the three players in Stage 1 and 2 is no more than $C$ over the input distribution and public randomness.

**Remark.** The public randomness used in Stage 2 does not help the players in general. Nevertheless, we still keep it in the definition for the sake of neatness of our proof.

$Z(x, y)$ can be viewed as a question that the players want to ask Merlin about their joint input. One can potentially design more efficient protocols in this model than in the classic communication model if verifying the answer to this question is easier than computing it.

*Comparison with other communication models.* One crucial difference between popular nondeterministic models (for example, $\mathsf{NP}^{\mathsf{cc}} \cap \mathsf{coNP}^{\mathsf{cc}}$, $\mathsf{MA}^{\mathsf{cc}} \cap \mathsf{coMA}^{\mathsf{cc}}$) and our new model is the uniqueness of the certificate $Z$. In the classic nondeterministic models, the protocol must satisfy that the function value is

1 (or is 0) if and only there is a certificate that makes the players accept (with high probability). It is possible that some inputs have multiple such certificates. In terms of proving communication lower bound by the rectangle argument, multiple certificates correspond to a *covering* of the communication matrix by rectangles, while unique certificate corresponds to a *partitioning* of the matrix. When error is allowed in the protocol, partitioning of the communication matrix into rectangles allows us to "distribute" the error into rectangles, and concludes that most rectangles must be nearly monochromatic. However, this argument does not hold if it is a covering, because error part of the matrix may appear in many different rectangles. It is possible that the overall error probability is low, but every rectangle has a large error portion, thus is hard to prove a lower bound in this way.

Among all communication models people have studied, the one that is closest to our model is perhaps $\mathsf{UP}^{\mathsf{cc}} \cap \mathsf{coUP}^{\mathsf{cc}}$ [27, 11]. $\mathsf{UP}^{\mathsf{cc}}$ is the *ambiguous* version of $\mathsf{NP}^{\mathsf{cc}}$, i.e., the function value is 1 if and only if there is a unique certificate. $\mathsf{coNP}^{\mathsf{cc}}$ is defined in a similar way. One major difference between $\mathsf{UP}^{\mathsf{cc}} \cap \mathsf{coUP}^{\mathsf{cc}}$ and the new model is that our model allows error, but in a very specific way. They players can always tell whether Merlin is lying (no error in this part!), but when they decide to accept and output a value, the output can be wrong with small probability. In some sense, this corresponds to that the players can always simulate the data structure perfectly, but the data structure itself might be wrong, which leads to an incorrect output. There are also several other insignificant differences. For example, we work with a fixed input distribution, we do not restrict to binary functions, etc.

With respect to this communication model, on the one hand, we can show that given a "good" batch partial sum data structure, we can solve the communication game efficiently (Lemma 1). On the other hand, we prove a communication lower bound for it (Lemma 2). Combining these two lemmas, we conclude that no "good" data structure exists.

LEMMA 1. *Given a deterministic batch partial sum data structure for the cell-probe model with word size $w$ that is correct on all queries in a random operation sequence drawn from $\mathcal{D}$ with probability $\geq 90\%$, we can solve the communication game $G(I_A, I_B)$ with error $0.1$ and cost*

$$O\left(\mathop{\mathbb{E}}_{\mathcal{D}}\left[|P_A| + |P_B| + |P_A \cap P_B| \cdot w\right]\right)$$

*in model $\mathcal{M}$, where $P_A$ ($P_B$ resp.) is the (random) set of cells probed in time period $I_A$ ($I_B$ resp.) by the data structure.*

PROOF. We prove the lemma by showing the following protocol is efficient in terms of $P_A$ and $P_B$.

Step 1: (Merlin sends the key information)

Merlin first simulates the data structure up to the beginning of $I_B$, which is also the end of $I_A$, and records the set $P_A$, i.e., all the cells that were probed in time period $I_A$. Then Merlin continues his simulation on the operations in $I_B$. In the meanwhile, every time he probes a memory cell, he checks whether this cell was probed in $I_B$ before and whether it was probed in $I_A$ (in set $P_A$). If this is the first time probing this cell since the beginning of $I_B$, Merlin will send one bit to Alice and Bob. He sends "1" if the cell was probed in $I_A$, and sends "0" otherwise. Basically, Merlin sends one bit for every cell in $P_B$ indicating whether it is also in $P_A$. These bits are sent in the chronological order of their first probes in $I_B$.

Step 2: (Alice and Bob simulate the data structure to accomplish the task)

Alice simulates the data structure up to the beginning of $I_B$, and records $P_A$. Since Bob does not have any information about operations in $I_A$, he instead simulates up to the beginning of $I_A$, then tries to skip $I_A$ and simulate the operations in $I_B$ directly. Of course, the memory state Bob holds might be very different from what it should look like at the beginning of $I_B$. But with the help of Merlin's message, Bob will be able to figure out the difference.

As Bob is simulating the data structure, every time he needs to probe a cell, he first checks if this is the first time probing this cell since the beginning of $I_B$. If it is not, Bob knows its content from the latest probe to it. Otherwise, he looks at the next bit of Merlin's message. If it is a "0", Merlin is claiming that this cell was not probed in $I_A$. Thus, its content has not been changed since the beginning of $I_A$. Bob has the information in his own copy of memory. If it is a "1", Bob sends the address of this cell to Alice, Alice will send back its content. At the same time, Alice checks if the cell was actually probed in $I_A$. If not, they report "Merlin is cheating" (they reject), and abort the protocol. At last, Bob updates this cell in his own copy of memory, and records that it has been probed in $I_B$.

If there are no more bits left in Merlin's message when Bob needs to look at the next bit, or there are still unread bits when Bob has finished the simulation, the players reject.

Step 3: (Players verify that Merlin is truthful)

They are going to check if there is any memory cell in $P_A$ but Bob does not ask for its content during the simulation. Alice generates the set $P_A$. Bob generates the set of cells that are probed in $I_B$ in Step 2, but Merlin claims that they are not in $P_A$ (and thus did not ask Alice for the contents). They check if these two sets of cells are disjoint (all cells that Merlin claims to be not probed in $I_A$ are actually not) using the *zero-error* sparse set disjointness protocol in Theorem 3 of Håstad and Wigderson. If the two sets intersect, they report "Merlin is cheating" (reject), otherwise they report "Merlin is truthful" (accept) and Bob returns the answers he has computed for all queries in $I_B$.

Step 1 above describes what Merlin is *supposed* to do, and thus defines $Z(x, y)$. The following shows that the above protocol is a valid protocol in model $\mathcal{M}$, and solves the communication game efficiently.

1. If Merlin tells the truth ($z = Z(x, y)$), it is not hard to see that the players will always accept. If $z$ is a prefix of $Z(x, y)$ or $Z(x, y)$ is a prefix of $z$, Bob will detect it in Step 2 and reject. Otherwise let the $i$-th bit be the first bit that $z$ and $Z(x, y)$ differ. As the first $i - 1$ bits are the same in $z$ and $Z(x, y)$, Bob will correctly simulate the data structure up to that point, right before a probe that causes Bob to read the $i$-th bit of $z$. Thus the cell probed by the data structure corresponding to the $i$-th bit will be the same in Bob's simulation and in the actual execution. If $Z_i(x, y) = 0, z_i = 1$, the cell is not probed in $I_A$ but Merlin claims it is. The players can detect the mistake and will reject in Step 2. If $Z_i(x, y) = 1, z_i = 0$, Merlin claims the cell is not probed in $I_A$ but it is. In this case, the cell will belong to both Alice's and Bob's sets in Step 3. Therefore, the sparse set disjointness protocol will return "intersect". The players will

reject. This proves that Alice and Bob accept if and only if $z = Z(x, y)$.

2. When $z = Z(x, y)$, Bob will successfully simulate the data structure on all operations in $I_B$. As the data structure correctly answers all queries simultaneously with $\geq 90\%$ probability, in particular, it answers all queries in $I_B$ correctly with $\geq 90\%$ probability. Thus, the error probability is no more than 0.1.

3. When $z = Z(x, y)$, Merlin sends exactly one bit for each cell in $P_B$, $|z| = |P_B|$. In Step 2, the players send $O(w)$ bits for every "1" in $z$, which is $O(|P_A \cap P_B| \cdot w)$ in total. Note that by definition of the model, we only measure the communication cost when Merlin follows the protocol. In this case, the two sets generated in Step 3 must be disjoint. By Theorem 3, Step 3 costs $O(|P_A| + |P_B|)$ bits in expectation to compute sparse set disjointness. Thus, in total the protocol uses

$$O\left(\mathop{\mathbb{E}}_{\mathcal{D}}\left[|P_A| + |P_B| + |P_A \cap P_B| \cdot w\right]\right)$$

bits of communication as we claimed.

This proves the lemma. □

Let $s$ be a binary string of length less than $b = \log B$. Define $I_s$ to be the interval consisting operations $\{U_t, Q_t : s \text{ is a prefix of } t\}$. Let $G(s) = G(I_{s0}, I_{s1})$ be the communication game defined by $I_{s0}$ and $I_{s1}$, e.g., in game $G(\emptyset)$, Alice receives all operations in the first half of the sequence as her input, and Bob receives the second half, in game $G(0)$, Alice receives the first quarter and the second half, Bob receives the second quarter and the second half.

LEMMA 2. *For $p \geq B$ large enough, the communication game $G(s)$ requires communication cost at least $\Omega(2^{-|s|}KB(b - |s|))$ for any protocol with error $0.1$ in model $\mathcal{M}$, where $|s|$ is the length of string $s$.*

We will defer the proof of Lemma 2 to Section 4. Using these two lemmas, we are ready to prove our data structure lower bound.

PROOF OF THEOREM 2. Fix a (randomized) data structure for batch partial sum problem, which errors with probability no more than 0.05, and in expectation, probes $T$ cells on an operation sequence drawn from $\mathcal{D}$. By Markov's inequality and union bound, there is a way to fix the random bits used by the data structure, such that the error probability is no more than 0.1, and probes at most $3T$ cells in expectation. In the following, we show that for such deterministic data structure, $T$ must be large.

For binary string $s$ of length no more than $\log B$, define $P_s$ to be the (random) set of cells probed by the data structure in $I_s$. For every $s$, Lemma 1 and Lemma 2 together implies that

$$\mathop{\mathbb{E}}_{\mathcal{D}}\left[|P_{s0}| + |P_{s1}| + |P_{s0} \cap P_{s1}| \cdot w\right] \geq \Omega\left(2^{-|s|}KB(b - |s|)\right).$$

Now sum up the two sides over all binary strings $s$ of length at most $b - 1$. For the left-hand-side, fix an operation sequence. In the sum $\sum_s (|P_{s0}| + |P_{s1}|)$, every probe will be counted at most $\log B$ times, because the probes during $U_t$ or $Q_t$ will be counted only when $s$ is a prefix of $t$. In the sum $\sum_s |P_{s0} \cap P_{s1}|$, for each cell in $|P_{s0} \cap P_{s1}|$, we refer it to its first probe in $I_{s1}$. Every probed will be referred to at most once: consider a probe during $U_t$ or $Q_t$, assume the last probe to this cell happened during $U_{t'}$ or $Q_{t'}$ for some $t' < t$, this probe will be referred only when $s0$ is a prefix of $t'$ and $s1$ is a prefix $t$, i.e., $s$ is the longest common prefix of $t'$ and $t$. Therefore, the left-hand-side sums up to at most $3T \cdot (w + \log B)$. The right-hand-side sums up to

$$\sum_{s:|s|<b} \Omega(KB2^{-|s|}(b - |s|)) = \sum_{|s|=0}^{b-1} \Omega(KB(b - |s|)) = \Omega(KB \log^2 B)$$

This implies $T \geq \Omega(KB \log^2 B/(w + \log B))$, which proves the theorem. □

# 4. COMMUNICATION LOWER BOUND

Before proving the communication lower bound for the game $G(s)$ itself, we first do a "clean-up" to make the problem more accessible. In particular, we show that the communication game we want to prove a lower bound for is essentially the *multi-index* problem.

In the multi-index problem, Alice is given a vector $x \in \mathbb{F}_p^{LK}$. Bob is given an $L$-tuple of vectors $y = (y_1, y_2, \ldots, y_L)$, such that $y_i \in \mathbb{F}_p^{LK}$ for each $i \in [L]$. Moreover, if we divide the $LK$ coordinates into $K$ blocks of $L$ coordinates each in the most natural way (first block is the first $L$ coordinates, second block is the next $L$ coordinates, etc), each $y_i$ will be a $\{0, 1\}$-vector with *at most* one 1 in each block. Their goal is to compute the $L$ inner products over $\mathbb{F}_p$: $f(x, y) = (\langle x, y_1\rangle, \langle x, y_2\rangle, \ldots, \langle x, y_L\rangle)$. In other words, Alice gets an array, Bob gets $L$ sets of indices of the array (of some restricted form). They want to figure out together for each set, what is the sum of elements in the corresponding entries.

Fix a communication game $G(s)$ defined by $I_A = I_{s0}$ and $I_B = I_{s1}$. By the way we set up the hard distribution $\mathcal{D}$, every update operation will always update the set of entries, only the values change. Therefore, the only thing about the sequences Bob does not know is the values in the entries that are updated in $I_A$. As Bob knows the values in all other entries right before each query, the players' actual goal is to figure out the prefix sums as if there were only those unknown entries, which can be formulated as an instance of the multi-index problem. Moreover, input distribution $\mathcal{D}$ will induce an input distribution for the multi-index problem. We just need to prove a communication lower bound under that distribution.

More specifically, let $L = B/2^{|s|+1} = |I_A| = |I_B|$. Define the following function $F$ which maps a sequence of operations $(U_0, Q_0, \ldots, U_{B-1}, Q_{B-1})$, which is a possible outcome of $\mathcal{D}$, to an instance of the multi-index problem. Consider all updates in interval $I_A = I_{s0}$, let $\mathcal{E}_k$ be the set of entries of sequence $k$ which are updated in $I_A$. To get a multi-index instance, we set the $l$-th coordinate in $k$-th block of Alice's input $x$ to be the sum of values in $l$ first entries in $\mathcal{E}_k$ (the $l$ entries with smallest indices), for $l \in [L]$ and $k \in [K]$. For Bob's input $y_i$, consider the $i$-th query in $I_B$, let it be $\texttt{query}(j_{i,1}, \ldots, j_{i,K})$, querying the sum of first $j_{i,k}$ numbers in sequence $k$. For $k \in [K]$, assume there are $l$ entries in $\mathcal{E}_k$ with indices at most $j_{i,k}$, which will be summands in the $i$-th query. If $l = 0$, we set all coordinates in the $k$-th block of $y_i$ to 0, otherwise, we set the $l$-th coordinate in the block to 1. This defines the function $F$. It is not hard to see that the inner product $\langle x, y_i\rangle$ encodes the dependence of $i$-th query in $I_B$ on updates in $I_A$. Moreover, it is easy to verify that under the mapping of $F$, the distribution over operation sequences $\mathcal{D}$ induces a distribution over the input pairs $(x, y)$ for multi-index, with some probability measure $\mu$, such that

1. $x$ and $y$ are independent under $\mu$, i.e., $\mu = \mu_x \times \mu_y$ is a product distribution;

2. $\mu_x$ is the uniform distribution over $\mathbb{F}_p^{LK}$;

3. $\mu_y$ is close to being uniform: all $K$ blocks in all $L$ vectors in $y$ are independent, and in each block, each one of the $L+1$ possibilities will occur with probability no more than $1/L$, as adjacent elements in $\mathcal{E}_k$ are spaced by exactly $B/L-1$ numbers. In particular, $\mu_y(\{y\}) \le L^{-LK}$ for any singleton.

In the following, we will only use the above three properties of $\mu$ in the proofs. In this setting, Alice's input carries $O(LK \log p)$ bits of information. Bob's input carries $O(LK \log L)$ bits of information. The following lemma shows that the best strategy is just to let one of the players send the whole input to the other, even with Merlin's help in model $\mathcal{M}$.

LEMMA 3. *For large enough $p$ and $L$, solving the multi-index problem in model $\mathcal{M}$ with error $0.15$ requires communication cost $\Omega(\min\{LK \log p, LK \log L\})$ under input distribution $\mu$.*

Before proving this lemma, we first show that it implies Lemma 2.

PROOF OF LEMMA 2. Fix a protocol $P$ for $G(s)$ with error $0.1$ and cost $C$. We are going to use it to solve multi-index. Let us first assume that there is a sequence of public random bits that all three parties can see. We will first design a protocol in this setting, then try to get rid of this extra requirement by fixing the random bits.

For an input pair $(x, y) \sim \mu$ for the multi-index problem, consider the following protocol:

*Preprocessing.* Use the public randomness to sample an operation sequence from $\mathcal{D}$ conditioned on that $F$ maps it to $(x, y)$. It is easy to verify that all operations outside $I_B$ does not dependent on $y$ and all operations outside $I_A$ does not depend on $x$. Therefore, with no communication, all three parties get their inputs for $G(s)$.

*Simulate $P$.* Alice and Bob run protocol $P$ to compute all answers to queries in $I_B$.

*Postprocessing.* Bob knows all updates outside $I_A$, and from the value returned by the communication game, he gets to know the answers to all queries. Therefore, Bob can compute for $i$-th query in $I_B$, the sum of all entries updated in $I_A$ that are summands of the query, which is exactly $\langle x, y_i \rangle$, by subtracting all other summands from the answer. With no further communication, Bob can figure out the solution to the multi-index problem.

After Preprocessing, the inputs the players get for $G(s)$ will be distributed as $\mathcal{D}$. Therefore, in Simulate $P$, the communication cost will be $C$ in expectation, and error probability will be $0.1$ over the randomness of $P$, input $(x, y)$ and random bits $r$ used in Preprocessing. By Markov's inequality and union bound, there is a way to fix $r$, such that the communication cost is at most $4C$ in expectation and error probability is at most $0.15$ over the randomness of $P$ and $(x, y)$. To show that the protocol can be implemented in model $\mathcal{M}$, we hardwire $r$ and define for each input pair $(x, y)$, $Z(x, y)$ to be the message Merlin is supposed to send in Simulate $P$ when Preprocessing uses random bits $r$. Alice and Bob accept if and only if they were to accept in $P$.

It is easy to verify that the above protocol solves the multi-index problem under input distribution $\mu$ with error $0.15$ and cost $4C$. However, by Lemma 3, we have a lower bound of

$$\Omega(\min\{LK \log p, LK \log L\})$$

on the communication cost. Together with $p \ge B \ge L$, we have $C \ge \Omega(LK \log L) = \Omega(2^{-|s|} KB(b - |s|))$. This proves the lemma. □

To prove a communication lower bound in model $\mathcal{M}$, the main idea is to use the uniqueness of the certificate ($Z(x, y)$) and the perfect completeness and soundness. To start with, let us first consider the case where the protocol is deterministic, and the communication cost $C$ is defined in worst case.

In this case, fix one Merlin's possible message $z$, it defines a communication problem between Alice and Bob in the deterministic model: check whether $Z(x, y) = z$. By definition, the players can solve this task with zero error. By the classic monochromatic rectangle argument, we can partition the matrix $M(f)$ (defined in Section 2) into exponentially in $C$ many combinatorial rectangles, such that in each rectangle, either $Z(x, y) = z$ for every pair or $Z(x, y) \ne z$ for every pair. In particular, it partitions the set $Z^{-1}(z)$ into combinatorial rectangles. Moreover, for each rectangle with $Z(x, y) = z$, the protocol associates it with a value, which is the value returned in Stage 4. Now we go over all possible $z$'s, which is again exponentially in $C$ many. Every input pair belongs to exactly one of the $Z^{-1}(z)$'s. By cutting all $Z^{-1}(z)$, along with their partitioning into rectangles, and pasting into one single matrix, it induces a partitioning of the *whole* matrix $M(f)$ into $2^{O(C)}$ rectangles. The values associated with the rectangles should match the actual function values with high probability. Therefore, there must be large *nearly* monochromatic rectangles in $M(f)$.

A natural final step of the proof, as in many communication complexity lower bound proofs, would be to show that all nearly monochromatic rectangles are small. However, in the multi-index problem, there do exist large monochromatic rectangles.

Fix a set $S \subseteq [LK]$ of coordinates, such that it has $L/\log L$ coordinates in each block. Let $X = \{x : \forall j \in S, x_j = 0\}$, $Y = \{(y_1, \ldots, y_L) : \forall j \notin S, i \in [L], y_{i,j} = 0\}$. $X \times Y$ is a monochromatic rectangle with value $(0, \ldots, 0)$, and $\mu(X \times Y) = \Theta(p^{-LK/\log L} \cdot (\log L)^{-LK}) = \Theta(2^{-LK(\log p/\log L + \log\log L)})$. In particular, for $p = O(L)$, we can only prove lower bounds no better than $\Omega(LK \log\log L)$ using this approach only.

However, these $y$'s are not what a "typical" Bob's input should look like. Since the ones in $\{y_1, \ldots, y_L\}$ only appear in $(1/\log L)$-fraction of the coordinates, while a random input with very high probability should have ones appearing in a constant fraction of the coordinates. This motivates the following definition of *evenly-spreading*:

DEFINITION 5. *Define Bob's input to be* evenly-spreading, *if for any set of coordinates of size at most $0.1LK$, the number of ones in all $L$ vectors in Bob's input in these coordinates is no more than $0.9LK$.*

LEMMA 4. *For $L \cdot K$ large enough, with probability $\ge 95\%$, Bob's input is evenly-spreading.*

PROOF. Draw a random input $(x, y)$ from $\mu$, and try to upper-bound the probability that it is not evenly-spreading. Fix a set of coordinates $S = S_1 \cup S_2 \cup \cdots \cup S_K$ of size at most $0.1LK$, where $S_k$ is a subset of coordinates in block $k$. Let $\xi_{ki}$ be the random variable $\langle 1_{S_k}, y_i \rangle$, which is the number of ones in $y_i$ that is in $S_k$. By the properties of $\mu$, in $k$-th block of $y_i$, each of the $L+1$ possibilities occurs with probability no more than $1/L$, and is independent of all other blocks and vectors. Thus, we have that $\mathbb{E}[\xi_{ki}] \le |S_k|/L$, $\xi_{ki} \in [0, 1]$, and $\xi_{ki}$'s are independent. Let $\xi = \sum_{k=1}^{K} \sum_{i=1}^{L} \xi_{ki}$, be the number of ones in all $L$ vectors in $S$. We have $\mathbb{E}[\xi] \le 0.1LK$. Thus, by Hoeffding's inequality,

$$\Pr[\xi > 0.9LK] \le \Pr[\xi - E[\xi] > 0.8LK] \le e^{-32LK/25}.$$

However, the number of possible set $S$'s is no more than $2^{LK}$. By union bound, the probability that there exists an $S$ violating the

constraint is at most $e^{-32LK/25} \cdot 2^{LK} < 0.05$, for large enough $L \cdot K$. $\square$

Instead of upper-bounding the number of input pairs a nearly monochromatic rectangle can contain, we are going to upper-bound the measure of evenly-spreading inputs in it. Define set $E$ to be the all input pairs $(x, y)$ that $y$ is *not* evenly-spreading. By Lemma 4, $\mu(E) \leq 0.05$. The following lemma shows that there are no large nearly monochromatic rectangles if we ignore all elements in $E$.

LEMMA 5. *For large enough $p, L$, every 0.7-monochromatic rectangle $R$, which is disjoint from $E$, must have*

$$\mu(R) \leq \max\{2^{-\Omega(LK \log L)}, 2^{-\Omega(LK \log p)}\}.$$

PROOF. Fix a combinatorial rectangle $R = X \times Y$, such that there is a value $v = (v_1, \ldots, v_L)$ that $\mu(R \cap f^{-1}(v)) \geq 0.7\mu(R)$. We want to prove that $\mu(R)$ must be small. First, without loss of generality, we can assume that for every $y \in Y$,

$$\mu((X \times \{y\}) \cap f^{-1}(v)) \geq 0.6\mu(X \times \{y\}),$$

i.e., every column in $R$ is 0.6-monochromatic. Since in general, by Markov's inequality, at least $1/4$ (with respect to $\mu_y$) of the columns in $R$ are 0.6-monochromatic, we can just apply the following argument to the subrectangle induced by $X$ and these columns, and only lose a factor of 4.

Let $|Y| = q, Y = \{y_1, y_2, \ldots, y_q\}$, $y_i = (y_{i1}, \ldots, y_{iL})$. Let $r_i$ be the dimension of subspace in $\mathbb{F}_p^{LK}$ spanned by vectors in first $i$ $L$-tuples: $\{y_{jl} : 1 \leq j \leq i, l \in [L]\}$, and $r_0 = 0$. Let $r = r_q$ be the dimension of the subspace spanned by all vectors in $Y$. Define $\nu$ to be the probability distribution over $Y$ such that $\nu(y_i) = (r_i - r_{i-1})/r$. $\nu(y_i)$ is proportional to "the number of new dimensions $y_i$ introduces". Thus, $\nu$ is supported on no more than $r$ elements in $Y$.

Since every column in $Y$ is 0.6-monochromatic under $\mu_x$, $R$ will also be 0.6-monochromatic under $\mu_x \times \nu$. By Markov's inequality again, at least 1/5 of the rows are 0.5-monochromatic in $R$ under $\mu_x \times \nu$. However, when $r$ is large, there cannot be too many such rows, even in the whole matrix. For some 0.5-monochromatic row $x$, let $S \subseteq Y$ be the set of columns with value $v$ in that row. By definition, we have $\nu(S) \geq 0.5$. By the way we set up the distribution $\nu$, the linear space spanned by all vectors in $S$ must have dimension at least $r/2$. This adds at least $r/2$ independent linear constraints on $x$. There can be at most $p^{-r/2}$-fraction of $x$'s satisfying all linear constraints in the whole matrix. By taking a union bound on all possible $S$'s, we obtain an upper bound on the measure of 0.5-monochromatic rows in $R$ under $\mu_x \times \nu$. More formally, we have

$$\begin{aligned}
\frac{1}{5}\mu_x(X) &\leq \mu_x\left(\{x : \nu(\{y : f(x,y) = v\}) \geq 0.5\}\right) \\
&= \Pr_{x \sim \mu_x}\left[\Pr_{y=(y_1,\ldots,y_L)\sim\nu}[\forall l \in [L], \langle x, y_l\rangle = v_l] \geq 0.5\right] \\
&= \Pr_{x \sim \mu_x}[\exists S \subseteq \mathrm{supp}(\nu), \nu(S) \geq 0.5, \\
&\qquad \forall y \in S, \forall l \in [L], \langle x, y_l\rangle = v_l] \\
&\leq \sum_{\substack{S \subseteq \mathrm{supp}(\nu) \\ \nu(S) \geq 0.5}} \Pr_{x \sim \mu_x}[\forall y \in S, \forall l \in L, \langle x, y_l\rangle = v_l] \\
&\leq \sum_{\substack{S \subseteq \mathrm{supp}(\nu) \\ \nu(S) \geq 0.5}} p^{-r/2} \\
&\leq 2^r p^{-r/2} \leq 2^{-\Omega(r \log p)}.
\end{aligned}$$

Therefore, if $r \geq 0.1LK$, we have

$$\mu(R) \leq \mu_x(X) \leq 2^{-\Omega(r \log p)} \leq 2^{-\Omega(LK \log p)}.$$

Otherwise, $r \leq 0.1LK$. In this case, we are going to show that, it is impossible to pack too many vectors the the form of Bob's inputs into any subspace of small dimension. In particular, we will upper bound $\mu_y(Y)$, the measure of Bob's evenly-spreading inputs, when their span has dimension $r$. Fix a basis of the span of vectors in $Y$, consisting of $r$ vectors in $\mathbb{F}_p^{LK}$.[7] Without loss of generality, we can assume that for each basis vector, there is a coordinate in which this vector has value 1, and all other basis vectors have value 0, because we can always run a standard Gaussian elimination to transform the basis into this form. Let $T$ be this set of $r$ coordinates. As each vector in the subspace is a linear combination of the basis, fixing the values in coordinates in $T$ uniquely determines a vector in the subspace. By definition of evenly-spreading, and $|T| = r \leq 0.1LK$, each $L$-tuple $y_i \in Y$ can have at most $0.9LK$ 1's in $T$. For all $LK$ blocks in the $L$ vectors, there are $\binom{LK}{\leq 0.9LK}$ choices for the set of blocks with ones. Moreover, each $y_i$ can have at most one 1 in each block. If a vector has a 1 in a block, there will be at most $L$ different choices to place the 1. Otherwise, we must set all coordinates in $T$ in that block to be 0. After fixing values all coordinates in $T$, there can be at most one such vector in the subspace with matching values. Thus, we have

$$\begin{aligned}
\mu_y(Y) &\leq \binom{LK}{\leq 0.9LK} \cdot L^{0.9LK} \cdot L^{-LK} \\
&\leq L^{-0.1LK} \cdot 2^{LK} \leq 2^{-\Omega(LK \log L)}.
\end{aligned}$$

Therefore, in this case, we have $\mu(R) \leq \mu_y(Y) \leq 2^{-\Omega(LK \log L)}$.

Combining the two cases, we conclude that

$$\mu(R) \leq \max\{2^{-\Omega(LK \log L)}, 2^{-\Omega(LK \log p)}\}.$$

$\square$

Using the above lemma, we can prove a communication lower bound for multi-index.

PROOF OF LEMMA 3. Fix a protocol that solves the multi-index problem with error 0.15 and cost $C$ in model $\mathcal{M}$. As we are working with a fixed input distribution, randomness in the protocol shall not help. In particular, by Markov's inequality and union bound, there is a way to fix the internal (public) randomness of the protocol, such that the success probability is at least $80\%$ and communication cost no more than $5C$. From now on, let us assume the protocol is deterministic, and success probability is at least $80\%$.

For some message $z$ sent by Merlin, let $T_z = Z^{-1}(z)$ be set of the input pairs $(x, y)$ such that $z$ is message Merlin is supposed to send when the players get these input pairs. As Alice and Bob are always able to decide whether $z = Z(x, y)$, the classical combinatorial rectangle argument induces a way to partition each $T_z$ into rectangles based on the transcript between Alice and Bob. Moreover, the set $\{T_z\}_{z \in \{0,1\}^*}$ induces a partition of all possible input pairs. Therefore, provided that Merlin tells the truth, the entire transcript $\gamma(x, y)$, which includes both Merlin's message and the transcript between Alice and Bob, induces a partition of the matrix $M(f)$ into combinatorial rectangles $\{R_i\}$. For each $R_i$ with transcript $\gamma_i$, the players will return a fixed answer $v_i$ for every pair of inputs in the rectangle.

By the definition of communication cost, we have

$$\mathop{\mathbb{E}}_{(x,y)\sim\mu}[|\gamma(x, y)|] \leq 5C.$$

[7] These vectors do not have to be from Bob's inputs.

Thus, by Markov's inequality

$$\sum_{R_i:|\gamma_i|\le 50C} \mu\left(R_i \setminus E\right) \ge \sum_{R_i:|\gamma_i|\le 50C} \mu\left(R_i\right) - \mu(E) > 0.9 - 0.05 = 0.85 \quad (1)$$

By the definition of probability of computing $f$ correctly,

$$\sum_{R_i} \mu(R_i \cap f^{-1}(v_i)) \ge 0.8.$$

By Markov's inequality,

$$\begin{aligned}
&\sum_{\substack{R_i:\mu((R_i\setminus E)\cap f^{-1}(v_i)) \\ <0.7\mu(R_i\setminus E)}} \mu(R_i \setminus E) \\
&= \sum_{\substack{R_i:\mu(R_i\setminus E\setminus f^{-1}(v_i)) \\ >0.3\mu(R_i\setminus E)}} \mu(R_i \setminus E) \\
&\le \frac{1}{0.3} \sum_{\substack{R_i:\mu(R_i\setminus E\setminus f^{-1}(v_i)) \\ >0.3\mu(R_i\setminus E)}} \mu(R_i \setminus E \setminus f^{-1}(v_i)) \\
&\le \frac{1}{0.3} \sum_{R_i} \mu(R_i \setminus f^{-1}(v_i)) \\
&= \frac{1}{0.3}\left(1 - \sum_{R_i} \mu(R_i \cap f^{-1}(v_i))\right) \\
&\le 2/3
\end{aligned}$$

Thus, we have

$$\sum_{\substack{R_i:\mu((R_i\setminus E)\cap f^{-1}(v_i)) \\ \ge 0.7\mu(R_i\setminus E)}} \mu(R_i \setminus E) \ge 1 - \mu(E) - 2/3 > 0.25 \quad (2)$$

Let $\mathcal{R}$ be the (disjoint) union of all $R_i$ with value $v_i$ and transcript $\gamma_i$, such that

$$\mu((R_i \setminus E) \cap f^{-1}(v_i)) \ge 0.7\mu(R_i \setminus E)$$

and $|\gamma_i| \le 50C$. By (1) and (2), we have $\mu(\mathcal{R} \setminus E) \ge 0.1$. However, there can be only $2^{O(C)}$ different such $\gamma_i$'s, and thus $2^{O(C)}$ such rectangles. There must be some transcript $\gamma_i$ such that $\mu(R_i \setminus E) \ge 2^{-O(C)}$ and $R_i \setminus E$ is 0.7-monochromatic rectangle under distribution $\mu$. Therefore, by Lemma 5, we have

$$C \ge \Omega(\min\{LK \log L, LK \log p\}),$$

which proves the lemma. □

# 5. APPLICATIONS TO DYNAMIC GRAPH PROBLEMS

In this section, we present three applications of our main theorem to dynamic graph problems. See Appendix B for formal definitions of the problems.

COROLLARY 2. *For the following three dynamic graph problems:*

*(a) dynamic #SCC,*

*(b) dynamic planar s-t min-cost flow,*

*(c) dynamic weight s-t shortest path,*

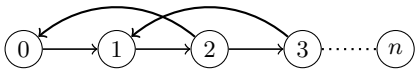


**Figure 1: dynamic #SCC**

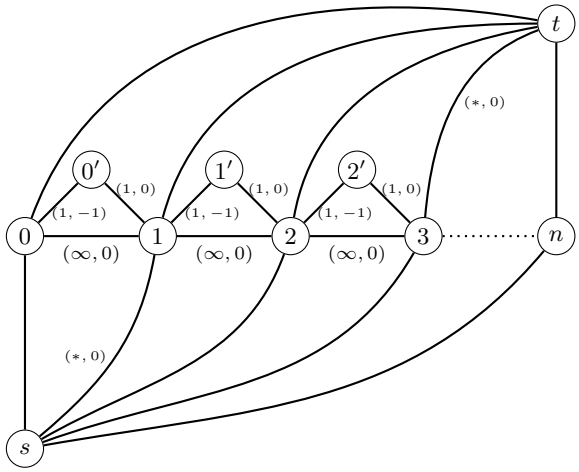


**Figure 2: dynamic planar s-t min-cost flow**

*any data structure with amortized expected update time $o(\log n)$, and error probability $\le 5\%$ under polynomially many operations must have amortized expected query time $n^{1-o(1)}$.*

PROOF. (sketch) To prove the lower bounds, we are going to give three reductions from dynamic interval union. The corollary follows from Corollary 1.

(a) To solve dynamic interval union, we maintain the following graph $G$: $G$ has a Hamiltonian path $0 \to 1 \to \cdots \to n$; for every $[a, b] \in \mathcal{I}$, $G$ has an edge $b \to a$. It is not hard to see that the graph can be maintained efficiently given a dynamic #SCC data structure, and the total length of the union of $\mathcal{I}$ is exactly $n + 1$ minus the number of strongly connected components in $G$. See Figure 1.

(b) The underlying graph is shown as Figure 2. The edges connecting vertices $i$ and $i + 1$ have infinite capacities and zero cost. The edges connect $i$ and $i'$ have capacities 1 and cost -1. The edges connecting $i'$ and $i + 1$ have capacities 1 and cost 0. All edges connecting to $s$ or $t$ have cost 0. The only values that may change are the capacities of edges connecting to $s$ or $t$. More specifically, we maintain the graph such that the capacity from $s$ to vertex $i$ always equals to the number of intervals in $\mathcal{I}$ with left endpoint $i$, the capacity from $i$ to $t$ always equals to the number of intervals with right endpoint $i$. It is easy to verify that given a dynamic planar s-t min-cost flow data structure, we can efficiently maintain these invariants. To query the total length of the union of $\mathcal{I}$, we query the min-cost flow in $G$ with flow value $|\mathcal{I}|$. For each $i$, the amount of flow from $i$ to $i + 1$ (also counting flow going through $i'$) is exactly the number of intervals containing $[i, i + 1]$. As $i \to i' \to i + 1$ has a smaller cost than going to $i + 1$ directly from $i$, min-cost flow will prefer to use the former path. Each $[i, i + 1]$ contained in any intervals in $\mathcal{I}$ adds a cost of $-1$ to the flow. Therefore, the negate of the cost is exactly the length of the union.

(c) We maintain a graph $G$ such that there is an edge from $s$ to 0 with weight 0, an edge from $n$ to $t$ with weight 0, edges from $i$

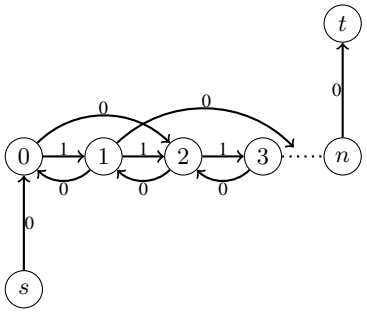


**Figure 3: dynamic weight s-t shortest path**

to $i+1$ with weight 1, and edges from $i+1$ to $i$ with weight 0. Moreover, for each interval $[a, b] \in \mathcal{I}$, the graph has an edge $a \to b$ with weight 0. The shortest path from $s$ to $t$ is exactly $n$ minus the length of the union, because for $[i, i+1]$ contained in any interval in $\mathcal{I}$, we can go from $i$ to $i+1$ with zero cost: go to the left endpoint of the interval, then go to the right endpoint using one edge, and go to $i+1$. See Figure 3.

□

# 6. FINAL REMARKS

In [6], Clifford, Grønlund and Larsen mentioned a $\log m \log n$ barrier for dynamic data structure lower bounds, where $m$ is the number of different queries (including parameters), and $n$ is the number operations in the sequence we analyse. In some sense, our main result can also be viewed as a "$(\log m \log n)$-type" lower bound, as the query takes only $O(1)$ bits to describe. The way we prove this type of lower bound is very different from [16, 17, 6]. We obtain this kind of the lower bound via reduction. Assume we want to prove a lower bound for problem $A$. We first prove a lower bound for some other problem $B$ with large $m$, and get a high lower bound. Then we find a way to implement updates of $B$ using updates of $A$, and queries of $B$ using updates and queries of $A$, and thus derive a lower bound for $A$. Note that it is important that we implement queries in problem $B$ using *both* updates and queries in the original problem. Because if we only use queries to implement queries, in order to keep all the information in the query of $B$, it has to be decomposed into many queries of $A$. A simple calculation shows that in this case, we cannot break the barrier for problem $A$ unless we have already broken it for problem $B$. However, if we use both updates and queries of $A$, it is possible to "hide" information in the updates. A good example is the reduction in Proposition 1 in Appendix A. However, using this approach, we still cannot beat $\log m' \log n$, where $m'$ is the number of different *updates* (including parameters). Nevertheless, it gives us a potential way to break the $\log m \log n$ barrier for problems with $m' > m$ if we can combine it with the previous techniques.

Pătraşcu has used the communication lower bound for lopsided set disjointness, set disjointness problem of a special form, to prove a collection of (static) data structure lower bounds [19, 20]. In this paper, we applied communication *protocol (upper bound)* for sparse set disjointness, set disjointness of a different special form, to prove a dynamic data structure *lower bound*. In some sense, this can be viewed as an analogue of the recent development in duality between algorithm and complexity [25, 26, 2] in the communication complexity and data structure world. It would be interesting to see examples where both communication lower bound and upper bound for the exact same problem can be used to prove data structure lower bounds.

On Klee's measure problem, our result is an unconditional lower bound for one certain type of algorithms. From Theorem 1, we can generate a hard input distribution for the sweep-line algorithm, such that if the algorithm only sorts the rectangles, goes through the entire area row by row and computes the number of grids in the union only based on the rectangles intersecting the current row or previous rows, then it cannot beat Bentley's algorithm. However, our hard distribution is not very robust, in the sense that if we do the sweep-line from a different direct, the distribution over inputs becomes really easy. At least, it still shows us what an $o(N \log N)$ time algorithm for computing 2D Klee's measure problem in range $[0, N^{2/3}] \times [0, N^{2/3}]$ should *not* look like, if it existed.

# 7. ACKNOWLEDGEMENTS

The author would like to thank Yuqing Ai and Jian Li for introducing Klee's measure problem to me during a discussion, and would like to thank Timothy Chan for telling me the state-of-the-art.

The author also wishes to thank Ryan Williams for helpful discussions on applications to dynamic graph problems and in paper-writing.

# 8. REFERENCES

[1] 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.

[2] Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 218–230, 2015.

[3] 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.

[4] Jon L. Bentley. Algorithms for Klee's rectangle problems. *Unpublished manuscript*, 1977.

[5] Timothy M. Chan. Klee's measure problem made easy. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*, pages 410–419, 2013.

[6] Raphaël Clifford, Allan Grønlund, and Kasper Green Larsen. New unconditional hardness results for dynamic and online problems. In *56rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2015*, 2015.

[7] Raphaël Clifford and Markus Jalsenius. Lower bounds for online integer multiplication and convolution in the cell-probe model. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011*, pages 593–604, 2011.

[8] Raphaël Clifford, Markus Jalsenius, and Benjamin Sach. Tight cell-probe bounds for online hamming distance computation. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013*, pages 664–674, 2013.

[9] Raphaël Clifford, Markus Jalsenius, and Benjamin Sach. Cell-probe bounds for online edit distance and other pattern matching problems. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 552–561, 2015.

[10] 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.

[11] Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015*, pages 1077–1088, 2015.

[12] Johan Håstad and Avi Wigderson. The randomized communication complexity of set disjointness. *Theory of Computing*, 3(1):211–219, 2007.

[13] Haim Kaplan, Or Zamir, and Uri Zwick. The amortized cost of finding the minimum. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 757–768, 2015.

[14] Victor Klee. Can the measure of $\cup_1^n [a_i, b_i]$ be computed in less than $O(n \log n)$ steps? *The American Mathematical Monthly*, 84(4):284–285, 1977.

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

[16] 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.

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

[18] 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.

[19] Mihai Pătraşcu. (Data) STRUCTURES. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008*, pages 434–443, 2008.

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

[21] Mihai Pătraşcu and Erik D. Demaine. Lower bounds for dynamic connectivity. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 546–553, 2004.

[22] 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.

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

[24] 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.

[25] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013.

[26] Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Symposium on Theory of Computing, STOC 2014*, pages 664–673, 2014.

[27] Mihalis Yannakakis. Expressing combinatorial optimization problems by linear programs. *J. Comput. Syst. Sci.*, 43(3):441–466, 1991.

[28] 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.

[29] 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.

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

# APPENDIX

## A. REDUCTION FROM PARTIAL SUM

The partial sum problem is to maintain a sequence of $n$ numbers $(A_i)$ over $[n]$, supporting the following operations:

- `update(i, v)`: set $A_i$ to value $v$;
- `query(l)`: return $\sum_{i \le l} A_i$.

Pătraşcu and Demaine [22] showed that at least one of the operations needs to take $\Omega(\log n)$ time in the cell-probe model with word size $w = \Theta(\log n)$.

PROPOSITION 1. *Any data structure for the dynamic interval union problem with insertion time $t_i$, deletion time $t_d$ and query time $t_q$ must have $\max\{t_i, t_d, t_q\} \geq \Omega(\log n)$ in the cell-probe model with word size $w = \Theta(\log n)$.*

PROOF. Consider the partial sum problem with $\sqrt{n}$ numbers over $[\sqrt{n}]$. Any data structure will take $\Omega(\log n)$ time to update or query. Fix a dynamic interval union data structure over $[0, n]$, we will use it to solve the partial sum problem. First partition $[0, n]$ into $\sqrt{n}$ blocks of length $\sqrt{n}$ each. The $i$-th block will correspond to number $A_i$. We maintain the invariant that for each number $A_i$, there is an interval in the corresponding block of length equal to the value of $A_i$.

More specifically, every time we need to update $A_i$ from value $v'$ to $v$, we delete the previous interval $[(i-1)\sqrt{n}, (i-1)\sqrt{n}+v']$, then insert a new interval $[(i-1)\sqrt{n}, (i-1)\sqrt{n}+v]$. When we need to query the sum of first $l$ numbers, we first insert an interval $[l\sqrt{n}, n]$, which covers all blocks from the $(l+1)$-th to the last, then query the length of the union, delete the interval inserted earlier. We know that the temporarily inserted interval has length $(\sqrt{n}-l)\sqrt{n}$. Subtracting it from the answer returned, we get the total length of intervals in the first $l$ blocks, which is exactly the sum of first $l$ numbers.

Every update of partial sum can be implemented using an insertion and a deletion of dynamic interval union, every query can be implemented using an insertion, a deletion and a query. Therefore, at least one of the operations has to take $\Omega(\log n)$ time. □

## B. CATALOGUE OF DYNAMIC PROBLEMS IN OUR APPLICATION

The *dynamic number of strongly connected components* problem is to maintain a directed graph $G$, supporting:

- `insert(u, v)`: insert an edge $(u, v)$;
- `delete(u, v)`: delete an (existing) edge $(u, v)$;
- `query()`: return the number of strongly connected components in $G$.

The *dynamic planar s-t min-cost flow* problem is to maintain an undirected planar flow network $G$ with edge cost, supporting:

- `update(u, v, cap)`: update the capacity of (an existing) edge $(u, v)$ to `cap`;

- `query(f)`: return the min-cost flow from a fixed source $s$ to a fixed sink $t$ with flow value $f$.

The *dynamic weighted s-t shortest path* problem is to maintain a weighted directed graph $G$, supporting:

- `insert(u, v, w)`: insert an edge $(u, v)$ with weight $w$;
- `delete(u, v)`: delete an (existing) edge $(u, v)$;
- `query()`: return the shortest path from a fixed source $s$ to a fixed target $t$.