Tight Dynamic Problem Lower Bounds from Generalized BMM and OMv*

Ce Jin†
cejin@mit.edu
MIT
Cambridge, MA, USA

Yinzhan Xu‡
xyzhan@mit.edu
MIT
Cambridge, MA, USA

ABSTRACT

Popular fine-grained hypotheses have been successful in proving conditional lower bounds for many dynamic problems. Two of the most widely applicable hypotheses in this context are the *combinatorial Boolean Matrix Multiplication (BMM) hypothesis* and the closely-related *Online Matrix Vector Multiplication (OMv) hypothesis*. The main theme of this paper is using k-dimensional generalizations of these two hypotheses to prove new *tight* conditional lower bounds for dynamic problems.

The *combinatorial k-Clique hypothesis*, which is a standard hypothesis in the literature, naturally generalizes the combinatorial BMM hypothesis. In this paper, we prove tight lower bounds for several dynamic problems under the combinatorial k-Clique hypothesis. For instance, we show that the *Dynamic Range Mode* problem has no combinatorial algorithms with $\text{poly}(n)$ pre-processing time, $O(n^{2/3-\varepsilon})$ update time and $O(n^{2/3-\varepsilon})$ query time for any $\varepsilon > 0$, matching the known upper bounds for this problem. Previous lower bounds only ruled out algorithms with $O(n^{1/2-\varepsilon})$ update and query time under the OMv hypothesis. We also show that the *Dynamic Subgraph Connectivity* problem on undirected graphs with m edges has no combinatorial algorithms with $\text{poly}(m)$ pre-processing time, $O(m^{2/3-\varepsilon})$ update time and $O(m^{1-\varepsilon})$ query time for $\varepsilon > 0$, matching the upper bound given by Chan, Pătraşcu, and Roditty [SICOMP'11], and improving the previous update time lower bound (based on OMv) with exponent 1/2. Other examples include tight combinatorial lower bounds for *Dynamic 2D Orthogonal Range Color Counting*, *Dynamic 2-Pattern Document Retrieval*, and *Dynamic Range Mode* in higher dimensions.

Furthermore, we propose the OuMv_k hypothesis as a natural generalization of the OMv hypothesis. Under this hypothesis, we prove tight lower bounds for various dynamic problems. For instance, we show that the *Dynamic Skyline Points Counting* problem in $(2k-1)$-dimensional space has no algorithm with $\text{poly}(n)$ pre-processing time and $O(n^{1-1/k-\varepsilon})$ update and query time for $\varepsilon > 0$, even if the updates are semi-online. Other examples include tight conditional lower bounds for (semi-online) Dynamic Klee's measure for unit cubes, and high-dimensional generalizations of Erickson's problem and Langerman's problem.

CCS CONCEPTS

• **Theory of computation → Data structures design and analysis**.

KEYWORDS

fine-grained complexity, dynamic data structures

ACM Reference Format:
Ce Jin and Yinzhan Xu. 2022. Tight Dynamic Problem Lower Bounds from Generalized BMM and OMv. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC '22), June 20–24, 2022, Rome, Italy.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3519935.3520036

1 INTRODUCTION

In dynamic (data structure) problems, we need to maintain some data D (e.g., graphs, sequences, geometric objects) that undergoes small updates, and to support querying $f(D)$ for some function f. Such problems are motivated by practical scenarios where we want to maintain large data sets that are constantly changing, such as social network graphs, large collaborative documents, or real-time flight trackers. A large body of work in theoretical computer science has been devoted to designing efficient data structures to solve dynamic problems. These data structures are not only useful on their own, but also turn out to have applications in solving static problems in many areas of computer science, e.g., computational geometry [87], optimization [38], and graph theory [20, 52].

Some dynamic problems have efficient data structures that only require sub-polynomial time for each update and query. One such example is the Graph Connectivity problem, where we need to maintain an undirected graph under edge insertions and deletions, and support querying whether two vertices are connected [35, 51, 58, 59, 64, 81, 88, 95]. However, many other dynamic problems only have way slower data structures that run in polynomial time in the data size. For instance, if we change the graph in the Graph Connectivity problem from undirected to directed (known as the Dynamic Reachability problem), the current best data structure runs in $O(n^{1.407})$ time per update or query [86, 90]. It is thus natural to seek lower bounds for such problems. Unfortunately, proving unconditional super poly-logarithmic data structure lower bounds is beyond the reach of current techniques [36].

People have thus tried to prove conditional lower bounds for dynamic problems. An important tool for proving conditional lower bounds is *fine-grained complexity* (see [91] for a survey), which uses fine-grained reductions to prove conditional lower bounds for various computational problems under some hypotheses. There has

*The full version of this paper is available at https://arxiv.org/abs/2202.11250.
†Supported by NSF Grant CCF-2129139.
‡Supported by NSF Grant CCF-2129139.

STOC '22, June 20–24, 2022, Rome, Italy

ACM ISBN 978-1-4503-9264-8/22/06.
https://doi.org/10.1145/3519935.3520036

been a great success in proving dynamic problem lower bounds under various popular hypotheses, including the 3SUM hypothesis [4, 5, 39, 66, 80], the APSP hypothesis [2, 4, 5, 39, 50, 84, 93], the Strong Exponential Time Hypothesis (SETH) [4, 5, 7, 39], the combinatorial Boolean Matrix Multiplication (BMM) hypothesis [4, 37, 84] and the Online Matrix Vector Multiplication (OMv) hypothesis [13, 37, 39, 50, 57, 69].

The combinatorial BMM hypothesis and the closely-related OMv hypothesis are two versatile hypotheses that have been used in proving conditional lower bounds for various dynamic problems.

In the BMM problem, one is asked to compute the product of two given $n \times n$ matrices over the Boolean semiring $\{0, 1\}$. We could of course use any fast matrix multiplication algorithm to solve BMM in $O(n^\omega)$ time, where $\omega < 2.37286$ [6] denotes the square matrix multiplication exponent. However, fast matrix multiplication algorithms use "Strassen-like" techniques (see e.g. [9]) that do not perform well in practice. This has motivated the study of "combinatorial" algorithms for BMM that do not use any heavy algebraic techniques, in the hope of getting a both theoretically and practically fast algorithm for BMM. Unfortunately, despite considerable amount of efforts [8, 10, 25, 96], the current fastest combinatorial BMM algorithm runs in $n^3(\log\log n)^{O(1)}/(\log n)^4$ time [96], still not gaining any polynomial speed-up over the brute-force $O(n^3)$ time algorithm. Therefore, the *combinatorial BMM hypothesis*, which states that no combinatorial algorithm for BMM can run in $O(n^{3-\varepsilon})$ time for $\varepsilon > 0$, is popular in fine-grained complexity[1].

Historically, the combinatorial BMM hypothesis was first used to prove conditional lower bounds for static problems. Lee [70] first used the combinatorial BMM hypothesis to show a lower bound for Context Free Grammar Parsing. Following this work, BMM-based conditional lower bounds have found a wide range of applications, including Colored Orthogonal Range Counting [63], Range Mode [27], 2-Pattern Document Retrieval [67], 6-Cycle Detection in undirected graphs [40], and many more, e.g., [32, 44, 92]. The combinatorial BMM hypothesis is also widely used in the context of dynamic problems. For instance, under the combinatorial BMM hypothesis, Roditty and Zwick [84] showed the hardness of partially dynamic unweighted Single-Source Shortest Paths problems, Abboud and Vassilevska Williams [4] showed the hardness of a variety of dynamic graph problems and some dynamic set problems, and Clifford, Grønlund, Larsen, and Starikovskaya [37] showed the hardness of some string problems.

Another related hypothesis that has been successful in proving dynamic problem lower bounds is the OMv hypothesis. In the OMv problem, we first pre-process an $n \times n$ Boolean matrix M, and then receive multiple length-n Boolean vectors arriving one by one, and we are asked to compute the Boolean product Mv for each received vector v in an online fashion. The *OMv hypothesis*, proposed by Henzinger, Krinninger, Nanongkai, and Saranurak [57], states that there is no algorithm for OMv with $\text{poly}(n)$ pre-processing time and $O(n^{1-\varepsilon})$ query time for $\varepsilon > 0$.[2] The OMv problem can be viewed as an online version of BMM, and it was proposed in order to remove the "combinatorial" notion in the combinatorial BMM hypothesis. Another key advantage of OMv-based lower bounds for dynamic problems is that they hold even when the algorithms are allowed to have arbitrary polynomial pre-processing time [57]. Prior to [57], this type of results were only seen in some SETH-based lower bounds in [4].

People have established a wide range of hardness results based on the OMv hypothesis. Henzinger, Krinninger, Nanongkai, and Saranurak [57] showed over 15 tight hardness results under the OMv hypothesis, including many dynamic graph problems, Erickson's problem, Pagh's problem, and the Multiphase problem. Following this work, the OMv hypothesis has been applied to more problems, such as database query problems [13], dynamic string problems [37], 2D range query problems [69], and Dynamic Longest Increasing Subsequence [50].

In this paper, we study natural high-dimensional generalizations of the BMM hypothesis and the OMv hypothesis, and show *tight* conditional lower bounds for a wide range of dynamic problems under these hypotheses.

Combinatorial k-Clique hypothesis. It is known that via combinatorial reductions, BMM is subcubically equivalent to the Triangle Detection problem, which asks to determine whether an n-node graph contains a triangle [92]. Therefore, the combinatorial BMM hypothesis is equivalent to the hypothesis stating that there is no truly subcubic combinatorial algorithm for Triangle Detection.

The natural generalization of Triangle Detection is k-Clique Detection, which asks to determine whether an n-node graph contains a k-clique, for any constant $k \geq 3$. Although the current fastest algorithm for k-Clique Detection runs in $O(n^{\omega(\lfloor k/3\rfloor,\lceil k/3\rceil,\lceil (k-1)/3\rceil)})$ time [45, 60, 76], where $\omega(a, b, c)$ denotes the exponent for multiplying an $n^a \times n^b$ matrix with an $n^b \times n^c$ matrix, the algorithm heavily relies on fast matrix multiplication, and is thus not efficient in practice. If we restrict the algorithm to be combinatorial, then there is currently no combinatorial algorithm for k-Clique that runs polynomially faster than $O(n^k)$-time brute-force, for any constant k. Therefore, the following combinatorial k-Clique hypothesis is a popular natural generalization of the combinatorial BMM hypothesis.

Hypothesis 1.1 (Combinatorial k-Clique Hypothesis)**.** *There is no $O(n^{k-\varepsilon})$ time* combinatorial *algorithm for k-Clique Detection on n-vertex graphs, for any $\varepsilon > 0$.*

We remark that it is not new to study k-Clique Detection in the context of fine-grained complexity (e.g. some previous works include [1, 3, 16, 17, 23, 33, 56, 71]). For instance, Chan [23] reduced k-Clique Detection to the k-dimensional Klee's measure problem, showing a matching combinatorial lower bound for the latter problem; Bringmann, Grønlund, and Larsen [16] reduced k-Clique Detection to the Word Break problem; Abboud, Backurs, Vassilevska Williams [1] reduced k-Clique Detection to Context-Free Grammar Parsing. Nonetheless, previous applications of k-Clique Detection to dynamic problems are much rarer. To the best of our knowledge, the only known example is a result by Gutenberg, Vassilevska Williams, and Wein [56], who reduced 4-Clique Detection to partially dynamic Single Source Shortest Path.

[1] Throughout this work, we consider the word-RAM model of computation with $O(\log n)$-bit words.

[2] The original version of the OMv hypothesis is defined for the OMv problem with n queries, but it was shown to be equivalent to the version with an arbitrary polynomial number of queries [57].

OuMv$_k$ hypothesis. A problem that is often used as an intermediate step in showing OMv-based lower bounds is the OuMv problem. In OuMv, we need to first pre-process an $n \times n$ Boolean matrix M. Then for each pair of length n Boolean vectors u, v that arrive in an online fashion, we need to compute $u^T Mv$. The OuMv hypothesis states that there is no algorithm for OuMv with polynomial pre-processing time and $O(n^{2-\varepsilon})$ query time for $\varepsilon > 0$. It was shown that the OMv hypothesis is equivalent to the OuMv hypothesis [57].

We propose the following OuMv$_k$ problem for any constant integer $k \geq 2$, which is a natural high-dimensional generalization of the OuMv problem (OuMv is equivalent to OuMv$_2$).

Definition 1.2 (OuMv$_k$ Problem)**.** During pre-processing we are given a subset $M \subseteq [n]^k$.[3] Then we receive online queries each specifying k sets $U^{(1)}, U^{(2)}, \ldots, U^{(k)} \subseteq [n]$, and we need to answer whether $U^{(1)} \times U^{(2)} \times \cdots \times U^{(k)}$ has a non-empty intersection with M.

Clearly, we can handle each OuMv$_k$ query in $O(n^k)$ time, by explicitly computing $U^{(1)} \times U^{(2)} \times \cdots \times U^{(k)}$ and then comparing it with M. We propose the following OuMv$_k$ hypothesis which states that the $O(n^k)$ time brute-force algorithm is essentially the best.

Hypothesis 1.3 (OuMv$_k$ Hypothesis)**.** *There is no algorithm for the OuMv$_k$ problem with n queries in $O(n^{1+k-\varepsilon})$ total time (pre-processing time plus total query time) for any $\varepsilon > 0$.*

Using techniques similar to [57], we can show that the following hypothesis is equivalent. We include a proof in the full version.

Hypothesis 1.4. *There is no algorithm for the OuMv$_k$ problem with* poly(n) *pre-processing time and $O(n^{\gamma+k-\varepsilon})$ total query time for n^γ queries, for any $\gamma, \varepsilon > 0$.*

In the following, we explain why we believe that the OuMv$_k$ Hypothesis is plausible. The OuMv$_k$ problem can be viewed as a variant of the $(k+1)$-Clique Detection problem in $(k+1)$-partite graphs. Imagine we have k vertex parts $V_1, \ldots, V_k$ each of size n, and we add a hyperedge among $(v_1, \ldots, v_k)$ if and only if $(v_1, \ldots, v_k) \in M$. Each OuMv$_k$ query represents a vertex u in a $(k+1)$-th vertex part, and for each $i \in [k]$, we connect u with $v_i \in V_i$ if and only if $v_i \in U^{(i)}$. Clearly, the answer to the OuMv$_k$ query is YES if and only if u is in a "$(k+1)$-clique" with some vertices $v_1, \ldots, v_k$, where there is a hyperedge among $(v_1, \ldots, v_k)$ and there is an edge between u and v_i for every $i \in [k]$. Since hyperedges are more powerful than edges, and online vertices are harder than static vertices, the OuMv$_k$ problem is clearly harder than $(k+1)$-Clique Detection.

All known algorithms [45, 60, 76] for $(k+1)$-Clique Detection that are polynomially faster than brute-force use the following idea: grouping the vertex parts to three groups, reducing $(k+1)$-Clique Detection to Triangle Detection where each group corresponds to one vertex part in the Triangle Detection instance, and finally using fast (rectangular) matrix multiplication to solve the Triangle Detection instance. If we try to apply this idea to OuMv$_k$, we have to assign $V_1, \ldots, V_k$ to at most 2 groups, since otherwise there is no way to encode the hyperedges. This leaves V_{k+1} to its own group.

[3] We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$.

Recall the vertices in V_{k+1} arrive in an online fashion. Thus, we have essentially reduced OuMv$_k$ to a Triangle Detection instance in a tripartite graph, where vertices in one vertex part arrive in an online fashion. It can be further viewed as a possibly rectangular instance of OuMv, which is known to be equivalent to OuMv [57].

Therefore, to solve OuMv$_k$ polynomially faster than the $O(n^k)$ time per query brute-force algorithm, we either need a new algorithm for $(k+1)$-Clique Detection that is drastically different from all previous algorithms, or a polynomially faster algorithm for OuMv. Thus, it is natural to consider the OuMv$_k$ hypothesis.

Gutenberg, Vassilevska Williams, and Wein [56] studied another generalization of the OuMv hypothesis, the OMv3 hypothesis, which was used to show conditional lower bound for partially dynamic Single Source Shortest Paths. In contrast to our OuMv$_3$ problem defined on 3-dimensional tensors, their OMv3 problem is defined on matrices, and admits speedup via fast matrix multiplication. Hence, their hypothesis is based on an easier problem with a lower hypothesized running time exponent, and is not directly comparable to our OuMv$_3$ hypothesis.

1.1 Our Contributions

Tight combinatorial lower bounds based on the k-Clique hypothesis. We show tight combinatorial lower bounds for dynamic problems such as Dynamic Range Mode, Dynamic Subgraph Connectivity, and Dynamic 2D Orthogonal Range Color Counting. These tight lower bounds are not known to be possible under either the BMM hypothesis or the OMv hypothesis. Moreover, all these lower bounds hold even if the data structures are allowed to use arbitrary polynomial pre-processing time. Interestingly, the static variants of many problems we study had tight combinatorial lower bounds based on the BMM hypothesis, such as Range Mode [27] and 2D Orthogonal Range Color Counting [63]; we in turn design tight combinatorial lower bounds for their dynamic variants under the combinatorial 4-Clique hypothesis. This identifies an intriguing pattern that relates k-Clique-based lower bound for a static problem and $(k+1)$-Clique-based lower bound for its dynamic variant. We believe this pattern could potentially be useful for designing combinatorial clique-based lower bounds for many other dynamic problems in future works.

We also show that many previous BMM-based lower bounds for dynamic problems [4] can be easily strengthened to arbitrary polynomial pre-processing time under the combinatorial 4-Clique hypothesis, without lowering the combinatorial lower bounds on update or query time.

Tight lower bounds based on the OuMv$_k$ hypothesis. There are many problems that are parameterized by some constant integer parameters and become much harder when the integer parameters increase. For instance, such parameters could be the dimension for computational geometry problems or tensor problems, or edge cardinality for hypergraph problems. It is thus natural to seek conditional lower bounds parameterized by such integer parameters. However, it is unclear how to use the OMv hypothesis to explain the increased difficulties of this type of problems when the parameters increase. Using our proposed OuMv$_k$ hypothesis, we are able to show increasing conditional lower bounds for such problems when their parameters increase. Such examples include Dynamic

Skyline Points Counting, Dynamic Klee's measure for unit hypercubes, high-dimensional Erickson's problem and high-dimensional Langerman's problem. We believe the OuMv$_k$ hypothesis could potentially have further applications in proving dynamic problem lower bounds.

1.1.1 Lower Bounds based on k-Clique.

Dynamic Range Mode. Given an integer sequence $a_1, a_2, \ldots, a_n$, a range mode query l, r asks to report the integer that appears most frequently (breaking ties arbitrarily) among $a_l, a_{l+1}, \ldots, a_r$. In the Dynamic Range Mode problem, we need to maintain an integer sequence that undergoes insertions and deletions, and support range mode queries. In a certain batched version of static range mode, Batch Range Mode [27], we are given a length n sequence and n range mode queries, and need to answer these n queries all at once.

The time complexity of combinatorial algorithms for Batch Range Mode is quite well-understood. It is known that we can solve Batch Range Mode in $\tilde{O}(n^{1.5})$ time[4] by combinatorial algorithms, and any polynomially faster ($O(n^{1.5-\varepsilon})$ time for $\varepsilon > 0$) combinatorial algorithm will contradict the combinatorial BMM hypothesis [27]. Faster algorithms are known for Batch Range Mode if fast matrix multiplication is allowed [53, 94].

In contrast, time complexity of combinatorial algorithms for Dynamic Range Mode was much less understood. There are in fact multiple combinatorial algorithms for Dynamic Range Mode with $\tilde{O}(n^{2/3})$ update and query times [27, 46], and the $\tilde{O}(n^{2/3})$ bound was a seeming barrier faced by these algorithms. On the lower bound side, the combinatorial $n^{1.5-o(1)}$ lower bound of Batch Range Mode under the combinatorial BMM hypothesis [27] can be easily adapted to show an $n^{0.5-o(1)}$ per update and query (non-combinatorial) lower bound under the OMv hypothesis, which has a big gap from the $\tilde{O}(n^{2/3})$ upper bound. People have used fast matrix multiplication to design $O(n^{2/3-\varepsilon})$ time (for $\varepsilon > 0$) algorithms for Dynamic Range Mode [53, 85], but besides the lack of progress of purely combinatorial algorithms, there was no other evidence why fast matrix multiplication is necessary. In fact, even an $\tilde{O}(n^{1/2})$ time combinatorial algorithm could exist under previous knowledge.

We finally resolve this gap between the upper and lower bounds for combinatorial Dynamic Range Mode. We show that the previous $\tilde{O}(n^{2/3})$ seeming barrier for combinatorial Dynamic Range Mode algorithms is actually supported by a strong reason: assuming the combinatorial 4-Clique hypothesis, no combinatorial algorithm for Dynamic Range Mode can have poly(n) pre-processing time and $O(n^{2/3-\varepsilon})$ update and query time for any $\varepsilon > 0$.

Our techniques can also show conditional lower bound for a similar problem, Dynamic Range Minority [28, 46], which asks for the least frequent integer (that appears at least once) in the query range $a_l, a_{l+1}, \ldots, a_r$ for a dynamic sequence a.

Theorem 1.5. *Assuming the combinatorial 4-Clique hypothesis, there is no combinatorial data structure that solves Dynamic Range Mode in* poly(n) *pre-processing time, $O(n^{2/3-\varepsilon})$ amortized query time and $O(n^{2/3-\varepsilon})$ amortized update time for $\varepsilon > 0$. The same lower bound also holds for the Dynamic Range Minority problem.*

[4] In this paper, $\tilde{O}$ hides poly-logarithmic factors in the input size.

Our techniques generalize to high-dimensional Range Mode as well, which was studied in [27].

Subgraph Connectivity. In the Subgraph Connectivity problem (SubConn) [22, 48], we need to pre-process a static undirected graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, and maintain a dynamic vertex subset $S \subseteq V$ that undergoes insertions and deletions. For each query specified by vertices s, t, we need to report whether s and t are connected in the induced subgraph of S in G.

By running breadth-first search for every query or update, it is trivial to solve SubConn in $O(m)$ query time and $O(1)$ update time, or $O(1)$ query time and $O(m)$ update time respectively. The first nontrivial solution was an algorithm given by Chan [22] that uses fast matrix multiplication, with $\tilde{O}(m^{0.94})$ amortized update time and $\tilde{O}(m^{1/3})$ worst-case query time. The algorithm with current fastest update time, due to Chan, Pătraşcu, and Roditty [31], has $\tilde{O}(m^{2/3})$ amortized update time and $\tilde{O}(m^{1/3})$ worst-case query time, and does not need fast matrix multiplication. There exist other algorithms [12, 34, 42, 43] that achieve some combinations of almost linear space, worst-case update time guarantee, or different update-query time trade-off, but the $\tilde{O}(m^{2/3})$ time per update bound remains unbeaten.

The algorithm of Chan, Pătraşcu, and Roditty [31] also supports the following trade-off: for any parameter $1 \le \Delta \le n$, their data structure can achieve $\tilde{O}(\Delta^2 + m/\Delta)$ update time and $\tilde{O}(\Delta)$ query time (with $\tilde{O}(m\Delta)$ pre-processing time). An interesting question is whether this trade-off curve is tight; in particular, it was asked in [31] as an open question whether the $m^{2/3}$ update time can be improved (while keeping a sublinear query time).

Previous conditional lower bounds have ruled out algorithms for SubConn with any of the following running times (for any $\varepsilon > 0$):

(1) (under 3SUM [4]) $\tilde{O}(m^{4/3-\varepsilon})$ pre-processing, $O(m^{a-\varepsilon})$ update and $O(m^{2/3-a-\varepsilon})$ query time, for any $a \in [\frac{1}{6}, \frac{1}{3}]$.
(2) (under OMv [57]) polynomial pre-processing, $O(m^{a-\varepsilon})$ update and $O(m^{1-a-\varepsilon})$ query time, for any $a \in (0, 1)$.
(3) (under OMv [57]) polynomial pre-processing, $O(m^{1/2-\varepsilon})$ update and $O(m^{1-\varepsilon})$ query time.

Item 1 and Item 3 also apply to the easier st-Subconn problem, where each query involves two fixed vertices s, t given during pre-processing. Item 2 partly matches the trade-off curve, showing that the product of update time and query time cannot be much smaller than m. However, it remains open whether we can achieve $\tilde{O}(m^{2/3-\alpha})$ update time and $\tilde{O}(m^{1/3+\alpha})$ query time for some $\alpha > 0$; Item 3 only ruled out the possibility of $\alpha > 1/6$.

We answer this open question, showing that Chan, Pătraşcu, and Roditty's combinatorial algorithm for SubConn [31] is near-optimal under the combinatorial 4-Clique hypothesis. Our lower bound also holds for the easier st-SubConn problem.

Theorem 1.6. *Assuming the combinatorial 4-Clique hypothesis, there is no combinatorial algorithm that solves st-SubConn in poly(m) pre-processing time, $O(m^{2/3-\varepsilon})$ amortized update time, and $O(m^{1-\varepsilon})$ amortized query time for $\varepsilon > 0$.*

We leave it as an open problem to either improve the 2/3 exponent in the update time using fast matrix multiplication or determine it's impossible.

Table 1: Our lower bounds for dynamic problems. The lower bounds based on the k-Clique hypothesis work for combinatorial algorithms and the lower bounds based on the OuMv$_k$ hypothesis work for arbitrary algorithms.
The lower bounds state that there are no algorithms achieving the stated pre-processing time, update time and query time simultaneously for $\varepsilon > 0$, under the corresponding hypotheses, even if the algorithms have amortized update and query time, and the updates are semi-online.
All our lower bounds have matching upper bounds unless otherwise stated. The upper bounds column references algorithms that run in the stated pre-processing time, update time and query time simultaneously for $\varepsilon = 0$, up to poly-logarithmic factors. All algorithms work for fully dynamic inputs with worst-case time guarantees, unless otherwise stated.

Problems	Lower Bounds			Hypotheses	References	Upper Bounds
	Pre-processing	Update	Query			
Dynamic Range Mode	$\text{poly}(n)$	$n^{2/3-\varepsilon}$		4-Clique	Thm. 1.5	[27, 46]
Dynamic Range Minority	$\text{poly}(n)$	$n^{2/3-\varepsilon}$		4-Clique	Thm. 1.5	[28, 46]
Dynamic d-Dimensional Orthogonal Range Mode	$\text{poly}(n)$	$n^{1-1/(2d+1)-\varepsilon}$		$(2d+2)$-Clique	Thm. 3.3	Prop. 3.2
st Subgraph Connectivity	$\text{poly}(m)$	$m^{2/3-\varepsilon}$	$m^{1-\varepsilon}$	4-Clique	Thm. 1.6	[31] amortized
Dynamic 2-Pattern Document Retrieval	$\text{poly}(n)$	$n^{2/3-\varepsilon}$		4-Clique	Thm. 1.7	Full Paper
Dynamic 2D Orthogonal Range Color Counting	$\text{poly}(n)$	$n^{2/3-\varepsilon}$		4-Clique	Thm. 1.8	Full Paper
Dynamic st-Reachability	$\text{poly}(n)$	$n^{2-\varepsilon}$		4-Clique	Thm. 1.9	trivial
Dynamic Strong Connectivity						
Dynamic Bipartite Perfect Matching						
Dynamic Skyline Points Counting in $\mathbb{R}^{2k-1}$	$\text{poly}(n)$	$n^{1-1/k-\varepsilon}$		OuMv$_k$	Thm. 1.10	Prop. 4.4 semi-online
Dynamic Klee's measure for unit hypercubes in $\mathbb{R}^{2k-1}$	$\text{poly}(n)$	$n^{1-1/k-\varepsilon}$		OuMv$_k$	Thm. 1.11	[21] semi-online only for $k = 2$
Chan's Halfspace problem in $\mathbb{R}^k$	$\text{poly}(n)$	$n^{1-1/k-\varepsilon}$		OuMv$_k$	Thm. 1.12	[21] amortized
Dynamic s-k-Uniform $(k+1)$-Hyperclique	$\text{poly}(n)$	$n^{1-\varepsilon}$	$n^{k-\varepsilon}$	OuMv$_k$	Full Paper	trivial
k-Dimensional Erickson's problem	$\text{poly}(n)$	$n^{k-1-\varepsilon}$	$n^{k-\varepsilon}$	OuMv$_k$	Full Paper	trivial
$(k-1)$-Dimensional Langerman's problem	$\text{poly}(n)$	$n^{(k-1)^2/k-\varepsilon}$		OuMv$_k$	Full Paper	Full Paper

Dynamic 2-Pattern Document Retrieval. In the 2-Pattern Document Retrieval problem, one is given a list of strings $S_1, \ldots, S_D$ of total length $\sum_i |S_i| = n$, and needs to support the following query: given a pair of strings (T_1, T_2), report/count all indices i where S_i contains both T_1 and T_2. The reporting variant was first considered by Muthukrishnan [74], who gave a combinatorial data structure with $\tilde{O}(n^{1.5})$ pre-processing time and $O(|T_1| + |T_2| + \sqrt{n} + output)$ query time where $output$ is the output size, by using a previous algorithm due to Ferragina, Koudas, Muthukrishnan, and Srivastava [47] for a related problem. Larsen, Munro, Nielsen, and Thankachan [67] studied the counting variant of the 2-Pattern Document Retrieval problem and noted that for the counting variant, the query time can be improved to $O(|T_1| + |T_2| + \sqrt{n})$. They also provided a conditional lower bound for the counting variant of the 2-Pattern Document Retrieval problem, showing that any combinatorial algorithm answering $O(n)$ queries requires $n^{1.5-o(1)}$ time under the combinatorial BMM hypothesis, even if the algorithm is only required to determine if the counts are zeros.

Document Retrieval has a wide range of applications (see [75] for a survey) in many scenarios such as web search [79], bioinformatics [11], software repositories [72], chemoinformatics [19] and symbolic music sequences [89] . For instance, in the important web search application, each document string S_i can represent each website, and the patterns can represent keywords sent to a search engine. It is also natural to formalize this application as a dynamic problem instead of a static one: websites are constantly down and up, and it makes sense for a search engine to have the option to only search for websites that are currently online. Thus, we propose the following natural dynamic variant of the 2-Pattern Document Retrieval problem.

Problem 1 (Dynamic 2-Pattern Document Retrieval)**.** *Given a list of strings $S_1, \ldots, S_D$ of total length $\sum_{i=1}^{D} |S_i| = n$, where each string is on or off, maintain a data structure that supports the following operations:*

- *Turn on or turn off a string;*
- *Given a pair of strings (T_1, T_2), count the number of i such that S_i is on and contains both T_1 and T_2.*

We show (in the full paper) that this problem can be solved by a combinatorial data structure with $\tilde{O}(n^{2/3})$ time per update and $\tilde{O}(|T_1| + |T_2| + n^{2/3})$ time per query. Under the combinatorial 4-Clique hypothesis, this data structure is in fact optimal among combinatorial ones.

Theorem 1.7. *Assuming the combinatorial 4-Clique hypothesis, there is no combinatorial data structure that solves the Dynamic 2-Pattern Document Retrieval problem in $\text{poly}(n)$ pre-processing time, $O(n^{2/3-\varepsilon})$ amortized query time and $O(n^{2/3-\varepsilon})$ amortized update time for $\varepsilon > 0$, even when all patterns have lengths $O(1)$ and the algorithm is only required to determine if the counts are zeros.*

We defer the proof of Theorem 1.7 to the full paper.

Dynamic 2D Orthogonal Range Color Counting. In the Orthogonal Range Color Counting problem, we are given a set of n points in $\mathbb{R}^d$, each associated with a color. Each query is given as an axis-aligned box, asking the number of distinct colors of points in the box. The Orthogonal Range Color Counting problem and its reporting variants have been extensively studied, e.g., [14, 29, 30, 49, 54, 55, 61, 63, 68]. In this paper, we focus on the 2-dimensional case.

There are data structures with $\tilde{O}(n^2)$ pre-processing time and $\tilde{O}(1)$ query time for static 2D Orthogonal Range Color Counting [49, 54, 63, 73]. More generally, Kaplan, Rubin, Sharir, and Verbin [63] gave a data structure for static 2D Orthogonal Range Color Counting with a trade-off between pre-processing time and query time.

Kaplan, Rubin, Sharir, and Verbin [63] also showed that, assuming the combinatorial BMM hypothesis, no algorithm can answer n static 2D Orthogonal Range Color Counting queries in $O(n^{1.5-\varepsilon})$ time for $\varepsilon > 0$. This is tight for combinatorial algorithms as we can use their trade-off to obtain a combinatorial algorithm that solves n static 2D Orthogonal Range Color Counting queries in $\tilde{O}(n^{1.5})$ total time.

As we will show in the full paper, their trade-off also implies a combinatorial data structure with $\tilde{O}(n^{2/3})$ update and query time for the dynamic version of 2D Orthogonal Range Color Counting where it is allowed to insert or delete points. Based on ideas from the reduction in [63], we show in the full paper that this combinatorial data structure is essentially optimal under the combinatorial 4-Clique hypothesis.

Theorem 1.8. *Assuming the combinatorial 4-Clique hypothesis, there is no combinatorial data structure that solves Dynamic 2D Orthogonal Range Color Counting in* $\mathrm{poly}(n)$ *pre-processing time,* $O(n^{2/3-\varepsilon})$ *amortized query time and* $O(n^{2/3-\varepsilon})$ *amortized update time for* $\varepsilon > 0$.

High pre-processing time bound for other dynamic graph problems. One main weakness of the combinatorial BMM hypothesis is that it traditionally does not imply update and query lower bounds for data structures that can use arbitrary polynomial pre-processing time. Using reductions from [57], we could obtain lower bounds for data structures with arbitrary polynomial pre-processing time under the OMv hypothesis (and thus the same combinatorial lower bounds under the combinatorial BMM hypothesis), but the lower bounds for update and query times might get lower.

For instance, under the combinatorial BMM hypothesis, Abboud and Vassilevska Williams [4] showed that no combinatorial algorithm can achieve $O(n^{3-\varepsilon})$ pre-processing time and $O(n^{2-\varepsilon})$ update and query times for the Dynamic st-Reachability problem (st-Reach), in which one needs to maintain a directed graph undergoing edge insertions and deletions, and needs to answer whether a fixed node s can reach a fixed node t. Their lower bound for update and query times are very high, showing that any combinatorial algorithm essentially needs to run breadth-first search from scratch for each update or query. On the other hand, their lower bound for the pre-processing time is less desirable: their bound does not rule out combinatorial algorithms with, say, $O(n^3)$ pre-processing time and $\tilde{O}(1)$ update and query times.

Henzinger, Krinninger, Nanongkai, and Saranurak [57] improved the lower bound for pre-processing, by showing that under the OMv hypothesis, there is no algorithm for st-Reach that achieves $\mathrm{poly}(n)$ time pre-processing, $O(n^{1-\varepsilon})$ time update and $O(n^{2-\varepsilon})$ time query for $\varepsilon > 0$. Note that because st-Reach has fast algebraic algorithms that run in $\mathrm{poly}(n)$ pre-processing and $O(n^{1.407})$ time per update and query [90], the drop of the update time bound is inevitable for conditional lower bounds of general algorithms.

We resolve the gap of pre-processing time for st-Reach, in the world of combinatorial algorithms. We show that, under the combinatorial 4-Clique hypothesis, no combinatorial algorithm can achieve $\mathrm{poly}(n)$ pre-processing time and $O(n^{2-\varepsilon})$ update and query times for st-Reach. We also show similar results for other dynamic graph problems such as Dynamic Strong Connectivity. See the full paper for the definitions of these problems.

Theorem 1.9. *Assuming the combinatorial 4-Clique hypothesis, there is no combinatorial data structure that solves st-Reach, Dynamic Strong Connectivity, or Dynamic Bipartite Perfect Matching, in* $\mathrm{poly}(n)$ *pre-processing time,* $O(n^{2-\varepsilon})$ *amortized query time and* $O(n^{2-\varepsilon})$ *amortized update time for* $\varepsilon > 0$.

We defer the proof of Theorem 1.9 to the full paper.

1.1.2 Lower Bounds based on OuMv$_k$.

Skyline Points Counting. Given a set P of points in $\mathbb{R}^d$, a point $p \in P$ is called a *skyline point* if there does not exist another point $q \in P$ such that $p_i \le q_i$ for every $i \in [d]$ (a.k.a. q dominates p). In the Dynamic Skyline Points Counting problem, we need to maintain a set $P \subseteq \mathbb{R}^d$ of at most n points that undergoes insertions and deletions, and query the number of skyline points in P.

The Skyline Counting problem (and its variants) has been studied in various settings, e.g., [18, 26, 62, 83]. For $d \le 2$, Dynamic Skyline Points Counting can be solved in amortized $\mathrm{poly}\log(n)$ time per update [77]. For $d = 3$, Chan [26] designed a data structure for Skyline Points Counting in $\mathbb{R}^3$ with $\tilde{O}(n)$ pre-processing and $\tilde{O}(n^{2/3})$ amortized insertion and deletion time. No nontrivial upper bound is known for the fully dynamic Skyline Counting problem when $d > 3$.

Better upper bounds are possible in the easier *semi-online model*, where during the insertion of a point we are told when the point is to be deleted. By adapting the techniques of Chan [21], it is possible to solve Skyline Points Counting in $\mathbb{R}^{2k-1}$ with $\tilde{O}(n^{1-1/k})$ time per semi-online update for any $k \ge 2$. We show that this upper bound is tight (for odd dimension) under the OuMv$_k$ conjecture.

Theorem 1.10. *Fix any integer* $k \ge 2$. *Assuming the OuMv*$_k$ *hypothesis, there is no data structure for Dynamic Skyline Points Counting in* $\mathbb{R}^{2k-1}$ *with* $\mathrm{poly}(n)$ *pre-processing time,* $O(n^{1-1/k-\varepsilon})$ *amortized update and query time for* $\varepsilon > 0$, *even in the semi-online model.*

We leave it as an open problem to determine the correct exponent for even dimensions $d \ge 4$.

Independent to our work, Dallant and Iacono [41] recently showed an $n^{1/2-o(1)}$ update and query lower bound for Dynamic Skyline Points Counting in $\mathbb{R}^3$ based on the OMv hypothesis, which is the special case of Theorem 1.10 for $k = 2$. They also showed an $n^{1/3-o(1)}$ update and query lower bound for the same problem under the 3SUM hypothesis.

Klee's measure. The Klee's measure problem [65] is an important problem in computational geometry. In the original Klee's measure problem, one is given n axis-aligned boxes in $\mathbb{R}^d$, and needs to determine the volume of their union. For $d \leq 2$, this problem can be solved in $O(n \log n)$ time [82]. In higher dimensions, the best algorithms run in $\tilde{O}(n^{d/2})$ time [23, 24, 78]. It is known that no combinatorial algorithm can improve this bound significantly, under the combinatorial k-Clique hypothesis [23]. As an important special case, when the input boxes are guaranteed to be unit hypercubes, the upper bound can be improved to $n^{d/3+O(1)}$ time [15, 24].

In this paper, we consider the Dynamic Klee's measure problem for unit hypercubes: maintain a data structure for a set of at most n axis-aligned unit hypercubes in $\mathbb{R}^d$ that supports inserting a unit hypercube, deleting a unit hypercube, and querying the volume of the union of the unit hypercubes.

In the semi-online model, where during the insertion of a hypercube we are told when the hypercube is to be deleted, Dynamic Klee's measure for unit cubes can be solved in $\tilde{O}(\sqrt{n})$ update time in $\mathbb{R}^3$ [21, Theorem 6.1]. We show (in the full paper) that the $\tilde{O}(\sqrt{n})$ upper bound in the semi-online model is tight under the OuMv_k hypothesis. (We remark that our lower bound is not tight for dimensions higher than 3.)

Theorem 1.11. *Let $k \geq 2$ be a positive integer. Assuming the OuMv_k hypothesis, there is no data structure for Dynamic Klee's measure for unit hypercubes in $\mathbb{R}^{2k-1}$ with $\text{poly}(n)$ pre-processing time, $O(n^{1-1/k-\varepsilon})$ amortized update and query time for $\varepsilon > 0$, even in the semi-online model.*

Relatedly, Dallant and Iacono [41] obtained various conditional lower bounds for Dynamic Klee's Measure Problem with Squares in $\mathbb{R}^2$ of arbitrary side lengths. This is different from our problem which requires the hypercubes to have unit side lengths.

Chan's Halfspace problem. The following problem appears in [21, Section 4]. We need to maintain a dynamic set H of hyperplanes in $\mathbb{R}^d$, and a dynamic set Q of points in $\mathbb{R}^d$. Each update operation can insert (resp. delete) a hyperplane to (resp. from) H or a point to (resp. from) Q. Define mapping $c_H : \mathbb{R}^d \to \mathbb{R}$ where $c_H(q)$ is the number of hyperplanes in H that contain q. We need to implicitly maintain the multiset of numbers $c_H(Q) = \{c_H(q) : q \in Q\}$. More precisely, Chan [21] originally considered outputting $\square c_H(Q)$ for any fixed operator $\square$ that is decomposable and allows computing $\square(S + j)$ from $\square S$ in constant time (where $S + j = \{i + j : i \in S\}$). For instance, $\square c_H(Q)$ can be defined as the minimum value in $c_H(Q)$ or the sum of all numbers in $c_H(Q)$. We call this problem Chan's Halfspace problem.

Chan [21] solved this problem in $\tilde{O}(n^{1-1/d})$ amortized time per update using $O(n)$ space. Furthermore, he used this problem as an intermediate step in obtaining faster algorithms for a wide range of computational geometry problems such as the decision version of Dynamic Hausdorff Distance and Dynamic Bichromatic Nearest Neighbor Search. Therefore, it is important to understand the computational complexity of this problem, since any improvements to it would carry over to various other problems.

Unfortunately, we show (in the full paper) that the $\tilde{O}(n^{1-1/d})$ upper bound is essentially optimal, under the OuMv_k hypothesis.

Theorem 1.12. *Let $k \geq 2$ be a positive integer. Assuming the OuMv_k hypothesis, there is no data structure for Chan's Halfspace problem in $\mathbb{R}^k$ with $\text{poly}(n)$ pre-processing time, $O(n^{1-1/k-\varepsilon})$ amortized update and query time for $\varepsilon > 0$, even if we only need to output $\min c_H(Q)$ for each query.*

Generalizations of Known OMv-Hard Problems. A wide range of dynamic problems were shown to be hard under the OMv hypothesis [57]. Among these problems, many of them have natural generalizations (e.g. graph problems generalize to hypergraph problems, matrix problems generalize to tensor problems). We show that many of these generalizations in fact have tight conditional lower bounds under the OuMv_k hypothesis.

For instance, Henzinger, Krinninger, Nanongkai, and Saranurak [57] showed that no algorithm for the Dynamic s-Triangle Detection problem, in which one needs to maintain a graph undergoing edge insertions and deletions, and needs to answer whether a fixed node s is in a triangle, has $\text{poly}(n)$ pre-processing time, $O(n^{1-\varepsilon})$ update time and $O(n^{2-\varepsilon})$ query time for $\varepsilon > 0$. Its natural generalization to k-uniform hypergraphs, Dynamic s-k-Uniform $(k+1)$-Hyperclique, is the following: given a k-uniform hypergraph that undergoes hyperedge insertions and deletions, determine whether a fixed node s is in a k-uniform $(k+1)$-hyperclique. We show (in the full paper) that under the OuMv_k hypothesis, no algorithm for Dynamic k-uniform $(k+1)$-hyperclique has $\text{poly}(n)$ pre-processing time, $O(n^{1-\varepsilon})$ amortized update time, and $O(n^{k-\varepsilon})$ amortized query time for $\varepsilon > 0$.

We also show (in the full paper) tight conditional lower bounds for natural generalizations of Erickson's problem and Langerman's problem [80].

1.2 Further Related Works

Pătraşcu [80] was arguably the first to systematically study fine-grained conditional lower bounds for dynamic problems. In this groundbreaking work, Pătraşcu first reduced the 3SUM problem to some triangle reporting problem, which is then further reduced to many dynamic problems such as Dynamic Reachability, Dynamic Shortest Paths and Subgraph Connectivity. This series of reductions show polynomial lower bounds for dynamic problems under the 3SUM hypothesis. This work was later generalized by, for instance, Abboud and Vassilevska Williams [4], and Kopelowitz, Pettie, and Porat [66] to show polynomial lower bounds for more problems under the 3SUM hypothesis. Both [4] and [66] use some variants of the aforementioned triangle reporting problem as intermediate steps in their reductions.

The APSP hypothesis is also widely used to show conditional lower bounds for dynamic problems [2, 4, 50, 84]. For instance, Abboud and Dahlgaard [2] showed hardness for Dynamic APSP and Dynamic Maximum Weight Bipartite Matching in planar graphs under the APSP hypothesis. Based on their techniques, Gawrychowski and Janczewski [50] proved conditional hardness for Dynamic Longest Increasing Subsequence. Vassilevska Williams and Xu [93] related the APSP hypothesis and the 3SUM hypothesis in the context of dynamic problem lower bounds. In [93], they showed that the above-mentioned variants of triangle reporting problems are actually also hard under the APSP hypothesis. Combined with previous reductions from versions of triangle reporting to many dynamic

problems, e.g., [4, 66, 80], these dynamic problems get polynomial lower bounds under the APSP hypothesis as well.

SETH is another popular hypothesis for proving dynamic problem lower bounds. Under SETH, Abboud and Vassilevska Williams [4] showed tight lower bounds for some dynamic problems such as Dynamic Strongly Connected Components Counting. [7] showed hardness for Dynamic Approximate Diameter and related problems under SETH.

Abboud, Vassilevska Williams, and Yu [5] considered an extremely weak hypothesis, which states that at least one of the 3SUM hypothesis, the APSP hypothesis and SETH is true. Under this hypothesis, they first showed conditional lower bound for the so-called Triangle Collection problem, and then used Triangle Collection as an intermediate step to show conditional lower bounds for many dynamic problems such as the counting version of Dynamic Single Source Reachability. Dahlgaard [39] later used Triangle Collection to show conditional lower bounds for dynamic and static diameter approximating problems.

2 PRELIMINARIES

In a graph $G = (V, E)$, we use $\mathcal{N}(v)$ to denote the set of neighbors of $v \in V$. For any subset $U \subseteq V$, we use $\mathcal{N}_U(v)$ to denote $\mathcal{N}(v) \cap U$.

By known techniques (e.g. [92]), the combinatorial k-Clique hypothesis is equivalent to the following unbalanced version.

Hypothesis 2.1 (Combinatorial k-Clique Hypothesis, unbalanced version)**.** *Let $d_1, d_2, \ldots, d_k > 0$ be constant real numbers. There is no $O(n^{d_1+d_2+\cdots+d_k-\varepsilon})$-time* combinatorial *algorithm for k-Clique on k-partite graphs $(V_1 \cup V_2 \cup \cdots \cup V_k, E)$ where $|V_i| = n^{d_i}$ for $i \in [k]$, for any $\varepsilon > 0$.*

Fact 2.2. *For any fixed k and fixed $d_1, d_2, \ldots, d_k > 0$, Hypothesis 1.1 is equivalent to Hypothesis 2.1.*

3 LOWER BOUNDS UNDER THE k-CLIQUE HYPOTHESIS

In this section, we show tight combinatorial lower bounds for Dynamic Range Mode and *st* Subgraph Connectivity under the combinatorial 4-Clique hypothesis. We defer the proofs for Dynamic 2-Pattern Document Retrieval and Dynamic 2D Orthogonal Range Color Counting to the full paper.

Previously, there exist known combinatorial lower bounds under the combinatorial BMM hypothesis for the static versions of Range Mode [27], 2-Pattern Document Retrieval [67] and 2D Orthogonal Range Color Counting [63]. Based on these previous reductions, we show higher lower bounds for the dynamic variants of these problems. Intuitively, the static variants of these problems are able to simulate triangles by known reductions, and the dynamic operations are able to simulate the 4th vertex in a 4-clique. We also show hardness for the d-dimensional generalizations of static and dynamic Range Mode, based on the combinatorial $(2d+1)$-Clique hypothesis and combinatorial $(2d+2)$-Clique hypothesis respectively. Our results showcase an intriguing pattern that relates k-Clique-based lower bound for a static problem and $(k+1)$-Clique-based lower bound for its dynamic variant. We believe this pattern could be useful for designing combinatorial clique-based lower bounds in the future.

3.1 Range Mode

We first recall the definition of Dynamic Range Mode.

Problem 2 (Dynamic Range Mode)**.** *Maintain a data structure for an integer array a of size at most n, and support the following operations:*

- *Insert or delete an integer;*
- *For each query specified by l, r, report the most frequent integer appearing in $a_l, a_{l+1}, \ldots, a_r$, breaking ties arbitrarily.*

In this section, we prove Theorem 1.5:

Theorem 1.5. *Assuming the combinatorial 4-Clique hypothesis, there is no combinatorial data structure that solves Dynamic Range Mode in* poly(n) *pre-processing time, $O(n^{2/3-\varepsilon})$ amortized query time and $O(n^{2/3-\varepsilon})$ amortized update time for $\varepsilon > 0$. The same lower bound also holds for the Dynamic Range Minority problem.*

Proof. Suppose for the sake of contradiction that there is a combinatorial data structure for Dynamic Range Mode in poly(n) pre-processing time, $O(n^{2/3-\varepsilon})$ query time and $O(n^{2/3-\varepsilon})$ update time. Let the pre-processing time of the data structure be $O(n^t)$ for some fixed constant t.

We reduce from an unbalanced instance of 4-Clique Detection, where the 4 vertex parts A, B, C, D have sizes $n^{1/3}, n^{1/3}, n^t, n^{2/3}$ respectively. By Fact 2.2, combinatorial algorithms for such a unbalanced instance of 4-Clique Detection requires $n^{t+4/3-o(1)}$ time under the combinatorial 4-Clique hypothesis.

We initialize an array of size $|A||D| + |B||D|$ as follows. The array will consist of $|A| + |B|$ blocks, where each block corresponds to a permutation of D. For each $a \in A$, we create a permutation P_a of D where the neighbors of a in D all occur *before* the non-neighbors of a in D. Similarly, for each $b \in B$, we create a permutation Q_b of D where the neighbors of b in D all occur *after* the non-neighbors of b in D. The resulting array is the concatenation of all P_a for $a \in A$, followed by all Q_b for $b \in B$. This array has size $O(n)$ and thus running the pre-processing phase of the assumed data structure for Dynamic Range Mode on it takes $O(n^t)$ time.

Then for every $c \in C$, we start a phase by performing the following operations on the data structure. First, we insert all neighbors of c in D into the "middle" of the array where the inserted elements are after all the P_a but before all the Q_b. Then for every pair $a \in A, b \in B$, we perform a range mode query on the range that starts with the first neighbor of a in P_a and ends with the last neighbor of b in Q_b. If the mode is a common neighbor of a, b, c and a, b, c form a triangle, then we have found a 4-clique; otherwise, we declare that there is no 4-clique involving a, b, c and continue to the next pair of (a, b). After we are done with c, we remove all neighbors of c inserted in the phase for c.

To show the correctness of this reduction, it suffices to show that if vertices a, b, c have a common neighbor in D, then the range mode query corresponding to a, b, c will find a common neighbor of them. This is clearly true because the range we query consists of the neighbors of a in D, the neighbors of b in D, the neighbors of c in D and some full permutations of D.

The total number of updates is $O(|C||D|) = O(n^{t+2/3})$ and the total number of queries is $O(|A||B||C|) = O(n^{t+2/3})$. Therefore, the running time of the reduction is $O(n^t + n^{t+2/3} \cdot n^{2/3-\varepsilon}) =$

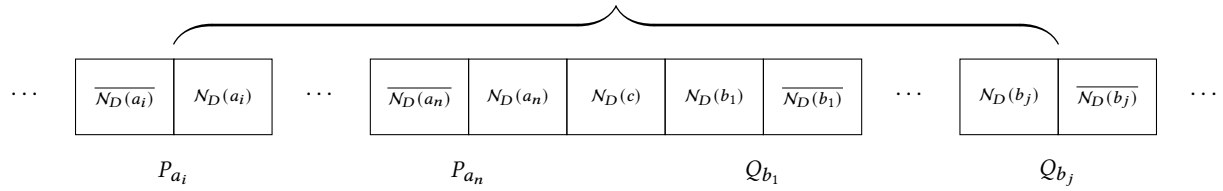

Figure 1: This figure depicts the range mode query corresponding to vertices a_i, b_j and c. Here, $\mathcal{N}_D(v)$ denotes the set of neighbors of vertex v in D, and $\overline{\mathcal{N}_D(v)}$ denotes the set of non-neighbors of vertex v in D.

$O(n^{t+4/3-\varepsilon})$, contradicting the combinatorial 4-Clique hypothesis. Therefore, such an efficient combinatorial data structure for Dynamic Range Mode cannot exist under the combinatorial 4-Clique hypothesis, leading to the claimed lower bound.

A similar reduction works for the Dynamic Range Minority problem as well. For conciseness, we only list the main differences. First, we need to swap the order between the neighbors and non-neighbors inside each permutation P_a for $a \in A$ and Q_b for $b \in B$. To account for the fact that range minority queries ask for the least frequent element that needs to appear in a given range, we insert an arbitrary permutation of D after all P_a for $a \in A$ and before all Q_b for $b \in B$, so that all elements appear in every query we make in the reduction. Then in phase c, we insert the non-neighbors of c in D instead of the neighbors. The query range for a, b, c becomes the range that starts with the first non-neighbor of a in P_a and ends with the last non-neighbor of b in Q_b. The other parts of the reduction remain more or less the same. □

High-Dimensional Range Mode. A natural high-dimensional variant of Range Mode with orthogonal range queries was studied in [27], which gave a combinatorial data structure for the static version of d-Dimensional Range Mode with $\tilde{O}(ns^{2d-1})$ pre-processing time and $\tilde{O}(n^2/s)$ query time for any parameter $s \in [1, n]$ (their pre-processing time is implicit). By setting s to be $n^{1/2d}$, their data structure implies an $\tilde{O}(n^{2-1/2d})$ time algorithm for the following Batch d-Dimensional Orthogonal Range Mode problem.

Problem 3 (Batch d-Dimensional Orthogonal Range Mode)**.** *Given n points in $\mathbb{R}^d$ each labeled with an integer, and n queries specified by $l_1, r_1, l_2, r_2, \ldots, l_d, r_d$, we need to report the most frequent label appearing in the axis-aligned box $[l_1, r_1] \times [l_2, r_2] \times \cdots \times [l_d, r_d]$ for each query, breaking ties arbitrarily.*

We show that the $\tilde{O}(n^{2-1/2d})$ time algorithm is in fact nearly-optimal under the combinatorial $(2d+1)$-Clique hypothesis.

Theorem 3.1. *Assuming the $(2d+1)$-Clique hypothesis, there is no combinatorial data structure that solves Batch d-Dimensional Orthogonal Range Mode in $O(n^{2-1/2d-\varepsilon})$ time for $\varepsilon > 0$.*

Proof. We reduce from an unbalanced instance of $(2d+1)$-Clique Detection, where the first $2d$ parts $V_1, \ldots, V_{2d}$ all have sizes $n^{1/2d}$, while the last part V_{2d+1} has size $n^{1-1/2d}$. By Fact 2.2, combinatorial algorithms for such unbalanced instances of $(2d+1)$-Clique Detection require $n^{2-1/2d-o(1)}$ time under the combinatorial $(2d+1)$-Clique hypothesis.

For each $i \in [2d]$, we will first create an array A_i of size $O(n)$ as follows. The elements in the array will be identified by vertices in V_{2d+1}. For each $v_i \in V_i$, we create a permutation of V_{2d+1}, such that the neighbors of v_i in V_{2d+1} appear before the non-neighbors of v_i in V_{2d+1}. The array A_i is then the concatenation of all the permutations.

We can split each of the d axes in $\mathbb{R}^d$ at the origin to get a total of $2d$ half-axes. We will put each A_i on one of the half-axes as follows. For each odd $i \in [2d]$ and each $j \in [|A_i|]$, we add a point whose $\lceil i/2 \rceil$-th coordinate is j and whose other coordinates are all zeros. We assign a label $A_i[j]$ to this point. For each even $i \in [2d]$ and each $j \in [|A_i|]$, we add a point whose $\lceil i/2 \rceil$-th coordinate is $-j$ and whose other coordinates are all zeros. We similarly assign a label $A_i[j]$ to this point.

Fix a tuple $(v_1, \ldots, v_{2d}) \in V_1 \times \cdots \times V_{2d}$. For every $i \in [2d]$, we use b_i to denote the index in A_i of the last neighbor of v_i in the permutation corresponding to $\mathcal{N}_{V_{2d+1}}(v_i)$. Then we ask a range mode query on the orthogonal range defined as the following:

$$\begin{cases} x_{\lceil i/2 \rceil} \le b_i & : i \in [2d] \text{ is odd,} \\ x_{\lceil i/2 \rceil} \ge -b_i & : i \in [2d] \text{ is even.} \end{cases}$$

It is not hard to see that the multi-set of labels in this orthogonal range is exactly

$$\{A_i[j] : 1 \le i \le 2d, 1 \le j \le b_i\}.$$

By construction of the arrays A_i, this multiset is the union of several full permutations of V_{2d+1}, and the neighborhoods of $v_1, \ldots, v_{2d}$ in V_{2d+1}. Thus, if $v_1, \ldots, v_{2d}$ have a common neighbor in V_{2d+1}, the mode of the orthogonal range will also be a common neighbor.

Therefore, by asking $O(n)$ range mode queries on this instance, we are able to determine whether each tuple $(v_1, \ldots, v_{2d}) \in V_1 \times \cdots \times V_{2d}$ has a common neighbor in V_{2d+1}, so that we can solve the $(2d+1)$-Clique Detection instance in $O(n)$ additional time. This concludes the lower bound proof for Batch d-Dimensional Orthogonal Range Mode. □

Similar to Dynamic Range Mode, the Batch d-Dimensional Orthogonal Range Mode has a natural dynamic variant.

Problem 4 (Dynamic d-Dimensional Orthogonal Range Mode)**.** *Maintain a data structure for a set of at most n points in $\mathbb{R}^d$ each labeled with an integer and support the following operations:*

- *Insert or delete a point;*
- *For each query specified by $l_1, r_1, l_2, r_2, \ldots, l_d, r_d$, report the most frequent label appearing in the axis-aligned box $[l_1, r_1] \times [l_2, r_2] \times \cdots \times [l_d, r_d]$, breaking ties arbitrarily.*

In the full paper, we will show an $\tilde{O}(n^{1-1/(2d+1)})$ time per operation data structure for Dynamic d-Dimensional Orthogonal Range

Mode, and then show that this data structure is essentially optimal under the combinatorial $(2d+2)$-Clique hypothesis.

Proposition 3.2. *There is a combinatorial data structure that solves Dynamic d-Dimensional Orthogonal Range Mode in $\tilde{O}(n^{2-2/(2d+1)})$ pre-processing time, $\tilde{O}(n^{1-1/(2d+1)})$ query time and $\tilde{O}(n^{1-1/(2d+1)})$ update time.*

Theorem 3.3. *Assuming the combinatorial $(2d+2)$-Clique hypothesis, there is no combinatorial data structure that solves Dynamic d-Dimensional Orthogonal Range Mode in $\mathrm{poly}(n)$ pre-processing time, $O(n^{1-1/(2d+1)-\varepsilon})$ amortized query time and $O(n^{1-1/(2d+1)-\varepsilon})$ amortized update time for $\varepsilon > 0$.*

3.2 *st* Subgraph Connectivity

Recall the definition of *st* Subgraph Connectivity (*st*-SubConn)

Problem 5 (*st*-SubConn)**.** *Maintain a data structure for a static undirected graph $G = (V, E)$ with $|V| = n$ and $|E| = m$ with two fixed vertices $s, t \in V$, and a dynamic vertex subset $S \subseteq V$. Support the following operations:*

- *Insert or delete a vertex to or from S;*
- *Report if s is connected to t in the subgraph induced by S.*

We will show the following lower bound for combinatorial algorithms, matching the best known upper bound [31].

Theorem 1.6. *Assuming the combinatorial 4-Clique hypothesis, there is no combinatorial algorithm that solves st-SubConn in $\mathrm{poly}(m)$ pre-processing time, $O(m^{2/3-\varepsilon})$ amortized update time, and $O(m^{1-\varepsilon})$ amortized query time for $\varepsilon > 0$.*

Proof. Suppose there is a combinatorial data structure for *st*-Subconn with $\mathrm{poly}(m)$ pre-processing time, $O(m^{2/3-\varepsilon})$ update time and $O(m^{1-\varepsilon})$ query time. Let the pre-processing time of the data structure be $O(m^r)$ for some fixed constant r.

We reduce from an unbalanced instance of 4-Clique Detection, where the 4 vertex parts A, B, C, D have sizes $m^{2/3}, m^{1/3}, m^{1/3}, m^r$ respectively. Let E denote the edge set of this 4-Clique Detection input instance. By Fact 2.2, any combinatorial algorithm solving such a unbalanced instance of 4-Clique Detection requires $m^{r+4/3-o(1)}$ time under the combinatorial 4-Clique hypothesis.

We create an undirected graph G with "source vertex" s, "sink vertex" t, and $O(m)$ edges as follows. The graph consists of disjoint vertex parts (or, "layers" from left to right)

$$\{s\} \cup V_B \cup U_B \cup U_D \cup U_C \cup V_C \cup \{t\},$$

where $|V_B| = |B| = m^{1/3}, |V_C| = |C| = m^{1/3}, |U_B| = |U_D| = |U_C| = |A| = m^{2/3}$. We assume a natural bijection between B and V_B, which maps $b \in B$ to $b^{V_B} \in V_B$. Similarly, $c \in C$ maps to $c^{V_C} \in V_C$, and $a \in A$ maps to $a^{U_B} \in U_B, a^{U_C} \in U_C, a^{U_D} \in U_D$. The undirected edges in G, which are defined as follows, only connect vertices between adjacent layers.

- For every $b \in B$, add an edge (s, b^{V_B}).
- For every $c \in C$, add an edge (c^{V_C}, t).
- For every $b \in B, a \in A$ such that $(b, a) \in E$, add an edge (b^{V_B}, a^{U_B}).
- For every $c \in C, a \in A$ such that $(c, a) \in E$, add an edge (a^{U_C}, c^{V_C}).
- For every $a \in A$, add two edges (a^{U_B}, a^{U_D}) and (a^{U_D}, a^{U_C}).

To solve the input 4-Clique Detection instance, we use Algorithm 1 with the help of an *st*-SubConn data structure on G maintaining an active vertex subset S.

Algorithm 1: the reduction from 4-Clique Detection to *st*-SubConn

```
1  Initialize the st-SubConn data structure on G, letting S
     contain all vertices.
2  for d ∈ D do
3      for a ∈ A do
4          Let a^{U_D} ∈ S if and only if (a, d) ∈ E.
5      for b ∈ B such that (b, d) ∈ E do
6          Let b^{V_B} ∈ S.
7          for b' ∈ B \ {b} do
8              Let (b')^{V_B} ∉ S.
9          for c ∈ C do
10             Let c^{V_C} ∈ S if and only if (c, d), (c, b) ∈ E.
11         if s, t are connected in the induced subgraph of S
             then
12             return True
13 return False
```

Now we prove the correctness of Algorithm 1 solving 4-Clique Detection. First, assume the input graph contains a 4-clique with vertices a_0, b_0, c_0, d_0. Then, at Line 11 when $d = d_0, b = b_0$, we must have $a_0^{U_D} \in S$ (due to Line 4), $b_0^{V_B} \in S$ (due to Line 6), and $c_0^{V_C} \in S$ (due to Line 10). From the definition of G, we also know that G contains edges $(b_0^{V_B}, a_0^{U_B}), (a_0^{U_C}, c_0^{V_C})$ since $(b, a), (a, c) \in E$. Hence, there is a path $s \to b_0^{V_B} \to a_0^{U_B} \to a_0^{U_D} \to a_0^{U_C} \to c_0^{V_C} \to t$ in the induced subgraph of S, and the query at Line 11 will return True.

Conversely, suppose the query at Line 11 returns True. We will show that it implies the existence of a 4-clique. Let p be the shortest path from s to t in the induced subgraph of S. Then, the shortest path p must visit V_B at most once, since otherwise we could take the last visit $b_{last}^{V_B} \in V_B$ and directly go from s to $b_{last}^{V_B}$ along the edge $(s, b_{last}^{V_B})$. Also, any path from s to t must use at least one vertex in V_B, since the removal of V_B disconnects s and t. Hence, p visits exactly one vertex b^{V_B} in V_B. By a similar argument, p visits exactly one vertex c^{V_C} in V_C. Then, inspecting the structure of the middle layers U_B, U_D, U_C, we see that between b^{V_B} and c^{V_C} the path p must visit $a^{U_B} \to a^{U_D} \to a^{U_C}$ for some $a \in A$.

From Lines 6-8 we observe that b^{V_B} is the only vertex in $S \cap V_B$, and $(b, d) \in E$. Then, from Line 10 and $c^{V_C} \in S$ we know $(c, d), (c, b) \in E$. From Line 4 and $a^{U_D} \in S$ we know $(a, d) \in E$. Finally, from the definition of G, we know $(b, a), (a, c) \in E$ from the existence of edges $(b^{V_B}, a^{U_B}), (a^{U_C}, c^{V_C})$. So a, b, c, d form a 4-clique.

It remains to analyze the time complexity of Algorithm 1. Line 4 contributes $O(|D| \cdot |A|) = O(m^{2/3+r})$ update operations in total. Lines 6-10 contribute $O(|D| \cdot |B| \cdot (|B| + |C|)) = O(m^{2/3+r})$ update operations in total. Line 11 contributes $O(|D| \cdot |B|) = O(m^{1/3+r})$

query operations in total. Hence, the overall running time of this algorithm is asymptotically at most

$$m^r + m^{2/3+r} \cdot m^{2/3-\varepsilon} + m^{1/3+r} \cdot m^{1-\varepsilon} \leq m^{4/3+r-\varepsilon},$$

contradicting the combinatorial 4-Clique hypothesis. □

4 GEOMETRIC PROBLEMS AND OuMv$_k$ HYPOTHESIS

In this section, we will show OuMv$_k$-based conditional lower bounds for Dynamic Skyline Points Counting. We defer the proofs for Dynamic Klee's measure for unit hypercubes, and Chan's Halfspace problem to the full paper. For certain low-dimensional cases of these problems, our lower bounds are actually based on the OMv hypothesis (i.e., OuMv$_2$). We remark that even these OMv-based lower bounds for the low-dimension problems were not known previously in the literature.

All our conditional lower bounds based on the OuMv$_k$ hypothesis hold for all algorithms (not necessarily combinatorial).

4.1 Skyline Points Counting

In this section, we study the Dynamic Skyline Points Counting problem. We first give its formal definition.

Definition 4.1. Given a set of points P in $\mathbb{R}^d$, a point $p \in P$ is called a *skyline point* or *maximal point* if there is no point $q \in P \setminus \{p\}$ such that $p_i \leq q_i$ for every $i \in [d]$ (a.k.a. q dominates p).

Problem 6 (Dynamic Skyline Points Counting)**.** *For a constant integer parameter $d \geq 1$, maintain a data structure for a set of at most n points in $\mathbb{R}^d$ and support inserting a point, deleting a point, and querying the number of skyline (maximal) points.*

We first show an upper bound for Dynamic Skyline Points Counting in $\mathbb{R}^{2k-1}$ in the semi-online model, which needs the following two lemmas.

Lemma 4.2 ([21, Lemma 2.1])**.** *Consider a problem Π with the following property, where $\alpha \geq 1$ and $0 < \beta \leq 1$ are constants: there exists a data structure that can pre-process a set S of n points in $\tilde{O}(n)$ time, such that given any additional set S' of b points, the data structure can solve Π on the set $S \cup S'$ (block query) in $\tilde{O}(b^\alpha n^{1-\beta})$ time.*

Then, we can solve Π on a set of n points under semi-online updates in $\tilde{O}(n^{1-\beta/(1+\alpha)})$ time per update.

Lemma 4.3 ([63, Theorem 2.1])**.** *Let A be a set of n points in $\mathbb{R}^{2k-1}$. For $a = (x_1, x_2, \ldots, x_{2k-1}) \in A$, let $Q(a)$ denote the orthant $(-\infty, x_1] \times (-\infty, x_2] \times \cdots \times (-\infty, x_{2k-1}]$.*

We can decompose $\bigcup_{a \in A} Q(a)$ into $O(n^{k-1})$ pairwise disjoint boxes in $\tilde{O}(n^{k-1})$ time.

Proposition 4.4. *For any integer $k \geq 2$, there exists a data structure for Dynamic Skyline Points Counting in $\mathbb{R}^{2k-1}$ in the semi-online model with $\mathrm{poly}(n)$ pre-processing time and $\tilde{O}(n^{1-1/k})$ update and query time.*

Proof. We verify that the Skyline Points Counting problem satisfies the property required by Lemma 4.2 with $\beta = 1$ and $\alpha = k-1$, which would directly imply the statement.

Given a set S of n points in $\mathbb{R}^{2k-1}$, we first remove all the points that are dominated by some other points, and let S_0 denote the remaining points (i.e., S_0 contains all the skyline points of S). To check whether a point is dominated, we can use standard $(2k-1)$-dimensional range trees in $\mathrm{poly}\log(n)$ time per query after an $\tilde{O}(n)$ time pre-processing. Hence, S_0 can be constructed in $\tilde{O}(n)$ time.

Then, given any additional set S' of b points, we solve the Skyline Point Counting problem on $S \cup S'$ as follows. First, remove all points in S' that are dominated by some other points in $S' \cup S$, and let S'_0 denote the remaining points. The set S'_0 can be similarly computed as before, in $\tilde{O}(b)$ time. Then, observe that the Skyline Points of $S' \cup S$ consist of

- The points in S'_0.
- Points in S_0 that are not in $\bigcup_{p \in S'_0} Q(p)$.

We use Lemma 4.3 to decompose $\bigcup_{p \in S'_0} Q(p)$ into $\tilde{O}(b^{k-1})$ disjoint boxes in $\tilde{O}(b^{k-1})$ time. For each of the boxes, we count the number of points in S_0 it contains, using the $(2k-1)$-dimensional range tree. Hence, we can count the total number of skyline points of $S' \cup S$ in $\tilde{O}(b^{k-1})$ time. □

Then we show that the upper bound is nearly-optimal in the semi-online model:

Theorem 1.10. *Fix any integer $k \geq 2$. Assuming the OuMv$_k$ hypothesis, there is no data structure for Dynamic Skyline Points Counting in $\mathbb{R}^{2k-1}$ with $\mathrm{poly}(n)$ pre-processing time, $O(n^{1-1/k-\varepsilon})$ amortized update and query time for $\varepsilon > 0$, even in the semi-online model.*

Proof. Assume for contradiction that such an efficient data structure exists. We will reduce from an OuMv$_k$ instance of dimension $N = \Theta(n^{1/k})$. Let $M \subseteq [N]^k$ be the input set of OuMv$_k$.

Let $\delta = o(1/N)$ be a sufficiently small positive real number. For every tuple $(a_1, \ldots, a_k) \in M$, we create a point

$$(a_1 - \delta a_k, N - a_1, a_2, N - a_2, \ldots, a_{k-1}, N - a_{k-1}, a_k) \in \mathbb{R}^{2k-1}$$

for the Dynamic Skyline Points Counting instance. We call these points *initial points*. Then we use the pre-processing part of the assumed data structure to pre-process these points in $\mathrm{poly}(n)$ time.

We first show these initial points do not dominate each other.

Claim 4.5. *Two distinct initial points do not dominate each other.*

Proof. Suppose

$$(a_1 - \delta a_k, N - a_1, a_2, N - a_2, \ldots, a_{k-1}, N - a_{k-1}, a_k)$$

is dominated by $(b_1 - \delta b_k, N - b_1, b_2, N - b_2, \ldots, b_{k-1}, N - b_{k-1}, b_k)$. For each integer $i \in [2, k-1]$, if we consider the $(2i-1)$-th and $(2i)$-th coordinates, we must have $a_i \leq b_i$ and $N - a_i \leq N - b_i$, which lead to $a_i = b_i$. Then we consider the first, second and last coordinates. We have $a_1 - \delta a_k \leq b_1 - \delta b_k$, $N - a_1 \leq N - b_1$ and $a_k \leq b_k$. Since a_1, a_k, b_1, b_k are all integers from $[N]$ and $\delta \ll 1/N$, these inequalities imply $a_1 = b_1$ and $a_k = b_k$.

Thus, two points can dominate each other only if they are the same point. □

For every OuMv$_k$ query $U^{(1)} \times \cdots \times U^{(k)}$, we perform the following phase. For every $i \in [k-1]$, and $j' \in [N] \setminus U^{(i)}$, we insert the following point to the data structure:

$$(\underbrace{\infty, \ldots, \infty}_{2i-2 \text{ coordinates of } \infty}, j', N - j', \infty, \ldots, \infty),$$

i.e., it is a point where the $(2i-1)$-th and $(2i)$-th coordinates are j' and $N-j'$ respectively, and all other coordinates are ∞. Then, for each $j \in \{0,\dots,N\}$, we perform an insertion, a query, and a deletion, as follows: first insert a point

$$(\infty,\dots,\infty,j),$$

i.e., it is a point where the last coordinate is j and all other coordinates are ∞. After this insertion, we query the data structure for the number of maximal points, and denote the answer of the query by c_j. After the query, we delete the point $(\infty,\dots,\infty,j)$ and proceed to the next j. After we finish for all $j \in \{0,\dots,N\}$, we delete all points added in the current phase and end the phase.

Then we show that given $c_1,\dots,c_N$, we can determine whether $U^{(1)}\times\cdots\times U^{(k)}$ intersects M in $O(N)$ additional time. We first show the following claim:

Claim 4.6. *For any $j \in [N]$,*

$$c_j = -\left(\sum_{1\le i\le k-1}|U_i|\right) + (k-1)N + 1 + \left|M\cap\left(U^{(1)}\times\cdots\times U^{(k-1)}\times[j+1,N]\right)\right|.$$

Proof. First, it is easy to verify that the points added within each phase are always maximal points, which contributes $\sum_{1\le i\le k-1}(N-|U_i|)+1$ to c_j.

We then analyze which of the initial points are maximal points. Since initial points do not dominate each other, it suffices to consider how the points added within each phase dominate the initial points.

For every $i \in [k-1]$, and $j' \notin U^{(i)}$, we inserted a point where the $(2i-1)$-th and $(2i)$-th coordinates are j' and $N-j'$ respectively, and all other coordinates are ∞. By our construction of initial points, these points precisely dominate those initial points whose $(2i-1)$-th and $(2i)$-th coordinates are j' and $N-j'$ respectively. These points in turn correspond to tuples in M whose i-th entries equal j'. In the query for c_j, we also inserted another point $(\infty,\dots,\infty,j)$, which precisely dominates those initial points whose last coordinate is at most j. These points in turn correspond to tuples in M whose last entries are at most j. Therefore, the set of undominated initial points has a one-to-one correspondence with $M\cap\left(U^{(1)}\times\cdots\times U^{(k-1)}\times[j+1,N]\right)$. This gives the final term of c_j in the claim statement. □

By Claim 4.6, for every $j \in [N]$, the value $c_{j-1}-c_j$ is equal to $\left|M\cap\left(U^{(1)}\times\cdots\times U^{(k-1)}\times\{j\}\right)\right|$. Therefore,

$$\begin{aligned}&\left|M\cap\left(U^{(1)}\times\cdots\times U^{(k-1)}\times U^{k}\right)\right|\\ =&\sum_{j\in U^{(k)}}\left|M\cap\left(U^{(1)}\times\cdots\times U^{(k-1)}\times\{j\}\right)\right|\\ =&\sum_{j\in U^{(k)}}(c_{j-1}-c_j),\end{aligned}$$

which can be computed in $O(N)$ additional time. This concludes the correctness proof of the reduction.

The pre-processing time of the reduction is clearly $\mathrm{poly}(n)=\mathrm{poly}(N)$. For each OuMv_k query, we spend $O(N)$ data structure updates and queries, which take $O(N\cdot n^{1-1/k-\varepsilon}) = O(N^{k-k\varepsilon})$ time. We also spend $O(N)$ additional time, so the running time for each query is $O(N+N^{k-k\varepsilon})$. This clearly contradicts the OuMv_k hypothesis. Therefore, assuming the OuMv_k hypothesis, there is no data structure for Dynamic Skyline Points Counting in $\mathbb{R}^{2k-1}$ with $\mathrm{poly}(n)$ pre-processing time, $O(n^{1-1/k-\varepsilon})$ update and query time for $\varepsilon>0$.

Clearly, the lower bound also works for data structures in the semi-online model, since in fact, we know the deletion time of all points when they are inserted. □

ACKNOWLEDGMENTS

We would like to thank Virginia Vassilevska Williams for many helpful discussions during the early phase of this project. We also thank her for valuable comments on a draft of this paper.

REFERENCES

[1] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. 2018. If the current clique algorithms are optimal, so is Valiant's parser. *SIAM J. Comput.* 47, 6 (2018), 2527–2555. https://doi.org/10.1137/16M1061771

[2] Amir Abboud and Søren Dahlgaard. 2016. Popular Conjectures as a Barrier for Dynamic Planar Graph Algorithms. In *Proceedings of the 57th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2016)*. 477–486. https://doi.org/10.1109/FOCS.2016.58

[3] Amir Abboud, Loukas Georgiadis, Giuseppe F. Italiano, Robert Krauthgamer, Nikos Parotsidis, Ohad Trabelsi, Przemysław Uznański, and Daniel Wolleb-Graf. 2019. Faster Algorithms for All-Pairs Bounded Min-Cuts. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. 7:1–7:15. https://doi.org/10.4230/LIPIcs.ICALP.2019.7

[4] Amir Abboud and Virginia Vassilevska Williams. 2014. Popular Conjectures Imply Strong Lower Bounds for Dynamic Problems. In *Proceedings of the 55th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2014)*. 434–443. https://doi.org/10.1109/FOCS.2014.53

[5] Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. 2018. Matching Triangles and Basing Hardness on an Extremely Popular Conjecture. *SIAM J. Comput.* 47, 3 (2018), 1098–1122. https://doi.org/10.1137/15M1050987

[6] Josh Alman and Virginia Vassilevska Williams. 2021. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA 2021)*. SIAM, 522–539. https://doi.org/10.1137/1.9781611976465.32

[7] Bertie Ancona, Monika Henzinger, Liam Roditty, Virginia Vassilevska Williams, and Nicole Wein. 2019. Algorithms and Hardness for Diameter in Dynamic Graphs. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. 13:1–13:14. https://doi.org/10.4230/LIPIcs.ICALP.2019.13

[8] V. L. Arlazarov, E. A. Dinic, M. A. Kronrod, and I. A. Faradzev. 1970. On economical construction of the transitive closure of a directed graph. *Sov. Math. Dokl.* 11, 5 (1970), 1209–1210.

[9] Grey Ballard, James Demmel, Olga Holtz, and Oded Schwartz. 2013. Graph expansion and communication costs of fast matrix multiplication. *J. ACM* 59, 6 (2013), 1–23. https://doi.org/10.1145/2395116.2395121

[10] Nikhil Bansal and R. Ryan Williams. 2012. Regularity Lemmas and Combinatorial Algorithms. *Theory Comput.* 8, 4 (2012), 69–94. https://doi.org/10.4086/toc.2012.v008a004

[11] Annekathrin Bartsch, Boyke Bunk, Isam Haddad, Johannes Klein, Richard Münch, Thorsten Johl, Uwe Kärst, Lothar Jänsch, Dieter Jahn, and Ida Retter. 2011. GeneReporter—sequence-based document retrieval and annotation. *Bioinformatics* 27, 7 (2011), 1034–1035. https://doi.org/10.1093/bioinformatics/btr047

[12] Surender Baswana, Shreejit Ray Chaudhury, Keerti Choudhary, and Shahbaz Khan. 2016. Dynamic DFS in undirected graphs: breaking the O(m) barrier. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016)*. 730–739. https://doi.org/10.1137/17M114306X

[13] Christoph Berkholz, Jens Keppeler, and Nicole Schweikardt. 2017. Answering conjunctive queries under updates. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS 2017)*. 303–318. https://doi.org/10.1145/3034786.3034789

[14] Panayiotis Bozanis, Nectarios Kitsios, Christos Makris, and Athanasios Tsakalidis. 1995. New upper bounds for generalized intersection searching problems. In *Proceedings of the 22nd International Colloquium on Automata, Languages, and Programming (ICALP 1995)*. Springer, 464–474. https://doi.org/10.1007/3-540-60084-1_97

[15] Karl Bringmann. 2010. Klee's measure problem on fat boxes in time $O(n^{(d+2)/3})$. In *Proceedings of the 26th Annual Symposium on Computational Geometry (SoCG 2010)*. 222–229. https://doi.org/10.1145/1810959.1810999

[16] Karl Bringmann, Allan Grønlund, and Kasper Green Larsen. 2017. A dichotomy for regular expression membership testing. In *Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2017)*. 307–318. https://doi.org/10.1109/FOCS.2017.36

[17] Karl Bringmann and Philip Wellnitz. 2017. Clique-Based Lower Bounds for Parsing Tree-Adjoining Grammars. In *Proceedings of the 28th Annual Symposium on Combinatorial Pattern Matching (CPM 2017)*. 12:1–12:14. https://doi.org/10.4230/LIPIcs.CPM.2017.12

[18] Gerth Stølting Brodal and Kasper Green Larsen. 2014. Optimal Planar Orthogonal Skyline Counting Queries. In *Proceedings of the 14th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2014)*. 110–121. https://doi.org/10.1007/978-3-319-08404-6_10

[19] Frank Brown. 2005. Editorial opinion: chemoinformatics-a ten year update. *Curr. Opin. Drug Discov. Dev.* 8, 3 (2005), 298–302.

[20] Sergio Cabello. 2019. Subquadratic Algorithms for the Diameter and the Sum of Pairwise Distances in Planar Graphs. *ACM Trans. Algorithms* 15, 2 (2019), 21:1–21:38. https://doi.org/10.1145/3218821

[21] Timothy M. Chan. 2003. Semi-Online Maintenance of Geometric Optima and Measures. *SIAM J. Comput.* 32, 3 (2003), 700–716. https://doi.org/10.1137/S0097539702404389

[22] Timothy M. Chan. 2006. Dynamic Subgraph Connectivity with Geometric Applications. *SIAM J. Comput.* 36, 3 (2006), 681–694. https://doi.org/10.1137/S009753970343912X

[23] Timothy M. Chan. 2010. A (slightly) faster algorithm for Klee's measure problem. *Comput. Geom.* 43, 3 (2010), 243–250. https://doi.org/10.1016/j.comgeo.2009.01.007

[24] Timothy M. Chan. 2013. Klee's Measure Problem Made Easy. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2013)*. 410–419. https://doi.org/10.1109/FOCS.2013.51

[25] Timothy M. Chan. 2014. Speeding up the four russians algorithm by about one more logarithmic factor. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2014)*. SIAM, 212–217. https://doi.org/10.1109/FOCS.2009.76

[26] Timothy M. Chan. 2020. Dynamic Geometric Data Structures via Shallow Cuttings. *Discret. Comput. Geom.* 64, 4 (2020), 1235–1252. https://doi.org/10.1007/s00454-020-00229-5

[27] Timothy M. Chan, Stephane Durocher, Kasper Green Larsen, Jason Morrison, and Bryan T. Wilkinson. 2014. Linear-Space Data Structures for Range Mode Query in Arrays. *Theory Comput. Syst.* 55, 4 (2014), 719–741. https://doi.org/10.1007/s00224-013-9455-2

[28] Timothy M. Chan, Stephane Durocher, Matthew Skala, and Bryan T. Wilkinson. 2015. Linear-Space Data Structures for Range Minority Query in Arrays. *Algorithmica* 72, 4 (2015), 901–913. https://doi.org/10.1007/s00453-014-9881-9

[29] Timothy M. Chan, Qizheng He, and Yakov Nekrich. 2020. Further Results on Colored Range Searching. In *Proceedings of the 36th International Symposium on Computational Geometry (SoCG 2020)*. 28:1–28:15. https://doi.org/10.4230/LIPIcs.SoCG.2020.28

[30] Timothy M. Chan and Yakov Nekrich. 2020. Better Data Structures for Colored Orthogonal Range Reporting. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA 2020)*. 627–636. https://doi.org/10.1137/1.9781611975994.38

[31] Timothy M. Chan, Mihai Pătraşcu, and Liam Roditty. 2011. Dynamic Connectivity: Connecting to Networks and Geometry. *SIAM J. Comput.* 40, 2 (2011), 333–349. https://doi.org/10.1137/090751670

[32] Timothy M. Chan, Saladi Rahul, and Jie Xue. 2020. Range closest-pair search in higher dimensions. *Comput. Geom.* 91 (2020), 101669. https://doi.org/10.1016/j.comgeo.2020.101669

[33] Yi-Jun Chang. 2019. Hardness of RNA folding problem with four symbols. *Theor. Comput. Sci.* 757 (2019), 11–26. https://doi.org/10.1016/j.tcs.2018.07.010

[34] Lijie Chen, Ran Duan, Ruosong Wang, Hanrui Zhang, and Tianyi Zhang. 2018. An Improved Algorithm for Incremental DFS Tree in Undirected Graphs. In *Proceedings of the 16th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2018)*. 16:1–16:12. https://doi.org/10.4230/LIPIcs.SWAT.2018.16

[35] Julia Chuzhoy, Yu Gao, Jason Li, Danupon Nanongkai, Richard Peng, and Thatchaphol Saranurak. 2020. A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond. In *Proceedings of the 61st IEEE Annual Symposium on Foundations of Computer Science (FOCS 2020)*. IEEE, 1158–1167. https://doi.org/10.1109/FOCS46700.2020.00111

[36] Raphaël Clifford, Allan Grønlund, and Kasper Green Larsen. 2015. New Unconditional Hardness Results for Dynamic and Online Problems. In *Proceedings of the 56th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2015)*. 1089–1107. https://doi.org/10.1109/FOCS.2015.71

[37] Raphaël Clifford, Allan Grønlund, Kasper Green Larsen, and Tatiana Starikovskaya. 2018. Upper and Lower Bounds for Dynamic Data Structures on Strings. In *Proceedings of the 35th Symposium on Theoretical Aspects of Computer Science (STACS 2018)*. 22:1–22:14. https://doi.org/10.4230/LIPIcs.STACS.2018.22

[38] Michael B. Cohen, Yin Tat Lee, and Zhao Song. 2021. Solving Linear Programs in the Current Matrix Multiplication Time. *J. ACM* 68, 1 (2021), 3:1–3:39. https://doi.org/10.1145/3424305

[39] Søren Dahlgaard. 2016. On the Hardness of Partially Dynamic Graph Problems and Connections to Diameter. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. 48:1–48:14. https://doi.org/10.4230/LIPIcs.ICALP.2016.48

[40] Søren Dahlgaard, Mathias Bæk Tejs Knudsen, and Morten Stöckel. 2017. Finding even cycles faster via capped k-walks. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017)*. 112–120. https://doi.org/10.1145/3055399.3055459

[41] Justin Dallant and John Iacono. 2021. Conditional Lower Bounds for Dynamic Geometric Measure Problems. *CoRR* abs/2112.10095 (2021). https://doi.org/10.48550/arXiv.2112.10095

[42] Ran Duan. 2010. New Data Structures for Subgraph Connectivity. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP 2010)*. 201–212. https://doi.org/10.1007/978-3-642-14165-2_18

[43] Ran Duan and Le Zhang. 2017. Faster Randomized Worst-Case Update Time for Dynamic Subgraph Connectivity. In *Proceedings of the 15th International Symposium on Algorithms and Data Structures (WADS 2017)*. 337–348. https://doi.org/10.1007/978-3-319-62127-2_29

[44] Lech Duraj, Krzysztof Kleiner, Adam Polak, and Virginia Vassilevska Williams. 2020. Equivalences between triangle and range query problems. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA 2020)*. 30–47. https://doi.org/10.1137/1.9781611975994.3

[45] Friedrich Eisenbrand and Fabrizio Grandoni. 2004. On the complexity of fixed parameter clique and dominating set. *Theor. Comput. Sci.* 326, 1-3 (2004), 57–67. https://doi.org/10.1016/j.tcs.2004.05.009

[46] Hicham El-Zein, Meng He, J. Ian Munro, and Bryce Sandlund. 2018. Improved Time and Space Bounds for Dynamic Range Mode. In *Proceedings of the 26th Annual European Symposium on Algorithms (ESA 2018)*. 25:1–25:13. https://doi.org/10.4230/LIPIcs.ESA.2018.25

[47] Paolo Ferragina, Nick Koudas, S Muthukrishnan, and Divesh Srivastava. 2003. Two-dimensional substring indexing. *J. Comput. Syst. Sci.* 66, 4 (2003), 763–774. https://doi.org/10.1016/S0022-0000(03)00028-X

[48] Daniele Frigioni and Giuseppe F. Italiano. 2000. Dynamically Switching Vertices in Planar Graphs. *Algorithmica* 28, 1 (2000), 76–103. https://doi.org/10.1007/s004530010032

[49] Younan Gao and Meng He. 2021. Space Efficient Two-Dimensional Orthogonal Colored Range Counting. In *Proceedings of the 29th Annual European Symposium on Algorithms (ESA 2021)*. 46:1–46:17. https://doi.org/10.4230/LIPIcs.ESA.2021.46

[50] Paweł Gawrychowski and Wojciech Janczewski. 2021. Conditional Lower Bounds for Variants of Dynamic LIS. *CoRR* abs/2102.11797 (2021). https://doi.org/10.48550/arXiv.2102.11797

[51] David Gibb, Bruce M. Kapron, Valerie King, and Nolan Thorn. 2015. Dynamic graph connectivity with improved worst case update time and sublinear space. *CoRR* abs/1509.06464 (2015). https://doi.org/10.48550/arXiv.1509.06464

[52] Andrew V. Goldberg and Robert Endre Tarjan. 1988. A new approach to the maximum-flow problem. *J. ACM* 35, 4 (1988), 921–940. https://doi.org/10.1145/48014.61051

[53] Yuzhou Gu, Adam Polak, Virginia Vassilevska Williams, and Yinzhan Xu. 2021. Faster Monotone Min-Plus Product, Range Mode, and Single Source Replacement Paths. In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*. 75:1–75:20. https://doi.org/10.4230/LIPIcs.ICALP.2021.75

[54] Prosenjit Gupta, Ravi Janardan, and Michiel Smid. 1995. Further results on generalized intersection searching problems: counting, reporting, and dynamization. *Journal of Algorithms* 19, 2 (1995), 282–317. https://doi.org/10.1006/jagm.1995.1038

[55] Prosenjit Gupta, Ravi Janardan, and Michiel Smid. 1997. A technique for adding range restrictions to generalized searching problems. *Inf. Process. Lett.* 64, 5 (1997), 263–269. https://doi.org/10.1016/S0020-0190(97)00183-X

[56] Maximilian Probst Gutenberg, Virginia Vassilevska Williams, and Nicole Wein. 2020. New algorithms and hardness for incremental single-source shortest paths in directed graphs. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2020)*. ACM, 153–166. https://doi.org/10.1145/3357713.3384236

[57] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. 2015. Unifying and Strengthening Hardness for Dynamic Problems via the Online Matrix-Vector Multiplication Conjecture. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC 2015)*. ACM, 21–30. https://doi.org/10.1145/2746539.2746609

[58] Monika R. Henzinger and Valerie King. 1999. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *J. ACM* 46, 4 (1999), 502–516. https://doi.org/10.1145/320211.320215

[59] Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. 2001. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree,

2-edge, and biconnectivity. *J. ACM* 48, 4 (2001), 723–760. https://doi.org/10.1145/502090.502095

[60] Alon Itai and Michael Rodeh. 1978. Finding a Minimum Circuit in a Graph. *SIAM J. Comput.* 7, 4 (1978), 413–423. https://doi.org/10.1137/0207033

[61] Ravi Janardan and Mario Lopez. 1993. Generalized intersection searching problems. *Int. J. Comput. Geom. Appl.* 3, 01 (1993), 39–69. https://doi.org/10.1142/S021819599300004X

[62] Anil Kishore Kalavagattu, Ananda Swarup Das, Kishore Kothapalli, and Kannan Srinathan. 2011. On Finding Skyline Points for Range Queries in Plane. In *Proceedings of the 23rd Annual Canadian Conference on Computational Geometry (CCCG 2011)*. http://www.cccg.ca/proceedings/2011/papers/paper61.pdf

[63] Haim Kaplan, Natan Rubin, Micha Sharir, and Elad Verbin. 2008. Efficient Colored Orthogonal Range Counting. *SIAM J. Comput.* 38, 3 (2008), 982–1011. https://doi.org/10.1137/070684483

[64] Bruce M. Kapron, Valerie King, and Ben Mountjoy. 2013. Dynamic graph connectivity in polylogarithmic worst case time. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2013)*. 1131–1142. https://doi.org/10.1137/1.9781611973105.81

[65] Victor Klee. 1977. Can the Measure of $\cup_1^n [a_i, b_i]$ be Computed in Less than $O(n \log n)$ Steps? *Am. Math. Mon.* 84, 4 (1977), 284–285. https://doi.org/10.2307/2318871

[66] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. 2016. Higher lower bounds from the 3SUM conjecture. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016)*. SIAM, 1272–1287. https://doi.org/10.1137/1.9781611974331.ch89

[67] Kasper Green Larsen, J. Ian Munro, Jesper Sindahl Nielsen, and Sharma V. Thankachan. 2015. On hardness of several string indexing problems. *Theor. Comput. Sci.* 582 (2015), 74–82. https://doi.org/10.1016/j.tcs.2015.03.026

[68] Kasper Green Larsen and Freek van Walderveen. 2013. Near-optimal range reporting structures for categorical data. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2013)*. SIAM, 265–276. https://doi.org/10.1137/1.9781611973105.20

[69] Joshua Lau and Angus Ritossa. 2021. Algorithms and hardness for multidimensional range updates and queries. In *Proceedings of the 12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, Vol. 185. 35:1–35:20. https://doi.org/10.4230/LIPIcs.ITCS.2021.35

[70] Lillian Lee. 2002. Fast context-free grammar parsing requires fast boolean matrix multiplication. *J. ACM* 49, 1 (2002), 1–15. https://doi.org/10.1145/505241.505242

[71] Jason Li. 2019. Faster Minimum k-cut of a Simple Graph. In *Proceedings of the 60th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2019)*. 1056–1077. https://doi.org/10.1109/FOCS.2019.00068

[72] Erik Linstead, Sushil Bajracharya, Trung Ngo, Paul Rigor, Cristina Lopes, and Pierre Baldi. 2009. Sourcerer: Mining and Searching Internet-Scale Software Repositories. *Data Min. Knowl. Discov.* 18, 2 (apr 2009), 300–336. https://doi.org/10.1007/s10618-008-0118-x

[73] J. Ian Munro, Yakov Nekrich, and Sharma V. Thankachan. 2015. Range Counting with Distinct Constraints. In *Proceedings of the 27th Canadian Conference on Computational Geometry (CCCG 2015)*. http://research.cs.queensu.ca/cccg2015/CCCG15-papers/44.pdf

[74] Shanmugavelayutham Muthukrishnan. 2002. Efficient algorithms for document retrieval problems. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2002)*. 657–666.

[75] Gonzalo Navarro. 2014. Spaces, trees, and colors: The algorithmic landscape of document retrieval on sequences. *ACM Computing Surveys (CSUR)* 46, 4 (2014), 1–47. https://doi.org/10.1145/2535933

[76] Jaroslav Nešetřil and Svatopluk Poljak. 1985. On the complexity of the subgraph problem. *Comment. Math. Univ. Carol.* 026, 2 (1985), 415–419.

[77] Mark H. Overmars and Jan van Leeuwen. 1981. Maintenance of Configurations in the Plane. *J. Comput. Syst. Sci.* 23, 2 (1981), 166–204. https://doi.org/10.1016/0022-0000(81)90012-X

[78] Mark H. Overmars and Chee-Keng Yap. 1991. New upper bounds in Klee's measure problem. *SIAM J. Comput.* 20, 6 (1991), 1034–1045. https://doi.org/10.1109/SFCS.1988.21971

[79] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

[80] Mihai Pătraşcu. 2010. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the 42nd ACM Symposium on Theory of computing (STOC 2010)*. 603–610. https://doi.org/10.1145/1806689.1806772

[81] Mihai Pătraşcu and Mikkel Thorup. 2007. Planning for fast connectivity updates. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007)*. IEEE, 263–271. https://doi.org/10.1109/FOCS.2007.59

[82] Franco P. Preparata and Michael I. Shamos. 2012. *Computational geometry: an introduction.* Springer Science & Business Media.

[83] Saladi Rahul and Ravi Janardan. 2012. Algorithms for range-skyline queries. In *Proceedings of the 2012 International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2012)*. 526–529. https://doi.org/10.1145/2424321.2424406

[84] Liam Roditty and Uri Zwick. 2011. On Dynamic Shortest Paths Problems. *Algorithmica* 61, 2 (2011), 389–401. https://doi.org/10.1007/s00453-010-9401-5

[85] Bryce Sandlund and Yinzhan Xu. 2020. Faster Dynamic Range Mode. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*. 94:1–94:14. https://doi.org/10.4230/LIPIcs.ICALP.2020.94

[86] Piotr Sankowski. 2004. Dynamic Transitive Closure via Dynamic Matrix Inverse (Extended Abstract). In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2004)*. 509–517. https://doi.org/10.1109/FOCS.2004.25

[87] Michael Ian Shamos and Dan Hoey. 1976. Geometric Intersection Problems. In *Proceedings of the 17th Annual Symposium on Foundations of Computer Science (SFCS 1976)*. IEEE Computer Society, 208–215. https://doi.org/10.1109/SFCS.1976.16

[88] Mikkel Thorup. 2000. Near-optimal fully-dynamic graph connectivity. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC 2000)*. 343–350. https://doi.org/10.1145/335305.335345

[89] Rainer Typke, Frans Wiering, and Remco C. Veltkamp. 2005. A survey of music information retrieval systems. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*. 153–160.

[90] Jan van den Brand, Danupon Nanongkai, and Thatchaphol Saranurak. 2019. Dynamic matrix inverse: Improved algorithms and matching conditional lower bounds. In *Proceedings of the 60th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2019)*. 456–480. https://doi.org/10.1109/FOCS.2019.00036

[91] Virginia Vassilevska Williams. 2018. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018*. World Scientific, 3447–3487. https://doi.org/10.1142/9789813272880_0188

[92] Virginia Vassilevska Williams and R. Ryan Williams. 2018. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM* 65, 5 (2018), 1–38. https://doi.org/10.1145/3186893

[93] Virginia Vassilevska Williams and Yinzhan Xu. 2020. Monochromatic triangles, triangle listing and APSP. In *Proceedings of the 2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS 2020)*. IEEE, 786–797. https://doi.org/10.1109/FOCS46700.2020.00078

[94] Virginia Vassilevska Williams and Yinzhan Xu. 2020. Truly subcubic min-plus product for less structured matrices, with applications. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2020)*. SIAM, 12–29. https://doi.org/10.1137/1.9781611975994.2

[95] Christian Wulff-Nilsen. 2017. Fully-dynamic minimum spanning forest with improved worst-case update time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017)*. ACM, 1130–1143. https://doi.org/10.1145/3055399.3055415

[96] Huacheng Yu. 2018. An improved combinatorial algorithm for boolean matrix multiplication. *Inf. Comput.* 261 (2018), 240–247. https://doi.org/10.1007/978-3-662-47672-7_89