Hardness of sequence alignment from fine-grained complexity

Tianqi Yang
IIIS, Tsinghua University
Beijing, China
yangtq19@mails.tsinghua.edu.cn

Yiding Zhang
IIIS, Tsinghua University
Beijing, China
zhangyd19@mails.tsinghua.edu.cn

January 8, 2022

Abstract

We study the hardness of sequence alignment problem in the context of data structure lower bounds. In particular, the problem is given a database of many sequences, and on each query sequence, find the sequence from the database that best aligns to the query. This characterizes the task of finding similar species from a huge biological database. We show that assuming Orthogonal Vectors Conjecture in fine-grained complexity, even approximately finding the optimal answer is very hard, in the sense that there could not be a data structure that simultaneously achieves $\mathrm{poly}(n)$ space and $n^{1-\varepsilon}$ query time, where n is the length of the database. This means that essentially we need to enumerate over the whole database to get the answer.

1 Introduction

Sequence alignment problem asks to find an "alignment" of two DNA sequences so that as much as essential pieces are matched together. Besides revealing the common genes of the two species, it also characterizes the similarity between them. As a result, sequence alignment is a fundamental problem in computational biology, not only serves as the starting point of analyzing the difference between species, but is used to find similar species from a database for an unknown new specimen as well.

In this note, we mainly consider the problem of finding similar sequences from a huge database, which is required for the second application. Informally, we are given a database of sequences of hundreds of millions of species, and given a query, we are asked to find the sequence that best aligns to this query. We measure the goodness of an alignment via edit distance, i.e., the minimum number of insertions, deletions, and modifications to make the sequences equals. We note that there are many other measures (such as a penalty matrix for different types of misses), but they are in general more complicated to edit distance measure, and hence harder to solve. Since we are interested in hardness results, this is not a problem. One can refer to Definition 3.2 for a formal definition of the problem.

Sequence alignment Given a database of n sequences, each of length at most d, build a data structure that, on a query sequence of length d, find the sequence with minimum edit distance from the given n sequences in the database.

To make the data structure problem interesting, we assume $d \ll n$, since the database is required to be much bigger than the query length to make preprocessing nontrivial. And this is indeed most of the case in practice, since we usually have a super large database of lots of species, but the query sequence is only a particular segment from one sequence. For the rest of introduction we assume $d = \mathsf{polylog} n$ for simplicity.

There are two obvious ways of solving this problem. In one hand, we can do nothing at preprocessing stage, and answer queries by enumerating over the whole database and find the nearest sequence. This data structure takes $O(n)$ space but $O(nd^2)$ query time. On the other hand, we can preprocess and store the answer of all possible queries in advance, and answer all queries due to the preprocessed answer. This results in a data structure with constant query time but $O(2^d)$ space, which is super-polynomial to n as long as $d = \omega(\log n)$. Both these data structures are inefficient since they either requires super-polynomial amount of space, or requires super-polynomial time to answer the queries.

For years, people tried to find faster data structures for this problem, but we are still not able to improve either of the two trivial solutions significantly. So we are still quite far from the ideal solution of $\mathsf{poly}(n)$ space and $\mathsf{poly}(d)$ query time. In practice, we have some good heuristic algorithms such as BLAST [AGMML90], but none of them can have guarantees on the optimality of the result. Hence, it is natural to think of the possibility that such data structures would not exist.

1.1 Our results

We study the essence of why people are not able to give more efficient data structures, and show evidence that these data structures might not exist. In particular, based on well-formed assumptions in fine-grained complexity, such as SETH or OV conjecture (see Section 2.1 for introduction on these conjectures), we are able to show that even finding a result close to optimal does not admit an efficient data structures. In fact, even a data structure slightly more efficient than the trivial ones described above could not exist. We give an state our main theorem here, and will prove it in Section 3. One may also refer to the beginning of Section 3 for a rigorous definition of approximate sequence alignment problem.

Theorem 1.1 (Main theorem). There exists a sufficiently small constant $\varepsilon > 0$ such that the following holds. Assuming SETH or OV conjecture, sequence alignment problem with approximation ratio $(1 + \varepsilon)$ cannot be solved by a data structure of polynomial size and $n^{0.99}$ query time. $\diamondsuit$

Essentially, conditioned on widely believed assumptions, this means that even finding a near-optimal result would be very hard, so we should not expect to find more efficient algorithm, or prove the correctness of the heuristic algorithms. This also shed lights towards proving unconditional lower bounds for this data structure problem, giving known results in data structure lower bounds (see, e.g., [Yin16]).

1.2 Related works

1.2.1 Complexity of pairwise sequence alignment

Besides formalizing sequence alignment as a data structure problem, pairwise sequence alignment is also important since it can be used to find similar pieces of two sequences. Hence finding a fast algorithm has been of great interest in the past forty years [MP80; SW81; CLZ03; BF08; AWY15]. Although there is a straight-forward algorithm running in time $O(n^2)$ using dynamic programming, so this problem is already known to be in P, we still hope the algorithm to run as fast as

possible. However, despite plenty of research, we are still not able to get a provably correct algorithm running in $O(n^{1.999})$ time. Indeed, the fastest algorithm currently is developed by Abboud, Williams, and Yu [AWY15] building on the all-pair shortest path algorithm by Williams [Wil18], and it runs in $O(n^2/\exp(\Omega(\sqrt{\log n})))$ time.

Recently, Backurs and Indyk [BI18] showed that assuming SETH, pairwise sequence alignment problem cannot be solved by an algorithm of truly sub-quadratic time (i.e., $O(n^{2-\varepsilon})$ time for some $\varepsilon > 0$). This complements the line of works on algorithm design for this problem, and showed the inherent difficulty in improving known algorithms.

1.2.2 Hardness of data structures from fine-grained complexity

Prior to our work, Chen and Williams [CW19] studied the hardness of many data structure problems based on the conjectures in fine-grained complexity. In particular, they proved that the following three problems are equivalent in the existence of a polynomial space and $O(n^{0.99})$ query time data structure.

Online OV Given a database of n vectors in $\mathbb{F}_2^d$, for each query of another vector, decides whether there exists a vector in the database orthogonal to it.

Partial Match Given a database of n strings over alphabet $\{0, 1, \star\}$, each of length d. For each query of length-d string over alphabet $\{0, 1\}$, decides whether there exists a string in the database that matches to it, where $\star$ can be matched arbitrarily to 0 or 1.

Onelin Max-IP Given a database of n vectors in $\{0, 1\}^d$, for each query of another vector, find the vector in the database with maximum inner product with it.

Approximate NNS Given a database of n vector in $\mathbb{R}^d$, for each query of another vector, find a vector in the database with almost minimum ℓ_1 distance to the query vector.

We note that the online OV is actually the online version of the classical OV problem, and indeed does not admit such data structures if OV conjecture holds. Hence, all the above problems do not have efficient data structures assuming fine-grained conjectures.

2 Preliminaries

2.1 Fine-grained conjectures

We begin by giving a brief introduction to fine-grained complexity, and the conjectures used in this note. Fine-grained complexity is a recently developed line of work aiming at capturing the hardness of problems already in P. A typical example is the *orthogonal vector* problem (OV), that asks to find an orthogonal pair from a set of vectors. Suppose that the input set has n vectors of dimension d, where $d \ll n$ (a typical setting is $d = c \log n$), enumerating over all pairs of vectors requires $O(n^2 d)$ time. However, improving this algorithm even non-trivially has been an appealing open problem for decades. Indeed, we are still not able to obtain a truly subquadratic algorithm (e.g., $O(n^{1.99}\text{poly}(d)$ time algorithm), despite great efforts. Hence, some researchers believe that this problem is inherently not solvable in truly subquadratic time.

Conjecture 2.1 (OV conjecture). There does not exist constant $\varepsilon > 0$ such that orthogonal vector problem with $d = n^{o(1)}$ can be solved in $O(n^{2-\varepsilon})$ time. ♢

Using truly subquadratic reduction, Erickson [Eri95] showed many interesting problem to be OV-hard, in the sense that if OV conjecture breaks, then they have $O(n^{2-\varepsilon})$ algorithms as well. Such results give evidence that OV conjecture is true, since breaking it would imply significantly better algorithms for many problems, all of which are actively studied for years.

Another central problem considered is CNF-SAT, which asks whether a formula in conjunctive normal form has satisfying assignments. The famous *Strong Exponential Time Hypothesis* (SETH) asserts the non-existence of $2^{(1-\varepsilon)n}$ time CNF-SAT algorithm on n variables for any $\varepsilon > 0$. The folklore reduction from CNF-SAT to OV also put evidence that OV conjecture is true.

Theorem 2.2 (see, e.g., [WY14]). If strong exponential time hypothesis is true, then OV conjecture is true as well. ♢

Still, it is worth mentioning the skepticism of SETH. Recently, some results showed that some stronger versions of SETH are false (see, e.g., [Wil16], [VW21]), so the likelihood of SETH is sometimes doubted. Nevertheless, OV conjecture is weaker than SETH, and it seems plausible that OV conjecture is true but SETH is false. One may refer to [Wil18] for more discussion.

2.2 Formalization of data structure problems

We now formally define the data structure problems discussed in this note. Generally, a data structure problem has a database $\mathcal{D}$ as input, and preprocessing algorithm is required to generate a data structure. Then on any query q, we need an algorithm to compute the answer of this query based on the data structure built.

Definition 2.3. For $n, d \in \mathbb{N}$, we define the following data structure problems:

Online OV Given a database $\mathcal{D}$ of n points in $\{0,1\}^d$, preprocess a data structure of these given points such that, for all query of the form $q \in \{0,1\}^d$, either report a point $x \in \mathcal{D}$ that satisfies $\langle x, q\rangle = 0$, or report that no such x exists.

Approximate ℓ_p**-NNS** Given a database $\mathcal{D}$ of n points in $\mathbb{R}^d$, preprocess a data structure of these given points such that, for all query of the form $q \in \mathbb{R}^d$, report a point $x \in \mathcal{D}$ that satisfies $\|x - q\|_p \le (1+\varepsilon) \cdot \min_{y \in \mathcal{D}} \|y - q\|_p$. Here $(1+\varepsilon)$ is called the approximation ratio. ♢

For data structure problems, the database is usually extremely large compared to the query length, so it is a reasonable to assume that $d = n^{o(1)}$, similar to the OV conjecture. Trivial algorithms for these problems require either $\exp(d)$ space to store the answer of all possible queries in advance or $O(n\mathrm{poly}(d))$ to scan the whole database when answering a query. Chen and Williams [CW19] proved that these two problems are equivalently hard, if we want a better algorithm.

Theorem 2.4 ([CW19], Theorem 7.1). The following are equivalent.

1. There is a $\delta > 0$ such that for all constant c, there is a data structure for Online OV with $d = c \log n$ uses $\mathrm{poly}(n)$ space and allows $n^{1-\delta}$ query time.
2. For some constant $p \in [1,2]$, there is a $\delta > 0$ such that for all $\varepsilon > 0$, there is a data structure for approximate ℓ_p-NNS with approximation ratio $(1+\varepsilon)$ uses $\mathrm{poly}(n)$ space and allows $n^{1-\delta}$ query time.

Moreover, assuming OV conjecture, both algorithms above do not exist. ♢

2.3 Locality-sensitive hashing

Locality-sensitive hashing (LSH) is a useful technique for approximate NNS, and we will use it in our main reduction. The formal definition is shown below.

Definition 2.5 (Locality-sensitive hashing). For a family $\mathcal{H}$ of functions $h : \mathcal{X} \to \mathcal{S}$, we say that $\mathcal{H}$ is a (R, cR, p_1, p_2)-LSH iff

1. for all $x, y \in \mathcal{X}$ with $\|x - y\| \leq R$, we have $\Pr_{h \leftarrow \mathcal{H}}[h(x) = h(y)] \geq p_1$, and
2. for all $x, y \in \mathcal{X}$ with $\|x - y\| \geq cR$, we have $\Pr_{h \leftarrow \mathcal{H}}[h(x) = h(y)] \leq p_2$

where $R > 0$, $c \geq 1$, $0 \leq p_2 < p_1 \leq 1$, and $\|\cdot\|$ is any kind of distance measure. ♢

Known results have shown that there exists locality-sensitive hashing for commonly used distance measures like ℓ_p distance.

Lemma 2.6 ([DIIM04], see also [CW19]). For any constant $p \in [1, 2]$, for any real number $R > 0$, $\varepsilon \in (0, 0.1)$, there exists a $(R, (1+\varepsilon) \cdot R, p_1, p_2)$-LSH under ℓ_p distance such that $p_1 - p_2 = \Omega(\varepsilon)$.♢

3 Proof of the main theorem

In this section, we prove that problem of finding a sequence from a database that best matches the query sequence is OV-hard, even in the approximating case. We first formally define the sequence alignment problem as nearest neighbor search over edit distance.

Definition 3.1 (Edit distance). Let $0 < \alpha \leq 2$. For two strings S and T, we allow the following two operations.

- Insert a character into S or T with cost 1.
- Modify a character in S or T with cost α.

We define the *α-edit distance of S and T*, denoted by $\mathsf{ED}_\alpha(S, T)$, to be the minimum cost to make S and T equal. ♢

We note that when $\alpha > 2$, the definition would be meaningless since we can use two insertions to simulate a modification. So we assume for simplicity that $\alpha \leq 2$.

Definition 3.2 (Sequence alignment, or NNS over α-edit distance). Let Σ be an alphabet of constant size. Given a database $\mathcal{D}$ of n sequences of length d over Σ, preprocess a data structure such that, for all query q of length-d sequences over alphabet Σ, report a sequence $x \in \mathcal{D}$ with minimum α-edit distance with q. ♢

Since we are also interested in the hardness of approximation, we also need the approximate version of this problem.

Definition 3.3 (Approximate NNS over α-edit distance). Let Σ be an alphabet of constant size. Given a database $\mathcal{D}$ of n sequences of length d over Σ, preprocess a data structure such that, for all query q of length-d sequences over alphabet Σ, report a sequence $x \in \mathcal{D}$ that satisfies $\mathsf{ED}_\alpha(q, x) \leq (1+\varepsilon) \cdot \min_{y \in \mathcal{D}} \mathsf{ED}_\alpha(q, y)$. Here $(1+\varepsilon)$ is called the approximation ratio. ♢

To prove the OV-hardness of approximate NNS over edit distance, we show an reduction from approximate NNS in ℓ_1 space. Theorem 1.1 then directly follows from Theorem 2.4 this reduction.

Theorem 3.4. Assume that there exist $\alpha, \delta > 0$ such that for all $\varepsilon > 0$, there exists a data structure for approximate NNS over α-edit distance with approximation ratio $(1+\varepsilon)$, $\mathsf{poly}(n)$ space, and $n^{1-\delta}$ query time. Then there exists $\delta' > 0$ such that for all $\varepsilon' > 0$, there is a data structure for approximate ℓ_1-NNS with approximation ratio $(1+\varepsilon')$, $\mathsf{poly}(n)$ space, and $n^{1-\delta'}$ query time. ♢

The rest of this section is devoted to proving this theorem. We first reduce the standard NNS to NNS in $\mathbb{F}_2$, then reduce the later to the edit distance case.

3.1 From ℓ_1-NNS to $\mathbb{F}_2$-NNS

We first define an intermediate problem in order to make the underlying field discrete. That is, we define the $\mathbb{F}_2$-NNS to be similar to the nearest neighbor search, but instead of taking ℓ_1 distance in $\mathbb{R}$, we take the Hamming distance in the Boolean cube $\{0,1\}^d$.

Definition 3.5 (Approximate $\mathbb{F}_2$-NNS). Given a database $\mathcal{D}$ of n points in $\{0,1\}^d$, preprocess a data structure of these given points such that, for all query of the form $q \in \{0,1\}^d$, report a point $x \in \mathcal{D}$ that satisfies $\|x-q\|_0 \le (1+\varepsilon) \cdot \min_{y\in\mathcal{D}} \|y-q\|_0$. Here $(1+\varepsilon)$ is called the approximation ratio. ♢

Then we can show that $\mathbf{F}_2$-NNS is as hard as the original problem.

Lemma 3.6. Assume that there is a $\delta > 0$ such that for all $\varepsilon > 0$, there exists a data structure that can approximate $\mathbb{F}_2$-NNS with $(1+\varepsilon)$ approximation ratio, $\mathsf{poly}(n)$ space, and $n^{1-\delta}$ query time. Then there is a $\delta' > 0$ such that for all $\varepsilon' > 0$, we can construct a data structure for approximate ℓ_1-NNS with $(1+\varepsilon')$ approximation ratio, $\mathsf{poly}(n)$ space, and $n^{1-\delta'}$ query time. ♢

Proof. The proof can be done in almost the same way as the reduction from [CW19].

First, we can "discretize" the ℓ_1-NNS problem into the following version: given $R = (1+\varepsilon/3)^k$ for some $k \in \mathbb{Z}$, construct a data structure $\mathcal{D}$ for the given n points in $\mathbb{R}^d$ such that for each query $q \in \mathbb{R}^d$, reports a point $x \in \mathcal{D}$ satisfying $\|x-q\|_1 \le (1+\varepsilon/3) \cdot R$ if $\min_{x\in\mathcal{D}} \|x-q\|_1 \le R$, and reports a failure if $\min_{x\in\mathcal{D}} \|x-q\|_1 > R$ (note that its behavior can be arbitrary if neither case holds). We can see that if there is a sublinear algorithm for this version of problem, we can solve the original ℓ_1-NNS in sublinear query time by enumerating $k \in \mathbb{Z}$ and running the algorithm with $R = (1+\varepsilon/3)^k$ until a point is reported. When we get to the smallest k such that $(1+\varepsilon/3)^k \ge \min_{x\in\mathcal{D}} \|x-q\|_1$, the algorithm reports a point $x \in \mathbb{R}^d$ with distance $\|x-q\|_1 \le (1+\varepsilon/3) \cdot R \le (1+\varepsilon/3)^2 \cdot \min_{x\in\mathcal{D}} \|x-q\|_1 \le (1+\varepsilon) \cdot \min_{x\in\mathcal{D}} \|x-q\|_1$.

Then we reduce the modified version to $\mathbf{F}_2$-NNS by applying LSH. Let $\mathcal{H}$ be a family of $(R, (1+\varepsilon/3)\cdot R, p_1, p_2)$-LSH. Then we construct a mapping from $\mathbb{R}^d$ to $\mathbf{F}_2^{d'}$ in the following way (d' will be determined later):

1. Independently sample d' functions $h_1, h_2, \ldots, h_{d'}$ from $\mathcal{H}$.
2. Independently sample d' random mappings $\phi_1, \phi_2, \ldots, \phi_{d'}$ from $Range(\mathcal{H})$ to $\{0,1\}$ (i.e., map each element in $Range(\mathcal{H})$ independently to 0 or 1).
3. Let the mapping be $g : \mathbb{R}^d \to \mathbf{F}_2^{d'}$, where the i-th bit is $g_i(x) = \phi_i(h_i(x))$.

Consider two points $x, y \in \mathbb{R}^d$. By the definition of LSH, for any $i \in [d']$, we have

1. $\Pr[g_i(x) = g_i(y)] \geq p_1 + (1-p_1)/2 = 1/2 + p_1/2$ if $\|x-y\|_1 \leq R$.
2. $\Pr[g_i(x) = g_i(y)] \leq p_2 + (1-p_2)/2 = 1/2 + p_2/2$ if $\|x-y\|_1 \geq (1+\varepsilon/3)\cdot R$.

So we can set $d' = c \log n$ for some constant c. Then from Chernoff bound, we know that for any $0 < \delta < (p_1 - p_2)/2$,

1. $\Pr[\|g(x) - g(y)\|_0 \leq \frac{1}{2} - \frac{1}{2}(p_1 - \delta)] \geq 1 - 2^{\Omega(\delta^2 d')}$ if $\|x-y\|_1 \leq R$.
2. $\Pr[\|g(x) - g(y)\|_0 \geq \frac{1}{2} - \frac{1}{2}(p_2 + \delta)] \geq 1 - 2^{\Omega(\delta^2 d')}$ if $\|x-y\|_1 \geq (1+\varepsilon/3)\cdot R$.

Therefore, if we have a sublinear algorithm for $\mathbf{F}_2$-NNS with approximation ratio $\frac{1/2-(p_2+\delta)/2}{1/2-(p_1-\delta)/2} = 1 + \frac{p_1-p_2-2\delta}{1-p_1+\delta}$, we can solve ℓ_1-NNS with approximation ratio $1+\varepsilon$ and sublinear query time. □

3.2 From $\mathbb{F}_2$-NNS to NNS over edit distance

We now reduce $\mathbb{F}_2$-NNS to our sequence alignment problem, which completes our proof.

Lemma 3.7. Assume that there exist $\alpha, \delta > 0$ such that for all $\varepsilon > 0$, there exists a data structure for approximate NNS over α-edit distance with approximation ratio $(1+\varepsilon)$, poly(n) space, and $n^{1-\delta}$ query time. Then there is a $\delta' > 0$ such that for all $\varepsilon' > 0$, there exists a data structure that can approximate $\mathbb{F}_2$-NNS with $(1+\varepsilon')$ approximation ratio, poly(n) space, and $n^{1-\delta'}$ query time. ♢

Proof. Let the dimension of $\mathbb{F}_2$-NNS be d (note that $d = n^{o(1)}$). Then for each $x \in \{0,1\}^d$, we pad it into a string in $\{0,1,\#\}^{d'}$ by adding $\lceil \alpha \cdot (d+1) \rceil$ #'s between each pair of adjacent characters. Thus we still have $d' = d + (d-1)\cdot\lceil \alpha \cdot (d+1)\rceil = n^{o(1)}$. Note that after the padding, each insertion/deletion of 0, 1 causes a cascade of adjacent #'s required to be inserted/deleted for alignment, and the total cost has exceeded the cost of modifying the whole sequence. Therefore, the edit distance between any two padded strings does not change if we forbid insertion and deletion, and it turns into the ℓ_0 distance between original strings in $\{0,1\}^d$ as a result. Above all, if we can find a sublinear solution for α-edit distance, we can also find a sublinear solution for approximate $\mathbb{F}_2$-NNS. □

References

[AWY15] Amir Abboud, Richard Ryan Williams, and Huacheng Yu. "More Applications of the Polynomial Method to Algorithm Design". In: *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*. Ed. by Piotr Indyk. SIAM, 2015, pp. 218–230. DOI: 10.1137/1.9781611973730.17. URL: https://doi.org/10.1137/1.9781611973730.17 (cit. on pp. 2, 3).

[AGMML90] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. "Basic local alignment search tool". In: *Journal of molecular biology* 215.3 (1990), pp. 403–410 (cit. on p. 2).

[BI18] Arturs Backurs and Piotr Indyk. "Edit Distance Cannot Be Computed in Strongly Subquadratic Time (Unless SETH is False)". In: *SIAM J. Comput.* 47.3 (2018), pp. 1087–1097. DOI: 10.1137/15M1053128. URL: https://doi.org/10.1137/15M1053128 (cit. on p. 3).

[BF08] Philip Bille and Martin Farach-Colton. "Fast and compact regular expression matching". In: *Theor. Comput. Sci.* 409.3 (2008), pp. 486–496. DOI: 10.1016/j.tcs.2008.08.042. URL: https://doi.org/10.1016/j.tcs.2008.08.042 (cit. on p. 2).

[CW19] Lijie Chen and Ryan Williams. "An Equivalence Class for Orthogonal Vectors". In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*. Ed. by Timothy M. Chan. SIAM, 2019, pp. 21–40. DOI: 10.1137/1.9781611975482.2. URL: https://doi.org/10.1137/1.9781611975482.2 (cit. on pp. 3, 4, 5, 6).

[CLZ03] Maxime Crochemore, Gad M. Landau, and Michal Ziv-Ukelson. "A Subquadratic Sequence Alignment Algorithm for Unrestricted Scoring Matrices". In: *SIAM J. Comput.* 32.6 (2003), pp. 1654–1673. DOI: 10.1137/S0097539702402007. URL: https://doi.org/10.1137/S0097539702402007 (cit. on p. 2).

[DIIM04] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. "Locality-sensitive hashing scheme based on p-stable distributions". In: *Proceedings of the 20th ACM Symposium on Computational Geometry, Brooklyn, New York, USA, June 8-11, 2004*. Ed. by Jack Snoeyink and Jean-Daniel Boissonnat. ACM, 2004, pp. 253–262. DOI: 10.1145/997817.997857. URL: https://doi.org/10.1145/997817.997857 (cit. on p. 5).

[Eri95] Jeff Erickson. "On the relative complexities of some geometric problems". In: *Proceedings of the 7th Canadian Conference on Computational Geometry, Quebec City, Quebec, Canada, August 1995*. Carleton University, Ottawa, Canada, 1995, pp. 85–90. URL: http://www.cccg.ca/proceedings/1995/cccg1995%5C_0014.pdf (cit. on p. 4).

[MP80] William J. Masek and Mike Paterson. "A Faster Algorithm Computing String Edit Distances". In: *J. Comput. Syst. Sci.* 20.1 (1980), pp. 18–31. DOI: 10.1016/0022-0000(80)90002-1. URL: https://doi.org/10.1016/0022-0000(80)90002-1 (cit. on p. 2).

[SW81] Temple F Smith and Michael S Waterman. "Identification of common molecular subsequences". In: *Journal of molecular biology* 147.1 (1981), pp. 195–197 (cit. on p. 2).

[VW21] Nikhil Vyas and R. Ryan Williams. "On Super Strong ETH". In: *J. Artif. Intell. Res.* 70 (2021), pp. 473–495. DOI: 10.1613/jair.1.11859. URL: https://doi.org/10.1613/jair.1.11859 (cit. on p. 4).

[Wil18] R. Ryan Williams. "Faster All-Pairs Shortest Paths via Circuit Complexity". In: *SIAM J. Comput.* 47.5 (2018), pp. 1965–1985. DOI: 10.1137/15M1024524. URL: https://doi.org/10.1137/15M1024524 (cit. on pp. 3, 4).

[Wil16] Richard Ryan Williams. "Strong ETH Breaks With Merlin and Arthur: Short Non-Interactive Proofs of Batch Evaluation". In: *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*. Ed. by Ran Raz. Vol. 50. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, 2:1–2:17. DOI: 10.4230/LIPIcs.CCC.2016.2. URL: https://doi.org/10.4230/LIPIcs.CCC.2016.2 (cit. on p. 4).

[WY14] Ryan Williams and Huacheng Yu. “Finding orthogonal vectors in discrete structures”. In: *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*. Ed. by Chandra Chekuri. SIAM, 2014, pp. 1867–1877. DOI: 10.1137/1.9781611973402.135. URL: https://doi.org/10.1137/1.9781611973402.135 (cit. on p. 4).

[Yin16] Yitong Yin. “Simple Average-Case Lower Bounds for Approximate Near-Neighbor from Isoperimetric Inequalities”. In: *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*. Ed. by Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi. Vol. 55. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, 84:1–84:13. DOI: 10.4230/LIPIcs.ICALP.2016.84. URL: https://doi.org/10.4230/LIPIcs.ICALP.2016.84 (cit. on p. 2).