Theoretical Computer Science Department
Faculty of Mathematics and Computer Science
Jagiellonian University

Hardness in theory of computing

Adam Polak

Ph.D. Thesis
Advisor: Paweł Idziak
Kraków, 2018

To my father

Preface

This thesis presents lower and upper bounds, which prove hardness of certain problems in algorithmics and combinatorics. It consists of a series of published papers, which explore two research directions: computational complexity and extremal combinatorics.

> Adam Polak. **Why is it hard to beat $\mathcal{O}(n^2)$ for Longest Common Weakly Increasing Subsequence?** *Information Processing Letters*, volume 132, pages 1-5, 2018.

> Lech Duraj, Marvin Künnemann, and Adam Polak. **Tight Conditional Lower Bounds for Longest Common Increasing Subsequence**. Extended abstract in *12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*, Leibniz International Proceedings in Informatics (LIPIcs), volume 89, pages 15:1-15:13, 2018.
> Full version available online: `https://arxiv.org/abs/1709.10075`.

> Grzegorz Guśpiel, Piotr Micek, and Adam Polak. **On an Extremal Problem for Poset Dimension**. *Order – A Journal on the Theory of Ordered Sets and its Applications*, doi: 10.1007/s11083-017-9444-1, 2017.

In the first two papers, we study the computational complexity of problems related to calculating similarity between sequences, i.e. the Longest Common (Weakly) Increasing Subsequence problems. We prove that beating, by a polynomial factor, a simple quadratic time dynamic programming algorithms for these problems would require refuting the Strong Exponential Time Hypothesis.

The second direction of our research has an extremal combinatorics flavour. In the final of the three papers, we consider a problem related to the Dushnik-Miller dimension of a partial order, and prove that it is impossible to guarantee finding a two-dimensional subposet of size asymptotically larger than $n^{2/3}$ in every poset of size n.

Recently, we extended our study to the realm of online algorithms. In an unpublished manuscript, we introduce a variant of the online graph coloring problem restricted to the class of intersection graphs of intervals with lengths in the fixed range $[1, \sigma]$, a natural generalization of interval and unit-interval graph classes. We prove that no algorithm beats the $5/2$ asymptotic competitive ratio for all, arbitrarily large, values of σ.

> Grzegorz Gutowski, Konstanty Junosza-Szaniawski, Patryk Mikos, Adam Polak, and Joanna Sokół. **Online Coloring of Short Intervals**. Manuscript, 2018. Available online: `https://arxiv.org/abs/1802.09503`.

Acknowledgements

First, let me thank my advisor, Paweł Idziak, for his many insightful suggestions. He gave me a lot of encouragement to try and fail on my way to find out what really interests me. I am also very grateful for his colossal effort and countless hours he spent teaching me how to write mathematics. I have just rewritten this paragraph so that it is no longer a single very long sentence. Does it mean his time has not been wasted?

As trivial as it sounds, this thesis would not look as it looks if not for my wonderful co-authors: Lech Duraj, Grzegorz Guśpiel, Grzegorz Gutowski, Konstanty Junosza-Szaniawski, Marvin Künnemann, Piotr Micek, Patryk Mikos, and Joanna Sokół. With some of you I spent many days staring at a blank whiteboard, with others the collaboration was short yet intense. Still, from each of you I learned something important. Thank you!

All my colleagues from Theoretical Computer Science Department contributed to the excellent work atmosphere, which makes me happy every time I come to the office. I owe special thanks to Marcin Kozik and Bartosz Walczak, who generously offered me their advice and encouragement, always when I needed it.

Let me also mention I am enormously indebted to my wife Kinga, who did her best to be very understanding these many times I spent a night writing on a deadline.

Last but not least, I wish to thank two people, who might be unaware of their contribution to my thesis, yet without them I would not develop my interest in fine-grained complexity. I owe it to Arturs Backurs, who gave a wonderful talk at HALG 2016, and Ola Svensson, who once urged me to give a seminar talk which required studying Arturs's paper very carefully.

The drawing on the title page was created by Felix Reidl. Thank you, Felix! I hope I will develop my scientific toolbox so that not everything looks to me like a nail.

Contents

Introduction

Proving hardness or impossibility plays an important role in theoretical computer science. On one hand, such proofs allow to avoid spending time on improving algorithms that cannot be substantially improved. On the other, discovering reasons for which a particular problem is hard gives us a good insight into the structure of objects involved in the problem. There is a wide range of techniques used to prove hardness in a variety of different settings. In this thesis we explore three research directions and establish hardness results, demonstrating some of those settings.

The first problem we address lies within the scope of computational complexity. In this territory unconditional hardness results, such as e.g. $\Omega(n \log n)$ lower bound for comparison-based sorting, are very rare exceptions. In general, we still lack tools to prove that certain problems cannot be solved with fast algorithms – we do not even know if CNF-SAT, which is hypothesized to require exponential time, cannot be solved in linear time. It is thus common to focus on reductions between computational problems. If a problem A can be reduced to a problem B, then a faster algorithm for B yields a faster algorithm for A. The reduction lets us lift a conjectured hardness of A to hold also for B. Depending on the choice of problem A, this does not necessarily mean that fast algorithms for B do not exist, but it gives us a better understanding of both the underlying structure of the problems and reasons why solving B efficiently is difficult. When reductions go both ways, we can form equivalence classes. Each such class isolates a number of problems, often very different from each other, but computationally hard for some single basic reason.

The second question we address has the extremal combinatorics flavour. In this setting, the basic research question usually concerns guarantees on the size of certain regular substructures that can always be found within an arbitrary larger structure. Therefore, proving hardness amounts to constructing an infinite family of counterexamples.

Our last research direction involves online algorithms, specifically for graph coloring problems. In order to prove that no online algorithm can get close to the optimal offline solution, it is often useful to describe the problem as a combinatorial game between two players, Presenter and Algorithm. In each round Presenter reveals a vertex, together with adjacent edges, and Algorithm immediately and irrevocably assigns it a color. While Algorithm tries to minimize the number of colors, the Presenter's goal is to force Algorithm to use as many colors as possible. A strategy for Presenter implies a lower bound on the performance of any algorithm solving the problem.

1 Computational complexity

Theoretical research in algorithmics focuses primarily on the worst-case time complexity of computational problems. While algorithm design brings complexity upper bounds simply by demonstrating algorithms with provable worst-case running time, complexity theory usually needs much more indirect methods in order to deliver lower bounds.

Admittedly, time hierarchy theorems ascertain that for every (time-constructible) function $t(n)$ there exists a computational problem that can be solved in $t(n)$ time but not in $t(n)^{1-\varepsilon}$ time. However, it is usually very hard to prove this kind of statement for a specific problem of our interest.

One of very rare examples of unconditional complexity lower bounds is $\Omega(n \log n)$ lower bound for sorting n keys. However, it holds only in a restricted model, where a pairwise comparison is the only operation that is allowed to be performed on keys. In practice, sorting is often applied to numbers and one can perform arithmetic operations on them, thus the above lower bound does not apply, and it is in fact possible to sort numbers faster (see, e.g., [24]).

For the moment, we still very much lack tools to prove meaningful unconditional complexity lower bounds. Perhaps the most striking example of our helplessness is the CNF-SAT problem: Given a Boolean formula in *conjunctive normal form*, i.e. a conjunction of disjunctions of literals, decide whether variables can be assigned the values `TRUE` or `FALSE` in a way that the formula evaluates to `TRUE`. We do not know how to prove that CNF-SAT cannot be solved in linear time, even though almost everybody believes it is true, and most even believe that the problem requires exponential time.

For this reason, we settle for conditional lower bounds, i.e. hardness proofs under the assumption of a widely believed hypothesis, usually regarding hardness of a well-studied problem. Arguably the most popular such assumption is $\mathsf{P} \neq \mathsf{NP}$, which can be liberally rephrased as *CNF-SAT cannot be solved in polynomial time.*

Under this assumption, problems which admit reductions from CNF-SAT in polynomial time, i.e. NP-hard problems, cannot be solved in polynomial time, and thus they are considered *computationally hard.* On the other hand, problems in P, i.e. those solvable in polynomial time, are considered *computationally easy.*

For a long time this classic paradigm has remained dominant, for many good reasons: (1) it is model-independent because a polynomial time algorithm for a Turing Machine translates to a polynomial time algorithm in the RAM model and vice versa; (2) it generates a good structure because a composition of polynomial time algorithms yields a polynomial time algorithm; (3) when a problem admits a polynomial time algorithm, it often admits one that is fast in practice, and we rarely see natural examples of $\mathcal{O}(n^{100})$ time algorithms. Nonetheless, this distinction between easy and hard problems remains very coarse.

Indeed, even if a problem is NP-hard and a polynomial time algorithm should not be expected to exist, one might still be interested in obtaining the fastest possible exponential time algorithm. Besides, for problems in P, it happens that even a quadratic time algorithm is too slow in practice, and one looks for a faster one, preferably running in (near-)linear time. Without right tools to prove tight complexity lower bounds one never knows if further speedups are possible or if the current algorithms are optimal. The emergence of *fine-grained complexity* theory brought some such tools.

1.1 Hardness assumptions in fine-grained complexity

Recall that k-CNF-SAT denotes the CNF-SAT problem restricted to the formulas with each clause being a disjunctions of at most k literals. Let

$$s_k = \inf\{\delta \mid k\text{-CNF-SAT can be solved in } \mathcal{O}(2^{\delta n}) \text{ time}\},$$

and let $s_\infty = \lim_{k\to\infty} s_k$. Our current knowledge about CNF-SAT suggests that the following two statements might be true.

Hypothesis 1 (Exponential Time Hypothesis, ETH)**.**

$$s_3 > 0.$$

Hypothesis 2 (Strong Exponential Time Hypothesis, SETH)**.**

$$s_\infty = 1.$$

The two hypotheses were introduced by Impagliazzo, Paturi and Zane [25, 26], who also proved that SETH implies ETH. Naturally, both imply $\mathsf{P} \neq \mathsf{NP}$, and thus they are more bald assumptions[1].

Since their introduction, ETH and SETH have been used as conditional assumptions to prove a large number of lower bounds, which are way more precise than a simple statement that an NP-hard problem is unlikely to admit a polynomial time algorithm. For example, unless ETH fails, there is no $2^{o(n+m)}$ time algorithm for Vertex Cover on n-nodes m-edges graphs [26], as well as no $2^{o(\sqrt{n})}$ time algorithm for 3-Coloring on n-nodes planar graphs [7], while, unless SETH fails, Dominating Set on n-nodes graphs of treewidth t cannot be solved in $(3-\varepsilon)^t\text{poly}(n)$ time [34]. The book [11] offers a good overview of ETH- and SETH-based lower bounds for parameterized and exact complexity.

The use of SETH as an assumption for hardness proofs for problems in P was initiated by Williams [45]. After breakthrough works, of Roditty and Vassilevska Williams for graph problems [42] and Bringmann for sequence problems [5], many polynomial time algorithms were proved to be optimal under SETH (see, e.g., [47] for a survey).

Many of the SETH-based lower bounds go through an intermediate problem, the Orthogonal Vectors problem (OV), defined as follows: Given two sets of d-dimensional $(0,1)$-vectors $\mathcal{U}, \mathcal{V} \subseteq \{0,1\}^d$, both of the same size $|\mathcal{U}| = |\mathcal{V}| = n$, determine whether there is a pair of vectors $(u, v) \in \mathcal{U} \times \mathcal{V}$ which are orthogonal, i.e., their inner product $u \cdot v := \sum_{i=0}^{d-1}(u_i v_i)$ equals 0. The following conjecture is implied by SETH.

Hypothesis 3 (OV-Hypothesis)**.** There is no $\mathcal{O}(n^{2-\varepsilon}\text{poly}(d))$ time algorithm for OV, for any constant $\varepsilon > 0$.

For completeness, we recall the short proof of the above implication, using the split-and-list technique of Williams [45].

[1]It is an author's perception that roughly half of the theoretical computer science community believes SETH is true, while only very few doubt ETH. Ryan Williams, known for his disbelief in SETH [46], is attributed to say that chances he would be proved wrong during his lifetime are nil, and the author, although himself a SETH-believer, cannot disagree.

Theorem 4 (Williams [45])**.** *SETH implies OV-Hypothesis.*

Proof. Take any CNF-SAT instance with n variables and m clauses, and split the variables into two sets, each of size $n/2$. Consider every possible assignment to the variables from the first set, and for each such assignment create an m-dimensional $(0,1)$-vector whose i-th coordinate equals 0 if the assignment satisfies the i-th clause. Let $\mathcal{U}$ denote the just constructed set of $2^{n/2}$ vectors. Repeat the same procedure for the second half of variables to obtain set $\mathcal{V}$. Now, observe that a satisfying assignment to all variables can be composed out of two partial assignments, one to the first half and one to the second half, such that any clause is satisfied by at least one of them. This corresponds to the condition that for each $i \in [m]$ at least one of the two vectors, representing the two partial assignments, has 0 as its i-th coordinate. That is, an assignment is satisfying if and only if the two vectors are orthogonal. Therefore, if there is an $O(n^{2-\varepsilon}\text{poly}(d))$ time algorithm for OV, applying it to the sets $\mathcal{U}, \mathcal{V}$ yields an $O(2^{(n/2)\cdot(2-\varepsilon)}\text{poly}(m))$ algorithm for CNF-SAT, which refutes SETH. □

Note that lower bounds proved by a reduction from OV should be more believable than those that follow directly from SETH. It is entirely possible that SETH is false while OV-Hypothesis remains true.

Naturally, SETH is not the only conjecture commonly used to prove conditional lower bounds for problems in P. Another popular assumption is the 3SUM-Hypothesis.

Hypothesis 5 (3SUM-Hypothesis)**.** For any constant $\varepsilon > 0$, there is no $\mathcal{O}(n^{2-\varepsilon})$ time algorithm for the 3SUM problem, defined as follows: Given a set A of n integers, determine whether there are $a, b, c \in A$ such that $a + b + c = 0$.

3SUM-based lower bounds are known since 1990s, when Gajentaan and Overmars [20] introduced the 3SUM-Hypothesis and proved that, assuming this hypothesis is true, a large number of computational geometry primitives – such as, e.g., given a set of points in the plane, determine whether any three are colinear – require quadratic time. However, only after a seminal paper by Pătraşcu [40] and follow-up works [30] – which used linear hashing arguments to establish a connection between 3SUM and set-intersection-like problems – the 3SUM-Hypothesis became a hardness assumption in domains other than computational geometry.

Other problems whose conjectured hardness is used to prove complexity lower bounds for problems in P include, e.g., All Pairs Shortest Paths [43], Combinatorial Boolean Matrix Multiplication [32], $(\min, +)$-convolution [12].

None of such conditional lower bounds should be considered an impossibility result – certainly, any of the assumed hypotheses may turn out false. However, they give us a better understanding of reasons why solving a given problem efficiently is difficult. With the growing number of such results we start building a big picture of complexity within P. It is interesting if some of the above hypotheses can be reduced to others, so that we would have a single unifying hardness assumptions. However, recent results [8] suggest that such a reduction is unlikely. It seems that the complexity landscape within P is much more intricate than within NP, and, in particular, there is no single reason why hard problems are hard.

1.2 Sequence comparison problems

The Longest Common Subsequence problem (LCS) and its variants are computational primitives with a variety of applications, e.g. in spell checking, DNA sequence comparison, or determining the differences of text files in the UNIX `diff` utility. The Wagner-Fischer algorithm [44] for LCS is commonly taught at undergraduate level algorithms courses. It is a simple example of $\mathcal{O}(n^2)$ time dynamic programming. The state-of-the-art $\mathcal{O}(n^2/\log^2 n)$ time algorithm [36] dates back to the 1980s. Since then, the research community asked whether a polynomial improvement over the quadratic time LCS algorithm is possible.

Recently, based on a line of research relating the CNF-SAT problem to quadratic time problems [45, 42, 5, 3], it has been shown that unless SETH fails, there is no strongly subquadratic time algorithm for LCS [1, 6]. Subsequent work [2] strengthens these lower bounds to hold already under weaker assumptions, and even provides surprising consequences of sufficiently strong polylogarithmic improvements.

1.3 Longest Common Increasing Subsequence

While the progress on LCS was stalled for many years, numerous related problems were proposed and studied in the meantime, among them the Longest Common Increasing Subsequence problem (LCIS), defined as follows: Given two integer sequences X and Y, of length n, determine the length of the longest sequence Z such that Z is strictly increasing and Z is a subsequence of both X and Y. The problem is loosely motivated by biological sequence comparison tasks, and can also be seen as a generalization of the well-studied Longest Increasing Subsequence (LIS) problem, which has an $\mathcal{O}(n \log n)$ time solution and a matching lower bound in the decision tree model [19].

LCIS was originally proposed by Yang, Huang, and Chao [48], who gave a quadratic time algorithm using dynamic programming, leaving open the natural question whether there exists a way to extend the near-linear time solution for LIS to a near-linear time solution for two sequences. Only a partial progress on the question was possible. Let us denote by r the number of *matching pairs*, i.e. $(i, j) \in [n]^2$ such that $X[i] = Y[j]$, and by ℓ the length of the solution size. There is an algorithm for LCIS running in $\mathcal{O}(r \log \ell \log \log n + n \log n)$ time [9], and another one running in $\mathcal{O}(n\ell \log \log n + n \log n)$ time [31].

Even though all the algorithms mentioned above are devised to compute the Longest Common Increasing Subsequence, they can be easily modified to also compute the Longest Common Weakly Increasing Subsequence (LCWIS). While in the general case LCIS and LCWIS seem very similar, they begin to differ after restricting the alphabet size.

For a constant size alphabet LCIS is trivially solvable in linear time – by enumerating a constant number of all possible increasing sequences, and checking each of them in linear time. Even if the number of symbols is not constant, but can be bounded by a sublinear function of the input size, one can use the $\mathcal{O}(n\ell \log \log n + n \log n)$ time algorithm [31] to find LCIS in subquadratic time, since the output size cannot exceed the alphabet size.

The length of LCWIS is not bounded by the alphabet size, and the above solutions for small alphabets do not apply. For 3-symbols alphabet Kutz et al. [31] proposed an $\mathcal{O}(n \log \log n)$ time LCWIS algorithm, which was later improved to $O(n)$ by Duraj [15]. However, already for 4-symbols alphabet nothing better than the general case quadratic time algorithm is known. As we will see later, LCIS and LCWIS also differ in terms

of lower bounds, however the difference is in the proof techniques, not in the resulting bounds itself.

To the best of our knowledge, LCIS and LCWIS are the only previously studied variants of LCS which at the same time have the best known algorithms running in quadratic time and whose quadratic-time hardness does not follow immediately from the quadratic-time hardness of LCS [1, 6]. As such, it was open to determine whether there are (strongly) subquadratic algorithms for LCIS or LCWIS, or whether such algorithms can be ruled out under SETH.

1.4 Our results

We prove that neither LCIS nor LCWIS can be solved in strongly subquadratic time unless SETH fails. This lower bound admits several generalizations, which we discuss later.

Our proof follows the general outline of previous hardness proofs for other sequence comparison problems, e.g. Fréchet Distance [5], Edit Distance [3], or LCS [1, 6]. We provide fine-grained reductions from OV to LCIS and LCWIS. Our reductions are built of two main ingredients: (1) relatively straightforward *vector gadgets*, encoding vector inner product in the language of LCIS/LCWIS, and (2) a more involved *glue*, which let us combine many vector gadgets into a single sequence.

First, we present a short proof of quadratic time lower bound for LCWIS, which uses a relatively simple *glue*, and serves as a gentle introduction to the general scheme of fine-grained reductions from OV to sequence comparison problems. It is arguably simpler than earlier hardness proofs for similar problems. Unfortunately, this approach does not seem sufficient to generalize the lower bound neither to LCIS nor to the variant of the problem with more than two sequences.

In order to address this issue, we develop a more involved gadgetry, so called *separator sequences*, which let us prove the lower bound in full generality. We show that, unless SETH fails, for every $k \geqslant 2$ there is no $\mathcal{O}(n^{k-\varepsilon})$ time algorithm neither for LCIS nor for LCWIS on k sequences, for any $\varepsilon > 0$. We also prove that the $\mathcal{O}(n\ell \log\log n + n \log n)$ time algorithm [31] is optimal, by showing a matching $(n\ell)^{1-\varepsilon}$ lower bound.

2 Extremal combinatorics

Extremal combinatorics investigates the minimal (or maximal) size of structures with certain properties. It often delivers results of the form: *Given a structure of size n one is guaranteed to always find a substructure of size at least $f(n)$ satisfying certain property.* One of the most famous such results is Ramsey's theorem, stating that, given a complete graph on $n \geqslant R(k, k)$ vertices, with every edge colored either red or blue, one can always find either an entirely red complete graph on $k = f(n)$ vertices or an entirely blue complete graph on n vertices.

For such type of problems, a hardness proof amounts to demonstrating that the guaranteed size of a substructure cannot be substantially improved. In this thesis we consider an extremal-type question for poset dimension, and prove an upper bound on the guaranteed size of the largest subposet of a fixed dimension.

2.1 Posets, dimension, and an extremal-type question

A *partially ordered set* (or *poset* for short) is a set together with a binary relation which is reflexive, antisymmetric, and transitive. The rich structure of posets make them a popular subject of study in combinatorics and computer science. Despite many connections and analogies to the graph theory, the theory of posets seems more complex and, so far, less understood.

By the k^d-grid we mean the poset with the universe $\{1, 2, \ldots, k\}^d$ and the natural product order, i.e. $(x_1, x_2, \ldots, x_d) \leqslant (y_1, y_2, \ldots, y_d)$ if $x_i \leqslant y_i$ for all i. The *dimension* of a finite poset $(P, \leqslant_P)$ is the least integer d such that P is a subposet of the k^d-grid, for some $k \in \mathbb{N}$, i.e. the elements of P can be embedded into the grid with a function $h : P \to \{1, 2, \ldots, k\}^d$ such that $h(x) \leqslant h(y) \iff x \leqslant_P y$ for all $x, y \in P$. The notion of poset dimension was introduced in 1941 by Dushnik and Miller [16], and over the years it proved itself to be an important measure of poset complexity, and, to an extent, an analogue of the chromatic number of a graph.

The *standard example* of dimension d, denoted by S_d, is a subposet of the inclusion order of subsets of $\{1, 2, \ldots, d\}$ consisting of all singletons and their complements. The poset S_d is considered an analogue of the d-clique. Every poset containing S_d as a subposet has the dimension at least d. However, standard examples are not the sole reason for a poset to have a high dimension. Similarly to triangle-free graphs with an arbitrarily large chromatic number, there are examples of S_3-free posets with an arbitrarily large dimension. Moreover, both d-coloring and deciding whether a poset has dimension (at most) d are solvable in polynomial time for $d = 2$ and become NP-complete for every $d \geqslant 3$.

A *chain* in a poset is a subset of elements in which every two elements are comparable. In turn, an *antichain* is a subset of elements in which no two distinct elements are comparable. Dilworth's theorem [13] states that elements of a poset can be partitioned into a family of w, where w is the size of the largest antichain in this poset. It implies that every poset on n elements contains a chain or an antichain of size at least $\sqrt{n}$. The same corollary can also be obtained from an easier dual version of Dilworth's theorem, attributed to Mirsky [38]. Note that every chain is a poset of dimension one and every antichain (on at least two elements) is a poset of dimension two. Thus, every poset on n elements contains a subposet of dimension (at most) two on $\sqrt{n}$ elements. It is natural to ask if this guarantee can be improved.

Formally, let $f(n)$ be the largest integer such that every poset on n elements has a subposet on $f(n)$ elements of dimension (at most) two. Clearly, $n^{1/2} \leqslant f(n) \leqslant n$. What is the asymptotics of $f(n)$? This natural extremal-type question was posed in 2010 by François Dorais [14]. The first sublinear upper bound is by Reiniger and Yeager [41], who proved that $f(n) = \mathcal{O}(n^{0.8295})$. For the proof they analyzed two-dimensional subposets of lexicographic powers of standard examples S_{10}.

The Dorais's question gets even more interesting for higher dimensions. Namely, for $d \geqslant 2$, let $f_d(n)$ be the largest integer such that every poset on n elements has a subposet on $f_d(n)$ elements of dimension (at most) d. What is the asymptotics of $f_d(n)$?

Surprisingly, no asymptotically better lower bound than the trivial $\Omega(n^{1/2}) \leqslant f_2(n) \leqslant f_d(n)$ is known. It is possible to improve the constant hidden in the Ω notation to $\sqrt{d}$ by using the argument based on Dilworth's theorem and observing that the union of d

largest chains is a poset of dimension at most d. In terms of upper bounds, Reiniger and Yeager [41] proved that $f_d(n) = \mathcal{O}(n^{g(d)})$, where $g(d) = \min_{m \geqslant d+1} log_{2m}(m+d)$. The remaining gap between the lower and upper bounds is frustratingly large.

2.2 Our results

The main idea behind our contribution was our belief that a k^2-grid is asymptotically the largest two-dimensional subposet of the three-dimensional k^3-grid. Taking $k = n^{1/3}$, this would imply that

$$f(n) = \mathcal{O}(n^{2/3}).$$

First, we prove the above upper bound with a handcrafted construction, which also uses grid-based posets, although with a more involved order.

Then, we notice a link between poset dimension and the work of Marcus and Tardos [35] on permutation pattern avoidance and the Stanley-Wilf conjecture. Using a multidimensional version of the Marcus-Tardos theorem [29, 37], we prove that, for every $d \geqslant 2$, the k^d-grid is asymptotically the largest d-dimensional subposet of the k^{d+1}-grid. This confirms our initial belief, and implies that

$$f_d(n) = \mathcal{O}\left(n^{\frac{d}{d+1}}\right).$$

For dimensions d up to 7 this improves the best known $\mathcal{O}(n^{g(d)})$ upper bound of [41] (see Table 1). The remaining gap to the $\Omega(n^{1/2})$ lower bound stays frustrating.

d	2	3	4	5	6	7	8	9	10	100
$g(d)$	0.82948	0.84952	0.86076	0.86823	0.87370	0.87794	0.88136	0.88421	0.88663	0.92122
$\frac{d}{d+1}$	0.66667	0.75000	0.80000	0.83333	0.85714	0.87500	0.88889	0.90000	0.90909	0.99010

Table 1: Comparison of exponents in the two upper bounds for $f_d(n)$

3 Online algorithms

In many practical settings it is the case that even before the whole input data is known, already some decisions have to be made. This motivates the study of *online problems* and *online algorithms*, which model such settings. The input to an online problem is split into a sequence of *requests*, which are presented to the algorithm one by one. The algorithm has to produce partial output, related to the currently presented request, immediately, i.e. without knowing the following requests. Usually the output cannot be changed, or there is some penalty cost incurred by a later change.

While running time of online algorithms might be a secondary issue – sometimes we do not even care if the algorithm is polynomial-time – the standard performance measure, used to analyze online algorithms, is the *competitive ratio*. Informally, the competitive ratio is the worst-case ratio of the cost of the solution given by an online algorithm to the cost of the optimal offline solution (see Section 3.2 for a formal definition).

Given a particular optimization problem, we can ask what is the best possible competitive ratio of an online algorithm for this problem. Usually, first we want to know if it is possible to obtain any constant competitive ratio, independent of the input size. Then, if the answer is positive, we ask for the smallest possible constant.

There is an analogy between the competitive ratio for online problem and the approximation ratio for NP-hard optimization problems. While the best possible approximation ratio for a problem is the cost we have to pay to solve the problem in polynomial time, the competitive ratio is the cost we have to pay to solve the problem online.

3.1 Online graph coloring and interval graphs

In the *online graph coloring* problem the input graph is presented to the algorithm vertex by vertex, along with all the edges adjacent to the already presented vertices. Each vertex must be assigned a color, different than any of its neighbors, immediately and irrevocably at the moment it is presented, without any knowledge of the remaining part of the graph. The objective is to minimize the number of colors used. The problem and its variants attract much attention, both for theoretical properties and practical applications, e.g., in network multiplexing, resource allocation, or job scheduling.

In the general case, of online coloring of arbitrary graphs there is no hope for any algorithm with a constant competitive ratio. The best known algorithm [22] uses $\mathcal{O}(\chi \cdot n/\log n)$ colors for n-vertex χ-colorable graphs, i.e. it is $\mathcal{O}(n/\log n)$-competitive, and there is a lower bound [23] showing that no online graph coloring algorithm can be $o(n/\log^2 n)$-competitive. It is thus common to study the problem restricted to specific graph classes.

Having in mind the applications in scheduling, one of the important special cases is the class of *interval graphs*, i.e. intersection graphs of intervals on the real line. Interval graphs have been intensively studied since the sixties [4, 33], and, in particular, they are known to be *perfect*, i.e. the chromatic number χ of an interval graph always equals the size of the largest clique ω (see, e.g., [21]). To construct an optimal coloring offline it is enough to color the graph greedily in a nondecreasing order of the left ends of the intervals.

The most basic approach for online graph coloring is the FirstFit algorithm. FirstFit uses $\mathbb{N}$ as the set of colors, and greedily assigns to each vertex the smallest natural number not previously assigned to any of its neighbors. The exact value of the competitive ratio of this algorithm for interval graphs is unknown. After a series of papers, the most recent results state that FirstFit is at least 5- and at most 8-competitive [27, 39]. Kierstead and Trotter [28] designed a more involved online algorithm, which uses at most $3\omega - 2$ colors for ω-colorable interval graphs. They also proved a matching lower bound showing that any algorithm has to use exactly that number of colors. The same lower and upper bounds were obtained independently by Chrobak and Ślusarek [10, 49].

On the other hand, if we further restrict our attention, to the unit interval graphs, i.e. intersection graphs of intervals of unit length, FirstFit uses at most $2\omega - 1$ colors [17]. Currently no better algorithm is known. However, Epstein and Levy [17] proved that every algorithm has to use at least $\frac{3}{2}\omega$ colors.

3.2 Our results

Given the difference in online coloring between interval graphs and unit interval graphs, we ask, what happens in the intermediate graph classes, i.e. interval graphs with bounded length representations. The study of these classes was initiated by Fishburn and Graham [18]. However, they focused mainly on the combinatorial structure, and not its algorithmic applications. It seems a natural question whether it is possible to beat the 3-competitive Kierstead-Trotter algorithm [28] when we assume that interval lengths belong to a fixed range.

We answer this question in the affirmative, by providing an online algorithm which, given an ω-colorable set of intervals with lengths in $[1, \sigma]$, colors it using at most $(1 + \sigma) \cdot \omega + \mathcal{O}(1/\omega)$ colors. For $\sigma = 1$, i.e. unit interval graphs, the algorithm matches the state-of-the-art performance of FirstFit, and for $1 < \sigma < 2$ it beats the Kierstead-Trotter algorithm.

Similarly to computational complexity, algorithms can only give only upper bounds on the hardness of online problem. In order to prove lower bounds, it is often convenient to look at the problem as a combinatorial game between two players, Algorithm and Presenter. In our case, in each round Presenter reveals an interval, and Algorithm immediately and irrevocably assigns it a color. While Algorithm tries to minimize the number of different colors it assigns, the Presenter's goal is to force Algorithm to use as many colors as possible. A strategy for Presenter implies a lower bound on the competitive ratio of any algorithm solving the problem.

Before we proceed with the discussion of the lower bounds we proved, we need a formal definition of competitive ratio. There are two different commonly used notions: absolute competitive ratio and asymptotic competitive ratio. Let A be an online graph coloring algorithm, and let $A(\chi)$ denote the maximum number of colors A uses to color any graph which can be colored offline using χ-colors (i.e. its chromatic number is at most χ).

Definition 6. We say that A *has the absolute competitive ratio* α, if

$$\forall_\chi \ \frac{A(\chi)}{\chi} \leqslant \alpha.$$

Definition 7. We say that A *has the asymptotic competitive ratio* α, if

$$\limsup_{\chi \to \infty} \frac{A(\chi)}{\chi} \leqslant \alpha.$$

Kierstead and Trotter [28] give, for every $\omega \in \mathbb{N}_+$, a strategy for Presenter to construct an ω-colorable set of intervals while forcing Algorithm to use at least $3\omega - 2$ colors. However, the length of presented intervals increases with the increasing ω. For this reason, with the intervals lengths restricted to $[1, \sigma]$, their lower bound is only for the absolute competitive ratio and does not exclude, say, an algorithm that always uses at most $2\omega + \sigma^{10}$ colors.

With our lower bound we can rule out the existence of such an algorithm. We show that for every $\varepsilon > 0$ there is a length $\sigma \geqslant 1$ such that, for every number of colors $\omega \in \mathbb{N}_+$, there is a strategy for Presenter to construct an ω-colorable set of intervals with lengths in $[1, \sigma]$ while forcing Algorithm to use at least $(5/2 - \varepsilon) \cdot \omega$ colors. Therefore, there is

no algorithm with the asymptotic competitive ratio better than 5/2 that works for all $\sigma \geqslant 1$.

Our construction can be considered as a generalization of the 3/2 lower bound for online coloring of unit interval graphs by Epstein and Levy [17], and it borrows also from the work of Kierstead and Trotter [28]. However, in order to control the length of intervals independently of the number of colors, we cannot simply use the pigeonhole principle, as they did. Instead, we develop two combinatorial lemmas, which let us overcome this issue, at a cost of a worse bound for the competitive ratio, i.e. 5/2 instead of 3.

We complement the 5/2 lower bound with two lower bounds for small values of σ. Namely, we show that for every $\sigma > 1$ there is no online algorithm with the asymptotic competitive ratio less than 5/3, and for every $\sigma > 2$ there is no online algorithm with the asymptotic competitive ratio less than 7/4.

4 Bibliography

[1] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Quadratic-time hardness of LCS and other sequence similarity measures. In *Proc. 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS'15)*, pages 59–78, 2015.

[2] Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends or: A polylog shaved is a lower bound made. In *Proc. 48th Annual ACM Symposium on Symposium on Theory of Computing (STOC'16)*, pages 375–388, 2016.

[3] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proc. 47th Annual ACM Symposium on Theory of Computing (STOC'15)*, pages 51–58, 2015.

[4] Seymour Benzer. On the topology of the genetic fine structure. *Proceedings of the National Academy of Sciences of the United States of America*, 45(11):1607–1620, 1959.

[5] Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *Proc. 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS'14)*, pages 661–670, 2014.

[6] Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proc. 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS'15)*, pages 79–97, 2015.

[7] Liming Cai and David Juedes. On the existence of subexponential parameterized algorithms. *Journal of Computer and System Sciences*, 67(4):789–807, 2003.

[8] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, ITCS '16, pages 261–270, 2016.

[9] Wun-Tat Chan, Yong Zhang, Stanley P. Y. Fung, Deshi Ye, and Hong Zhu. Efficient algorithms for finding a longest common increasing subsequence. *Journal of Combinatorial Optimization*, 13(3):277–288, 2007.

[10] Marek Chrobak and Maciej Ślusarek. On some packing problem related to dynamic storage allocation. *RAIRO, Theoretical Informatics and Applications*, 22(4):487–499, 1988.

[11] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[12] Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. On Problems Equivalent to (min,+)-Convolution. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, pages 22:1–22:15, 2017.

[13] R. P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51(1):161–166, 1950.

[14] François G. Dorais. Subposets of small Dushnik-Miller dimension. MathOverflow, 2010. `http://mathoverflow.net/questions/29169`.

[15] Lech Duraj. A linear algorithm for 3-letter longest common weakly increasing subsequence. *Information Processing Letters*, 113(3):94–99, 2013.

[16] Ben Dushnik and E. W. Miller. Partially ordered sets. *American Journal of Mathematics*, 63(3):600–610, 1941.

[17] Leah Epstein and Meital Levy. Online interval coloring and variants. In *ICALP 2005: 32nd International Colloquim on Automata, Languages and Programming, Lisbon, Portugal, July 2005. Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 602–613, 2005.

[18] P. C. Fishburn and R. L. Graham. Classes of interval graphs under expanding length restrictions. *Journal of Graph Theory*, 9(4):459–472, 1985.

[19] Michael L. Fredman. On computing the length of longest increasing subsequences. *Discrete Mathematics*, 11(1):29–35, 1975.

[20] Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry*, 5(3):165–185, 1995.

[21] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57)*. Elsevier, 2 edition, 2004.

[22] Magnús M. Halldórsson. Parallel and on-line graph coloring. *Journal of Algorithms*, 23(2):265–280, 1997.

[23] Magnús M. Halldórsson and Mario Szegedy. Lower bounds for on-line graph coloring. *Theoretical Computer Science*, 130(1):163–174, 1994.

[24] Yijie Han. Deterministic sorting in $o(n \log \log n)$ time and linear space. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC'02, pages 602–608, 2002.

[25] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[26] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[27] H.A. Kierstead, David A. Smith, and W.T. Trotter. First-fit coloring on interval graphs has performance ratio at least 5. *European Journal of Combinatorics*, 51:236–254, 2016.

[28] Henry A. Kierstead and William T. Trotter. An extremal problem in recursive combinatorics. In *12th Southeastern Conference on Combinatorics, Graph Theory and Computing, Baton Rouge, LA, USA, March 1981. Proceedings, vol. II*, volume 33 of *Congressus Numerantium*, pages 143–153, 1981.

[29] Martin Klazar and Adam Marcus. Extensions of the linear bound in the Füredi-Hajnal conjecture. *Advances in Applied Mathematics*, 38(2):258–266, 2007.

[30] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3sum conjecture. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 1272–1287, 2016.

[31] Martin Kutz, Gerth Stølting Brodal, Kanela Kaligosi, and Irit Katriel. Faster algorithms for computing longest common increasing subsequences. *Journal of Discrete Algorithms*, 9(4):314–325, 2011.

[32] Lillian Lee. Fast context-free grammar parsing requires fast boolean matrix multiplication. *Journal of the ACM*, 49(1):1–15, 2002.

[33] C. Lekkeikerker and J. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1):45–64, 1962.

[34] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'11, pages 777–789, 2011.

[35] Adam Marcus and Gábor Tardos. Excluded permutation matrices and the Stanley-Wilf conjecture. *Journal of Combinatorial Theory, Series A*, 107(1):153–160, 2004.

[36] William J. Masek and Mike Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System Sciences*, 20(1):18–31, 1980.

[37] Abhishek Methuku and Dömötör Pálvölgyi. Forbidden hypermatrices imply general bounds on induced forbidden subposet problems. *Combinatorics, Probability and Computing*, 26(4):593–602, 2017.

[38] L. Mirsky. A dual of dilworth's decomposition theorem. *The American Mathematical Monthly*, 78(8):876–877, 1971.

[39] N. S. Narayanaswamy and R. Subhash Babu. A note on first-fit coloring of interval graphs. *Order*, 25(1):49–53, 2008.

[40] Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010*, pages 603–610, 2010.

[41] Benjamin Reiniger and Elyse Yeager. Large subposets with small dimension. *Order*, 33(1):81–84, 2016.

[42] Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proc. 45th Annual ACM Symposium on Symposium on Theory of Computing (STOC'13)*, pages 515–524, 2013.

[43] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science (FOCS'10)*, pages 645–654, 2010.

[44] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974.

[45] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357–365, 2005.

[46] Ryan Williams. Some estimated likelihoods for computational complexity. *Lecture Notes in Computer Science*, 10 000, 2018. `http://people.csail.mit.edu/rrw/likelihoods.pdf`.

[47] Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of International Congress of Mathematicians 2018*, To appear. Available online: http://people.csail.mit.edu/virgi/eccentri.pdf.

[48] I-Hsuan Yang, Chien-Pin Huang, and Kun-Mao Chao. A fast algorithm for computing a longest common increasing subsequence. *Information Processing Letters*, 93(5):249–253, 2005.

[49] Maciej Ślusarek. A coloring algorithm for interval graphs. In *Mathematical Foundations of Computer Science (MFCS 1989)*, pages 471–480, 1989.

Information Processing Letters 132 (2018) 1–5

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Why is it hard to beat $O(n^2)$ for Longest Common Weakly Increasing Subsequence?

Adam Polak [1]

Department of Theoretical Computer Science, Faculty of Mathematics and Computer Science, Jagiellonian University, Łojasiewicza 6, 30-348 Kraków, Poland

ARTICLE INFO

Article history:
Received 24 May 2017
Received in revised form 21 November 2017
Accepted 21 November 2017
Available online 2 December 2017
Communicated by Marcin Pilipczuk

Keywords:
Computational complexity
Longest common weakly increasing subsequence
Lower bound
SETH

ABSTRACT

The Longest Common Weakly Increasing Subsequence problem (LCWIS) is a variant of the classic Longest Common Subsequence problem (LCS). Both problems can be solved with simple quadratic time algorithms. A recent line of research led to a number of matching conditional lower bounds for LCS and other related problems. However, the status of LCWIS remained open.

In this paper we show that LCWIS cannot be solved in $O(n^{2-\varepsilon})$ time unless the Strong Exponential Time Hypothesis (SETH) is false.

The ideas which we developed can also be used to obtain a lower bound based on a safer assumption of NC-SETH, i.e. a version of SETH which talks about NC circuits instead of less expressive CNF formulas.

1. Introduction

Despite attracting interest of many researches, both from theoretical computer science and computational biology communities, for many years the classic Longest Common Subsequence problem (LCS) has not seen any significant improvement over the simple $O(n^2)$ dynamic programming algorithm. The current fastest, $O(n^2/\log^2 n)$ algorithm by Masek and Paterson [1], dates back to 1980.

Difficulties in making progress on the LCS inspired studying numerous related problems, among them the Longest Common Increasing Subsequence problem (LCIS), for which Yang, Huang, and Chao [2] found a quadratic time dynamic programming algorithm. Their algorithm was later improved by Sakai [3] to work in linear space. Even though both these algorithms are devised to compute the Longest Common Increasing Subsequence, they can be easily modified to compute the Longest Common Weakly Increasing Subsequence (LCWIS). The latter problem, first introduced by Kutz et al. [4], can be solved in linear time in the special case of a 3-letter alphabet, as proposed by Duraj [5]. However, despite some attempts over the last decade, no subquadratic time algorithm has been found for the general case of LCWIS.

A recent line of research led to a number of conditional lower bounds for polynomial time solvable problems. In particular Abboud, Backurs, and Vassilevska Williams [6], and independently Bringmann and Künnemann [7] proved that LCS cannot be solved in $O(n^{2-\varepsilon})$ time unless the Strong Exponential Time Hypothesis (SETH) is false.

Hypothesis 1 *(Strong Exponential Time Hypothesis). There is no $\varepsilon > 0$ such that for all $k \geqslant 3$, k-SAT on N variables can be solved in $O(2^{(1-\varepsilon)N})$ time.*

Moreover, Bringmann and Künnemann [7] proposed a general framework for proving quadratic time hardness of sequence similarity measures. Within this framework, it is sufficient to show that a similarity measure *admits an align-*

E-mail address: polak@tcs.uj.edu.pl.

[1] This work was supported by the Polish Ministry of Science and Higher Education program *Diamentowy Grant* under grant no. DI2012 018942.

https://doi.org/10.1016/j.ipl.2017.11.007
0020-0190/

ment gadget to prove that this similarity measure cannot be computed in $O(n^{2-\varepsilon})$ time unless SETH is false. Besides LCS many other similarity measures, e.g. Edit Distance and Dynamic Time Warping, fall into this framework. However, it seems that neither LCIS nor LCWIS admits an alignment gadget.

In this paper we show that LCWIS cannot be solved in $O(n^{2-\varepsilon})$ time unless SETH is false.

Theorem 2. *If the Longest Common Weakly Increasing Subsequence problem for two sequences of length n can be solved in $O(n^{2-\varepsilon})$ time, then given a CNF formula on N variables and M clauses it is possible to compute the maximum number of satisfiable clauses (MAX-CNF-SAT) in $O(2^{(1-\varepsilon/2)N}\text{poly}(M))$ time.*

Our reduction is modeled after previous hardness results based on SETH, in particular [6] and [8]. We go through the Most-Orthogonal Vectors problem, and construct *vector gadgets* such that two vector gadgets have large LCWIS if and only if the corresponding vectors have small inner product. The crucial ingredient, presented in Lemma 9, is a construction that lets us combine many vector gadgets into two sequences such that their LCWIS depends on the largest LCWIS among all pairs of vector gadgets.

Unfortunately, our uncomplicated techniques are not sufficient to prove similar lower bounds neither for LCIS nor for the generalization of LCWIS to more than two sequences. Recently, a more involved construction has been proposed to establish tight lower bounds for both these problems [9].

Unlike $\mathsf{P} \neq \mathsf{NP}$ and several other common assumptions for conditional lower bounds in computational complexity, SETH is considered by many not a very safe working hypothesis. Recently, Abboud et al. [10] came up with a weaker assumption, which still allows to prove many previous SETH-based lower bounds. More specifically, they propose a reduction from satisfiability of *Branching Programs* [11] (BP-SAT) to LCS (and, in general, any other similarity measure which admits an alignment gadget). Their reduction implies that the existence of a strongly subquadratic time algorithm for LCS would have much more remarkable consequences in computational complexity than just refuting SETH, e.g. an exponential improvement over brute-force algorithm for satisfiability of NC circuits. For an in-depth discussion of consequences of their reduction and motivations to study such reductions please refer to the original paper [10]. Their main development, which makes the reduction possible, is the construction of *reachability gadgets*, which encode computations of Branching Programs in the language of LCS and play a role analogous to vector gadgets in previous reductions from CNF-SAT via Orthogonal Vectors problem. It is easy to devise similar reachability gadgets for LCWIS, by adapting the original construction. Then, our Lemma 9 can be applied to these reachability gadgets to obtain a reduction from BP-SAT to LCWIS, giving even stronger evidence of the quadratic time hardness of LCWIS.

2. Preliminaries

Let us start with the formal definition of the LCWIS problem.

Definition 3 *(Longest Common Weakly Increasing Subsequence). Given two sequences A and B over an alphabet Σ with a linear order $\leqslant_\Sigma$, the* Longest Common Weakly Increasing Subsequence *problem asks to find a sequence C such that*

- *it is weakly increasing with respect to $\leqslant_\Sigma$,*
- *it is a subsequence of both A and B,*
- *and its length is maximum possible.*

We denote the length of C by LCWIS(A, B).

For example, LCWIS$(\langle 1,2,5,2,5,3\rangle, \langle 2,4,5,2,3,4\rangle) = 3$, and the optimal subsequence is $\langle 2,2,3\rangle$.

To simplify further arguments we introduce, as an auxiliary problem, the weighted version of LCWIS.

Definition 4 *(Weighted Longest Common Weakly Increasing Subsequence). Given two sequences A and B over an alphabet Σ with a linear order $\leqslant_\Sigma$ and the* weight function $w : \Sigma \longrightarrow \mathbb{N}_+$, *the* Weighted Longest Common Weakly Increasing Subsequence *problem (WLCWIS) asks to find a sequence C such that*

- *it is weakly increasing with respect to $\leqslant_\Sigma$,*
- *it is a subsequence of both A and B,*
- *and its total weight, i.e. $\sum_{i=1}^{|C|} w(C_i)$, is maximum possible.*

We denote the total weight of C by WLCWIS(A, B).

Lemma 5. *Computing the* WLCWIS *of two sequences, each of total weight at most W, can be reduced to computing the* LCWIS *of two sequences, each of length at most W.*

Proof. For a sequence $X = \langle X_1, X_2, \ldots, X_{|X|}\rangle$ let $\widehat{X}$ denote a sequence obtained from X by replacing each symbol a by its $w(a)$ many copies, i.e.

$$\widehat{X} = X_1^{w(X_1)}\, X_2^{w(X_2)} \ldots X_{|X|}^{w(X_{|X|})}.$$

We will show that WLCWIS$(A, B) =$ LCWIS$(\widehat{A}, \widehat{B})$. Every common weakly increasing subsequence C of A and B translates to a common weakly increasing subsequence $\widehat{C}$ of $\widehat{A}$ and $\widehat{B}$, and the length of $\widehat{C}$ equals the total weight of C, thus WLCWIS$(A, B) \leqslant$ LCWIS$(\widehat{A}, \widehat{B})$.

It remains to prove WLCWIS$(A, B) \geqslant$ LCWIS$(\widehat{A}, \widehat{B})$. Let $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_{|\Sigma|}\}$, $\sigma_i <_\Sigma \sigma_{i+1}$. Let us represent the longest common weakly increasing subsequence of $\widehat{A}$ and $\widehat{B}$ as $\sigma_1^{\alpha_1}\sigma_2^{\alpha_2}\ldots\sigma_{|\Sigma|}^{\alpha_{|\Sigma|}}$. Note that such a representation is possible because the subsequence is weakly increasing. Let

$$C := \sigma_1^{\lceil \alpha_1/w(\sigma_1)\rceil}\sigma_2^{\lceil \alpha_2/w(\sigma_2)\rceil}\ldots\sigma_{|\Sigma|}^{\lceil \alpha_{|\Sigma|}/w(\sigma_{|\Sigma|})\rceil}.$$

Note that the total weight of C with respect to w is at least LCWIS$(\widehat{A}, \widehat{B})$. To finish the proof observe that C is a subsequence of both A and B. Indeed, C is a subsequence of A because, for each i, α_i occurrences of σ_i in $\widehat{A}$ must originate from at least $\lceil \alpha_i/w(\sigma_i)\rceil$ different occurrences of σ_i in

A and all this occurrences must come after all corresponding occurrences of σ_j for $j < i$ and before all corresponding occurrences of σ_j for $j > i$. The same argument can be applied to show that C is a subsequence of B. □

3. Reduction from MAX-CNF-SAT to LCWIS

This section is devoted to proving Theorem 2. We do this by showing a reduction from the Most-Orthogonal Vectors problem, introduced by Abboud, Backurs, and Vassilevska Williams [6].

Definition 6 *(Most-Orthogonal Vectors). Given two sets of vectors $U, V \subseteq \{0,1\}^d$, both of the same size n, and an integer $r \in \{0, 1, \ldots, d\}$, are there two vectors $u \in U$ and $v \in V$ such that their inner product does not exceed r, i.e. $u \cdot v := \sum_{i=1}^{d} u_i \cdot v_i \leqslant r$?*

Lemma 7 *(Abboud, Backurs, Vassilevska Williams [6]). If Most-Orthogonal Vectors on n vectors in $\{0,1\}^d$ can be solved in $T(n,d)$ time, then given a CNF formula on N variables and M clauses, we can compute the maximum number of satisfiable clauses (MAX-CNF-SAT), in $O(T(2^{N/2}, M) \cdot \log M)$ time.*

First, we show how to encode vectors into sequences, called *vector gadgets*, such that two vector gadgets have large LCWIS if and only if the corresponding vectors have small inner product. For the encoding we use alphabet Σ consisting of the integers from 3 to $3d+2$.

Lemma 8. *There exist $\mathrm{VG}_1, \mathrm{VG}_2 : \{0,1\}^d \longrightarrow \Sigma^*$, both computable in $O(d)$ time, such that for any two vectors $u, v \in \{0,1\}^d$*

$$LCWIS(\mathrm{VG}_1(u), \mathrm{VG}_2(v)) = d - (u \cdot v).$$

Moreover, the lengths of $\mathrm{VG}_1(u)$ and $\mathrm{VG}_2(v)$ do not exceed $2d$.

Proof. First we define two kinds of *coordinate gadgets*: $\mathrm{CG}_1, \mathrm{CG}_2 : \{0,1\} \times \{1,2,\ldots,d\} \longrightarrow \Sigma^*$.

$$\mathrm{CG}_1(0,i) = \langle 3i, 3i+1 \rangle$$
$$\mathrm{CG}_1(1,i) = \langle 3i+2 \rangle$$
$$\mathrm{CG}_2(0,i) = \langle 3i, 3i+2 \rangle$$
$$\mathrm{CG}_2(1,i) = \langle 3i+1 \rangle$$

Observe that

$$LCWIS(\mathrm{CG}_1(x,i), \mathrm{CG}_2(y,i)) = \begin{cases} 0, & \text{if } x = 1 \text{ and } y = 1, \\ 1, & \text{otherwise.} \end{cases}$$

Now we are ready to define the *vector gadgets*:

$$\mathrm{VG}_1(u_1, u_2, \ldots, u_d) =$$
$$= \mathrm{CG}_1(u_1, 1)\ \mathrm{CG}_1(u_2, 2)\ \ldots\ \mathrm{CG}_1(u_d, d)$$
$$\mathrm{VG}_2(u_1, u_2, \ldots, u_d) =$$
$$= \mathrm{CG}_2(u_1, 1)\ \mathrm{CG}_2(u_2, 2)\ \ldots\ \mathrm{CG}_2(u_d, d)$$

Observe that, for $u = (u_1, u_2, \ldots, u_d)$, $v = (v_1, v_2, \ldots, v_d) \in \{0,1\}^d$,

$$LCWIS(\mathrm{VG}_1(u), \mathrm{VG}_2(v)) =$$
$$= \sum_{i=1}^{d} LCWIS(\mathrm{CG}_1(u_i, i), \mathrm{CG}_2(v_i, i)) =$$
$$= \sum_{i=1}^{d} (1 - u_i \cdot v_i) = d - (u \cdot v),$$

as desired. □

The next lemma helps us combine many vector gadgets into two sequences such that computing their LCWIS lets us find two vector gadgets with the largest LCWIS, which correspond to the pair of most orthogonal vectors.

Lemma 9. *Given two sets of sequences $S = \{s_1, s_2, \ldots, s_n\} \subseteq \Sigma^*$, $T = \{t_1, t_2, \ldots, t_n\} \subseteq \Sigma^*$, each sequence of length at most ℓ, one can construct, in $O(n\ell)$ time, two sequences such that the length of their LCWIS equals*

$$\max_{1 \leqslant i,j \leqslant n} \mathrm{LCWIS}(s_i, t_j) + (4n-2) \cdot \ell.$$

The length of the constructed sequences is $O(n\ell)$, and they are defined over an alphabet of size $|\Sigma| + O(1)$.

Proof. In order to simplify the proof we go through WLCWIS problem. We put weight 1 for every symbol originally in Σ, and augment the alphabet with four additional symbols A, B, Y, Z, such that $\mathrm{A} < \mathrm{B} < \sigma < \mathrm{Y} < \mathrm{Z}$ for all $\sigma \in \Sigma$. Let the weights of the new symbols be $w(\mathrm{A}) = w(\mathrm{Z}) = \ell$ and $w(\mathrm{B}) = w(\mathrm{Y}) = 2\ell$. Finally, define two sequences P_1 and P_2,

$$P_1 = \mathrm{A}^{2n}\ s_1\ \mathrm{YB}\ s_2\ \mathrm{YB}\ \ldots\ \mathrm{YB}\ s_n\ \mathrm{Z}^{2n},$$
$$P_2 = (\mathrm{ZYBA})^n\ t_1\ \mathrm{ZYBA}\ t_2\ \mathrm{ZYBA}\ \ldots\ \mathrm{ZYBA}\ t_n\ (\mathrm{ZYBA})^n.$$

Observe that the total weight of P_1 and P_2 is $O(n\ell)$. After we prove that

$$\mathrm{WLCWIS}(P_1, P_2) = \max_{1 \leqslant i,j \leqslant n} \mathrm{LCWIS}(s_i, t_j) + (4n-2) \cdot \ell,$$

it will remain to apply Lemma 5 to obtain the desired sequences.

The proof consists of two parts, first we prove that a common weakly increasing subsequence of P_1 and P_2 with the claimed total weight exists, then we prove that there is no such subsequence with a larger total weight.

Take i and j maximizing $\mathrm{WLCWIS}(s_i, t_j)$, denote by Q the corresponding subsequence of s_i and t_j, and consider the following sequence:

$$Q' = \mathrm{A}^{n+j-1-(i-1)}\ \mathrm{B}^{i-1}\ Q\ \mathrm{Y}^{n-i}\ \mathrm{Z}^{2n-j-(n-i)}.$$

The construction of Q' is motivated by the following intuition. We start with Q, and append at the beginning one B for each YB-fragment appearing before s_i in P_1. There

are $i-1$ such fragments. On the other hand, the number of ZYBA-fragments appearing before t_j in P_2 equals $n+j-1$. We devote the rightmost $i-1$ of these fragments to find matching B symbols, and for each of the remaining fragments we take one A and append it at the beginning of Q'. The A^{2n} prefix of P_1 is long enough to match all these A symbols. In an analogous way we append Y and Z elements at the end of Q'.

Note that Q' is weakly increasing and it appears in both P_1 and P_2 as a subsequence. The total weight of Q' equals

$$\begin{aligned}&\ell(n+j-1-(i-1))+2\ell(i-1)+\mathrm{WLCWIS}(s_i,t_j)+\\&\quad+2\ell(n-i)+\ell(2n-j-(n-i))=\\&=\ell\cdot 2n+2\ell\cdot(n-1)+\mathrm{WLCWIS}(s_i,t_j)=\\&=\mathrm{WLCWIS}(s_i,t_j)+(4n-2)\cdot\ell.\end{aligned}$$

This finishes the first part of the proof.

Now we prove that there is no common weakly increasing subsequence of P_1 and P_2 with a larger total weight. Consider any such subsequence. Denote by c_S the number of s_i-fragments in P_1 that contribute at least one symbol to that subsequence. Analogously, denote by c_T the number of t_j-fragments in P_2 contributing at least one symbol. Finally, denote by c_A, c_B, c_Y, c_Z the number of symbols A, B, Y, Z in the subsequence, respectively. Observe that the contribution of A, B, Y, Z symbols to the total weight of the subsequence is

$$\begin{aligned}&c_A\cdot w(A)+c_B\cdot w(B)+c_Y\cdot w(Y)+c_Z\cdot w(Z)=\\&\quad=c_A\cdot\ell+c_B\cdot 2\ell+c_Y\cdot 2\ell+c_Z\cdot\ell=\\&\quad=(c_A+c_B+c_Y+c_Z)\cdot\ell+(c_B+c_Y)\cdot\ell.\end{aligned}\quad(1)$$

Now the proof splits into two cases:

Case 1: $c_S\leqslant 1$ *and* $c_T\leqslant 1$. The contribution of s_i- and t_j-fragments to the total weight of the subsequence is at most $\max_{1\leqslant i,j\leqslant n}\mathrm{WLCWIS}(s_i,t_j)$. The remaining weight of the subsequence must come from the symbols A, B, Y, Z. Each ZYBA-fragment in P_2 can contribute at most one symbol to the, weakly increasing, subsequence. There are $3n-1$ such fragments, therefore

$$c_A+c_B+c_Y+c_Z\leqslant 3n-1.$$

Similarly, each YB-fragment in P_1 can contribute at most one symbol. There are $n-1$ such fragments in P_2, therefore

$$c_B+c_Y\leqslant n-1.$$

Combining these two inequalities with the equation (1) we get that the contribution of A, B, Y, Z symbols to the total weight of the subsequence is at most $(3n-1)\cdot\ell+(n-1)\cdot\ell=(4n-2)\cdot\ell$. This together with the contribution of the s_i- and t_j-fragments gives the required upper bound for $\mathrm{WLCWIS}(P_1,P_2)$.

Case 2: $c_S\geqslant 2$ *or* $c_T\geqslant 2$. The contribution of s_i- and t_j-fragments to the total weight of the subsequence is at most $\min(c_S,c_T)\cdot\ell$. Since $\min(c_S,c_T)=c_S+c_T-\max(c_S,c_T)$ and at least one of c_S,c_T is at least 2, we can bound $\min(c_S,c_T)$ from above by c_S+c_T-2. Thus the considered contribution is at most

$$((c_S-1)+(c_T-1))\cdot\ell.$$

If two s_i-fragments contribute at least one symbol each, no YB-fragment between them can contribute, because the subsequence has to be weakly increasing. Therefore, the number of YB-fragments contributing is at most $n-1-(c_S-1)$, thus

$$c_B+c_Y\leqslant n-1-(c_S-1).$$

Similarly

$$c_A+c_B+c_Y+c_Z\leqslant 3n-1-(c_T-1).$$

Using the equation (1) we get that the total weight of the subsequence is at most

$$\begin{aligned}&((c_S-1)+(c_T-1))\cdot\ell+\\&\quad+(3n-1-(c_T-1))\cdot\ell+(n-1-(c_S-1))\cdot\ell,\end{aligned}$$

which is $(4n-2)\cdot\ell$. □

Now we are ready to complete our reduction from Most-Orthogonal Vectors to LCWIS.

Lemma 10. *If LCWIS of two sequences of length n can be computed in $O(n^{2-\varepsilon})$ time, then Most-Orthogonal Vectors for n vectors in $\{0,1\}^d$ can be solved in $O((dn)^{2-\varepsilon})$ time.*

Proof. Given two sets of vectors $U,V\subseteq\{0,1\}^d$, both of size n, we use Lemma 8 to construct two sets of vector gadgets $S=VG_1(U)$, $T=VG_2(V)\subseteq\Sigma^*$, such that

$$\max_{s\in S,\,t\in T}\mathrm{LCWIS}(s,t)=d-\min_{u\in U,\,v\in V}(u\cdot v).$$

Each of the two sets S,T contains n vector gadgets of length at most $2d$. Now we apply Lemma 9 to these sets and construct two sequences, of length $O(dn)$, such that their LCWIS equals

$$\begin{aligned}&\max_{s\in S,\,t\in T}\mathrm{LCWIS}(s,t)+(4n-2)\cdot 2d\\&\quad=(4n-2)\cdot 2d+d-\min_{u\in U,\,v\in V}(u\cdot v).\end{aligned}$$

All the constructions so far can be done in linear time. Finally, we execute the hypothesized subquadratic LCWIS algorithm, which takes $O((dn)^{2-\varepsilon})$ time and lets us determine the inner product of the pair of most orthogonal vectors. □

Proof of Theorem 2. This theorem is a direct conclusion from Lemma 7 and Lemma 10. □

4. Open problems

The reduction which we give in this paper does not work with a constant size alphabet. In the case of LCS it is possible to obtain conditional quadratic time lower bounds even for binary strings [7]. It is very unlikely to obtain similar results for LCWIS given the linear time algorithm for the 3-letter alphabet [5]. However, it is an open problem to show quadratic time hardness of LCWIS for an alphabet of some larger constant size.

References

[1] W. Masek, M. Paterson, A faster algorithm computing string edit distances, J. Comput. Syst. Sci. 20 (1980) 18–31, https://doi.org/10.1016/0022-0000(80)90002-1.

[2] I.-H. Yang, C.-P. Huang, K.-M. Chao, A fast algorithm for computing a longest common increasing subsequence, Inf. Process. Lett. 93 (5) (2005) 249–253, https://doi.org/10.1016/j.ipl.2004.10.014.

[3] Y. Sakai, A linear space algorithm for computing a longest common increasing subsequence, Inf. Process. Lett. 99 (5) (2006) 203–207, https://doi.org/10.1016/j.ipl.2006.05.005.

[4] M. Kutz, G.S. Brodal, K. Kaligosi, I. Katriel, Faster algorithms for computing longest common increasing subsequences, J. Discret. Algorithms 9 (4) (2011) 314–325, https://doi.org/10.1016/j.jda.2011.03.013, selected papers from the 17th Annual Symposium on Combinatorial Pattern Matching (CPM 2006).

[5] L. Duraj, A linear algorithm for 3-letter longest common weakly increasing subsequence, Inf. Process. Lett. 113 (3) (2013) 94–99, https://doi.org/10.1016/j.ipl.2012.11.007.

[6] A. Abboud, A. Backurs, V.V. Williams, Tight hardness results for LCS and other sequence similarity measures, in: Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS), FOCS '15, 2015, pp. 59–78.

[7] K. Bringmann, M. Künnemann, Quadratic conditional lower bounds for string problems and dynamic time warping, in: Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS), FOCS '15, 2015, pp. 79–97.

[8] A. Backurs, P. Indyk, Edit distance cannot be computed in strongly subquadratic time (unless SETH is false), in: Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC '15, 2015, pp. 51–58.

[9] L. Duraj, M. Künnemann, A. Polak, Tight conditional lower bounds for longest common increasing subsequence, in: 12th International Symposium on Parameterized and Exact Computation (IPEC 2017), Leibniz International Proceedings in Informatics (LIPIcs), 2017.

[10] A. Abboud, T.D. Hansen, V.V. Williams, R. Williams, Simulating branching programs with edit distance and friends: or: a polylog shaved is a lower bound made, in: Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC '16, 2016, pp. 375–388.

[11] S. Arora, B. Barak, Computational Complexity: A Modern Approach, 1st edition, Cambridge University Press, New York, NY, USA, 2009.

Tight Conditional Lower Bounds for Longest Common Increasing Subsequence*

Lech Duraj[†1], **Marvin Künnemann**[2], **and Adam Polak**[‡3]

1 **Theoretical Computer Science, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland**
duraj@tcs.uj.edu.pl

2 **Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany**
marvin@mpi-inf.mpg.de

3 **Theoretical Computer Science, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland**
polak@tcs.uj.edu.pl

Abstract

We consider the canonical generalization of the well-studied Longest Increasing Subsequence problem to multiple sequences, called k-LCIS: Given k integer sequences $X_1, \ldots, X_k$ of length at most n, the task is to determine the length of the longest common subsequence of $X_1, \ldots, X_k$ that is also strictly increasing. Especially for the case of $k = 2$ (called LCIS for short), several algorithms have been proposed that require quadratic time in the worst case.

Assuming the Strong Exponential Time Hypothesis (SETH), we prove a tight lower bound, specifically, that no algorithm solves LCIS in (strongly) subquadratic time. Interestingly, the proof makes no use of normalization tricks common to hardness proofs for similar problems such as LCS. We further strengthen this lower bound to rule out $\mathcal{O}\left((nL)^{1-\varepsilon}\right)$ time algorithms for LCIS, where L denotes the solution size, and to rule out $\mathcal{O}\left(n^{k-\varepsilon}\right)$ time algorithms for k-LCIS. We obtain the same conditional lower bounds for the related Longest Common Weakly Increasing Subsequence problem.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases fine-grained complexity, combinatorial pattern matching, sequence alignments, parameterized complexity, SETH

Digital Object Identifier 10.4230/LIPIcs.IPEC.2017.15

1 Introduction

The longest common subsequence problem (LCS) and its variants are computational primitives with a variety of applications, which includes, e.g., uses as similarity measures for spelling correction [36, 42] or DNA sequence comparison [38, 5], as well as determining the differences of text files as in the UNIX DIFF utility [27]. LCS shares characteristics of both an easy and a hard problem: (*Easy*) A simple and elegant dynamic-programming algorithm computes an LCS of two length-n sequences in time $\mathcal{O}\left(n^2\right)$ [42], and in many practical settings, certain properties of typical input sequences can be exploited to obtain faster, "tailored" solutions

* The full version of this paper is available at: http://arxiv.org/abs/1709.10075.
† Partially supported by Polish National Science Center grant 2016/21/B/ST6/02165.
‡ Partially supported by Polish Ministry of Science and Higher Education program *Diamentowy Grant*.

(e.g., [26, 28, 7, 37]; see also [13] for a survey). (*Hard*) At the same time, no polynomial improvements over the classical solution are known, thus exact computation may become infeasible for very long general input sequences. The research community has sought for a resolution of the question *"Do subquadratic algorithms for LCS exist?"* already shortly after the formalization of the problem [20, 4].

Recently, an answer conditional on the Strong Exponential Time Hypothesis (SETH; see Section 2 for a definition) could be obtained: Based on a line of research relating the satisfiability problem to quadratic-time problems [43, 40, 14, 3] and following a breakthrough result for Edit Distance [9], it has been shown that unless SETH fails, there is no (strongly) subquadratic-time algorithm for LCS [1, 15]. Subsequent work [2] strengthens these lower bounds to hold already under weaker assumptions and even provides surprising consequences of sufficiently strong polylogarithmic improvements.

Due to its popularity and wide range of applications, several variants of LCS have been proposed. This includes the heaviest common subsequence (HCS) [31], which introduces weights to the problem, as well as notions that constrain the structure of the solution, such as the longest common increasing subsequence (LCIS) [45], LCSk [12], constrained LCS [41, 19, 8], restricted LCS [25], and many other variants (see, e.g., [18, 6, 32]). Most of these variants are (at least loosely) motivated by biological sequence comparison tasks. To the best of our knowledge, in the above list, LCIS is the only LCS variant for which (1) the best known algorithms run in quadratic time in the worst case and (2) its definition does not include LCS as a special case (for such generalizations of LCS, the quadratic-time SETH hardness of LCS [1, 15] would transfer immediately). As such, it is open to determine whether there are (strongly) subquadratic algorithms for LCIS or whether such algorithms can be ruled out under SETH. The starting point of our work is to settle this question.

1.1 Longest Common Increasing Subsequence (LCIS)

The Longest Common Increasing Subsequence problem on k sequences (k-LCIS) is defined as follows: Given integer sequences $X_1, \ldots, X_k$ of length at most n, determine the length of the longest sequence Z such that Z is a strictly increasing sequence of integers and Z is a subsequence of each $X_i, i \in \{1, \ldots, k\}$. For $k = 1$, we obtain the well-studied longest increasing subsequence problem (LIS; we refer to [21] for an overview), which has an $\mathcal{O}(n \log n)$ time solution and a matching lower bound in the decision tree model [24]. The extension to $k = 2$, denoted simply as LCIS, has been proposed by Yang, Huang, and Chao [45], partially motivated as a generalization of LIS and by potential applications in bioinformatics. They obtained an $\mathcal{O}\left(n^2\right)$ time algorithm, leaving open the natural question whether there exists a way to extend the near-linear time solution for LIS to a near-linear time solution for multiple sequences.

Interestingly, already a classic connection between LCS and LIS combined with a recent conditional lower bound of Abboud, Backurs and Vassilevska Williams [1] yields a partial negative answer assuming SETH.

▶ **Observation 1** (Folklore reduction, implicit in [28], explicit in [31]). *After $\mathcal{O}\left(kn^2\right)$ time preprocessing, we can solve k-LCS by a single call to $(k-1)$-LCIS on sequences of length at most n^2.*

Note that by the above reduction, an $\mathcal{O}\left(n^{\frac{3}{2}-\varepsilon}\right)$ time LCIS algorithm would give an $\mathcal{O}\left(n^{3-2\varepsilon}\right)$ time algorithm for 3-LCS, which would refute SETH by a result of Abboud et al. [1].

▶ **Corollary 2.** *Unless SETH fails, there is no $\mathcal{O}\left(n^{\frac{3}{2}-\varepsilon}\right)$ time algorithm for LCIS for any constant $\varepsilon > 0$.*

While this rules out near-linear time algorithms, still an unsatisfying large polynomial gap between best upper and conditional lower bounds persists.

1.2 Our Results

Our first result is a tight SETH-based lower bound for LCIS.

▶ **Theorem 3.** *Unless SETH fails, there is no* $\mathcal{O}\left(n^{2-\varepsilon}\right)$ *time algorithm for LCIS for any constant* $\varepsilon > 0$.

We extend our main result in several directions.

1.2.1 Parameterized Complexity I: Solution Size

Subsequent work [17, 34] improved over Yang et al.'s algorithm when certain input parameters are small. Here, we focus particularly on the solution size, i.e., the length L of the LCIS. Kutz et al. [34] provided an algorithm running in time $\mathcal{O}\left(nL \log\log n + n \log n\right)$. If L is small compared to its worst-case upper bound of n, say $L = n^{\frac{1}{2}\pm o(1)}$, this algorithm runs in strongly subquadratic time. Interestingly, exactly for this case, the reduction from 3-LCS to LCIS of Observation 1 already yields a matching SETH-based lower bound of $(Ln)^{1-o(1)} = n^{\frac{3}{2}-o(1)}$. However, for smaller L, this reduction yields no lower bound at all and only a non-matching lower bound for larger L. We remedy this situation by the following result.[1]

▶ **Theorem 4.** *Unless SETH fails, there is no* $\mathcal{O}\left((nL)^{1-\varepsilon}\right)$ *time algorithm for LCIS for any constant* $\varepsilon > 0$. *This even holds restricted to instances with* $L = n^{\gamma\pm o(1)}$, *for* arbitrarily chosen $0 < \gamma \leqslant 1$.

1.2.2 Parameterized Complexity II: k-LCIS

For constant $k \geqslant 3$, we can solve k-LCIS in $\mathcal{O}\left(n^k \text{polylog}(n)\right)$ time [17, 34], or even $\mathcal{O}\left(n^k\right)$ time (see the appendix in the full version). While it is known that k-LCS cannot be computed in time $\mathcal{O}\left(n^{k-\varepsilon}\right)$ for any constant $\varepsilon > 0, k \geqslant 2$ unless SETH fails [1], this does not directly transfer to k-LCIS, since the reduction in Observation 1 is not tight. However, by extending our main construction, we can prove the analogous result.

▶ **Theorem 5.** *Unless SETH fails, there is no* $\mathcal{O}\left(n^{k-\varepsilon}\right)$ *time algorithm for* k*-LCIS for any constant* $k \geqslant 3$ *and* $\varepsilon > 0$.

1.2.3 Longest Common Weakly Increasing Subsequence (LCWIS)

We consider a closely related variant of LCIS called the Longest Common Weakly Increasing Subsequence (k-LCWIS): Here, given integer sequences $X_1, \ldots, X_k$ of length at most n, the task is to determine the longest *weakly increasing* (i.e. non-decreasing) integer sequence Z that is a common subsequence of $X_1, \ldots, X_k$. Again, we write LCWIS as a shorthand for 2-LCWIS. Note that the seemingly small change in the notion of increasing sequence has a major impact on algorithmic and hardness results: Any instance of LCIS in which the input sequences are defined over a small-sized alphabet $\Sigma \subseteq \mathbb{Z}$, say $|\Sigma| = \mathcal{O}\left(n^{1/2}\right)$, can be solved in strongly subquadratic time $\mathcal{O}\left(nL \log n\right) = \mathcal{O}\left(n^{3/2} \log n\right)$ [34], by using the fact that $L \leqslant |\Sigma|$. In contrast, LCWIS is quadratic-time SETH hard already over slightly

[1] We mention in passing that a systematic study of the complexity of LCS in terms of such input parameters has been performed recently in [16].

superlogarithmic-sized alphabets [39]. We give a substantially different proof for this fact and generalize it to k-LCWIS.

▶ **Theorem 6.** *Unless SETH fails, there is no $\mathcal{O}\left(n^{k-\varepsilon}\right)$ time algorithm for k-LCWIS for any constant $k \geqslant 3$ and $\varepsilon > 0$. This even holds restricted to instances defined over an alphabet of size $|\Sigma| \leqslant f(n) \log n$ for any function $f(n) = \omega(1)$ growing arbitrarily slowly.*

1.3 Discussion, Outline and Technical Contributions

Apart from an interest in LCIS and its close connection to LCS, our work is also motivated by an interest in the *optimality of dynamic programming (DP) algorithms*[2]. Notably, many conditional lower bounds in P target problems with natural DP algorithms that are proven to be near-optimal under some plausible assumption (see, e.g., [14, 3, 9, 10, 1, 15, 11, 22, 33] and [44] for an introduction to the field). Even if we restrict our attention to problems that find optimal sequence alignments under some restrictions, such as LCS, Edit Distance and LCIS, the currently known hardness proofs differ significantly, despite seemingly small differences between the problem definitions. Ideally, we would like to classify the properties of a DP formulation which allow for matching conditional lower bounds.

One step in this direction is given by the *alignment gadget framework* [15]. Exploiting normalization tricks, this framework gives an abstract property of sequence similarity measures to allow for SETH-based quadratic lower bounds. Unfortunately, as it turns out, we cannot directly transfer the alignment gadget hardness proof for LCS to LCIS – some indication for this difficulty is already given by the fact that LCIS can be solved in strongly subquadratic time over sublinear-sized alphabets [34], while the LCS hardness proof already applies to binary alphabets. By collecting gadgetry needed to overcome such difficulties (that we elaborate on below), we hope to provide further tools to generalize more and more quadratic-time lower bounds based on SETH.

1.3.1 Technical Challenges

The known conditional lower bounds for global alignment problems such as LCS and Edit Distance work as follows. The reductions start from the quadratic-time SETH-hard Orthogonal Vectors problem (OV), that asks to determine, given two sets of $(0,1)$-vectors $\mathcal{U} = \{u_0, \ldots, u_{n-1}\}, \mathcal{V} = \{v_0, \ldots, v_{n-1}\} \subseteq \{0,1\}^d$ over $d = n^{o(1)}$ dimensions, whether there is a pair i, j such that u_i and v_j are orthogonal, i.e., whose inner product $(u_i \cdot v_j) := \sum_{k=0}^{d-1} u_i[k] \cdot v_j[k]$ is 0 (over the integers). Each vector u_i and v_j is represented by a (normalized) vector gadget $\mathrm{VG}_{\mathrm{X}}(u_i)$ and $\mathrm{VG}_{\mathrm{Y}}(v_j)$, respectively. Roughly speaking, these gadgets are combined to sequences X and Y such that each candidate for an optimal alignment of X and Y involves locally optimal alignments between n pairs $\mathrm{VG}_{\mathrm{X}}(u_i), \mathrm{VG}_{\mathrm{Y}}(v_j)$ – the optimal alignment exceeds a certain threshold if and only if there is an orthogonal pair u_i, v_j.

An analogous approach does not work for LCIS: Let $\mathrm{VG}_{\mathrm{X}}(u_i)$ be defined over an alphabet Σ and $\mathrm{VG}_{\mathrm{X}}(u_{i'})$ over an alphabet Σ'. If Σ and Σ' overlap, then $\mathrm{VG}_{\mathrm{X}}(u_i)$ and $\mathrm{VG}_{\mathrm{X}}(u_{i'})$ cannot both be aligned in an optimal alignment without interference with each other. On the other hand, if Σ and Σ' are disjoint, then each vector v_j should have its corresponding vector gadget $VG_{\mathrm{Y}}(v_j)$ defined over both Σ and Σ' to enable to align $\mathrm{VG}_{\mathrm{X}}(u_i)$ with $\mathrm{VG}_{\mathrm{Y}}(v_j)$ as well as $\mathrm{VG}_{\mathrm{X}}(u_{i'})$ with $\mathrm{VG}_{\mathrm{Y}}(v_j)$. The latter option drastically increases the size of vector gadgets.

[2] We refer to [46] for a simple quadratic-time DP formulation for LCIS.

Thus, we must define all vector gadgets over a common alphabet Σ and make sure that *only a single pair* $\mathrm{VG}_X(u_i), \mathrm{VG}_Y(v_j)$ is aligned in an optimal alignment (in contrast with n pairs aligned in the previous reductions for LCS and Edit Distance).

1.3.2 Technical Contributions and Proof Outline

Fortunately, a surprisingly simple approach works: As a key tool, we provide *separator sequences* $\alpha_0 \dots \alpha_{n-1}$ and $\beta_0 \dots \beta_{n-1}$ with the following properties: (1) for every $i, j \in \{0, \dots, n-1\}$ the LCIS of $\alpha_0 \dots \alpha_i$ and $\beta_0 \dots \beta_j$ has a length of $f(i+j)$, where f is a linear function, and (2) $\sum_i |\alpha_i|$ and $\sum_j |\beta_j|$ are bounded by $n^{1+o(1)}$. Note that existence of such a gadget is somewhat unintuitive: condition (1) for $i = 0$ and $j = n-1$ requires $|\alpha_0| = \Omega(n)$, yet still the total length $\sum_i |\alpha_i|$ must not exceed the length of $|\alpha_0|$ significantly. Indeed, we achieve this by a careful inductive construction that generates such sequences with heavily varying block sizes $|\alpha_i|$ and $|\beta_j|$.

We apply these separator sequences as follows. We first define simple vector gadgets $\mathrm{VG}_X(u_i), \mathrm{VG}_Y(v_j)$ over an alphabet Σ such that the length of an LCIS of $\mathrm{VG}_X(u_i)$ and $\mathrm{VG}_Y(v_j)$ is $d - (u_i \cdot v_j)$. Then we construct the separator sequences as above over an alphabet $\Sigma_<$ whose elements are strictly smaller than all elements in Σ. Furthermore, we create analogous separator sequences $\alpha'_0 \dots \alpha'_{n-1}$ and $\beta'_0 \dots \beta'_{n-1}$ which satisfy a property like (1) for all suffixes instead of prefixes, using an alphabet $\Sigma_>$ whose elements are strictly larger than all elements in Σ. Now, we define

$$X = \alpha_0 \mathrm{VG}_X(u_0) \alpha'_0 \dots \alpha_{n-1} \mathrm{VG}_X(u_{n-1}) \alpha'_{n-1},$$
$$Y = \beta_0 \mathrm{VG}_Y(v_0) \beta'_0 \dots \beta_{n-1} \mathrm{VG}_Y(v_{n-1}) \beta'_{n-1}.$$

As we will show in Section 3, the length of an LCIS of X and Y is $C - \min_{i,j}(u_i \cdot v_j)$ for some constant C depending only on n and d.

In contrast to previous such OV-based lower bounds, we use heavily varying separators (paddings) between vector gadgets.

1.4 Paper organization

After setting up conventions and introducing our hardness assumptions in Section 2, we give the main construction, i.e., the hardness of LCIS in Section 3. The proofs of Theorems 4, 5 and 6 can be found in the full version. We conclude with some open problems in Section 4.

2 Preliminaries

As a convention, we use capital or Greek letters to denote sequences over integers. Let X, Y be integer sequences. We write $|X|$ for the length of X, $X[k]$ for the k-th element in the sequence X ($k \in \{0, \dots, |X|-1\}$), and $X \circ Y = XY$ for the concatenation of X and Y. We say that Y is a subsequence of X if there exist indices $0 \leqslant i_1 < i_2 < \dots < i_{|Y|} \leqslant |X| - 1$ such that $X[i_k] = Y[k]$ for all $k \in \{0, \dots, |Y|-1\}$. Given any number of sequences $X_1, \dots, X_k$, we say that Y is a common subsequence of $X_1, \dots, X_k$ if Y is a subsequence of each $X_i, i \in \{1, \dots, k\}$. X is called strictly increasing (or weakly increasing) if $X[0] < X[1] < \dots < X[|X|-1]$ (or $X[0] \leqslant X[1] \leqslant \dots \leqslant X[|X|-1]$). For any k sequences $X_1, \dots, X_k$, we denote by $\mathrm{lcis}(X_1, \dots, X_k)$ the length of their longest common subsequence that is strictly increasing.

All of our lower bounds hold assuming the Strong Exponential Time Hypothesis (SETH), introduced by Impagliazzo and Paturi [29, 30]. It essentially states that no exponential speed-up over exhaustive search is possible for the CNF satisfiability problem.

▶ **Hypothesis 7** (Strong Exponential Time Hypothesis (SETH)). *There is no $\varepsilon > 0$ such that for all $d \geqslant 3$ there is an $\mathcal{O}\left(2^{(1-\varepsilon)n}\right)$ time algorithm for d-SAT.*

This hypothesis implies tight hardness of the k-Orthogonal Vectors problem (k-OV), which will be the starting point of our reductions: Given k sets $\mathcal{U}_1, \ldots, \mathcal{U}_k \subseteq \{0,1\}^d$, each with $|\mathcal{U}_i| = n$ vectors over $d = n^{o(1)}$ dimensions, determine whether there is a k-tuple $(u_1, \ldots, u_k) \in \mathcal{U}_1 \times \cdots \times \mathcal{U}_k$ such that $\sum_{\ell=0}^{d-1} \prod_{i=1}^{k} u_i[\ell] = 0$. By exhaustive enumeration, it can be solved in time $\mathcal{O}\left(n^k d\right) = n^{k+o(1)}$. The following conjecture is implied by SETH by the well-known split-and-list technique of Williams [43] (and the sparsification lemma [30]).

▶ **Hypothesis 8** (k-OV conjecture). *Let $k \geqslant 2$. There is no $\mathcal{O}\left(n^{k-\varepsilon}\right)$ time algorithm for k-OV, with $d = \omega(\log n)$, for any constant $\varepsilon > 0$.*

For the special case of $k = 2$, which we simply denote by OV, we obtain the following weaker conjecture.

▶ **Hypothesis 9** (OV conjecture). *There is no $\mathcal{O}\left(n^{2-\varepsilon}\right)$ time algorithm for OV, with $d = \omega(\log n)$, for any constant $\varepsilon > 0$. Equivalently, even restricted to instances with $|\mathcal{U}_1| = n$ and $|\mathcal{U}_2| = n^\gamma$, $0 < \gamma \leqslant 1$, there is no $\mathcal{O}\left(n^{1+\gamma-\varepsilon}\right)$ time algorithm for OV, with $d = \omega(\log n)$, for any constant $\varepsilon > 0$.*

A proof of the folklore equivalence of the statements for equal and unequal set sizes can be found, e.g., in [15].

3 Main Construction: Hardness of LCIS

In this section, we prove quadratic-time SETH hardness of LCIS, i.e., prove Theorem 3. We first introduce an *inflation* operation, which we then use to construct our separator sequences. After defining simple vector gadgets, we show how to embed an Orthogonal Vectors instance using our vector gadgets and separator sequences.

3.1 Inflation

We begin by introducing the inflation operation, which simulates weighing the sequences.

▶ **Definition 10.** For a sequence $A = \langle a_0, a_1, \ldots, a_{n-1} \rangle$ of integers we define:

$$\operatorname{inflate}(A) = \langle 2a_0 - 1, 2a_0, 2a_1 - 1, 2a_1, \ldots, 2a_{n-1} - 1, 2a_{n-1} \rangle .$$

▶ **Lemma 11.** *For any two sequences A and B, $\operatorname{lcis}(\operatorname{inflate}(A), \operatorname{inflate}(B)) = 2 \cdot \operatorname{lcis}(A, B)$.*

Proof. Let C be the longest common increasing subsequence of A and B. Observe that $\operatorname{inflate}(C)$ is a common increasing subsequence of $\operatorname{inflate}(A)$ and $\operatorname{inflate}(B)$ of length $2 \cdot |C|$, thus $\operatorname{lcis}(\operatorname{inflate}(A), \operatorname{inflate}(B)) \geqslant 2 \cdot \operatorname{lcis}(A, B)$.

Conversely, let $\bar{A}$ denote $\operatorname{inflate}(A)$ and $\bar{B}$ denote $\operatorname{inflate}(B)$. Let $\bar{C}$ be the longest common increasing subsequence of $\bar{A}$ and $\bar{B}$. If we divide all elements of $\bar{C}$ by 2 and round up to the closest integer, we end up with a weakly increasing sequence. Now, if we remove duplicate elements to make this sequence strictly increasing, we obtain C, a common increasing subsequence of A and B. At most 2 distinct elements may become equal after division by 2 and rounding, therefore C contains at least $\lceil \operatorname{lcis}(\bar{A}, \bar{B})/2 \rceil$ elements, so $2 \cdot \operatorname{lcis}(A, B) \geqslant \operatorname{lcis}(\bar{A}, \bar{B})$. This completes the proof. ◀

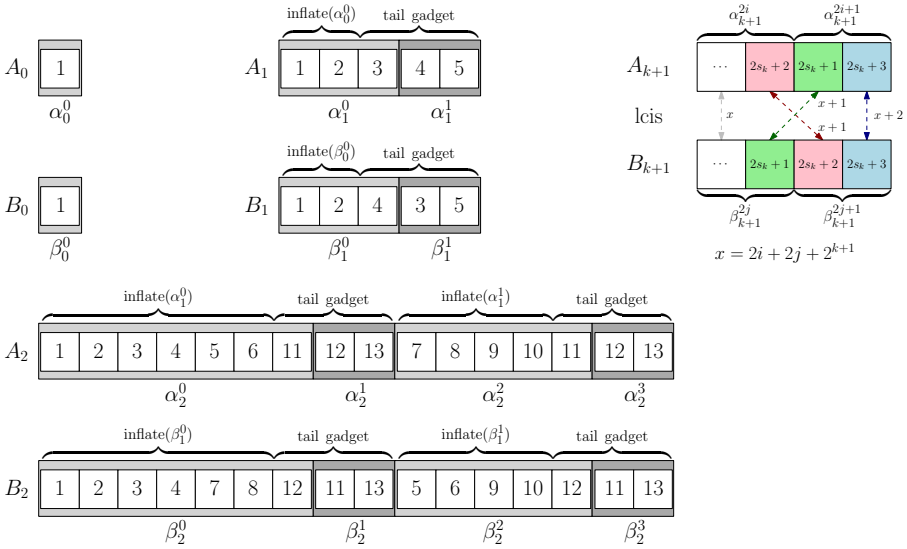

Figure 1 Initial steps of inductive construction of separator sequences (left), and intuition behind tail gadgets (right).

3.2 Separator sequences

Our goal is to construct two sequences A and B which can be split into n blocks, i.e. $A = \alpha_0\alpha_1 \dots \alpha_{n-1}$ and $B = \beta_0\beta_1 \dots \beta_{n-1}$, such that the length of the longest common increasing subsequence of the first i blocks of A and the first j blocks of B equals $i + j$, up to an additive constant. We call A and B *separator sequences*, and use them later to separate vector gadgets in order to make sure that only one pair of gadgets may interact with each other at the same time.

We construct the separator sequences inductively. For every $k \in \mathbb{N}$, the sequences A_k and B_k are concatenations of 2^k blocks (of varying sizes), $A_k = \alpha_k^0\alpha_k^1 \dots \alpha_k^{2^k-1}$ and $B_k = \beta_k^0\beta_k^1 \dots \beta_k^{2^k-1}$. Let s_k denote the largest element of both sequences. As we will soon observe, $s_k = 2^{k+2} - 3$.

The construction works as follows: for $k = 0$, we can simply set A_0 and B_0 as one-element sequences $\langle 1 \rangle$. We then construct A_{k+1} and B_{k+1} inductively from A_k and B_k in two steps. First, we inflate both A_k and B_k, then after each (now inflated) block we insert 3-element sequences, called *tail gadgets*, $\langle 2s_k + 2, 2s_k + 1, 2s_k + 3 \rangle$ for A_{k+1} and $\langle 2s_k + 1, 2s_k + 2, 2s_k + 3 \rangle$ for B_{k+1}. Formally, we describe the construction by defining blocks of the new sequences. For $i \in \{0, 1, \dots, 2^k - 1\}$,

$$\alpha_{k+1}^{2i} = \text{inflate}(\alpha_k^i) \circ \langle 2s_k + 2 \rangle, \qquad \alpha_{k+1}^{2i+1} = \langle 2s_k + 1, 2s_k + 3 \rangle,$$
$$\beta_{k+1}^{2i} = \text{inflate}(\beta_k^i) \circ \langle 2s_k + 1 \rangle, \qquad \beta_{k+1}^{2i+1} = \langle 2s_k + 2, 2s_k + 3 \rangle.$$

Note that the symbols appearing in tail gadgets do not appear in the inflated sequences. The largest element of both new sequences s_{k+1} equals $2s_k + 3$, and solving the recurrence gives indeed $s_k = 2^{k+2} - 3$.

Now, let us prove two useful properties of the separator sequences.

▶ **Lemma 12.** $|A_k| = |B_k| = \left(\frac{3}{2}k + 1\right) \cdot 2^k = \mathcal{O}\left(k2^k\right)$.

Proof. Observe that $|A_{k+1}| = 2|A_k| + 3 \cdot 2^k$. Indeed, to obtain A_{k+1} first we double the size of A_k and then add 3 new elements for each of the 2^k blocks of A_k. Solving the recurrence completes the proof. The same reasoning applies to B_k. ◀

▶ **Lemma 13.** *For every* $i, j \in \{0, 1, \ldots, 2^k - 1\}$, $\mathrm{lcis}(\alpha_k^0 \ldots \alpha_k^i, \beta_k^0 \ldots \beta_k^j) = i + j + 2^k$.

Proof. The proof is by induction on k. Assume the statement is true for k and let us prove it for $k+1$.

The "$\geqslant$" direction. First, consider the case when both i and j are even. Observe that $\mathrm{inflate}(\alpha_k^0 \ldots \alpha_k^{i/2})$ and $\mathrm{inflate}(\beta_k^0 \ldots \beta_k^{j/2})$ are subsequences of $\alpha_{k+1}^0 \ldots \alpha_{k+1}^i$ and $\beta_{k+1}^0 \ldots \beta_{k+1}^j$, respectively. Thus, using the induction hypothesis and inflation properties,

$$\begin{aligned}\mathrm{lcis}(\alpha_{k+1}^0 \ldots \alpha_{k+1}^i, \beta_{k+1}^0 \ldots \beta_{k+1}^j) &\geqslant \mathrm{lcis}(\mathrm{inflate}(\alpha_k^0 \ldots \alpha_k^{i/2}), \mathrm{inflate}(\beta_k^0 \ldots \beta_k^{j/2})) = \\ &= 2 \cdot \mathrm{lcis}(\alpha_k^0 \ldots \alpha_k^{i/2}, \beta_k^0 \ldots \beta_k^{j/2}) = 2 \cdot (i/2 + j/2 + 2^k) = i + j + 2^{k+1}.\end{aligned}$$

If i is odd and j is even, refer to the previous case to get a common increasing subsequence of $\alpha_{k+1}^0 \ldots \alpha_{k+1}^{i-1}$ and $\beta_{k+1}^0 \ldots \beta_{k+1}^j$ of length $i - 1 + j + 2^{k+1}$ consisting only of elements less than or equal to $2s_k$, and append the element $2s_k + 1$ to the end of it. Analogously, for i even and j odd, take such an LCIS of $\alpha_{k+1}^0 \ldots \alpha_{k+1}^i$ and $\beta_{k+1}^0 \ldots \beta_{k+1}^{j-1}$, and append $2s_k + 2$. Finally, for both i and j odd, take an LCIS of $\alpha_{k+1}^0 \ldots \alpha_{k+1}^{i-1}$ and $\beta_{k+1}^0 \ldots \beta_{k+1}^{j-1}$, and append $2s_k + 1$ and $2s_k + 3$.

The "$\leqslant$" direction. We proceed by induction on $i + j$. Fix i and j, and let L be a longest common increasing subsequence of $\alpha_{k+1}^0 \ldots \alpha_{k+1}^i$ and $\beta_{k+1}^0 \ldots \beta_{k+1}^j$.

If the last element of L is less than or equal to $2s_k$, L is in fact a common increasing subsequence of $\mathrm{inflate}(\alpha_k^0 \ldots \alpha_k^{\lfloor i/2 \rfloor})$ and $\mathrm{inflate}(\beta_k^0 \ldots \beta_k^{\lfloor j/2 \rfloor})$, thus, by the induction hypothesis and inflation properties, $|L| \leqslant 2 \cdot (\lfloor i/2 \rfloor + \lfloor j/2 \rfloor + 2^k) \leqslant i + j + 2^{k+1}$.

The remaining case is when the last element of L is greater than $2s_k$. In this case, consider the second-to-last element of L. It must belong to some blocks $\alpha_{k+1}^{i'}$ and $\beta_{k+1}^{j'}$ for $i' \leqslant i$ and $j' \leqslant j$, and we claim that $i = i'$ and $j = j'$ cannot hold simultaneously: by construction of separator sequences, if blocks α_{k+1}^i and β_{k+1}^j have a common element larger than $2s_k$, then it is the only common element of these two blocks. Therefore, it cannot be the case that both $i = i'$ and $j = j'$, because the last two elements of L would then be located in α_{k+1}^i and β_{k+1}^j. As a consequence, $i' + j' < i + j$, which lets us apply the induction hypothesis to reason that the prefix of L omitting its last element is of length at most $i' + j' + 2^{k+1}$. Therefore, $|L| \leqslant 1 + i' + j' + 2^{k+1} \leqslant i + j + 2^{k+1}$, which completes the proof. ◀

Observe that if we reverse the sequences A_k and B_k along with changing all elements to their negations, i.e. x to $-x$, we obtain sequences $\hat{A}_k$ and $\hat{B}_k$ such that $\hat{A}_k$ splits into 2^k blocks $\hat{\alpha}_k^0 \ldots \hat{\alpha}_k^{2^k-1}$, $\hat{B}_k$ splits into 2^k blocks $\hat{\beta}_k^0 \ldots \hat{\beta}_k^{2^k-1}$, and

$$\mathrm{lcis}(\hat{\alpha}_k^i \ldots \hat{\alpha}_k^{2^k-1}, \hat{\beta}_k^j \ldots \hat{\beta}_k^{2^k-1}) = 2 \cdot (2^k - 1) - i - j + 2^k. \tag{1}$$

Finally, observe that we can add any constant to all elements of the sequences A_k and B_k (as well as $\hat{A}_k$ and $\hat{B}_k$) without changing the property stated in Lemma 13 (and its analogue for $\hat{A}_k$ and $\hat{B}_k$, i.e. Equation (1)).

3.3 Vector gadgets

Let $\mathcal{U} = \{u_0, \ldots, u_{n-1}\}$ and $\mathcal{V} = \{v_0, \ldots, v_{n-1}\}$ be two sets of d-dimensional $(0,1)$-vectors.

For $i \in \{0, 1, \ldots, n-1\}$ let us construct the vector gadgets U_i and V_i as $2d$-element sequences, by defining, for every $p \in \{0, 1, \ldots, d-1\}$,

$$(U_i[2p-1], U_i[2p]) = \begin{cases} (2p-1, 2p) & \text{if } u_i[p] = 0, \\ (2p-1, 2p-1) & \text{if } u_i[p] = 1, \end{cases}$$

$$(V_i[2p-1], V_i[2p]) = \begin{cases} (2p, 2p-1) & \text{if } v_i[p] = 0, \\ (2p, 2p) & \text{if } v_i[p] = 1. \end{cases}$$

Observe that at most one of the elements $2p-1$ and $2p$ may appear in the LCIS of U_i and V_j, and it happens if and only if $u_i[p]$ and $v_j[p]$ are not both equal to one. Therefore, $\operatorname{lcis}(U_i, V_j) = d - (u_i \cdot v_j)$, and, in particular, $\operatorname{lcis}(U_i, V_j) = d$ if and only if u_i and v_j are orthogonal.

3.4 Final construction

To put all the pieces together, we plug vector gadgets U_i and V_j into the separator sequences from Section 3.2, obtaining two sequences whose LCIS depends on the minimal inner product of vectors u_i and v_j. We provide a general construction of such sequences, which will be useful for proving further results in the full version of the paper.

▶ **Lemma 14.** *Let $X_0, X_1, \ldots, X_{n-1}$, $Y_0, Y_1, \ldots, Y_{n-1}$ be integer sequences such that none of them has an increasing subsequence longer than δ. Then there exist sequences X and Y of length $\mathcal{O}(\delta \cdot n \log n) + \sum |X_i| + \sum |Y_j|$, constructible in linear time, such that:*

$$\operatorname{lcis}(X, Y) = \max_{i,j} \operatorname{lcis}(X_i, Y_j) + C$$

for a constant C that only depends on n and δ and is $\mathcal{O}(n\delta)$.

Proof. We can assume that $n = 2^k$ for some positive integer k, adding some dummy sequences if necessary. Recall the sequences A_k, B_k, $\hat{A}_k$ and $\hat{B}_k$ constructed in Section 3.2. Let $A, B, \hat{A}, \hat{B}$ be the sequences obtained from $A_k, B_k, \hat{A}_k, \hat{B}_k$ by applying inflation $\lceil \log_2 \delta \rceil$ times (thus increasing their length by a factor of $\ell = 2^{\lceil \log_2 \delta \rceil} \geqslant \delta$). Each of these four sequences splits into (now inflated) blocks, e.g. $A = \alpha_0 \alpha_1 \ldots \alpha_{n-1}$, where $\alpha_i = \operatorname{inflate}^{\lceil \log_2 \delta \rceil}(\alpha_k^i)$.

We subtract from A and B a constant large enough for all their elements to be smaller than all elements of every X_i and Y_j. Similarly, we add to A' and B' a constant large enough for all their elements to be larger than all elements of every X_i and Y_j. Now, we can construct the sequences X and Y as follows:

$$X = \alpha_0 X_0 \hat{\alpha}_0 \alpha_1 X_1 \hat{\alpha}_1 \ldots \alpha_{n-1} X_{n-1} \hat{\alpha}_{n-1},$$
$$Y = \beta_0 Y_0 \hat{\beta}_0 \beta_1 Y_1 \hat{\beta}_1 \ldots \beta_{n-1} Y_{n-1} \hat{\beta}_{n-1}.$$

We claim that

$$\operatorname{lcis}(X, Y) = \ell \cdot (4n-2) + M, \text{ where } M = \max_{i,j} \operatorname{lcis}(X_i, Y_j).$$

Let X_i and Y_j be the pair of sequences achieving $\operatorname{lcis}(X_i, Y_j) = M$. Recall that $\operatorname{lcis}(\alpha_0 \ldots \alpha_i, \beta_0 \ldots \beta_j) = \ell \cdot (i + j + n)$, with all the elements of this common subsequence preceding the elements of X_i and Y_j in X and Y, respectively, and being smaller than them. In the same way $\operatorname{lcis}(\hat{\alpha}_i \ldots \hat{\alpha}_{n-1}, \hat{\beta}_j \ldots \hat{\beta}_{n-1}) = \ell \cdot (2 \cdot (n-1) - (i+j) + n)$ with all the elements of LCIS being greater and appearing later than those of X_i and Y_j. By

concatenating these three sequences we obtain a common increasing subsequence of X and Y of length $\ell \cdot (4n-2) + M$.

We defer the simple remainder of the proof, i.e., proving $\text{lcis}(X,Y) \leqslant \ell \cdot (4n-2) + M$ to the full version of the paper. ◀

Proof of Theorem 3. Let $\mathcal{U} = \{u_0, \ldots, u_{n-1}\}$, $\mathcal{V} = \{v_0, \ldots, v_{n-1}\}$ be two sets of binary vectors in d dimensions. In Section 3.3 we constructed vector gadgets U_i and V_j, for $i, j \in \{0, 1, \ldots, n-1\}$, such that $\text{lcis}(U_i, V_j) = d - (u_i \cdot v_j)$. To these sequences we apply Lemma 14, with $\delta = 2d$, obtaining sequences X and Y of length $\mathcal{O}(n \log n \text{poly}(d))$ such that $\text{lcis}(X,Y) = C + d - \min_{i,j}(u_i \cdot v_j)$ for a constant C. This reduction, combined with an $\mathcal{O}\left(n^{2-\varepsilon}\right)$ time algorithm for LCIS, would yield an $\mathcal{O}\left(n^{2-\varepsilon}\text{polylog}(n)\text{poly}(d)\right)$ algorithm for OV, refuting Hypothesis 9 and, in particular, SETH. ◀

4 Conclusion and Open Problems

We prove a tight quadratic lower bound for LCIS, ruling out strongly subquadratic-time algorithms under SETH. It remains open whether LCIS admits mildly subquadratic algorithms, such as the Masek-Paterson algorithm for LCS [35]. Furthermore, we give tight SETH-based lower bounds for k-LCIS.

For the related variant LCWIS that considers weakly increasing sequences, strongly subquadratic-time algorithms are ruled out under SETH for slightly superlogarithmic alphabet sizes ([39] and Theorem 6). On the other hand, for binary and ternary alphabets, even linear time algorithms exist [34, 23]. Can LCWIS be solved in time $\mathcal{O}\left(n^{2-f(|\Sigma|)}\right)$ for some decreasing function f that yields strongly subquadratic-time algorithms for any constant alphabet size $|\Sigma|$?

Finally, we can compute a $(1+\varepsilon)$-approximation of LCIS in $\mathcal{O}\left(n^{3/2}\varepsilon^{-1/2}\text{polylog}(n)\right)$ time by an easy observation (see the appendix in the full version). Can we improve upon this running time or give a matching conditional lower bound? Note that a positive resolution seems difficult by the reduction in Observation 1: Any n^α, $\alpha > 0$, improvement over this running time would yield a strongly subcubic $(1+\varepsilon)$-approximation for 3-LCS, which seems hard to achieve, given the difficulty to find strongly subquadratic $(1+\varepsilon)$-approximation algorithms for LCS.

References

1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Quadratic-time hardness of LCS and other sequence similarity measures. In *Proc. 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS'15)*, pages 59–78, 2015.

2 Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends or: A polylog shaved is a lower bound made. In *Proc. 48th Annual ACM Symposium on Symposium on Theory of Computing (STOC'16)*, pages 375–388, 2016.

3 Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Proc. of 41st International Colloquium on Automata, Languages, and Programming (ICALP'14)*, pages 39–51, 2014.

4 Alfred V. Aho, Daniel S. Hirschberg, and Jeffrey D. Ullman. Bounds on the complexity of the longest common subsequence problem. *Journal of the ACM*, 23(1):1–12, 1976.

5 Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

6 Hsing-Yen Ann, Chang-Biau Yang, and Chiou-Ting Tseng. Efficient polynomial-time algorithms for the constrained LCS problem with strings exclusion. *Journal of Combinatorial Optimization*, 28(4):800–813, 2014.

7 Alberto Apostolico and Concettina Guerra. The longest common subsequence problem revisited. *Algorithmica*, 2(1):316–336, 1987.

8 Abdullah N. Arslan and Ömer Egecioglu. Algorithms for the constrained longest common subsequence problems. *International Journal of Foundations of Computer Science*, 16(6):1099–1109, 2005.

9 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proc. 47th Annual ACM Symposium on Theory of Computing (STOC'15)*, pages 51–58, 2015.

10 Arturs Backurs and Piotr Indyk. Which regular expression patterns are hard to match? In *Proc. 57th Annual Symposium on Foundations of Computer Science, (FOCS'16)*, pages 457–466, 2016.

11 Arturs Backurs and Christos Tzamos. Improving viterbi is hard: Better runtimes imply faster clique algorithms. In *Proc. 34th International Conference on Machine Learning (ICML'17)*, 2017. To appear.

12 Gary Benson, Avivit Levy, S. Maimoni, D. Noifeld, and B. Riva Shalom. Lcsk: A refined similarity measure. *Theoretical Computer Science*, 638:11–26, 2016.

13 Lasse Bergroth, Harri Hakonen, and Timo Raita. A survey of longest common subsequence algorithms. In *Proc. 7th International Symposium on String Processing and Information Retrieval (SPIRE'00)*, pages 39–48, 2000.

14 Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *Proc. 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS'14)*, pages 661–670, 2014.

15 Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proc. 56th Annual IEEE Symposium on Foundations of Compu ter Science (FOCS'15)*, pages 79–97, 2015.

16 Karl Bringmann and Marvin Künnemann. Multivariate fine-grained complexity of longest common subsequence. In *Proc. 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'18)*, 2018. To appear.

17 Wun-Tat Chan, Yong Zhang, Stanley P. Y. Fung, Deshi Ye, and Hong Zhu. Efficient algorithms for finding a longest common increasing subsequence. *Journal of Combinatorial Optimization*, 13(3):277–288, 2007.

18 Yi-Ching Chen and Kun-Mao Chao. On the generalized constrained longest common subsequence problems. *Journal of Combinatorial Optimization*, 21(3):383–392, 2011.

19 Francis Y. L. Chin, Alfredo De Santis, Anna Lisa Ferrara, N. L. Ho, and S. K. Kim. A simple algorithm for the constrained sequence problems. *Inf. Process. Lett.*, 90(4):175–179, 2004. doi:10.1016/j.ipl.2004.02.008.

20 Vaclav Chvatal, David A. Klarner, and Donald E. Knuth. Selected combinatorial research problems. Technical Report CS-TR-72-292, Stanford University, Department of Computer Science, 6 1972.

21 Maxime Crochemore and Ely Porat. Fast computation of a longest increasing subsequence and application. *Information & Computation*, 208(9):1054–1059, 2010.

22 Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. On problems equivalent to (min,+)-convolution. In *Proc. 44th International Colloquium on Automata, Languages, and Programming (ICALP'17)*, pages 22:1–22:15, 2017.

23 Lech Duraj. A linear algorithm for 3-letter longest common weakly increasing subsequence. *Information Processing Letters*, 113(3):94–99, 2013.

24 Michael L. Fredman. On computing the length of longest increasing subsequences. *Discrete Mathematics*, 11(1):29–35, 1975.

25 Zvi Gotthilf, Danny Hermelin, Gad M. Landau, and Moshe Lewenstein. Restricted LCS. In *Proc. 17th International Symposium on String Processing and Information Retrieval (SPIRE'10)*, pages 250–257, 2010.

26 Daniel S. Hirschberg. Algorithms for the longest common subsequence problem. *Journal of the ACM*, 24(4):664–675, 1977.

27 J. W. Hunt and M. D. McIlroy. An algorithm for differential file comparison. Computing Science Technical Report 41, Bell Laboratories, 1975.

28 James W. Hunt and Thomas G. Szymanski. A fast algorithm for computing longest subsequences. *Communications of the ACM*, 20(5):350–353, 1977.

29 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

30 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

31 Guy Jacobson and Kiem-Phong Vo. Heaviest increasing/common subsequence problems. In *Combinatorial Pattern Matching, Third Annual Symposium, CPM 92, Tucson, Arizona, USA, April 29 - May 1, 1992, Proceedings*, pages 52–66, 1992.

32 Tao Jiang, Guohui Lin, Bin Ma, and Kaizhong Zhang. The longest common subsequence problem for arc-annotated sequences. *Journal of Discrete Algorithms*, 2(2):257–270, 2004.

33 Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. On the Fine-grained Complexity of One-Dimensional Dynamic Programming. In *Proc. 44th International Colloquium on Automata, Languages, and Programming (ICALP'17)*, pages 21:1–21:15, 2017.

34 Martin Kutz, Gerth Stølting Brodal, Kanela Kaligosi, and Irit Katriel. Faster algorithms for computing longest common increasing subsequences. *Journal of Discrete Algorithms*, 9(4):314–325, 2011.

35 William J. Masek and Mike Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System Sciences*, 20(1):18–31, 1980.

36 Howard L. Morgan. Spelling correction in systems programs. *Communications of the ACM*, 13(2):90–94, 1970.

37 Eugene W. Myers. An $O(ND)$ difference algorithm and its variations. *Algorithmica*, 1(2):251–266, 1986.

38 Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

39 Adam Polak. Why is it hard to beat $O(n^2)$ for longest common weakly increasing subsequence? *CoRR*, abs/1703.01143, 2017.

40 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proc. 45th Annual ACM Symposium on Symposium on Theory of Computing (STOC'13)*, pages 515–524, 2013.

41 Yin-Te Tsai. The constrained longest common subsequence problem. *Information Processing Letters*, 88(4):173–176, 2003.

42 Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974.

43 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357–365, 2005.

44 Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *Proc. 10th International Symposium on Parameterized and Exact Computation (IPEC'15)*, pages 17–29, 2015.

45 I-Hsuan Yang, Chien-Pin Huang, and Kun-Mao Chao. A fast algorithm for computing a longest common increasing subsequence. *Information Processing Letters*, 93(5):249–253, 2005.

46 Daxin Zhu, Lei Wang, Tinran Wang, and Xiaodong Wang. A simple linear space algorithm for computing a longest common increasing subsequence. *CoRR*, abs/1608.07002, 2016.

Order
https://doi.org/10.1007/s11083-017-9444-1

On an Extremal Problem for Poset Dimension

Grzegorz Guśpiel[1] · **Piotr Micek**[1] · **Adam Polak**[1]

Received: 22 May 2017 / Accepted: 1 November 2017

Abstract Let $f(n)$ be the largest integer such that every poset on n elements has a 2-dimensional subposet on $f(n)$ elements. What is the asymptotics of $f(n)$? It is easy to see that $f(n) >= n^{1/2}$. We improve the best known upper bound and show $f(n) = \mathcal{O}(n^{2/3})$. For higher dimensions, we show $f_d(n) = \mathcal{O}\left(n^{\frac{d}{d+1}}\right)$, where $f_d(n)$ is the largest integer such that every poset on n elements has a d-dimensional subposet on $f_d(n)$ elements.

Keywords Partially ordered sets · Poset dimension · Extremal combinatorics · Permutation matrices

1 Introduction

Every partially ordered set on n elements has a chain or an antichain of size at least $n^{1/2}$, this is an immediate consequence of Dilworth's Theorem or its easier dual counterpart. Chains and antichains are very special instances of 2-dimensional posets. Surprisingly, the following simple problem is open:

Grzegorz Guśpiel was partially supported by the Polish Ministry of Science and Higher Education grant DI2013 000443. Piotr Micek was partially supported by the National Science Center of Poland under grant no. 2015/18/E/ST6/00299. Adam Polak was partially supported by the Polish Ministry of Science and Higher Education program "Diamentowy Grant".

✉ Grzegorz Guśpiel
guspiel@tcs.uj.edu.pl

Piotr Micek
micek@tcs.uj.edu.pl

Adam Polak
polak@tcs.uj.edu.pl

[1] Theoretical Computer Science Department, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland

Published online: 22 November 2017

Let $f(n)$ be the largest integer such that every poset on n elements has a 2-dimensional subposet on $f(n)$ elements. What is the asymptotics of $f(n)$?

Although this sounds like a natural extremal-type question for posets, it was posed only in 2010, by François Dorais [1]. Clearly, $n^{1/2} \leqslant f(n) \leqslant n$. Reiniger and Yeager [5] proved a sublinear upper bound, that is $f(n) = \mathcal{O}(n^{0.8295})$. Their construction is a lexicographic power of standard examples.

The main idea behind our contribution was a belief that a $(k \times k)$-grid is asymptotically the largest 2-dimensional subposet of the $(k \times k \times k)$-cube. This led us to the following theorem:

Theorem 1

$$f(n) \leqslant 4n^{2/3} + o\left(n^{2/3}\right).$$

Recall that the *dimension* $\dim(P)$ of a poset P is the least integer d such that elements of P can be embedded into $\mathbb{R}^d$ in such a way that $x < y$ in P if and only if the point of x is below the point of y with respect to the product order on $\mathbb{R}^d$. Equivalently, the dimension of P is the least d such that there are d linear extensions of P whose intersection is P. By convention, whenever we say a poset is d-dimensional, we mean its dimension is at most d.

Reiniger and Yeager [5] also studied the guaranteed size of the largest d-dimensional subposet of poset on n elements. Let $f_d(n)$ be the largest integer such that every poset on n elements has a d-dimensional subposet on $f_d(n)$ elements. They proved, in particular, that $f_d(n) = \mathcal{O}(n^g)$, where $g = \log_{2d+2}(2d+1)$.

Let $[n]$ denote $\{0, 1, \ldots, n-1\}$. By the $\mathbf{n^d}$-grid we mean the poset on the ground set $[n]^d$ with the natural product order, i.e. $(x_1, x_2, \ldots, x_d) \leqslant (y_1, y_2, \ldots, y_d)$ if $x_i \leqslant y_i$ for all i. Note that the $\mathbf{n^d}$-grid is a d-dimensional poset. Moreover, it is easy to see that the $\mathbf{n^{d+1}}$-grid contains as a subposet the $\mathbf{n^d}$-grid – simply fix one coordinate to an arbitrary value. We prove that this is asymptotically the largest d-dimensional subposet of the $\mathbf{n^{d+1}}$-grid. For $d \leqslant 7$, this observation improves on the best known upper bound for the asymptotics of $f_d(n)$.

Theorem 2

$$f_d(n) = \mathcal{O}\left(n^{\frac{d}{d+1}}\right).$$

In order to show this we apply a multidimensional version of the theorem by Marcus and Tardos [3] saying that the number of 1-entries in an $n \times n$ $(0, 1)$-matrix that avoids a fixed permutation matrix P is $\mathcal{O}(n)$. The multidimensional version was proved by Klazar and Marcus [2], and then independently by Methuku and Pálvölgyi [4], who applied it to another extremal problem related to subposets, i.e. they proved that for every poset P the size of any family of subsets of $[n]$ that does not contain P as a subposet is at most $\mathcal{O}\left(\binom{n}{\lfloor n/2 \rfloor}\right)$.

2 Dimension Two

If we ignore a multiplicative constant, Theorem 1 becomes a special case of Theorem 2. Still, we provide a short and simple proof of Theorem 1, as we believe it might provide a better insight to the core of the problem.

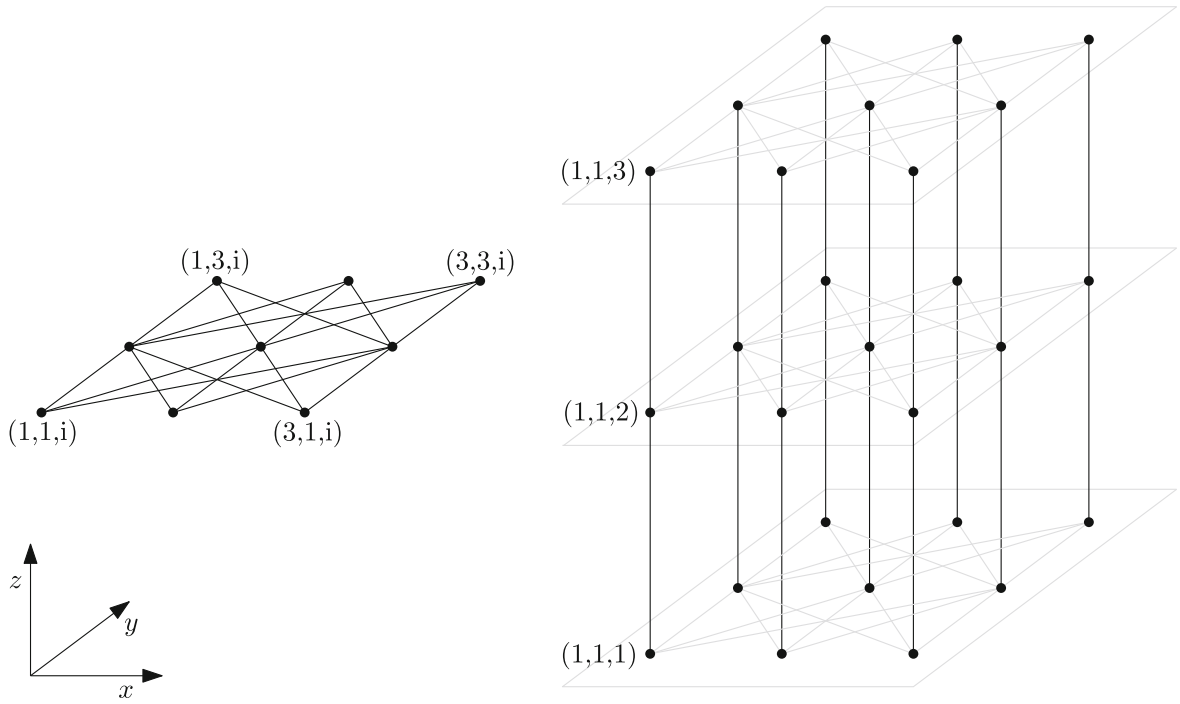

Fig. 1 A subposet of $\mathbf{C_3}$ composed of all elements with z-coordinate equal to i (on the left) and the poset $\mathbf{C_3}$ itself (on the right)

Proof of Theorem 1 First we argue for values of n such that $n = r^3$ for some $r \in \mathbb{N}$. Then at the end of the proof we address the general case.

Let $\mathbf{C_r}$ be the poset with the ground set $[r]^3$, where $(x_1, y_1, z_1) \leqslant_{\mathbf{C_r}} (x_2, y_2, z_2)$ if

$$(z_1 \leqslant z_2) \quad \text{and} \quad (y_1 < y_2 \text{ or } (y_1 = y_2 \text{ and } x_1 = x_2)),$$

see Fig. 1.

Consider any subposet S of $\mathbf{C_r}$ such that $|S| \geqslant 4r^2$. We will prove that $\dim(S) > 2$ by showing that S contains as a subposet the poset[1] of dimension 3 presented in Fig. 2.

Let S^1 be the poset obtained from S by removing every element (x, y, z) such that S contains no element (x, y, z') with $z' < z$. Note that $|S^1| \geqslant 3r^2$, as for every pair $(x, y) \in [r]^2$ at most one element is removed. Now by the pigeonhole principle, we get that S^1 contains a subposet S^2 on at least $3r$ elements such that all elements of S^2 have the same z-coordinate.

Let A be any point in S^2 with the minimal y-coordinate and let S^3 be the subposet of S^2 obtained by removing all points with the same y-coordinate as A. As there can be at most r points with the same y-coordinate, $|S^3| \geqslant 2r$. By the pigeonhole principle for $r - 1$ containers, S^3 contains three points with the same y-coordinate, say $B_1 = (x_1, y, z)$, $B_2 = (x_2, y, z)$, $B_3 = (x_3, y, z)$. Thanks to the removal rule that led to the creation of S^1, the poset S contains points $C_1 = (x_1, y, z_1)$, $C_2 = (x_2, y, z_2)$, $C_3 = (x_3, y, z_3)$ for some $z_1, z_2, z_3 < z$.

One can easily verify that the subposet $\{A, B_1, B_2, B_3, C_1, C_2, C_3\}$ of S is the poset in Fig. 2. Since it has dimension 3, we have $\dim(S) > 2$, which concludes the proof for n being a perfect cube.

[1] This is one of the 3-irreducible posets, which are listed in [6].

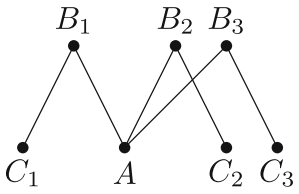

Fig. 2 A poset of dimension 3 found in any subposet of $\mathbf{C_r}$ of size at least $4r^2$

Now, fix any $n \in \mathbb{N}$, and let $r = \lceil \sqrt[3]{n} \rceil$. Note that f is non-decreasing, thus

$$f(n) \leqslant f(r^3) \leqslant 4r^2 \leqslant 4(\sqrt[3]{n}+1)^2 = 4n^{2/3} + o\left(n^{2/3}\right).$$

□

With a more tedious analysis, which involves one more forbidden subposet and removal of both lowest and highest z-coordinate points in each (x, y)-column, we can prove a slightly stronger upper bound, i.e. $f(n) \leqslant 3n^{2/3} + o\left(n^{2/3}\right)$. However, we do not know how to improve on the asymptotics of f.

3 Higher Dimensions

In this section we prove Theorem 2. In order to do this we apply a multidimensional version of the theorem by Marcus and Tardos [3], proved by Klazar and Marcus [2]. First, we recall their result. The original terminology can be simplified because our argument does not use arbitrary sized matrices and we can focus only on multidimensional analogs of square matrices.

We call a subset of $[n]^d$ a *d-dimensional* $(0, 1)$*-matrix*.

For two d-dimensional $(0, 1)$-matrices $A \subseteq [n]^d$ and $B \subseteq [k]^d$, we say that A *contains* B if there exist d increasing injections $h_i : [k] \to [n]$, $i \in \{1, 2, \ldots, d\}$, such that

$$\text{if } (x_1, x_2, \ldots, x_d) \in B, \text{ then } (h_1(x_1), h_2(x_2), \ldots, h_d(x_d)) \in A,$$

for all $(x_1, x_2, \ldots, x_d) \in [k]^d$. Otherwise, we say that A *avoids* B.

We say that $A \subseteq [n]^d$ is a *d-dimensional permutation of* $[n]$

$$|A| = n \quad \text{and} \quad \forall_{\substack{x,y\in A \\ x\neq y}} \forall_{i\in\{1,2,\ldots,d\}}\, x_i \neq y_i.$$

In other words, the size of the projection of A onto the i-th dimension equals n for each $i \in \{1, 2, \ldots, d\}$.

Theorem 3 (Klazar–Marcus [2]) *For every fixed d-dimensional permutation P the maximum number of elements of a d-dimensional matrix $A \subseteq [n]^d$ that avoids P is $\mathcal{O}(n^{d-1})$.*

Now we are ready to prove the following statement, which clearly implies Theorem 2.

Theorem 4 *The largest d-dimensional subposet of the $\mathbf{n^{d+1}}$-grid has $\mathcal{O}(n^d)$ elements.*

Proof We fix any poset of dimension $d+1$, e.g. the standard example S_{d+1}, i.e. the inclusion order of singletons and d-element subsets of $[d + 1]$. Now, we fix a realizer of S_{d+1} of size

$d+1$, i.e. a set of $d+1$ linear orders $\{L_1, L_2, \ldots, L_{d+1}\}$ such that $L_1 \cap L_2 \cap \cdots \cap L_{d+1} = S_{d+1}$. Finally, we construct a $(d+1)$-dimensional permutation $P \subseteq [2(d+1)]^{d+1}$ such that $(x_1, x_2, \ldots, x_{d+1}) \in P$ if and only if there exists $x \in S_{d+1}$ such that x is the x_i-th element of L_i for each $i \in \{1, 2, \ldots, d+1\}$. Note that the natural product order of elements of P is isomorphic to S_{d+1}.

Now, take any d-dimensional subposet of the $\mathbf{n^{d+1}}$-grid and denote by A the set of its elements. In particular, the subposet does not contain S_{d+1} as a subposet. Note that it implies that A avoids P, thus by Theorem 3 the size of the subposet is $\mathcal{O}(n^d)$. □

Note that the proof above does not exploit any specific properties of the standard example, apart from its dimension. In particular, it implies that every $(d+1)$-dimensional poset P can be found in every subposet of the $\mathbf{n^{d+1}}$-grid of size $\Omega(n^d)$, with the constant hidden in the asymptotic notation depending on the choice of P.

Acknowledgements We send thanks to Wojciech Samotij and Dömötör Pálvölgyi for pointing us to useful references.

References

1. Dorais, F.G.: Subposets of small Dushnik-Miller dimension. MathOverflow. http://mathoverflow.net/questions/29169 (2010). Accessed 13 Feb 2016
2. Klazar, M., Marcus, A.: Extensions of the linear bound in the Füredi-Hajnal conjecture. Adv. Appl. Math. **38**(2), 258–266 (2007). https://doi.org/10.1016/j.aam.2006.05.002
3. Marcus, A., Tardos, G.: Excluded permutation matrices and the Stanley-Wilf conjecture. J. Comb. Theory A **107**(1), 153–160 (2004). https://doi.org/10.1016/j.jcta.2004.04.002
4. Methuku, A., Pálvölgyi, D.: Forbidden hypermatrices imply general bounds on induced forbidden subposet problems. Combin. Probab. Comput. **26**(4), 593–602 (2017). https://doi.org/10.1017/S0963548317000013
5. Reiniger, B., Yeager, E.: Large subposets with small dimension. Order **33**(1), 81–84 (2016). https://doi.org/10.1007/s11083-015-9353-0
6. Trotter, W.T.: Combinatorics and Partially Ordered Sets: Dimension Theory. Johns Hopkins University Press, Baltimore (1992)

Appendix: Unpublished manuscript

ONLINE COLORING OF SHORT INTERVALS

GRZEGORZ GUTOWSKI, KONSTANTY JUNOSZA-SZANIAWSKI, PATRYK MIKOS,
ADAM POLAK, AND JOANNA SOKÓŁ

Abstract. We study the online graph coloring problem restricted to the intersection graphs of intervals with lengths in $[1, \sigma]$. For $\sigma = 1$ it is the class of unit interval graphs, and for $\sigma = \infty$ the class of all interval graphs. Our focus is on intermediary classes.

We present a $(1 + \sigma)$-competitive algorithm, which beats the state of the art for $1 < \sigma < 2$. For $\sigma = 1$ our algorithm matches the performance of FirstFit, which is 2-competitive for unit interval graphs. For $\sigma = 2$ it matches the Kierstead-Trotter algorithm, which is 3-competitive for all interval graphs.

On the lower bound side, we prove that no algorithm is better than 5/3-competitive for any $\sigma > 1$, nor better than 7/4-competitive for any $\sigma > 2$, nor better than 5/2-competitive for arbitrarily large values of σ.

1. Introduction

In the *online graph coloring* problem the input graph is presented to the algorithm vertex by vertex, along with all the edges adjacent to the already presented vertices. Each vertex must be assigned a color, different than any of its neighbors, immediately and irrevocably at the moment it is presented, without any knowledge of the remaining part of the graph. The objective is to minimize the number of used colors. The problem and its variants attract much attention, both for theoretical properties and practical applications in network multiplexing, resource allocation, and job scheduling.

The standard performance measure, used to analyze online algorithms, is the *competitive ratio*, i.e., the worst-case guarantee on the ratio of the solution given by an online algorithm to the optimal offline solution (see Sect. 1.1 for a formal definition).

In the general case, of online coloring of arbitrary graphs there is no hope for any algorithm with a constant competitive ratio. The best known algorithm [5] uses $O(\chi \cdot n/\log n)$ colors for n-vertex χ-colorable graphs, i.e. it is $O(n/\log n)$-competitive, and there is a lower bound [6] showing that no online graph coloring algorithm can be $o\big(n/\log^2 n\big)$-competitive. It is thus common to study the problem restricted to specific graph classes.

Having in mind the applications in scheduling, one of the important special cases is the class of *interval graphs*, i.e. intersection graphs of intervals on the real line. The classic result is by Kierstead and Trotter [9], who designed a 3-competitive algorithm and proved a matching lower bound. However, in the special case of *unit interval graphs*, i.e. intersection graphs of intervals of a fixed (unit, w.l.o.g.) length, even the simple greedy FirstFit algorithm is 2-competitive [2].

Grzegorz Gutowski was partially supported by the National Science Center of Poland under grant no. 2016/21/B/ST6/02165. Patryk Mikos was partially supported by the National Science Center of Poland under grant no. 2014/14/A/ST6/00138. Adam Polak was partially supported by the Polish Ministry of Science and Higher Education program *Diamentowy Grant* under grant no. DI2012 018942. Joanna Sokół was partially supported by the National Science Center of Poland under grant no. 2016/23/N/ST1/03181.

A natural question arises, what happens in between the interval and unit interval graph classes. In particular, we ask about the optimal competitive ratio of online coloring algorithms for intersection graphs of intervals of length restricted to a fixed range. Formally, let us introduce the σ-interval coloring problem.

Definition 1. For $\sigma \geqslant 1$, the *σ-interval coloring* problem asks: Given a sequence of closed intervals $[l_1, r_1], [l_2, r_2], \ldots, [l_n, r_n]$, such that $1 \leqslant (r_i - l_i) \leqslant \sigma$ for every $i \in [n]$, find a sequence of *colors*, $c_1, c_2, \ldots, c_n$, such that

$$\forall_{i \neq j} \left([l_i, r_i] \cap [l_j, r_j] \neq \emptyset \right) \Rightarrow (c_i \neq c_j),$$

minimizing the number of distinct colors $\left|\{c_1, c_2, \ldots, c_n\}\right|$.

We study the problem in the online setting, i.e., intervals are presented one by one, in an arbitrary order, and each interval has to be colored immediately and irrevocably after it is presented.

Note that we choose to include the interval representation in the input, instead of presenting the mere graph. It seems a plausible modelling choice given the scheduling applications. Moreover, it lets algorithms exploit geometric properties of the input, and not only structural graph properties. Naturally, any lower bound obtained for this variant of the problem transfers to the harder variant with no interval representation in the input.

1.1. **Our Results.** Before we state our results, let us give a formal definition of the competitive ratio. In this paper we focus on the *asymptotic* competitive ratio.

Definition 2. Let A be an online graph coloring algorithm, and let $A(\chi)$ denote the maximum number of colors A uses to color any graph which can be colored offline using χ-colors (i.e. its chromatic number is at most χ). We say that A has the asymptotic competitive ratio α (or that A is α-competitive, for short), if $\limsup_{\chi \to \infty} \frac{A(\chi)}{\chi} \leqslant \alpha$.

Another popular performance measure for online algorithms is the *absolute* competitive ratio, which requires that $\frac{A(\chi)}{\chi} \leqslant \alpha$ holds for *all* χ (and not only in the limit). The choice of the asymptotic, instead of absolute, competitive ratio for our analysis makes things easier for the algorithm and harder for the lower bounds. In our algorithm, sadly, we do not know how to get rid of a constant additive overhead, which vanishes only for large enough χ. The good side is, our lower bounds for the asymptotic competitive ratio imply the identical lower bounds for the absolute competitive ratio.

1.1.1. *Algorithm.* Our positive result is the existence of a $(1+\sigma)$-competitive algorithm.

Theorem 3. *For every $\sigma \in \mathbb{Q}$, there is an algorithm for online σ-interval coloring with $1+\sigma$ asymptotic competitive ratio.*

Note that for $\sigma' > \sigma$ every σ'-interval coloring algorithm is also a correct σ-interval coloring algorithm, with the same upper bound on its competitive ratio. Therefore, for $\sigma \in \mathbb{R} \setminus \mathbb{Q}$ Theorem 3 yields an online σ-interval coloring algorithm with a competitive ratio arbitrarily close to $1+\sigma$. This distinction between rational and irrational values of σ becomes somewhat less peculiar in the light of the results of Fishburn and Graham [3], who proved, among other things, that the classes of graphs with interval representation with lengths in $[1, \sigma]$ are right-continuous exactly at irrational σ.

Until now, the state-of-the art was the 2-competitive FirstFit algorithm [2] for $\sigma = 1$ and the 3-competitive Kierstead-Trotter algorithm [9] for $\sigma > 1$. Thus, our algorithm matches the performance of FirstFit for $\sigma = 1$, and beats the Kierstead-Trotter algorithm up until $\sigma = 2$.

1.1.2. *Lower Bounds.* In order to prove lower bounds for online problems, it is often convenient to look at the problem as a combinatorial game between two players, Algorithm and Presenter. In our case, in each round Presenter reveals an interval, and Algorithm immediately and irrevocably assigns it a color. While Algorithm tries to minimize the number of different colors it assigns, the Presenter's goal is to force Algorithm to use as many colors as possible. A strategy for Presenter implies a lower bound on the competitive ratio of any algorithm solving the problem.

Our negative results include a series of strategies for Presenter with the following consequences.

Theorem 4. *For every $\sigma > 1$ there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio less than $5/3$.*

Theorem 5. *For every $\sigma > 2$ there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio less than $7/4$.*

Theorem 6. *For every $\varepsilon > 0$ there is $\sigma \geqslant 1$ such that there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio $5/2 - \varepsilon$.*

The following, more illustrative, statement is a direct corollary of Theorem 6.

Corollary 7. *There is no online algorithm that works for all $\sigma \geqslant 1$ and uses at most $2.499 \cdot \omega + f(\sigma)$ colors for ω-colorable graphs (for any function f).*

1.2. **Methods.** Our algorithm is inspired by the recent result for online coloring of unit disk intersection graphs [7]. We cover the real line with overlapping blocks, grouped into a constant number of classes. Each class gets a private set of available colors. When an interval is presented, the algorithm chooses a block in a round-robin fashion, and greedily assigns a color from its class.

All our lower bounds can be considered as generalizations of the $3/2$ lower bound for online coloring of unit interval graphs by Epstein and Levy [2]. In particular, we heavily use their *separation strategy*. Our $5/2$ lower bound borrows also from the work of Kierstead and Trotter [9]. However, in order to control the length of intervals independently of the number of colors, we cannot simply use the pigeonhole principle, as they did. Instead, we develop Lemmas 18 and 19, which let us overcome this issue, at a cost of a worse bound for the competitive ratio, i.e. $5/2$ instead of 3.

1.3. **Related Work.** Interval graphs have been intensively studied since the sixties [1, 10], and, in particular, they are known to be *perfect*, i.e. the chromatic number χ of an interval graph always equals the size of the largest clique ω (see, e.g., [4]). To construct an optimal coloring offline it is enough to color the graph greedily in a nondecreasing order of the left ends of the intervals.

For the most basic approach for online coloring, that is the FirstFit algorithm, the competitive ratio for interval graphs is unknown. After a series of papers, the most recent results state that FirstFit is at least 5- and at most 8-competitive [8, 11]. Kierstead and Trotter [9] designed a more involved online coloring algorithm, which uses at most $3\omega - 2$ colors for ω-colorable interval graphs, and proved that there exists a strategy that forces any online coloring algorithm to use exactly that number of colors. For intersection graphs of intervals of unit length any online coloring algorithm uses at least $\frac{3}{2}\omega$ colors, and FirstFit uses at most $2\omega - 1$ colors [2].

It seems a natural question to ask if it is possible to improve the bound of $3\omega - 2$ by assuming that interval lengths belong to a fixed range. The study of interval graphs with

bounded length representations was initiated by Fishburn and Graham [3]. However, it focused mainly on the combinatorial structure, and not its algorithmic applications.

Kierstead and Trotter [9] give, for every $\omega \in \mathbb{N}_+$, a strategy for Presenter to construct an ω-colorable set of intervals while forcing Algorithm to use at least $3\omega - 2$ colors. However, the length of presented intervals increases with the increasing ω. For this reason, with the intervals lengths restricted to $[1, \sigma]$, their lower bound is only for the absolute competitive ratio and does not exclude, say, an algorithm that always uses at most $2\omega + \sigma^{10}$ colors. On the contrary, in Theorem 6 we rule out the existence of such an algorithm.

2. Algorithm

Theorem (Reminder of Theorem 3)**.** *For every $\sigma \in \mathbb{Q}$, there is an algorithm for online σ-interval coloring with $1 + \sigma$ asymptotic competitive ratio.*

Proof. Let us present an algorithm which, in principle, works for any real σ, however only for a rational σ it achieves the declared competitive ratio. The algorithm has a positive integer parameter b. Increasing the parameter brings the asymptotic competitive ratio closer to $1 + \sigma$ at the cost of increasing the additive constant. More precisely, given an ω-colorable set of intervals our algorithm colors it using at most $\lceil b \cdot (1 + \sigma) \rceil \cdot \left(\frac{\omega}{b} + b - 1\right)$ colors, and thus its competitive ratio is $\frac{\lceil b \cdot (1+\sigma) \rceil}{b} + O(1/\omega)$. For a rational σ, in order to obtain the declared competitive ratio it is sufficient to set b to the smallest possible denominator of a simple fraction representation of σ. Let $\varphi = \lceil b \cdot (1 + \sigma) \rceil$. The algorithm will use colors from the set $\{0, 1, \ldots, \varphi - 1\} \times \mathbb{N}$.

Now, let us consider the partition of the real line into *small blocks*. For $i \in \mathbb{Z}$, the i-th small block occupies interval $[i \cdot \frac{1}{b}, (i + 1) \cdot \frac{1}{b})$. Moreover, we define *large blocks*. The i-th large block occupies interval $[i \cdot \frac{1}{b}, i \cdot \frac{1}{b} + 1)$. See Fig. 1.

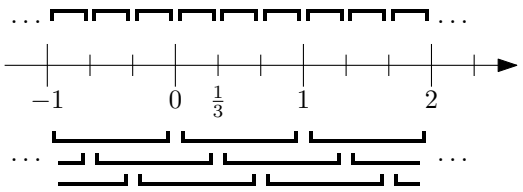

Figure 1. Small blocks (up), and large blocks (down), for $b = 3$

Let us point out certain properties of the blocks, which will be useful in the further analysis. Each large block is the sum of b consecutive small blocks, and each small block is a subset of b consecutive large blocks. Further, length of a large block is 1, and for any two intervals of length in $[1, \sigma]$ that both have the left endpoint in the same large block, the two intervals intersect. Thus, the intervals whose left endpoints belong to a fixed large block form a clique. Finally, if the indices of two large blocks differ by at least φ, then any two intervals – one with the left endpoint in one block, the other with the left endpoint in the other – do not intersect.

With each small block the algorithm associates a *small counter*, and with each large block the algorithm associates a *large counter*. Let S_i denote the small counter of the i-th small block, and L_i denote the large counter of the i-th large block. Initially, all the small and large counters are set to zero.

To assign a color to an interval, the algorithm proceeds as follows:

(1) Let i be the index of the small block containing the left endpoint of the interval.

(2) Let j be the index of the large block containing the left endpoint of the interval such that $j \equiv S_i \pmod{b}$. Note that there is exactly one such j.
(3) Assign to the interval the color $(j \bmod \varphi, L_j)$.
(4) Increase the small counter S_i by one.
(5) Increase the large counter L_j by one.

First let us argue that the algorithm outputs a proper coloring. Consider any two intervals which were assigned the same color. Let j_1 and j_2 denote the indices of the large blocks selected for these intervals by the algorithm. Since the colors of the two intervals have the same first coordinates, we have that $j_1 \equiv j_2 \pmod{\varphi}$. However, since the second coordinates, which are determined by large counters, are also the same, j_1 and j_2 must be different, and thus they differ by at least φ. It follows that the two intervals do not intersect and thus the coloring is proper.

It remains to bound the number of colors in terms of the clique number ω. Let j be the index of the maximum large counter L_j. Clearly, the algorithm used at most $\varphi \cdot L_j$ colors in total. Let C denote the set of intervals with the left endpoints in the j-th large block and colored with a color in $\{j \bmod \varphi\} \times \mathbb{N}$. Observe that $|C| = L_j$. Let x_k denote the number of intervals in C which have the left endpoint in the k-th small block. Recall that the j-th large block is the sum of b small blocks – indexed j, $j+1$, …, $j+b-1$ – and thus $L_j = x_j + x_{j+1} + \cdots + x_{j+b-1}$. Now, observe that, because of the round robin selection in the step 2 of the algorithm,

$$S_k \geqslant b \cdot (x_k - 1) + 1.$$

Let D denote the set of all intervals with the left endpoints in the j-th large block. We can bound the number of intervals in D

$$|D| = \left(\sum_{k=j}^{j+b-1} S_k \right) \geqslant b \cdot (L_j - b) + b.$$

Recall that D is a clique and thus the clique number ω of the input graph is at least the size of D. Therefore $L_j \leqslant \frac{\omega + b \cdot (b-1)}{b}$, and the algorithm used at most

$$\lceil b \cdot (1+\sigma) \rceil \cdot \left(\frac{\omega}{b} + b - 1 \right)$$

colors. □

3. Lower Bounds

In this section we show several strategies for Presenter that force Algorithm to use many colors while the introduced set of intervals is colorable with a smaller number of colors, and contains only short intervals. To properly capture asymptotic properties of those strategies we give the following formal definitions.

Definition 8. For $\omega, C \in \mathbb{N}_+$ and $\sigma, M \in \mathbb{R}_+$, an $\langle \omega, C, \sigma, M \rangle$-*strategy* is a strategy for Presenter that forces Algorithm to use at least C colors subject to the following constraints:

(1) the set of introduced intervals is ω-colorable,
(2) every introduced interval has length at least 1 and at most σ,
(3) every introduced interval is contained in the interval $[0, M]$.

We are interested in providing strategies that achieve the biggest possible ratio $\frac{C}{\omega}$ for large ω. This motivates the following definition.

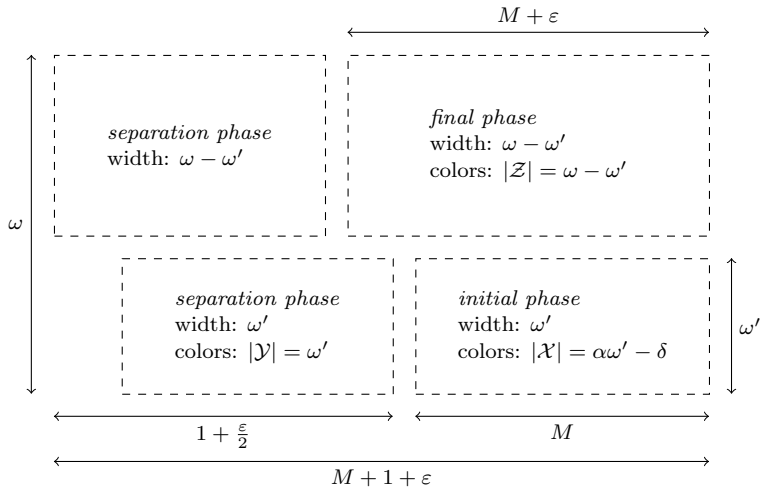

FIGURE 2. Strategy construction in Lemma 11

Definition 9. An $\langle\alpha,\sigma,M\rangle$-*schema* is a set of $\langle\omega,C_\omega,\sigma,M\rangle$-strategies for all $\omega\in\mathbb{N}_+$ such that $C_\omega=\alpha\omega-o(\omega)$.

Note that the existence of an $\langle\alpha,\sigma,M\rangle$-*schema* implies a lower bound of α for the asymptotic competitive ratio of any online algorithm solving the σ-interval coloring problem.

Kierstead and Trotter [9] give an $\langle\omega,3\omega-2,f(\omega),f(\omega)\rangle$-strategy for all $\omega\in\mathbb{N}_+$. However, their family of strategies does not yield an $\langle\alpha,\sigma,M\rangle$-schema, because the length of the presented intervals grows with ω.

Example 10 ($\langle1,1,1\rangle$-schema)**.** For any $\omega\in\mathbb{N}_+$, a strategy that introduces the interval $[0,1]$ in every round $1,\ldots,\omega$ is an $\langle\omega,\omega,1,1\rangle$-strategy. The set of these strategies is a $\langle1,1,1\rangle$-schema.

In the rest of this section we show a series of constructions that use an existing schema to create another schema with different parameters. The $\langle1,1,1\rangle$-schema given above is the initial step for those constructions.

Let S be an $\langle\omega,C,\sigma,M\rangle$-strategy. We say that Presenter uses strategy S in the interval $[x,x+M]$ meaning that Presenter plays according to S, presenting intervals shifted by x until Algorithm uses C colors.

3.1. **Warm-up.** Our first construction is a natural generalization of the strategy for unit intervals given by Epstein and Levy [2]. It is surpassed by more involved strategies coming later, but it serves as a gentle introduction to our framework.

Lemma 11. *If there is an* $\langle\alpha,\sigma,M\rangle$*-schema, then there is a* $\left\langle2-\frac{1}{\alpha+1},M+\varepsilon,M+1+\varepsilon\right\rangle$*-schema for every* $\varepsilon>0$.

Proof. Take arbitrary $\omega\in\mathbb{N}_+$, and let $\omega'=\left\lfloor\frac{\omega}{\alpha+1}\right\rfloor$. The $\langle\alpha,\sigma,M\rangle$-schema contains an $\langle\omega',\alpha\omega'-\delta,\sigma,M\rangle$-strategy S for some $\delta=o(\omega')$. The strategy for Presenter consists of three phases (see Fig. 2). In the first phase, called the *initial phase*, Presenter uses strategy S inside the interval $[1+\varepsilon,M+1+\varepsilon]$. Let $C=\alpha\omega'-\delta$ and let $\mathcal{X}$ denote the set of C colors used by Algorithm in the initial phase.

The second phase, borrowed from [2], is called the *separation phase*. In this phase, Presenter plays the following separation strategy for ω rounds. Let $l_1=0$. In the i-th round of the separation phase Presenter introduces the interval $[l_i,l_i+1]$. If Algorithm

colors the interval with one of the colors in $\mathcal{X}$, let $l_{i+1} = l_i + \frac{\varepsilon}{2^{i+1}}$. Otherwise, let $l_{i+1} = l_i$. Observe that all intervals introduced in the separation phase have length 1 and $l_\omega < \frac{\varepsilon}{2}$. Thus, every interval introduced in the separation phase is contained in $\left[0, 1 + \frac{\varepsilon}{2}\right]$ and any two of those intervals intersect. Furthermore, the choice of l_i's guarantees that for any two intervals x, y introduced in the separation phase, x colored with a color in $\mathcal{X}$, and y colored with a color not in $\mathcal{X}$, we have that the left end-point of x is to the left of the left end-point of y. Let Y be the set of ω' right-most intervals introduced in the separation phase. Let $\mathcal{Y}$ be the set of colors used by Algorithm on the intervals in Y. As $C + \omega' < \omega$, we get that sets of colors $\mathcal{X}$ and $\mathcal{Y}$ are disjoint.

For the last phase, called the *final phase*, let r be the left-most right end-point of an interval in Y. In the final phase Presenter introduces $\omega - \omega'$ times the same interval $[r, M + 1 + \varepsilon]$. This interval intersects all intervals introduced in the initial phase, all intervals in Y, and no other interval introduced in the separation phase. Thus, Algorithm must use $\omega - \omega'$ colors in the final phase that are different from the colors in both $\mathcal{X}$ and $\mathcal{Y}$. Let $\mathcal{Z}$ denote the set of colors used by Algorithm in the final phase.

The presented set of intervals is clearly ω-colorable and Algorithm used at least $|\mathcal{X}| + |\mathcal{Y}| + |\mathcal{Z}| = \alpha\omega' - \delta + \omega' + \omega - \omega' = \left(2 - \frac{1}{\alpha+1}\right)\omega - o(\omega)$ many colors. The longest interval presented has length $M + \varepsilon$, and all intervals are contained in $[0, M + 1 + \varepsilon]$. Thus, we have constructed a $\left\langle 2 - \frac{1}{\alpha+1}, M + \varepsilon, M + 1 + \varepsilon \right\rangle$-schema. □

Corollary 12. *There is a $\left\langle \frac{F_{2n+1}}{F_{2n}}, n + \varepsilon, n + 1 + \varepsilon \right\rangle$-schema, for every $n \in \mathbb{N}_+$ and every $\varepsilon > 0$, where F_n is the n-th Fibonacci number ($F_0 = F_1 = 1$, $F_{n+2} = F_{n+1} + F_n$).*

Proof. Starting with a $\langle 1, 1, 1\rangle$-schema and repeatedly applying Lemma 11 one can generate a family of schemas $\langle \alpha_n, \sigma_n + \varepsilon, M_n + \varepsilon\rangle$, such that $\alpha_{n+1} = 2 - \frac{1}{\alpha_n+1}$, $\sigma_{n+1} = M_n$, $M_{n+1} = M_n + 1$ and $\alpha_0 = \sigma_0 = M_0 = 1$. Solving the recurrence equations we get $\alpha_n = \frac{F_{2n+1}}{F_{2n}}$, $\sigma_n = n$, and $M_n = n + 1$. □

Note that this method cannot give a lower bound with the multiplicative factor better than $\lim_{n\to\infty} \frac{F_{2n+1}}{F_{2n}} = \frac{1+\sqrt{5}}{2} \approx 1.61803$. However, we can get arbitrarily close to this bound. That is, for every $\varepsilon > 0$ there is a σ and ω_0 such that for each $\omega \geqslant \omega_0$ there is a strategy for Presenter to present intervals of length at most σ and force Algorithm to use $\left(\frac{1+\sqrt{5}}{2} - \varepsilon\right) \cdot \omega$ colors on an ω-colorable set of intervals.

Observation 13. *There is no online algorithm that works for all $\sigma \geqslant 1$ and uses at most $1.618 \cdot \omega + f(\sigma)$ colors for ω-colorable graphs (for any function f).*

3.2. The 5/3 Lower Bound.

Lemma 14. *If there is an $\langle \alpha, \sigma, M\rangle$-schema, then there is a $\left\langle 2 - \frac{1}{\alpha+2}, M + \varepsilon, M + 2 + \varepsilon \right\rangle$-schema for every $\varepsilon > 0$.*

Proof. The proof of this lemma is very similar to the proof of Lemma 11, but now we have two separation phases instead of just one, see Fig. 3. Take arbitrary $\omega \in \mathbb{N}_+$, and let $\omega' = \left\lfloor \frac{\omega}{\alpha+2} \right\rfloor$. Let S be an $\langle \omega', \alpha\omega' - \delta, \sigma, M\rangle$-strategy for some $\delta = o(\omega')$.

In the initial phase, Presenter uses S inside interval $\left[1 + \frac{\varepsilon}{2}, M + 1 + \frac{\varepsilon}{2}\right]$, and forces Algorithm to use $C = \alpha\omega' - \delta$ colors. Let $\mathcal{X}$ denote the set of those colors.

In the separation phase, Presenter plays the separation strategy two times. First, Presenter plays the separation strategy for ω rounds in the region $\left[0, 1 + \frac{\varepsilon}{4}\right]$ pushing to the right colors not in $\mathcal{X}$. Let Y_1 be the set of ω' right-most intervals from this first separation. Let $\mathcal{Y}_1$ denote the set of colors used by Algorithm to color Y_1. Then, Presenter

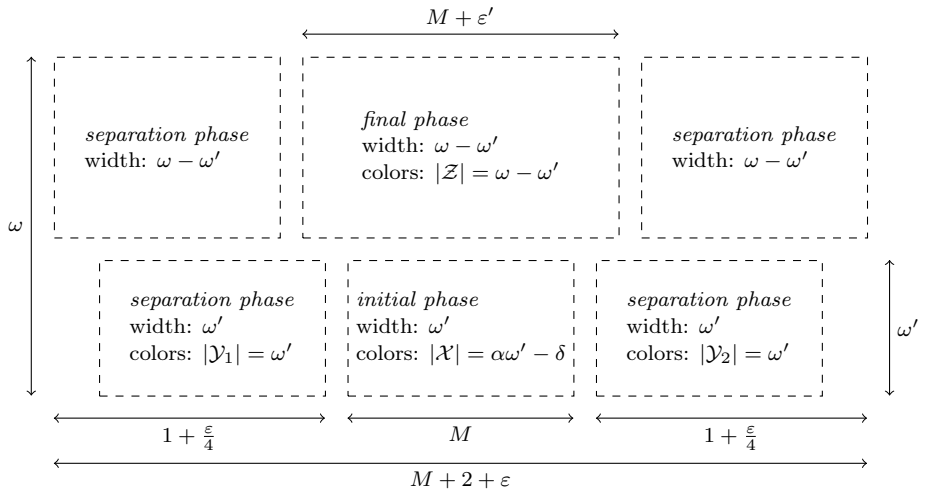

FIGURE 3. Strategy construction in Lemma 14

plays the separation strategy for ω rounds in the region $\left[M+1+\frac{3\varepsilon}{4}, M+2+\varepsilon\right]$ pushing to the left colors not in $\mathcal{X}\cup\mathcal{Y}_1$. Let Y_2 be the set of ω' left-most intervals from this second separation. Let $\mathcal{Y}_2$ denote the set of colors used by Algorithm to color Y_2.

Let r be the left-most right end-point of an interval in Y_1. Let l be the right-most left end-point of an interval in Y_2. In the final phase Presenter introduces $\omega-\omega'$ times the same interval $[r,l]$.

The presented set of intervals is clearly ω-colorable and Algorithm used at least $|\mathcal{X}|+|\mathcal{Y}_1|+|\mathcal{Y}_2|+|\mathcal{Z}| = \alpha\omega'-\delta+\omega'+\omega'+\omega-\omega' = \left(2-\frac{1}{\alpha+2}\right)\omega-o(\omega)$ many colors. The longest interval presented has length at most $M+\varepsilon$, and all intervals are contained in $[0, M+2+\varepsilon]$. Thus, we have constructed a $\left\langle 2-\frac{1}{\alpha+2}, M+\varepsilon, M+2+\varepsilon\right\rangle$-schema. □

Corollary 15. *There is an $\langle\alpha_n, 2n-1+\varepsilon, 2n+1+\varepsilon\rangle$-schema, for every $n\in\mathbb{N}_+$ and every $\varepsilon>0$, where*

$$\alpha_n = \frac{\left(\sqrt{3}-3\right)\left(\sqrt{3}-2\right)^n+\left(\sqrt{3}+3\right)\left(-\sqrt{3}-2\right)^n}{\left(\sqrt{3}-1\right)\left(\sqrt{3}-2\right)^n+\left(\sqrt{3}+1\right)\left(-\sqrt{3}-2\right)^n}.$$

Proof. The argument is similar to Corollary 12, but now we solve the recurrence equation $\alpha_0=1$, $\alpha_{n+1}=2-\frac{1}{\alpha_n+2}$. □

Note that, similarly to Observation 13, one could already use Corollary 15 to get a lower bound arbitrarily close to $\lim_{n\to\infty}\alpha_n=\sqrt{3}$ for the asymptotic competitive ratio of any online algorithm that work for all $\sigma\geqslant 1$. Nonetheless in Sect. 3.4 we prove a stronger $5/2$ lower bound.

Theorem (Reminder of Theorem 4)**.** *For every $\sigma>1$ there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio less than $5/3$.*

Proof. Observe that, for $n=1$, Corollary 15 gives a $\left\langle\frac{5}{3}, 1+\varepsilon, 3+\varepsilon\right\rangle$-schema. Thus, for any $\delta>0$, there is an ω_0 such that for each $\omega\geqslant\omega_0$ there is a strategy for Presenter that presents intervals of length in $[1,1+\varepsilon]$ and forces Algorithm to use $\left(\frac{5}{3}-\delta\right)\cdot\omega$ colors on an ω-colorable set of intervals. □

3.3. The $7/4$ Lower Bound.

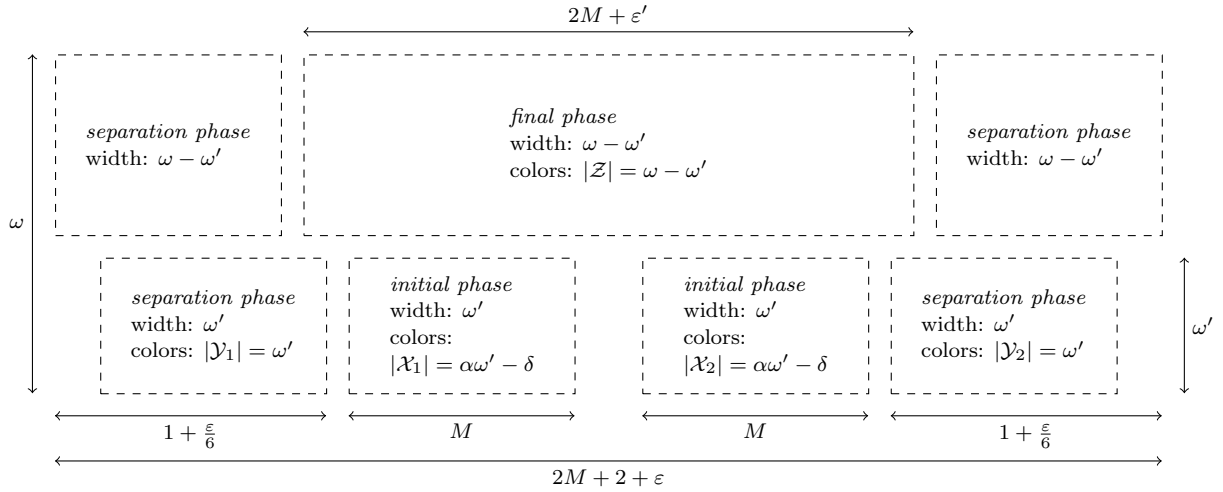

FIGURE 4. Lemma 16, Case 1: $|\mathcal{C}_2 \setminus \mathcal{C}_1| \geqslant \frac{\omega}{2\alpha+2}$

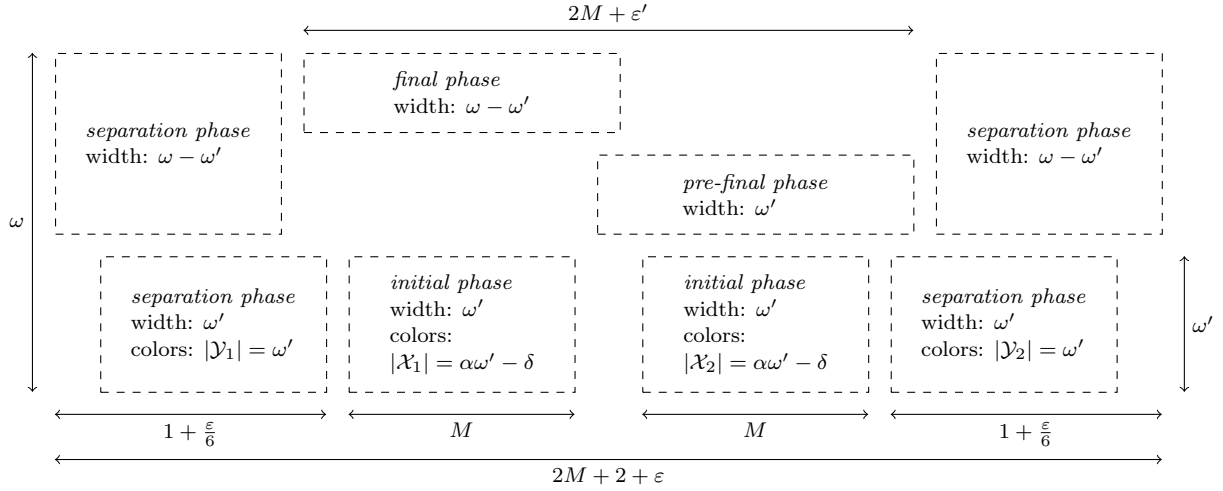

FIGURE 5. Lemma 16, Case 2: $|\mathcal{C}_2 \setminus \mathcal{C}_1| < \frac{\omega}{2\alpha+2}$

Lemma 16. *If there is an $\langle \alpha, \sigma, M \rangle$-schema, then there is a $\left\langle 2 - \frac{1}{2\alpha+2}, 2M + \varepsilon, 2M + 2 + \varepsilon \right\rangle$-schema for every $\varepsilon > 0$.*

Proof. The proof of this lemma is a bit more complicated than the previous ones, as we now have two initial phases, two separation phases and a strategy branching, see Fig. 4 and Fig. 5. Take arbitrary $\omega \in \mathbb{N}_+$, and let $\omega' = \left\lfloor \frac{\omega}{\alpha+1} \right\rfloor$. Let S be an $\langle \omega', \alpha\omega' - \delta, \sigma, M \rangle$-strategy for some $\delta = o(\omega')$.

In the initial phase, Presenter uses strategy S twice: first, inside interval $\left[1 + \frac{\varepsilon}{3}, M + 1 + \frac{\varepsilon}{3}\right]$, and then inside interval $\left[M + 1 + \frac{2\varepsilon}{3}, 2M + 1 + \frac{2\varepsilon}{3}\right]$. Algorithm uses $C = \alpha\omega' - \delta$ colors in each of these games. We get a set of colors $\mathcal{X}_1$ used by Algorithm in the first game, and a set of colors $\mathcal{X}_2$ used by Algorithm in the second game. Note that $\mathcal{X}_1 \cap \mathcal{X}_2$ might be non-empty.

In the separation phase, Presenter plays the separation strategy two times. First, Presenter plays the separation strategy for ω rounds in the region $\left[0, 1 + \frac{\varepsilon}{6}\right]$ pushing to the right colors not in $\mathcal{X}_1$. Let Y_1 be the set of ω' right-most intervals from the first separation phase. Let $\mathcal{Y}_1$ denote the set of colors used by Algorithm to color Y_1. Then, Presenter

plays the separation strategy for ω rounds in the region $\left[2M+1+\frac{5\varepsilon}{6}, 2M+2+\varepsilon\right]$ pushing to the left colors not in $\mathcal{X}_2$. Let Y_2 be the set of ω' left-most intervals from the second separation phase. Let $\mathcal{Y}_2$ denote the set of colors used by Algorithm to color Y_2. Let r be the left-most right end-point of an interval in Y_1. Let l be the right-most left end-point of an interval in Y_2.

There are two cases in the final phase. Let $\mathcal{C}_1 := \mathcal{X}_1 \cup \mathcal{Y}_1$, and analogously $\mathcal{C}_2 := \mathcal{X}_2 \cup \mathcal{Y}_2$. We have that $|\mathcal{C}_1| = |\mathcal{C}_2| = (\alpha+1)\omega' - \delta = \omega - o(\omega)$.

Case 1: If $|\mathcal{C}_2 \setminus \mathcal{C}_1| \geqslant \frac{\omega}{2\alpha+2}$, then Presenter introduces $\omega - \omega'$ times the same interval $[r, l]$.

Each interval introduced in the final phase intersects with all intervals from both initial phases and all intervals in $Y_1 \cup Y_2$. Thus, Algorithm is forced to use $|\mathcal{C}_1 \cup \mathcal{C}_2| + \omega - \omega' = |\mathcal{C}_1| + |\mathcal{C}_2 \setminus \mathcal{C}_1| + \omega - \omega' \geqslant \omega - o(\omega) + \frac{\alpha+\frac{1}{2}}{\alpha+1}\omega = \left(2 - \frac{1}{2\alpha+2}\right)\omega - o(\omega)$ colors in total.

Case 2: If $|\mathcal{C}_2 \setminus \mathcal{C}_1| < \frac{\omega}{2\alpha+2}$, then Presenter introduces ω' intervals, all of them having endpoints $[M+1+5\varepsilon/12, l]$. Let Q be the set of colors used by Algorithm in this *pre-final phase*. We have $\mathcal{C}_2 \cap Q = \emptyset$, and we assumed that $|\mathcal{C}_2 \setminus \mathcal{C}_1| \leqslant \frac{\omega}{2\alpha+2}$, thus we have $|Q \setminus \mathcal{C}_1| \geqslant \frac{\omega}{2\alpha+2}$, and now we are in the case 1 with $\mathcal{C}_2 \to Q$, see Fig. 5.

The longest interval introduced by Presenter in both cases has length strictly less than $2M+\varepsilon$, and the whole game is played in the region $[0, 2M+2+\varepsilon]$. □

Corollary 17. *There is an $\langle \alpha_n, 3\cdot 2^n - 4 + \varepsilon, 3\cdot 2^n - 2 + \varepsilon\rangle$-schema, for every $n \in \mathbb{N}_+$ and every $\varepsilon > 0$, where*

$$\alpha_n = \frac{\left(\sqrt{7}-4\right)\left(\sqrt{7}-3\right)^n + \left(\sqrt{7}+4\right)\left(-\sqrt{7}-3\right)^n}{\left(\sqrt{7}-1\right)\left(\sqrt{7}-3\right)^n + \left(\sqrt{7}+1\right)\left(-\sqrt{7}-3\right)^n}.$$

Proof. The argument is similar to Corollaries 12 and 15, but now we solve the recurrence equations $\alpha_0 = 1$, $\alpha_{n+1} = 2 - \frac{1}{2\alpha_n+2}$, and $M_0 = 1$, $M_{n+1} = 2M_n + 2$, $\sigma_0 = 1, \sigma_{n+1} = 2M_n$. □

Note that, similarly to Observation 13, one could already use Corollary 17 to get a lower bound arbitrarily close to $\lim_{n\to\infty}\alpha_n = \frac{1+\sqrt{7}}{2}$ for the asymptotic competitive ratio of any online algorithm that work for all $\sigma \geqslant 1$. Nonetheless in Sect. 3.4 we prove a stronger 5/2 lower bound.

Theorem (Reminder of Theorem 5)**.** *For every $\sigma > 2$ there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio less than 7/4.*

Proof. Observe that, for $n = 1$, Corollary 17 gives a $\left\langle \frac{7}{4}, 2+\varepsilon, 4+\varepsilon\right\rangle$-schema. For every $\delta > 0$, there is an ω_0 such that for each $\omega \geqslant \omega_0$ there is a strategy for Presenter that presents intervals of length in $[1, 2+\varepsilon]$ and forces Algorithm to use $\left(\frac{7}{4} - \delta\right)\cdot\omega$ colors on an ω-colorable set of intervals. □

3.4. **The 5/2 Lower Bound.** Before we present our main negative result, we need to prove two simple combinatorial lemmas.

Lemma 18. *Let $\gamma \in [0,1]$. For every four sets $X_1, \ldots, X_4$, each of size k, if their intersection is small: $\left|\bigcap_{i=1}^4 X_i\right| \leqslant (1-\gamma)\cdot k$, their sum is large: $\left|\bigcup_{i=1}^4 X_i\right| \geqslant \frac{3+\gamma}{3}\cdot k$.*

Proof. Each element which belongs to the sum but does not belong to the intersection can belong to at most three sets. Thus, we have

$$4\cdot\left(k - \left|\bigcap_{i=1}^4 X_i\right|\right) \leqslant 3\cdot\left(\left|\bigcup_{i=1}^4 X_i\right| - \left|\bigcap_{i=1}^4 X_i\right|\right),$$

and so

$$(3+\gamma)\cdot k \leqslant 4k - \left|\bigcap_{i=1}^{4} X_i\right| \leqslant 3\cdot\left|\bigcup_{i=1}^{4} X_i\right|.$$

□

Lemma 19. *Let $\gamma \in [0,1]$, and $X_1,\ldots,X_{4^n}$ be a family of 4^n sets, each of size k. Then, either*

$$\left|\bigcup_{i=1}^{4^n} X_i\right| \geqslant \left(\frac{3+\gamma}{3}\right)^n k,$$

or the sequence $1,2,\ldots,4^n$ can be covered with four disjoint intervals $[l_1,r_1],\ldots,[l_4,r_4]$, $l_1=1$, $l_{i+1}=r_i+1$, $r_i=4^n$, such that for $Y_i=\bigcup_{j=l_i}^{r_i} X_j$ the intersection of Y_i's is large:

$$|Y_1\cap Y_2\cap Y_3\cap Y_4| \geqslant (1-\gamma)\cdot k.$$

Proof. Consider $n+1$ families of sets defined as follows: $\mathcal{X}_i^0 := X_i$ for every $i\in[4^n]$, and $\mathcal{X}_i^j := \bigcup_{l=4i-3}^{4i}\mathcal{X}_l^{j-1}$ for every $j\in[n]$ and $i\in[4^{n-j}]$. See Fig. 6.

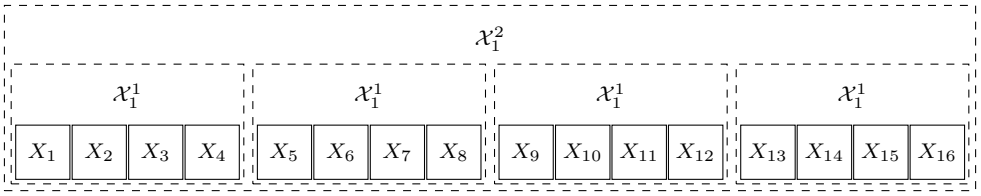

FIGURE 6. $\mathcal{X}_i^j$ sets in Lemma 19

If for some i,j we have $|\bigcap_{l=4i-3}^{4i}\mathcal{X}_l^j| \geqslant (1-\gamma)\cdot k$, then we are done. Thus, we assume that $\forall_{i,j} : |\bigcap_{l=4i-3}^{4i}\mathcal{X}_l^j| < (1-\gamma)\cdot k$. Let $\varrho := \frac{3+\gamma}{3} \in \left[1,\frac{4}{3}\right]$. We prove that $\forall_{j,i} : |\mathcal{X}_i^j| \geqslant \varrho^j k$, by induction on j. For $j=0$ the statement is obvious because $\forall_i : |\mathcal{X}_i^0| = |X_i| = k = \varrho^0 k$. For $j+1$ and arbitrary i, let $k' = \varrho^j k$ and observe that $|\mathcal{X}_{4i-3}^j|,\ldots,|\mathcal{X}_{4i}^j| \geqslant \varrho^j k = k'$. We may ignore some elements of those sets and assume that $|\mathcal{X}_{4i-3}^j| = \ldots = |\mathcal{X}_{4i}^j| = k'$, moreover we assumed that $|\mathcal{X}_{4i-3}^j\cap\ldots\cap\mathcal{X}_{4i}^j| < (1-\gamma)k = \frac{1-\gamma}{\varrho^j}\varrho^j k = (1-\gamma')k'$, where $\gamma'\in[0,1]$ and $\gamma'>\gamma$. We apply Lemma 18 and get $|\mathcal{X}_{4i-3}^j\cup\ldots\cup\mathcal{X}_{4i}^j| \geqslant \frac{3+\gamma'}{3}k'$. Thus, $|\mathcal{X}_i^{j+1}| \geqslant \frac{3+\gamma'}{3}k' > \frac{3+\gamma}{3}k' = \varrho k' = \varrho^{j+1}k$. □

Lemma 20. *If there is an $\langle\alpha,\sigma,M\rangle$-schema, then for every $\varepsilon>0$ and for every $\gamma\in(0,1)$, there is a $\left\langle\frac{5}{4}+\frac{1}{2}(1-\gamma)\alpha, 4^nM+\varepsilon, 4^nM+\varepsilon\right\rangle$-schema, for some $n := n(\gamma)$.*

Proof. Let $\omega\in\mathbb{N}_+$, and $\omega' = \left\lfloor\frac{\omega}{2}\right\rfloor$. Let S be an $\langle\omega',\alpha\omega'-\delta,\sigma,M\rangle$-strategy for some $\delta = o(\omega')$.

In the initial phase, Presenter uses strategy S inside each of 4^n disjoint intervals, $\left[(i-1)(M+\frac{\varepsilon}{4^n}),(i-1)(M+\frac{\varepsilon}{4^n})+M\right]$ for each $i\in[4^n]$. See Fig. 7. Algorithm uses $C = \alpha\omega'-\delta$ colors in each of these games. Let $\mathcal{X}_i$ denote the set of C colors used by Algorithm in the i-th game. Let $\mathcal{X}$ denote the set of all colors used in the initial phase, i.e., $\mathcal{X} = \bigcup_{i\in[4^n]}\mathcal{X}_i$.

We apply Lemma 19 to the family $\mathcal{X}_1,\ldots,\mathcal{X}_{4^n}$ and get that either the union of these sets has at least $\left(\frac{3+\gamma}{3}\right)^n\cdot C$ elements, or we get four disjoint consecutive subfamilies $\mathcal{Y}_1,\ldots,\mathcal{Y}_4$, $\mathcal{Y}_i = \bigcup_{j=l_i}^{r_i}\mathcal{X}_j$ such that the size of the intersection $\mathcal{Y}_1\cap\mathcal{Y}_2\cap\mathcal{Y}_3\cap\mathcal{Y}_4$ has at least $(1-\gamma)\cdot C$ elements.

Case 1: If the size of the union $|\mathcal{X}|$ is at least $\left(1+\frac{\gamma}{3}\right)^n \cdot C$, then Presenter introduces ω' intervals, all of them having endpoints $[0, 4^n M + \varepsilon]$. See Fig. 7. Each interval introduced in the final phase intersects with all intervals introduced in the initial phase. Thus, Algorithm is forced to use at least $|\mathcal{X}| + \omega' \geqslant \frac{1}{2}\left(\left(1+\frac{\gamma}{3}\right)^n \alpha + 1\right)\omega - o(\omega)$ colors in total. Easy calculation shows that for $\gamma \in (0,1)$, $\alpha \in [1,3]$ and for any $n \geqslant \log_{1+\frac{\gamma}{3}}(5/2-\gamma)$, we have $\frac{1}{2} + \frac{1}{2}\left(1+\frac{\gamma}{3}\right)^n \alpha \geqslant \frac{5}{4} + \frac{1}{2}(1-\gamma)\alpha$.

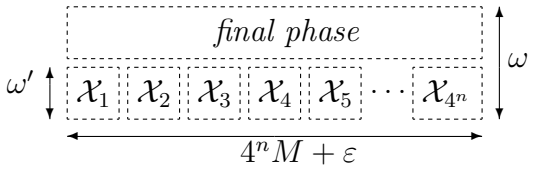

FIGURE 7. Case 1: $|\mathcal{X}|$ is large

Case 2: The size of the intersection $|\mathcal{Y}_1 \cap \ldots \cap \mathcal{Y}_4|$ is at least $(1-\gamma)\cdot C$. Let $\mathcal{Y} = \mathcal{Y}_1 \cap \mathcal{Y}_2 \cap \mathcal{Y}_3 \cap \mathcal{Y}_4$ denote the colors that appear in all four parts of the initial phase. Presenter introduces set Z_1 of ω' intervals, all of them having endpoints $\left[0, \frac{r_1+l_2}{2}\right]$. Intervals in Z_1 cover all intervals contributing to $\mathcal{Y}_1$ and are disjoint with intervals contributing to $\mathcal{Y}_2$. See Fig. 8. Let $\mathcal{Z}_1$ be the set of colors used by Algorithm to color Z_1.

Then Presenter introduces set Z_2 of ω' intervals, all of them having endpoints $\left[\frac{r_3+l_4}{2}, 4^n M + \varepsilon\right]$. Intervals in Z_2 cover all intervals contributing to $\mathcal{Y}_4$ and are disjoint with intervals contributing to $\mathcal{Y}_3$. Let $\mathcal{Z}_2$ be the set of colors used by Algorithm to color Z_2.

Clearly, $|\mathcal{Z}_1| = |\mathcal{Z}_2| = \omega'$, and $\mathcal{Z}_1 \cap \mathcal{Y} = \mathcal{Z}_2 \cap \mathcal{Y} = \emptyset$. Now we distinguish two subcases depending on the size of the set $\mathcal{Z}_2 \setminus \mathcal{Z}_1$.

Case 2.1: If $|\mathcal{Z}_2 \setminus \mathcal{Z}_1| \geqslant \frac{1}{4}\omega$, then Presenter introduces set W of ω' intervals, all of them having endpoints $\left[\frac{3r_1+l_2}{4}, \frac{r_3+3l_4}{4}\right]$. These intervals cover all the intervals contributing to $\mathcal{Y}_2$, and $\mathcal{Y}_3$, intersect all intervals in Z_1, and Z_2. Let $\mathcal{W}$ be the set of colors used by Algorithm to color W. By the definition, we have $\mathcal{W} \cap \mathcal{Y} = \mathcal{W} \cap \mathcal{Z}_1 = \mathcal{W} \cap \mathcal{Z}_2 = \emptyset$. Algorithm was forced to use $|\mathcal{W}| + |\mathcal{Z}_1| + |\mathcal{Z}_2 \setminus \mathcal{Z}_1| + |\mathcal{Y}| \geqslant \left(\frac{1}{2}+\frac{1}{2}+\frac{1}{4}\right)\omega + \frac{1}{2}(1-\gamma)\alpha\omega - o(\omega) = \left(\frac{5}{4} + \frac{1}{2}(1-\gamma)\alpha\right)\omega - o(\omega)$ colors in total. See Fig. 8.

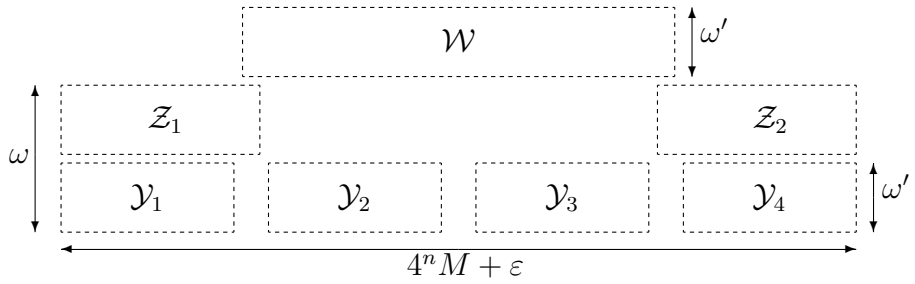

FIGURE 8. Case 2.1: $|\mathcal{Y}|$ is large and $|\mathcal{Z}_2 \setminus \mathcal{Z}_1| \geqslant \frac{1}{4}\omega$

Case 2.2: If $|\mathcal{Z}_2 \setminus \mathcal{Z}_1| < \frac{1}{4}\omega$, then let $\mathcal{Z} = \mathcal{Z}_1 \cap \mathcal{Z}_2$ and observe that $|\mathcal{Z}| \geqslant \left\lfloor \frac{\omega}{4} \right\rfloor$. Presenter introduces set W_1 of ω' intervals, all of them having endpoints $\left[\frac{3r_1+l_2}{4}, \frac{r_2+3l_3}{4}\right]$ and set W_2 of ω' intervals, all of them having endpoints $\left[\frac{3r_2+l_3}{4}, \frac{r_3+3l_4}{4}\right]$. Let $\mathcal{W}$ be the set of colors used by Algorithm to color intervals in $W_1 \cup W_2$. We have that $|\mathcal{W}| = 2\omega'$, and $\mathcal{W} \cap \mathcal{Y} = \mathcal{W} \cap \mathcal{Z} = \emptyset$. Algorithm was forced to use $|\mathcal{W}| + |\mathcal{Z}| + |\mathcal{Y}| \geqslant \left(1+\frac{1}{4}\right)\omega + \frac{1}{2}(1-\gamma)\alpha\omega - o(\omega) = \left(\frac{5}{4} + \frac{1}{2}(1-\gamma)\alpha\right)\omega - o(\omega)$ colors in total. See Fig. 9.

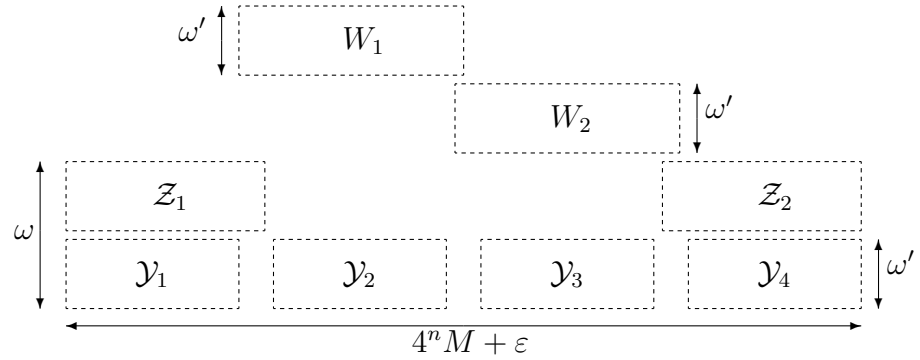

FIGURE 9. Case 2.2: $|\mathcal{Y}|$ is large and $|\mathcal{Z}_2 \setminus \mathcal{Z}_1| < \frac{1}{4}\omega$

□

Corollary 21. *There is an an* $\langle \alpha_n, 4^{nf(\gamma)} + \varepsilon, 4^{nf(\gamma)} + \varepsilon \rangle$*-schema, for every* $n \in \mathbb{N}_+$*, every* $\varepsilon > 0$*, and every* $\gamma \in (0, 1)$*, where*

$$\alpha_n = \frac{5}{2}\frac{1}{1+\gamma} - \frac{(3-2\gamma)}{2(1+\gamma)}\left(\frac{1-\gamma}{2}\right)^n, \quad f(\gamma) = \left\lceil \frac{\log\left(\frac{5}{2} - \gamma\right)}{\log\left(1 + \frac{\gamma}{3}\right)} \right\rceil.$$

Proof. The argument is similar to Corollaries 12, 15, and 17, but now we solve the recurrence equations $\alpha_0 = 1$, $\alpha_{n+1} = \frac{5}{4} + \frac{1}{2}(1-\gamma)\alpha_n$ for competitive ratio, and $L_0 = 1$, $L_{n+1} = 4^{f(\gamma)} L_n$, $a_n = L_n$ for interval and region lengths. □

Theorem (Reminder of Theorem 6)**.** *For every* $\varepsilon > 0$ *there is* $\sigma \geqslant 1$ *such that there is no online algorithm for* σ*-interval coloring with the asymptotic competitive ratio* $5/2 - \varepsilon$*.*

Proof. Setting γ small enough and n large enough, Corollary 21 gives us a $\langle \frac{5}{2} - \frac{\varepsilon}{2}, \sigma, \sigma \rangle$-schema, for some value of σ. Thus, there is ω_0 such that for each $\omega \geqslant \omega_0$ there is a strategy for Presenter that presents intervals of length at most σ and forces Algorithm to use $\left(\frac{5}{2} - \varepsilon\right) \cdot \omega$ colors on an ω-colorable set of intervals, and the theorem follows. □

4. Open Problems

There are still large gaps between the best known lower and upper bounds for the optimal competitive ratios for online σ-interval coloring problems. Figure 10 summarizes these bounds. It would be interesting to close the gap even for a single specific σ. For example, for $\sigma = 3/2$ the optimal online algorithm has the competitive ratio somewhere between $5/3$ and $5/2$.

Finally, let us conjecture that the lower bound of Theorem 6 is tight.

Conjecture 22. *There is a* $5/2$*-competitive online algorithm for* σ*-interval coloring, for every* $\sigma \geqslant 1$*.*

References

[1] Seymour Benzer. On the topology of the genetic fine structure. *Proceedings of the National Academy of Sciences of the United States of America*, 45(11):1607–1620, 1959.

[2] Leah Epstein and Meital Levy. Online interval coloring and variants. In *ICALP 2005: 32nd International Colloquim on Automata, Languages and Programming, Lisbon, Portugal, July 2005. Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 602–613, 2005.

[3] P. C. Fishburn and R. L. Graham. Classes of interval graphs under expanding length restrictions. *Journal of Graph Theory*, 9(4):459–472, 1985.

[4] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57)*. Elsevier, 2 edition, 2004.

[5] Magnús M. Halldórsson. Parallel and on-line graph coloring. *Journal of Algorithms*, 23(2):265–280, 1997.
[6] Magnús M. Halldórsson and Mario Szegedy. Lower bounds for on-line graph coloring. *Theoretical Computer Science*, 130(1):163–174, 1994.
[7] K. Junosza-Szaniawski, P. Rzążewski, J. Sokół, and K. Węsek. Online coloring and l(2,1)-labeling of unit disk intersection graphs. *SIAM Journal on Discrete Mathematics*, To appear.
[8] H.A. Kierstead, David A. Smith, and W.T. Trotter. First-fit coloring on interval graphs has performance ratio at least 5. *European Journal of Combinatorics*, 51:236–254, 2016.
[9] Henry A. Kierstead and William T. Trotter. An extremal problem in recursive combinatorics. In *12th Southeastern Conference on Combinatorics, Graph Theory and Computing, Baton Rouge, LA, USA, March 1981. Proceedings, vol. II*, volume 33 of *Congressus Numerantium*, pages 143–153, 1981.
[10] C. Lekkeikerker and J. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1):45–64, 1962.
[11] N. S. Narayanaswamy and R. Subhash Babu. A note on first-fit coloring of interval graphs. *Order*, 25(1):49–53, 2008.

(G. Gutowski, P. Mikos, A. Polak) Theoretical Computer Science Department, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland
E-mail address: {gutowski,mikos,polak}@tcs.uj.edu.pl

(K. Junosza-Szaniawski, J. Sokół) Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland
E-mail address: {k.szaniawski,j.sokol}@mini.pw.edu.pl

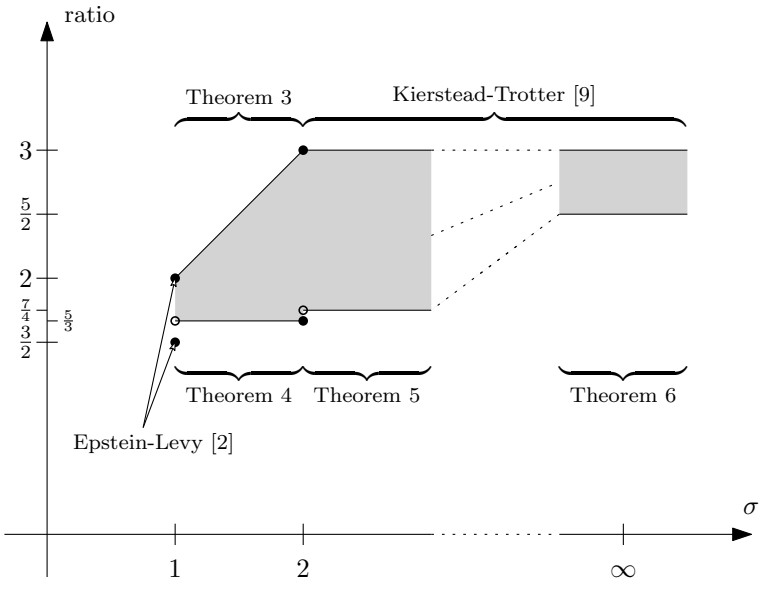

Figure 10. Current bounds for competitive ratio