Parameterized Lower Bounds for Problems in P via Fine-Grained Cross-Compositions

Klaus Heeger
Algorithmics and Computational Complexity, Technische Universität Berlin, Germany

André Nichterlein
Algorithmics and Computational Complexity, Technische Universität Berlin, Germany

Rolf Niedermeier
Algorithmics and Computational Complexity, Technische Universität Berlin, Germany

Abstract

We provide a general framework to exclude parameterized running times of the form $O(\ell^\beta + n^\gamma)$ for problems that have polynomial running time lower bounds under hypotheses from fine-grained complexity. Our framework is based on cross-compositions from parameterized complexity. We (conditionally) exclude running times of the form $O(\ell^{\gamma/(\gamma-1)-\varepsilon} + n^\gamma)$ for any $1 < \gamma < 2$ and $\varepsilon > 0$ for the following problems:

- Longest Common (Increasing) Subsequence: Given two length-n strings over an alphabet Σ (over $\mathbb{N}$) and $\ell \in \mathbb{N}$, is there a common (increasing) subsequence of length ℓ in both strings?
- Discrete Fréchet Distance: Given two lists of n points each and $k \in N$, is the Fréchet distance of the lists at most k? Here ℓ is the maximum number of points which one list is ahead of the other list in an optimum traversal.
- Planar Motion Planning: Given a set of n non-intersecting axis-parallel line segment obstacles in the plane and a line segment robot (called rod), can the rod be moved to a specified target without touching any obstacles? Here ℓ is the maximum number of segments any segment has in its vicinity.

Moreover, we exclude running times $O(\ell^{2\gamma/(\gamma-1)-\varepsilon} + n^\gamma)$ for any $1 < \gamma < 3$ and $\varepsilon > 0$ for:

- Negative Triangle: Given an edge-weighted graph with n vertices, is there a triangle whose sum of edge-weights is negative? Here ℓ is the order of a maximum connected component.
- Triangle Collection: Given a vertex-colored graph with n vertices, is there for each triple of colors a triangle whose vertices have these three colors? Here ℓ is the order of a maximum connected component.
- 2nd Shortest Path: Given an n-vertex edge-weighted digraph, vertices s and t, and $k \in \mathbb{N}$, has the second longest s-t-path length at most k? Here ℓ is the directed feedback vertex set number.

Except for 2nd Shortest Path all these running time bounds are tight, that is, algorithms with running time $O(\ell^{\gamma/(\gamma-1)} + n^\gamma)$ for any $1 < \gamma < 2$ and $O(\ell^{2\gamma/(\gamma-1)} + n^\gamma)$ for any $1 < \gamma < 3$, respectively, are known. Our running time lower bounds also imply lower bounds on kernelization algorithms for these problems.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases FPT in P, Kernelization, Decomposition

Digital Object Identifier 10.4230/LIPIcs.STACS.2023.35

Related Version *Full Version*: `https://arxiv.org/abs/2301.00797` [37]

Funding *Klaus Heeger*: Supported by DFG project NI 369/16 "FPTinP".

Acknowledgements In memory of Rolf Niedermeier, our colleague, friend, and mentor, who sadly passed away before this paper was finished.
We thank the anonymous reviewers for their thoughtful and constructive feedback.

© Klaus Heeger, André Nichterlein, and Rolf Niedermeier;
licensed under Creative Commons License CC-BY 4.0

1 Introduction

In recent years, many results in Fine-Grained Complexity showed that many decade-old textbook algorithms for polynomial-time solvable problems are essentially optimal: Consider as an example Longest Common Subsequence (LCS) where, given two input strings with n characters each, the task is to find a longest string that appears as subsequence in both input strings. The classic $O(n^2)$-time algorithm is often taught in introductory courses to dynamic programming [18]. Bringmann and Künnemann [14] and Abboud et al. [1] independently showed that an algorithm solving LCS in $O(n^{2-\varepsilon})$ time for any $\varepsilon > 0$ would refute the Strong Exponential Time Hypothesis (SETH). Such conditional lower bounds have been shown for many polynomial-time solvable problems in the recent years [50].

One approach to circumvent such lower bounds is "FPT in P" [3, 34]. For Longest Common Subsequence there is a (quite old) parameterized algorithm running in $O(kn + n \log n)$ time, where k is the length of the longest common subsequence [40]. Thus, if k is small (e. g. $O(n^{0.99})$), then the $O(n^2)$ barrier can be broken (without refuting the SETH). A natural question is whether we can do better. As $k \leq n$, an algorithm running in $O(k^{1-\varepsilon}n)$ time for any $\varepsilon > 0$ would break the SETH. However, there are no obvious arguments excluding a running time of $O(k^2+n)$. In fact, such additive running times are not only desirable (as again, for small k this would be faster than even $O(kn)$) but also quite common in parameterized algorithmics by employing kernelization: For Longest Common Subsequence the question would be whether there are linear-time applicable data reduction rules that shrink the input to size $O(k)$. Then we could simply apply the textbook algorithm to solve Longest Common Subsequence in overall $O(k^2 + n)$ time. Kernelization is well-studied in the parameterized community [5, 31] and also effective in practice for polynomial-time solvable problems such as Maximum Matching [43] or Minimum Cut [39].

Bringmann and Künnemann [15] showed in an extensive study that such an $O(k^2 + n)$-time algorithm (and indeed many other parameterized algorithms for Longest Common Subsequence) would refute the SETH. This also implies that no such kernelization algorithm as mentioned above is likely to exist. The results of Bringmann and Künnemann [15] are based on very carefully crafted reductions.

In this work, we follow a different route to obtain similar results for various problems. We provide an easy-to-apply, general framework to (conditionally) exclude algorithms with running time $O(k^\beta + n^\gamma)$ for problems admitting conditional running time lower bounds. Indeed we show for various string (including Longest Common Subsequence) and graph problems as well as problems from computational geometry tight trade-offs between β and γ. This shows that the trivial trade-offs are often the best one can hope for.

1.1 Related work

Fine-grained complexity is an active field of research with hundreds of papers. We refer to the survey of Vassilevska Williams [50] for an overview of the results and employed hypotheses.

Over the last couple of years there has been a lot of work in the direction of "FPT in P" for various problems such as Maximum Matching [19, 22, 30, 38, 41, 43, 44, 47], Hyperbolicity [19, 28], and Diameter [3, 19]. Parameterized lower bounds are rare in this line of work. Certain linear-time reductions can be used to exclude any kind of meaningful FPT-running times; this is also known as General-Problem-Hardness [9]. Using various carefully crafted reductions, Bringmann and Künnemann [15] show parameterized running time lower bounds (under SETH) for Longest Common Subsequence with respect to seven different parameters. In a similar fashion, Duraj et al. [24] show that solving Longest Common Increasing Subsequence in $O((n\ell)^{1-\epsilon})$ time where ℓ is the solution size for some $\epsilon > 0$ would refute SETH.

Fluschnik et al. [29] provide lower bounds for *strict* kernelization (i. e. kernels where the parameter is not allowed to increase) for subgraph detection problems such as Negative Weight Triangle and Triangle Collection. Conceptually, they use the *diminisher*-framework [17, 27] which was originally developed to exclude polynomial-size strict kernels under the assumption P $\neq$ NP. The basic idea is to iteratively apply a diminisher (an algorithm that reduces the parameter at a cost of increasing the instance size) and an (assumed) strict kernel (to shrink and control the instance size) to an instance I of an NP-hard problem. After a polynomial number of rounds, this overall polynomial-time algorithm will return a constant size instance which is equivalent to I, thus arriving at P = NP. Fluschnik et al. [29] applied the same idea to polynomial-time solvable problems. In contrast, we rely and adjust the composition-framework by Bodlaender et al. [11] which was developed to exclude (general) polynomial-size kernels under the stronger assumption NP $\not\subseteq$ coNP / poly.

The composition framework works as follows. Consider the example of the NP-hard problem Negative-Weight Clique: Given an edge-weighted graph G and an integer k, does G contain a negative-weight k-clique, that is, a clique on k vertices where the sum of the edge-weights of the edges within the clique is negative.

Let $(G_1, k), (G_2, k), \ldots, (G_t, k)$ be several instances of Negative-Weight Clique with the same k. Clearly, the graph G obtained by taking the disjoint union of all G_i contains a negative-weight k-clique if and only if some G_i contains a negative-weight k-clique. Moreover, the largest connected component of G has order $\max_{i\in[t]}\{|V(G_i)|\}$. Now assume that Negative-Weight Clique has a kernel of size $O(\ell^c)$ for some constant c where ℓ is the order of a largest connected component. By choosing $t = k^{c+1}$, it follows that kernelizing the instance (G, k) yields an instance of size less than ℓ, that is, less bits than the number of instances encoded in G. Given the NP-hardness of Negative-Weight Clique such a compression seems challenging; indeed it would imply NP $\subseteq$ coNP / poly [32], which in turn results in a collapse of the polynomial hierarchy. Compositions and their extension cross-composition [12] are extensively employed in the parameterized complexity literature. Moreover, to exclude kernels whose size is bounded by polynomials of a specific degree adjustments have been made to the composition framework [20].

Parameter trade-offs. For several of our running time lower bounds we have tight upper bounds that are derived from a simple case distinction argument.

▶ **Observation 1.1** (folklore). *If a problem $\mathcal{P}$ admits an $\widetilde{O}(\ell^\beta n^\gamma)$-time algorithm*[1]*, then it admits for every $\lambda > 0$ an $\widetilde{O}(\ell^{\beta+\frac{\gamma\cdot\beta}{\lambda}} + n^{\gamma+\lambda})$-time algorithm.*

Proof. If $\ell \leq n^{\frac{\lambda}{\beta}}$, then the $\widetilde{O}(\ell^\beta n^\gamma)$-time algorithm runs in $\widetilde{O}(n^{\gamma+\lambda})$ time. Otherwise $n \leq \ell^{\frac{\beta}{\lambda}}$. Then the $\widetilde{O}(\ell^\beta n^\gamma)$-algorithm then in $\widetilde{O}(\ell^{\beta+\frac{\gamma\cdot\beta}{\lambda}})$ time. ◀

1.2 Our Results & Technique

We provide a composition-based framework to establish parameterized running time lower bounds and apply the framework to Longest Common Subsequence, Longest Common (Weakly) Increasing Subsequence, Discrete Fréchet Distance, Planar Motion

[1] The $\widetilde{O}$ hides polylogarithmic factors.

Table 1 Overview of achievable running times. The upper part of the table lists the results for four problems that can be solved in $O(n^2)$ time but under SETH or 3SUM-hypothesis not in $O(n^{2-\varepsilon})$ time for any $\varepsilon > 0$. The lower part lists results for three graph problems that, based on the APSP-hypothesis, do not admit $O(n^{3-\varepsilon})$-time algorithms. The parameterized upper and lower bounds are visualized in Figure 1.

problems	LONGEST COMMON SUBSEQUENCE	$\ell \hat{=}$solution size
	LONGEST COMMON (WEAKLY) INCREASING SUBSEQUENCE	$\ell \hat{=}$solution size
	DISCRETE FRÉCHET DISTANCE	$\ell \hat{=}$maximum shift
	PLANAR MOTION PLANNING	$\ell \hat{=}$max. number of segments in the vincinty of any segment
results	upper bounds	lower bounds
	$O(n^2)$ [6, 7, 35, 52]	no $O(n^{2-\varepsilon})$ assuming SETH / 3SUM [1, 13, 14, 33]
	$\widetilde{O}(\ell n)$ [40, 45, 49]	
	$\widetilde{O}(\ell^{\gamma/(\gamma-1)} + n^\gamma)$ for each $\gamma > 1$	no $O(\ell^{\gamma/(\gamma-1)-\varepsilon} + n^\gamma)$ for any $\gamma < 2$ [15, 24]
	(Observation 1.1)	(Corollaries 4.6, 4.8, and 4.10)
problems	NEGATIVE TRIANGLE	$\ell \hat{=}$size of maximum component
	TRIANGLE COLLECTION	$\ell \hat{=}$size of maximum component
	2ND SHORTEST PATH (only lower bounds)	$\ell \hat{=}$directed feedback vertex set size
results	upper bounds	lower bounds
	$O(n^3/2^{\Theta(\log^{0.5} n)})$ [16, 53]	no $O(n^{3-\varepsilon})$ assuming APSP [51]
	$\widetilde{O}(\ell^2 n)$ (folklore)	
	$\widetilde{O}(\ell^{2\gamma/(\gamma-1)} + n^\gamma)$ for each $\gamma > 1$	no $O(\ell^{2\gamma/(\gamma-1)-\varepsilon} + n^\gamma)$ for any $\gamma < 3$
	(Observation 1.1)	(Corollaries 4.12 and 4.14 and Proposition 4.15)

PLANNING, NEGATIVE TRIANGLE, and 2ND SHORTEST PATH (see Section 1.3 for the problem definitions). Using similar ideas we obtain running time lower bounds for TRIANGLE COLLECTION. For all these problems except 2ND SHORTEST PATH parameterized by the directed feedback vertex set there exist matching running time upper bounds. We refer to Table 1 for an overview on the specific results and the parameterization. Moreover, we visualize in Figure 1 the trade-offs in the running times that are (im-)possible.

Framework. We adjust the cross-composition framework to obtain lower bounds for polynomial time solvable problems. As an example, consider NEGATIVE-WEIGHT TRIANGLE, that is, NEGATIVE-WEIGHT CLIQUE with k fixed to three. Assuming the APSP-hypothesis, NEGATIVE-WEIGHT CLIQUE cannot be solved in $O(n^{3-\varepsilon})$ time [51]. The first difference to the cross-composition framework is that we start with one instance G of NEGATIVE-WEIGHT TRIANGLE which we then decompose into many small instances as follows: Partition the vertices $V(G)$ of G into z many sets $V_1, \ldots, V_z$ of size n/z, where z is chosen depending on the running time we want to exclude (see the proof of Lemma 3.3 in Section 3 for the actual formula specifying z). Then, we create z^3 instances of NEGATIVE-WEIGHT CLIQUE: for each $(i, j, k) \in [z]^3$ take the graph $G[V_i \cup V_j \cup V_k]$. Clearly, we have that G contains a negative-weight triangle if and only if at least one of the created instances contains a negative-weight triangle.

Next, we apply the composition as explained above (the disjoint union) for NEGATIVE-WEIGHT CLIQUE to obtain an instance G' with $n' = z^3 \cdot {}^{n}/_{z} = z^2 n$ vertices. Note that the size ℓ of a largest connected component in G' is $3n/z$. Hence, an algorithm running in

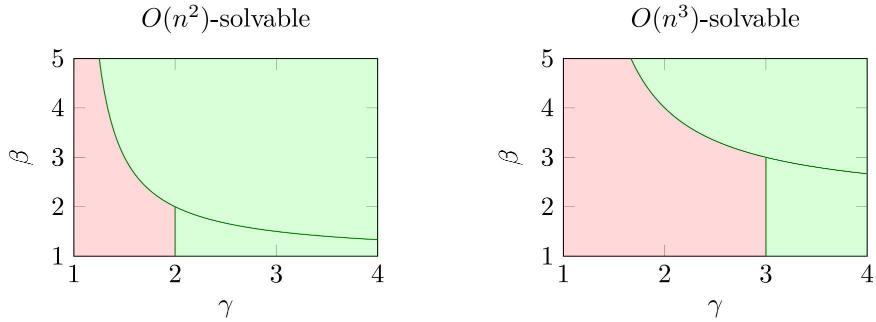

Figure 1 Overview on the possible (in green) and unlikely (in red) trade-offs in running times of the form $O(n^\gamma + \ell^\beta)$. Left: First category for $O(n^2)$-time solvable problems (upper part in Table 1); right: second category for $O(n^3)$-time solvable problems (lower part in Table 1).

time $O(n^\gamma + \ell^\beta)$ for Negative-Weight Triangle solves G' in time $O(z^{2\gamma} n^\gamma + (3n/z)^\beta)$. By carefully choosing z as a function in n, β, and γ, we get that this is in $O(n^{3-\varepsilon})$ for various combinations of γ and β.

The property that Negative-Weight Triangle can be decomposed as above is not unique to the problem. In fact, this has been observed already: "Many problems, like SAT, have a simple self-reduction proving that the "Direct-OR" version is hard, assuming the problem itself is hard" [2]. Our framework formalizes this notion of decomposition (see Section 2 for a definition) and adjusts the cross-composition definition. We furthermore show that commonly used "hard" problems such as Orthogonal Vectors, 3-Sum, and Negative-Weight k-Clique are decomposable. Thus, it remains to show cross-compositions in order to apply our framework and obtain lower bounds.

1.3 Preliminaries and Notation

Problem definitions. For $\ell \in \mathbb{N}$ we set $[\ell] := \{1, 2, \ldots, \ell\}$.

Orthogonal Vectors
Input: Two size-n sets $A, B \subseteq \{0,1\}^d$ for some $d \in \mathbb{N}$.
Question: Are there $a \in A$ and $b \in B$ so that a and b are orthogonal, i.e., for each $i \in [d]$, the i-th coordinate of a or the i-th coordinate of b is zero.

We denote the restriction of Orthogonal Vectors to instances with $d \leq O(\log n)$ as Orthogonal Vectors *with logarithmic dimension.*

3-Sum
Input: An array A of n integers.
Question: Are there $i, j, h \in [n]$ such that $A[i] + A[j] + A[h] = 0$?

Negative-Weight k-Clique
Input: An edge-weighted graph G on n vertices.
Question: Does G contain a k-clique of negative weight?

Longest Common Subsequence
Input: Two strings x^1 and x^2 of length n over an alphabet Σ and $k \in \mathbb{N}$.
Question: Decide whether there is a common subsequence of length k of x^1 and x^2.

Longest Common (Weakly) Increasing Subsequence

Input: Two strings x^1 and x^2 of length n over $\mathbb{N}$ and $k \in \mathbb{N}$.

Question: Decide whether there is a common subsequence y of length k of x^1 and x^2 with $y[i] < y[i+1]$ ($y[i] \leq y[i+1]$) for all $i \in [k-1]$.

Discrete Fréchet Distance

Input: Two lists $P = (p_1, \dots, p_n)$ and $Q = (q_1, \dots, q_n)$ of points in the plane and $k \in \mathbb{Q}$.

Question: Is the Fréchet distance of P and Q at most k, that is, there are two surjective, non-decreasing functions $f_P, f_Q : [2n] \to [n]$ with $f_P(1) = 1 = f_Q(1)$, $f_P(2n) = n = f_Q(2n)$ and $\max_{i \in [2n]} \operatorname{dist}(p_{f_P(i)}, q_{f_Q(i)}) \leq k$?

Planar Motion Planning

Input: A set of n non-intersecting, non-touching, axis-parallel line segment obstacles in the plane and a line segment robot (a rod or ladder), a given source, and a given goal.

Question: Can the rod be moved (allowing both translation and rotation) from the source to the goal without colliding with the obstacles?

2nd Shortest Path

Input: An n-vertex edge-weighted directed graph G, vertices s and t, and $k \in N$.

Question: Has the 2nd-shortest s-t-path length at most k?

Triangle Collection

Input: A vertex-colored graph G on n vertices.

Question: For each combination of three colors, does G contain a triangle whose vertices are colored with three colors?

Hypotheses. The conditional lower bounds in this work are based on SETH, 3-Sum-, and the APSP-Hypothesis (see Vassilevska Williams [50] for more details).

▶ **Hypothesis 1** (SETH). For every $\varepsilon > 0$ there exists a $k \in \mathbb{N}$ such that k-SAT cannot be solved in $O(2^{(1-\varepsilon)n})$ time, where n is the number of variables.

▶ **Hypothesis 2** (3SUM). 3-Sum on n integers in $\{-n^4, \dots, n^4\}$ cannot be solved in $O(n^{2-\varepsilon})$ time for any $\varepsilon > 0$.

▶ **Hypothesis 3** (APSP). All Pairs Shortest Path on n-vertex graphs with polynomial edge weights cannot be solved in $O(n^{3-\varepsilon})$ time for any $\varepsilon > 0$.

Parameterized Complexity. In many of the above problems, n is *not* the input size but a parameter and the input size is bounded by $n^{O(1)}$. A parameterized problem is a set of instances $(I, p) \in \Sigma^* \times \Sigma^*$, where Σ denotes a finite alphabet, I denotes the classical instance and p the parameter. A *kernelization* is a polynomial-time algorithm that maps any instance (I, p) to an equivalent instance (I', p') (the *kernel*) such that $|I'| + p' \leq f(p)$ for some computable function f. If f is a polynomial, then (I', p') is a polynomial-size kernel.

In this work we restrict ourselves to the following: Either $p = n$ (p is a single parameter) or $p = (n, \ell)$ (p is a combined parameter). Moreover, both n and ℓ are always nonnegative integers, n is related to the input size but ℓ is not (ℓ can be seen as "classical" parameter).

Due to space restrictions we defer proofs of results marked with a $\star$ to a full version [37].

2 Framework

Our framework has the following three steps (see Section 1.2 for a high-level description).

1. Start with an instance $(I, n_{\mathcal{P}})$ of a "hard" problem $\mathcal{P}$ and decompose it into the disjunction of t instances $(I_1, n_1), \dots, (I_t, n_t)$ of $\mathcal{P}$. In Section 3, we provide such decompositions for the frequently used hard problems 3-Sum, Orthogonal Vectors, and Negative Weight k-Clique.
2. Compose $(I_1, n_1), \dots, (I_t, n_t)$ into one instance $(J, (n, \ell))$ of the "target" problem using an OR-cross-composition. This step has to be done for the application at hand.
3. Apply the assumed $\widetilde{O}(n^\gamma + \ell^\beta)$-time algorithm to J. If the combination of γ and β is small enough, then the resulting algorithm will be faster than the lower bound for $\mathcal{P}$.

To give a more formal description of our framework, we first define decompositions and cross-compositions. Note that all mentioned problems are parameterized problems.

▶ **Definition 2.1** (OR-decomposition). *For $\alpha > 1$ an α-*OR-decomposition *for a problem $\mathcal{P}$ is an algorithm that, given $\lambda > 0$ and an instance (I, n) of $\mathcal{P}$, computes for some $\alpha' < \alpha$ in $\widetilde{O}(n^{\alpha'})$ time $t \in \widetilde{O}(n^{\alpha\lambda/(\alpha+\lambda)})$ many instances $(I_1, n_1), \dots, (I_t, n_t)$ of $\mathcal{P}$ such that*

- *$(I, n) \in \mathcal{P}$ if and only if $(I_i, n_i) \in \mathcal{P}$ for some $i \in [t]$, and*
- *$n_i \in \widetilde{O}(n^{\alpha/(\alpha+\lambda)})$ for all $i \in [t]$.*

We say a problem $\mathcal{P}$ is *α-OR-decomposable* if there exists an α-OR-decomposition for it. For some problems it is easier to show OR-decomposability than others. Thus, using appropriate reductions to transfer OR-decomposability can be desirable (we do so in Section 3 when showing that 3-Sum is 2-OR-decomposable). Quasi-linear time reductions that do not increase the parameter to much are one option. To this end, we say a reduction that given an instance $(I^{\mathcal{P}}, n^{\mathcal{P}})$ of $\mathcal{P}$ produces an instance $(I^{\mathcal{Q}}, n^{\mathcal{Q}})$ of $\mathcal{Q}$ is *parameter-preserving* if $n^{\mathcal{Q}} \in \widetilde{O}(n^{\mathcal{P}})$.

▶ **Proposition 2.2.** *Let $\mathcal{P}$ and $\mathcal{Q}$ be two problems such that there are quasi-linear-time parameter-preserving reductions from $\mathcal{P}$ to $\mathcal{Q}$ and from $\mathcal{Q}$ to $\mathcal{P}$. Then for any $\alpha > 1$, $\mathcal{P}$ is α-OR-decomposable if and only if $\mathcal{Q}$ is.*

Proof. Assume that $\mathcal{P}$ is α-OR-decomposable (the case that $\mathcal{Q}$ is α-OR-decomposable is symmetric). We now give an α-OR-decomposition for $\mathcal{Q}$. Given an instance $(I^{\mathcal{Q}}, n^{\mathcal{Q}})$ and $\lambda > 0$, we first reduce $(I^{\mathcal{Q}}, n^{\mathcal{Q}})$ to an instance $(I^{\mathcal{P}}, n^{\mathcal{P}})$. Afterwards, we apply the α-OR-decomposition from $\mathcal{P}$, resulting in instances $(I_1^{\mathcal{P}}, n_1^{\mathcal{P}}), \dots, (I_t^{\mathcal{P}}, n_t^{\mathcal{P}})$ of $\mathcal{P}$. Finally, we reduce each instance $(I_i^{\mathcal{P}}, n_i^{\mathcal{P}})$ to an instance $(I_i^{\mathcal{P}}, n_i^{\mathcal{P}})$. This clearly is an α-OR-decomposition for $\mathcal{Q}$. ◀

For the second step of our framework, we introduce fine-grained OR-cross-compositions:

▶ **Definition 2.3** (fine-grained OR-cross-composition). *For $\nu \geq 1, \mu \geq 0$ an (ν, μ)-*OR-cross-composition *from a problem $\mathcal{P}$ to a problem $\mathcal{Q}$ is an algorithm $\mathcal{A}$ which takes as an input t instances $(I_1^{\mathcal{P}}, n_1^{\mathcal{P}}), \dots, (I_t^{\mathcal{P}}, n_t^{\mathcal{P}})$ of $\mathcal{P}$, runs in $\widetilde{O}(t \cdot n_{\max}^\nu + \sum_{i=1}^t |I_i^{\mathcal{P}}|)$ time with $n_{\max} := \max_{i \in [t]} n_i^{\mathcal{P}}$, and computes an instance $(I^{\mathcal{Q}}, (n^{\mathcal{Q}}, \ell^{\mathcal{Q}}))$ of $\mathcal{Q}$ such that*

1. *$(I^{\mathcal{Q}}, (n^{\mathcal{Q}}, \ell^{\mathcal{Q}})) \in \mathcal{Q}$ if and only if $(I_i^{\mathcal{P}}, n_i^{\mathcal{P}}) \in \mathcal{P}$ for some $i \in [t]$, and*
2. *$n^{\mathcal{Q}} \in \widetilde{O}(t \cdot n_{\max}^\nu)$ and $\ell^{\mathcal{Q}} \in \widetilde{O}(n_{\max}^\mu)$.*

We say a problem $\mathcal{P}$ (ν, μ)-*OR-cross-composes* into a problem $\mathcal{Q}$ if there exists an (ν, μ)-OR-cross-composition from $\mathcal{P}$ to $\mathcal{Q}$.

▶ **Theorem 2.4.** *Let $\alpha > \nu \geq 1$, $\gamma > 1$, and $\mu > 0$ with $\alpha > \nu \cdot \gamma$. Let $\mathcal{P}$ be an α-OR-decomposable problem with parameter $n_{\mathcal{P}}$ that (ν, μ)-OR-cross-composes into a problem $\mathcal{Q}$ with parameters $n_{\mathcal{Q}}$ and $\ell_{\mathcal{Q}}$. If there is an $\widetilde{O}(n_{\mathcal{Q}}^{\gamma} + \ell_{\mathcal{Q}}^{\beta})$-time algorithm for $\mathcal{Q}$ and*

$$0 < \beta < \frac{\gamma \cdot (\alpha - \nu)}{(\gamma - 1) \cdot \mu},$$

then $\mathcal{P}$ can be solved in $O(n_{\mathcal{P}}^{\alpha-\varepsilon})$ time for some $\varepsilon > 0$ time.

Proof. Let $(I_{\mathcal{P}}, n_{\mathcal{P}})$ be an instance of $\mathcal{P}$. Our algorithm to solve $(I_{\mathcal{P}}, n_{\mathcal{P}})$ runs in the following steps:

1. Apply the α-OR-decomposition (with λ specified below) to obtain the instances (I_1, n_1), $\ldots, (I_t, n_t)$ with $\max_{i \in [t]} n_i \leq q := n_{\mathcal{P}}^{\alpha/(\alpha+\lambda)}$ and $t := n_{\mathcal{P}}^{\alpha\lambda/(\alpha+\lambda)} = q^{\lambda}$. Note that, by definition, this step runs in $\widetilde{O}(n_{\mathcal{P}}^{\alpha'})$ time for some $\alpha' < \alpha$.
2. Apply the (ν, μ)-OR-cross-composition to compute the instance $(I_{\mathcal{Q}}, (n_{\mathcal{Q}}, \ell_{\mathcal{Q}}))$ for $\mathcal{Q}$ from $(I_1, n_1), \ldots, (I_t, n_t)$. Note that the running time and $n_{\mathcal{Q}}$ is in $\widetilde{O}(t \cdot q^{\nu} + \sum_{i=1}^{t} |I_i^{\mathcal{P}}|) = \widetilde{O}(q^{\lambda+\nu} + n_{\mathcal{P}}^{\alpha'})$. Moreover, $\ell_{\mathcal{Q}} \in \widetilde{O}(q^{\mu})$.
3. Apply the algorithm with running time $\widetilde{O}(n_{\mathcal{Q}}^{\gamma} + \ell_{\mathcal{Q}}^{\beta})$. This requires $\widetilde{O}(q^{(\lambda+\nu)\gamma} + q^{\mu \cdot \beta})$ time.

It remains to show that all three steps run in $O(n_{\mathcal{P}}^{\alpha-\varepsilon})$ time for some $\varepsilon > 0$. To this end, we now specify $\lambda = \beta\mu/\gamma - \nu$. Note that $\lambda + \nu = \beta\mu/\gamma < \mu \cdot \beta$ and thus it suffices to show that the third step runs in $\widetilde{O}(n_{\mathcal{P}}^{\alpha-\epsilon})$ for some $\epsilon > 0$. The last step runs in $\widetilde{O}(q^{\mu \cdot \beta}) = \widetilde{O}(n_{\mathcal{P}}^{\alpha\mu \cdot \beta/(\alpha+\lambda)})$ time. The exponent is

$$\frac{\alpha\mu\beta}{\alpha + \lambda} = \frac{\alpha\mu\beta}{\alpha + \beta\mu/\gamma - \nu} = \frac{\alpha\beta\gamma\mu}{\gamma(\alpha - \nu) + \beta\mu} = \frac{\alpha\gamma\mu}{\gamma(\alpha - \nu)/\beta + \mu}.$$

By assumption we have $\beta < \frac{\gamma \cdot (\alpha - \nu)}{(\gamma - 1) \cdot \mu}$ and thus the exponent is

$$\frac{\alpha\gamma\mu}{\gamma(\alpha - \nu)/\beta + \mu} < \frac{\alpha\gamma\mu}{\gamma(\alpha - \nu)/\left(\frac{\gamma \cdot (\alpha - \nu)}{(\gamma - 1) \cdot \mu}\right) + \mu} = \frac{\alpha\gamma\mu}{(\gamma - 1) \cdot \mu + \mu} = \alpha.$$

There is still one thing left to do: We must ensure that $\lambda > 0$. This will not always be the case as when $\beta \to 0$, λ gets negative. However, an $\widetilde{O}(n_{\mathcal{Q}}^{\gamma} + \ell_{\mathcal{Q}}^{\beta})$-time algorithm also implies for any $\beta' > \beta$ an $\widetilde{O}(n_{\mathcal{Q}}^{\gamma} + \ell_{\mathcal{Q}}^{\beta'})$-time algorithm. Thus, we can simply pick some larger β' such that the corresponding λ' is larger than 0. To do so, let $\beta_{\max} := \frac{\gamma \cdot (\alpha - \nu)}{(\gamma - 1) \cdot \mu}$ the upper bound for β. Note that $\alpha > \nu \cdot \gamma$ implies that

$$\lambda_{\max} := \frac{\beta_{\max}\mu}{\gamma} - \nu = \frac{\frac{\gamma \cdot (\alpha - \nu)}{(\gamma - 1) \cdot \mu}\mu}{\gamma} - \nu = \frac{\alpha - \nu}{\gamma - 1} - \nu = \frac{\alpha - \nu - \nu \cdot (\gamma - 1)}{\gamma - 1} = \frac{\alpha - \nu \cdot \gamma}{\gamma - 1} > 0$$

Thus, we can pick $\beta < \beta' < \beta_{\max}$ such that $\lambda' := \frac{\beta'\mu}{\gamma} - \nu > 0$. ◀

Note that if for $\mathcal{P}$ there is a (conditional) running time lower bound of $\Omega(n^{\alpha})$, then Theorem 2.4 excludes (conditionally) running times of the form $\widetilde{O}(n_{\mathcal{P}} + \ell^{\beta})$ for any $\beta \in \mathbb{R}$ as $\lim_{\gamma \to 1} \frac{\gamma \cdot (\alpha - \nu)}{(\gamma - 1) \cdot \mu} = \infty$. This running time also excludes linear-time computable polynomial-size kernels. More precisely, we get the following.

▶ **Corollary 2.5.** *Let $\alpha > \nu \geq 1$, $\gamma > 1$, and $\mu > 0$ with $\alpha > \nu \cdot \gamma$. Let $\mathcal{P}$ be an α-OR-decomposable problem with parameter $n_{\mathcal{P}}$ that (ν, μ)-OR-cross-composes into a problem $\mathcal{Q}$ with parameters $n_{\mathcal{Q}}$ and $\ell_{\mathcal{Q}}$. Assume that there is an $\widetilde{O}(n_{\mathcal{Q}}^{\xi})$ algorithm for deciding $\mathcal{Q}$ and* *that $n_{\mathcal{Q}}$ is upper bounded by the input size. If there exists an $\widetilde{O}(\ell_{\mathcal{Q}}^{\beta})$-size $\widetilde{O}(n_{\mathcal{Q}}^{\gamma})$-time kernel for $\mathcal{Q}$ for some $\gamma > 1$, $\beta \in \mathbb{R}$, and*

$$0 < \beta < \frac{\gamma \cdot (\alpha - \nu)}{(\gamma - 1) \cdot \mu \cdot \xi},$$

then $\mathcal{P}$ can be solved in $O(n_{\mathcal{P}}^{\alpha-\varepsilon})$ time for some $\varepsilon > 0$.

Proof. An $\widetilde{O}(\ell_{\mathcal{Q}}^{\beta})$-size $\widetilde{O}(n_{\mathcal{Q}}^{\gamma})$-time kernel together with an $\widetilde{O}(n_{\mathcal{Q}}^{\xi})$-time algorithm solving $\mathcal{Q}$ yields an $\widetilde{O}(n_{\mathcal{Q}}^{\gamma} + \ell_{\mathcal{Q}}^{\beta\xi})$ algorithm for $\mathcal{Q}$. The corollary now follows from Theorem 2.4. ◀

Our general approach to apply our framework is follows. Start with a problem $\mathcal{P}$ that (under some hypothesis) cannot be solved in $O(n^{\alpha-\varepsilon})$ time for $\varepsilon > 0$. Then construct an α-decomposition for $\mathcal{P}$ followed by a (1,1)-OR-cross-composition into the target problem.

3 OR-decomposable problems

In order to apply our framework, we first need some OR-decomposable problems. We will observe that three fundamental problems from fine-grained complexity, namely Orthogonal Vectors, 3-SUM and Negative-Weight k-Clique, are OR-decomposable. These problems are also our source for running time lower bounds: Note that the former two problems cannot be solved in $O(n^{2-\varepsilon})$ time unless the SETH respectively 3-Sum-hypothesis fail [50]. We moreover use that Negative-Weight 3-Clique (= Negative Triangle) cannot be solved in $O(n^{3-\varepsilon})$ time unless APSP-hypothesis fails [51].

We will use that "Many problems, like SAT, have a simple self-reduction proving that the "Direct-OR" version is hard, assuming the problem itself is hard" [2]. This self-reduction is based on partitioning the instance into many small ones, with at least one of them containing the small desired structure (i.e., a pair of orthogonal vectors, three numbers summing to 0, or a negative triangle).

Orthogonal Vectors. We now show that Orthogonal Vectors is 2-OR-decomposable.

▶ **Lemma 3.1.** *Orthogonal Vectors parameterized by the number of vectors is 2-OR-decomposable.*

Proof. Let (I, n) be an instance of Orthogonal Vectors and $\lambda > 0$. Set $\epsilon := \frac{2}{2+\lambda}$. Partition A into $z := \lceil n^{1-\epsilon} \rceil$ many sets $A_1, \ldots, A_z$ of at most $\lceil n^{\epsilon} \rceil$ vectors each. Symmetrically, partition B into $B_1, \ldots, B_z$ of at most $\lceil n^{\epsilon} \rceil$ vectors each. We assume without loss of generality that $|A_i| = |B_j| = \lceil n^{\epsilon} \rceil =: n'$ (this can be achieved e.g. by adding the all-one vector). For each pair $(i, j) \in [z]^2$, create an instance $(I_{(i,j)}, n') = ((A_i, B_j), n')$ of Orthogonal Vectors. We claim that this constitutes a 2-OR-decomposition.

The number of vectors n' of each instance $I_{(i,j)}$ is $n' = O(n^{2/(2+\lambda)})$. The number of created instances is $z^2 = O(n^{2\cdot(1-\epsilon)}) = O(n^{2\cdot(1-2/(2+\lambda))}) = O(n^{(4+2\lambda-4)/(2+\lambda)}) = O(n^{2\lambda/(2+\lambda)})$. The running time to compute the decomposition is $\widetilde{O}(z^2 \cdot n') = \widetilde{O}(n^{2-\epsilon})$.

It remains to show that (I, n) is a "Yes"-instance if and only if $(I_{(i,j)}, n')$ is a "Yes"-instance for some $(i, j) \in [z]^2$. First assume that (I, n) is a "Yes"-instance. Then there exists some $a \in A$ and $b \in B$ such that a and b are orthogonal. Let $i \in [z]$ such that $a \in A_i$ and $j \in [z]$ such that $b \in B_j$. Then a and b are orthogonal vectors in $(I_{(i,j)}, n')$, showing that $(I_{(i,j)}, n')$ is a "Yes"-instance.

Finally, assume that there exists $(i^*, j^*) \in [z]^2$ such that $(I_{(i^*,j^*)}, n')$ is a "Yes"-instance. Then there exists $a \in A_{i^*}$ and $b \in B_{j^*}$ which are orthogonal. Consequently, a and b are orthogonal vectors in I, implying that (I, n) is a "Yes"-instance. ◀

▶ Remark 3.2. Note that the above decomposition does not change the dimension d. Thus, even restricted versions of Orthogonal Vectors with $d \in O(\log n)$ are 2-OR-decomposable. Further, we can assume that all constructed instances of Orthogonal Vectors have the same number of vectors and the same dimension.

Negative-Weight k-Clique. We now show that Negative-Weight k-Clique is k-OR-decomposable.

▶ **Lemma 3.3.** *For any $k \geq 3$, Negative-Weight k-Clique parameterized by the number of vertices is k-OR-decomposable.*

Proof. The proof follows the ideas from Lemma 3.1. Let $(I = (G, w), n)$ be an instance of Negative-Weight k-Clique and $\lambda > 0$. Set $\epsilon := \frac{k}{k+\lambda}$. Partition the set $V(G)$ of vertices into $z := \lceil n^{1-\epsilon} \rceil$ many sets $V_1, \ldots, V_z$ of size at most $\lceil n^\epsilon \rceil$. We assume without loss of generality that $|V_i| = \lceil n^\epsilon \rceil$ for all $i \in [z]$ (this can be achieved e.g. by adding isolated vertices). Let $n_{(i_1,\ldots,i_k)} := |\{i_1, \ldots, i_k\}| \cdot |V_1|$. For each tuple $(i_1, \ldots, i_k) \in [z]^k$, create an instance $(I_{(i_1,\ldots,i_k)} = G[A_{i_1} \cup \cdots \cup A_{i_k}], n_{(i_1,\ldots,i_k)})$ of Negative-Weight k-Clique. We claim that this constitutes a k-OR-decomposition.

Each instance $(I_{(i,j)}, n_{(i_1,\ldots,i_k)})$ has at most $O(k \cdot n^\epsilon) = O(n^{k/(k+\lambda)})$ vertices (note that k is a constant here). The number of created instances is $z^k = O(n^{k\cdot(1-\epsilon)}) = O(n^{k\cdot(1-k/(k+\lambda))}) = O(n^{(k^2+k\lambda-k^2)/(k+\lambda)}) = O(n^{k\lambda/(k+\lambda)})$. Further, the summed size of all created instances and therefore also the running time is $\widetilde{O}(z^k \cdot k \cdot n^{2\epsilon}) = \widetilde{O}(n^{k-k\epsilon+2\epsilon}) = \widetilde{O}(n^{k-(k-2)\epsilon}) = \widetilde{O}(n^{k'})$ for some $k' < k$.

It remains to show that (I, n) is a "Yes"-instance if and only if $(I_{(i_1,\ldots,i_k)}, n_{(i_1,\ldots,i_k)})$ is a "Yes"-instance for some $(i_1, \ldots, i_k) \in [z]^k$. First assume that (I, n) is a "Yes"-instance. Thus, G contains a negative-weight clique $C = \{v_1, \ldots, v_k\}$. Let i_j such that $v_j \in V_{i_j}$. Then C is a negative-weight k-clique in $(I_{(i_1,\ldots,i_k)}, n_{(i_1,\ldots,i_k)})$.

Finally, assume that there exists $(i_1, \ldots, i_k) \in [n]^k$ such that $(I_{(i_1,\ldots,i_k)}, n_{(i_1,\ldots,i_k)})$ is a "Yes"-instance. Then there exists a clique in $G[V_{i_1} \cup \cdots \cup V_{i_k}]$. Since G contains $G[V_{i_1} \cup \cdots \cup V_{i_k}]$, also G contains a negative-weight k-clique. ◀

3-Sum. Showing that 3-Sum is 2-OR-decomposable requires some more work. We defer these parts to a full version.

▶ **Lemma 3.4** ($\star$). *3-Sum parameterized by the number of numbers is 2-OR-decomposable.*

Applying the framework. The above results make our framework easier to apply. To apply Theorem 2.4 we only need to provide a suitable OR-cross composition from one of the three problems discussed above. We thus arrive at the following.

▶ **Proposition 3.5.** *Let Q be a problem with parameters n_Q and ℓ_Q. If Orthogonal Vectors resp. 3-Sum parameterized by n $(1,1)$-OR-cross-composes into Q, then an $O(\ell_Q^\beta + n_Q^\gamma)$-time algorithm for Q for any $2 > \gamma > 1$ and $\beta < \gamma/\gamma-1$ refutes the SETH respectively the 3-Sum-Hypothesis.*

4 Applications

We now apply our framework from Theorem 2.4 to several problems from different areas such as string problems (Section 4.1), computational geometry (Section 4.2), and subgraph isomorphism (Section 4.3).

4.1 String problems

We start with Longest Common Subsequence that can be solved in $O(n^2)$ time algorithm [18]. Assuming SETH, there is no algorithm solving Longest Common Subsequence in $O(n^{2-\varepsilon})$ time for any $\varepsilon > 0$ [14, 1]. However, Longest Common Subsequence can be solved in $O(kn + n \log n)$ time, where k is the length of the longest common subsequence [40]. Bringmann and Künnemann [15] proved that this running time is essentially optimal under SETH. Here we reprove this fact using our framework. We refer to Bringmann and Künnemann [15] for a more extensive literature review on Longest Common Subsequence.

A *string* over an alphabet Σ is an element from Σ^*. We access the i-th element of a string x via $x[i]$. A *subsequence* of a string x is a string y such that there is an injective, strictly increasing function f with $y[i] = x[f(i)]$ for all i. A *common subsequence* of two strings x and x' is a string which is a subsequence of both x and x'. For two strings x and y, we denote their concatenation (i.e. the string starting with x and ending with y) by $x \circ y$.

▶ **Lemma 4.1.** *Orthogonal Vectors with logarithmic dimension parameterized by the number of vectors* $(1, 1)$*-OR-cross-composes into Longest Common Subsequence parameterized by the length of the input strings and the length* k *of the longest common subsequence.*

Proof. Let $(I_1^{\mathrm{OV}}, n_1), \ldots, (I_t^{\mathrm{OV}}, n_t)$ be instances of Orthogonal Vectors with logarithmic dimension. We denote by d_i the dimension of I_i^{OV}. Abboud, Backurs, and Williams [1] gave a reduction from Orthogonal Vectors to Longest Common Subsequence which, given an instance of Orthogonal Vectors with n vectors of dimension d, constructs in $n \cdot d^{O(1)}$ time an equivalent instance of Longest Common Subsequence with strings of length $n \cdot d^{O(1)}$. We apply this reduction to each instance (I_i^{OV}, n_i) of OV, giving us an instance $I_i^{\mathrm{LCS}} = ((x_i^1, x_i^2), (n_i^{\mathrm{LCS}}, k_i))$ with $k_i = O(n_i \cdot d_i^{O(1)})$ and $n_i^{\mathrm{LCS}} = n_i \cdot d_i^{O(1)}$. We assume that $k_i = k_j$ for every i, j (this can be achieved by appending identical sequences of appropriate length to strings x_i^1 and x_i^2), and set $k := k_i$. Further, we assume that the alphabets used for I_i^{LCS} and I_j^{LCS} are disjoint for $i \neq j$. We define $x^1 := x_1^1 \circ x_2^1 \circ \cdots \circ x_t^1$ and $x^2 := x_t^2 \circ x_{t-1}^2 \circ \cdots \circ x_1^2$. The OR-cross-composition constructs the instance (x^1, x^2, k) (the parameter is (n^{LCS}, k) with $n^{\mathrm{LCS}} = \sum_{i=1}^t n_i^{\mathrm{LCS}}$).

We now show correctness of the reduction. First assume that (I_i^{OV}, n_i) is a "Yes"-instance for some $i \in [t]$. Then x_i^1 and x_i^2 contain a subsequence y of length $k_i = k$. This subsequence y is also a subsequence of x^1 and x^2, so (x^1, x^2, k) is a "Yes"-instance. Vice versa, assume that (x^1, x^2, k) is a "Yes"-instance, i.e., x^1 and x^2 contain a subsequence y of length k. Let $i \in [t]$ such that the first letter of y is contained in x_i^1. We claim that all letters from y are contained in x_i^1. Since the first letter $y[1]$ of y is contained in x_i^1, no letter of y can be contained in x_j^1 for $j < i$. For $j > i$, note that (as any letter from x_j^2 only appears in x_j^2 and x_j^1) any letter from x_j^1 appears only before x_i^2 in x_2 and thus would have to appear before $y[1]$ in y, a contradiction to $y[1]$ being the first letter of y. Thus, y is contained in x_i^1 and x_i^2. Consequently, (x_i^1, x_i^2, k) is a "Yes"-instance, implying that (I_i, n_i) is a "Yes"-instance.

Computing the OR-cross-composition can be done in $t \cdot \max_{i \in t} n_i \cdot (\max_{i \in [t]} d_i)^{O(1)} = \widetilde{O}(t \max_{i \in [t]} n_i)$ time and $k = O(\max_{i \in t} n_i \cdot (\max_{i \in t} d_i)^{O(1)}) = \widetilde{O}(\max_{i=1} n_i)$ (we use here that $d_i \in O(\log n_i)$). Further, we have $n^{\mathrm{LCS}} = \sum_{i=1}^t n_i^{\mathrm{LCS}} = O(\sum_{i=1}^t n_i \cdot d_i^{O(1)}) = \widetilde{O}(\sum_{i=1}^t)$. Thus, it is an $(1, 1)$-OR-cross-composition. ◀

We remark that the use of alphabets of non-constant size in the above composition is probably necessary as it excludes a linear-time computable kernel of polynomial size. In contrast, Longest Common Subsequence *with* constant alphabet size parameterized by solution size admits a linear-time computable polynomial-size kernel (note that the SETH-based $O(n^{2-\epsilon})$ lower bound also holds for binary alphabets [15]):

▶ **Proposition 4.2.** *Longest Common Subsequence with constant alphabet size parameterized by solution size k admits a linear-time computable polynomial-size kernel.*

Proof. Consider the following reduction rule which directly leads a polynomial kernel. A *substring* of a string x is a string y such that $y = (x[i], x[i+1], \dots, x[j])$ for some $i < j$.

▶ **Reduction Rule 4.3.** If, for some $t \in \mathbb{N}$ and $i \in \{1, 2\}$, a string x^i contains a substring x' of length at least $(k+1)^t$ over only t different letters and not containing a substring x'' of length $(k+1)^{t-1}$ over at most $t-1$ different letters, then we can delete all but the first $(k+1)^t$ letters of x'.

First, we argue that the reduction rule is safe. Let $I = (x^1, x^2, k)$ be an instance of Longest Common Subsequence. We assume without loss of generality that the reduction rule can be applied to x^1, i.e., x^1 contains a substring x' of length $(k+1)^t$ which contains only t different letters, and all substrings of x' containing only $t' < t$ different letters have length at most $(k+1)^{t'}$. Let $I_{\text{red}} = (x^1_{\text{red}}, x^2, k)$ be the instance arising from applying Reduction Rule 4.3. Clearly, if I is a "No"-instance, then also I_{red} is also a "No"-instance.

Now assume that I is a "Yes"-instance, i.e., x^1 and x^2 contain a common subsequence z of length k. By assumption x' contains no substring of length $(k+1)^{t-1}$ which contains at most $t-1$ different letters. Thus, in the first $(k+1)^{t-1} - 1$ letters of x', each of the t letters appears at least once. More generally, for any $i \in [k]$, each letter appears at least once among $x'[(i-1) \cdot (k+1)^{t-1} + 1], \dots, x'[i \cdot (k+1)^{t-1}]$. Consequently, each string of length k over the t appearing letters is a subsequence of x'. Hence, z is also a subsequence of x^1_{red} (and thus a common subsequence of x^1_{red} and x^2), showing that I_{red} is also a "Yes"-instance. Next, we analyze the time needed to exhaustively apply Reduction Rule 4.3.

▷ Claim 4.4. For constant alphabet size, Reduction Rule 4.3 can be exhaustively applied in linear time.

Proof. We process x^1 and x^2 from left to right. For each subset S of the alphabet, we have a variable storing the number of consecutive letters from S are before the current letter. If this number exceeds $(k+1)^{|S|}$ for some letter, then we delete this letter. Note that whenever we found a string of length $(k+1)^{|S|}$ containing only letters from S, then this string does not contain a substring of length $(k+1)^{|S|-1}$ containing only letters from $S \setminus \{\ell\}$ for some $\ell \in S$ as in this case, the number stored for $S \setminus \{\ell\}$ would have exceeded $(k+1)^{|S|-1}$.

The above procedure clearly runs in linear time as the alphabet size is constant. ◁

After the exhaustive application of Reduction Rule 4.3, the size of the instance is clearly $\mathcal{O}(k^{|\Sigma|})$. As $|\Sigma|$ is constant, this is a polynomial-sized kernel. ◀

Analogously to Longest Common Subsequence, we derive similar hardness results for the related problems Longest Common Weakly Increasing Subsequence and Longest Common Increasing Subsequence. Both problems can be solved in slightly subquadratic time [7, 23]. For Longest Common Increasing Subsequence, our lower bound was already shown by Duraj et al. [24]. We refer to Duraj et al. [24] and Duraj [23] for a broader literature review on Longest Common Subsequence.

▶ **Lemma 4.5** ($\star$). *Orthogonal Vectors with logarithmic dimension $(1,1)$-OR-cross-composes into Longest Common (Weakly) Increasing Subsequence parameterized by the length n of the strings and the length k of the longest common (weakly) increasing subsequence.*

Combining Proposition 3.5 and Lemmas 4.1 and 4.5, we get the following result:

▶ **Corollary 4.6.** *Unless the SETH fails, there is no $\widetilde{O}(n^\gamma + k^\beta)$-time algorithm for* Longest Common Subsequence *or* Longest Common (Weakly) Increasing Subsequence *parameterized by solution size k for $1 < \gamma < 2$ and $\beta < \frac{\gamma}{\gamma-1}$.*

We remark that the running time lower bounds are tight; the tight upper bound follows from the known $O(kn+n\log n)$ time algorithm for Longest Common Subsequence [40] or the $O(nk\log\log n + n\log n)$ time algorithm for Longest Common (Weakly) Increasing Subsequence [45] and Observation 1.1.

4.2 Computational Geometry

We now turn to problems from computational geometry. We will denote the Euclidean distance between two points p and q in the plane by $\mathrm{dist}(p, q)$. For a list of points P, a *tour* through P is a surjective, non-increasing function $f_P : [2n] \to [n]$ such that $f_P(1) = 1$ and $f_P(2n) = n$. The input of Discrete Fréchet Distance contains two lists of points $P = (p_1, \ldots, p_n)$ and $Q = (q_1, \ldots, q_n)$. A pair (f_P, f_Q) of tours through P and Q is called a *traversal*.

Discrete Fréchet Distance can be solved in $O(n^2 \cdot \log\log n/\log^2 n)$ time [6]. Bringmann [13] gave a linear-time reduction from Orthogonal Vectors[2] to Discrete Fréchet Distance, showing that assuming SETH, Discrete Fréchet Distance cannot be solved in $O(n^{2-\epsilon})$ time for any $\epsilon > 0$. We use this reduction to get a $(1, 1)$-OR-composition for the parameter $\max_{i\in[2n]} |f_P(i) - f_Q(i)|$, i.e., the maximum number of time steps which P may be ahead of Q (or vice versa) in the optimal solution. We will call the parameter $\max_{i\in[2n]} |f_P(i) - f_Q(i)|$ the *maximum shift* of the traversal (f_P, f_Q). For a more extensive literature review on Discrete Fréchet Distance, we refer to Agarwal et al. [6].

▶ **Lemma 4.7** ($\star$). *Orthogonal Vectors admits a $(1, 1)$-OR-cross-composition into Discrete Fréchet Distance parameterized by the length of the input lists of points and the maximum shift in a solution.*

Combining Proposition 3.5 and Lemma 4.7, yields the following:

▶ **Corollary 4.8.** *Unless the SETH fails, there is no $\widetilde{O}(n^\gamma + \ell^\beta)$-time algorithm* Discrete Fréchet Distance *parameterized by maximum shift ℓ for $1 < \gamma < 2$ and $\beta < \frac{\gamma}{\gamma-1}$.*

It is easy to extend the $O(n^2)$-algorithm for Discrete Fréchet Distance [25] to run in $O(n\ell)$ time, where ℓ is the minimal (over all solutions) maximum shift of an optimal solution [8]. This together with Observation 1.1 shows that the running time lower bound from Corollary 4.8 is tight.

We now give lower bounds for another problem from computational geometry, Planar Motion Planning, based not on the hardness of Orthogonal Vectors but on the hardness of 3-Sum. Planar Motion Planning can be solved in $O(n^2)$ time [52]. Assuming the 3-Sum conjecture, Planar Motion Planning cannot be solved in $O(n^{2-\epsilon})$ for any $\epsilon > 0$ [33]. We say a segment is in the *vicinity* of another segment if they have distance at most the length of the rod.

▶ **Lemma 4.9.** *3-Sum $(1, 1)$-OR-cross-composes into Planar Motion Planning parameterized by the maximum number of segments any segment has in its vicinity.*

[2] Bringmann [13] reduces from Satisfiability, but his reduction implicitly reduces Satisfiability to Orthogonal Vectors and then Orthogonal Vectors to Discrete Fréchet Distance.

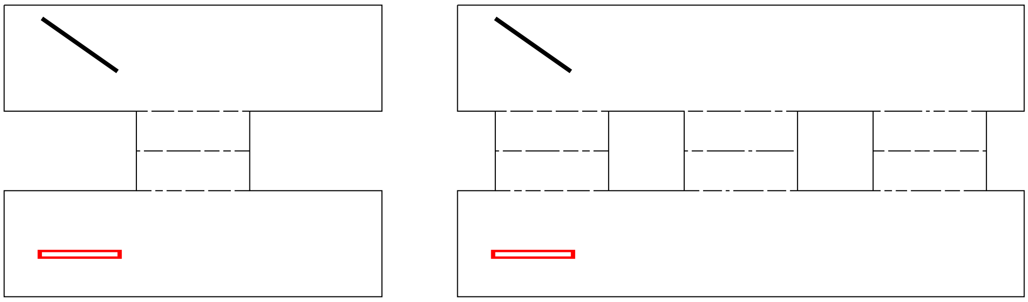

Figure 2 Left: Exemplary illustration of an instance of Planar Motion Planning constructed by the reduction from 3-Sum from Gajentaan and Overmars [33]. Right: An example for the OR-cross-composition of three instances of 3-Sum into Planar Motion Planning. The goal position of the rod is surrounded by red.

Proof sketch. Let $I_1^{\text{3-Sum}}, \dots, I_t^{\text{3-Sum}}$ be instances of 3-Sum. We denote by n_i the number of numbers of $I_1^{\text{3-Sum}}$. Gajentaan and Overmars [33] gave a reduction from 3-Sum to Planar Motion Planning which, given an instance of 3-Sum with n numbers, constructs in $\widetilde{O}(n)$ time an instance of Planar Motion Planning where the rod initially is in a large "upper" rectangle and has to reach a "lower" rectangle through a narrow passage (see Figure 2 for a proof by picture). We apply this reduction to each instance $I_i^{\text{3-Sum}}$ of 3-Sum, giving us an instance I_i^{PMP} with $\widetilde{O}(n_i)$ many segments. From these instances, we construct an instance of Planar Motion Planning as follows: We identify the large rectangles in which the rod starts from each I_i^{PMP}. The narrow passages are copied next to each other (if the large starting rectangle is not wide enough, then we make it wider.

The correctness of the reduction is obvious.

Any segment from an instance $I_i^{\text{3-Sum}}$ has distance at most the length of the rod except for the bounding boxes. By splitting the segments of the bounding box into many small segments not longer than the rod, we get that for each segment s there are at most $O(n)$ segments whose distance to s is at most the length of the rod. ◀

Combining Proposition 3.5 and Lemma 4.9 yields the following:

▶ **Corollary 4.10.** *Unless the 3SUM-hypothesis fails, there is no $\widetilde{O}(n^\gamma + \ell^\beta)$-time algorithm Planar Motion Planning parameterized by the maximum number ℓ of segments any segment has in its vicinity for $\beta < \frac{\gamma}{\gamma-1}$.*

The lower bound for the running time is tight by the $O(K^2 \log n)$ time algorithm from Sifrony and Sharir [49] (where K is the number of segment pairs whose distance is less than the length of the rod), the observation that $K^2 \leq n \cdot \ell$, and Observation 1.1.

4.3 Graph Problems

We now turn to graph problems. First, we consider Minimum Weight k-Clique parameterized by the maximum size of a connected component.

▶ **Proposition 4.11.** *Minimum Weight k-Clique $(1,1)$-OR-cross-composes into Minimum Weight k-Clique parameterized by the maximum size of a connected component.*

Proof. Let $G_1, \dots, G_t$ be instances of Minimum Weight k-Clique. The cross-composition just computes the disjoint union G of $G_1, \dots, G_t$.

Clearly, the cross-composition is correct, runs in linear time, and the maximum size of a connected component is bounded by the maximum size of $G_1, \ldots, G_s$. ◀

As Negative Triangle is the special case of Minimum Weight k-Clique, combining Proposition 3.5 and Proposition 4.11 yields the following lower bound:

▶ **Corollary 4.12.** *Unless the APSP-hypothesis fails, there is no $\widetilde{O}(n^\gamma + \ell^\beta)$-time algorithm for Negative Triangle parameterized by the maximum size ℓ of a connected component for $1 < \gamma < 3$ and $\beta < \frac{2\cdot\gamma}{\gamma-1}$.*

An algorithm running in $O(\ell^2 n)$ where ℓ is the maximum size of a connected component is trivial. This together with Observation 1.1 implies that the running time lower bound is tight.

Next, we turn to 2nd Shortest Path which can be solved in $\widetilde{O}(mn)$ [46] or in $O(Mn^\omega)$ time (where M is the largest edge weight) [36]. If the graph is undirected [42] or one aims to find a 2nd shortest walk [26], then there is a quasi-linear-time algorithm. For unweighted directed graphs, the problem can be solved in $\widetilde{O}(m\sqrt{n})$ time [48]. An ϵ-approximation can be computed in $\widetilde{O}(\frac{m}{\epsilon})$ time [10].

▶ **Lemma 4.13.** *Negative Triangle $(1,1)$-OR-cross-composes into 2nd Shortest Path parameterized by directed feedback vertex number.*

Proof. Let $(I_1^{\mathrm{NT}}, n_1), \ldots, (I_t^{\mathrm{NT}}, n_t)$ be instances of Negative Triangle. We assume without loss of generality that each instance of Negative Triangle has the same number n of vertices, i.e., $n_i = n$ for all $i \in [t]$, and that the largest absolute value of an edge weight is the same for all instances. Vassilevska Williams and Williams [51] gave a linear-time reduction from Negative Triangle to 2nd Shortest Path which, given an instance $(I_i^{\mathrm{NT}} = (G_i, w_i), n)$ of Negative Triangle, creates an instance I_i^{2SP} of 2nd Shortest Path as follows: For each vertex $v \in V(G_i)$, add three vertices a^v, b^v, and c^v. Further, add two vertices s^i and t^i as well as a path $s^i, p_1^i, p_2^i, \ldots, p_n^i, t^i$, all of whose edges have weight 0. Further, there are edges (p_i, a^v), (a^v, b^u), (b^u, c^v), and (c^v, p_i) for every $i \in [n], u, v \in V(G_i)$. All these edges have positive weight (depending on the edge-weights in $E(G_i)$). Identifying s^i with s^j, t^i with t^j, and p_ℓ^i with p_ℓ^j for all $i, j \in [t]$ and $j \in [\ell]$ then results in one instance of 2nd Shortest Path with a directed feedback vertex set of size n, namely $\{p_1, \ldots, p_n\}$. As the constructed instance is equivalent to the disjunction of $I_1^{\mathrm{2SP}}, \ldots, I_t^{\mathrm{2SP}}$ (it is never beneficial to leave $s, p_1, \ldots, p_n, t$ more than once as all edges leaving this path have positive weight), we have a $(1,1)$-OR-cross-composition. ◀

Combining Proposition 3.5 with Lemma 4.13 yields the following:

▶ **Corollary 4.14.** *Unless the APSP-hypothesis fails, there is no $\widetilde{O}(n^\gamma + \ell^\beta)$-time algorithm for 2nd Shortest Path parameterized by directed feedback vertex set ℓ parameterized by the maximum size ℓ of a connected component for $1 < \gamma < 3$ and $\beta < \frac{2\gamma}{\gamma-1}$.*

In contrast to the other problems studied in this paper, we do not know whether the running time lower bounds for 2nd Shortest Path are tight.

4.4 Triangle Collection

Last, we consider the Triangle Collection problem. Triangle Collection can trivially be solved in $O(n^3)$ time, but does not admit an $O(n^{3-\epsilon})$ time algorithm assuming SETH, the 3-Sum conjecture, *or* the APSP conjecture [4]. For this problem, we were unable to apply our framework directly, i.e., to find an OR-cross-composition from an OR-decomposable problem to Triangle Collection. However, we can still get a lower bound in a very similar fashion by combining decomposition and composition into one step. The difference is that the decomposition of Triangle Collection that we use in the proof of the following result does not admit the "OR"-property.

▶ **Proposition 4.15.** *Unless Orthogonal Vectors, 3-Sum, or APSP-hypothesis fails, there is no $\widetilde{O}(n^{\gamma}+\ell^{\beta})$-time algorithm for Negative Triangle parameterized by the maximum size ℓ of a connected component for $1 < \gamma < 3$ and $\beta < \frac{2\cdot\gamma}{\gamma-1}$.*

Proof. Fix $1 < \gamma < 3$. Let G be in an instance of Triangle Collection. Partition $V(G)$ into $z := n^{\lambda/3+\lambda}$ sets $V_1, \dots, V_z$ of size $q := n^{3/(3+\lambda)}$, where $\lambda := \beta/\gamma - 1$. For each $(i,j,k) \in [z]^3$, let $G_{(i,j,k)}$ be the graph induced by $V_i \cup V_j \cup V_k$. Finally, let H be the union of all $G_{(i,j,k)}$. Note that H corresponds to the output of the OR-decomposition in our framework. Clearly G has a triangle collection if and only if H has. Further, H can be computed in $\widetilde{O}(q^2 \cdot z^3) = \widetilde{O}(n^{3(\lambda+2)/(3+\lambda)})$ time (note that each of the z^3 graphs $G_{(i,j,k)}$ has size $O(q^2)$ as a q-vertex graph may have $O(q^2)$ many edges).

Now assume that there is a $\widetilde{O}(n^{\gamma} + \ell^{\beta})$-algorithm for Triangle Collection with $\beta < \frac{2\cdot\gamma}{\gamma-1}$ and apply this algorithm to H. This clearly solves H. The running time for this step is $\widetilde{O}((z^3 \cdot q)^{\gamma} + q^{\beta}) = \widetilde{O}(n^{\gamma\cdot(3\cdot(\lambda+1)/(3+\lambda))} + n^{\beta\cdot 3/(3+\lambda)})$ time. Note that $\gamma\cdot 3\cdot(\lambda+1)/(3+\lambda) = 3 \cdot \frac{\beta/\gamma}{2+\beta/\gamma} < 3$. Further, it holds that

$$\beta \cdot 3/(3+\lambda) = 3 \cdot \frac{\beta}{2+\beta/\gamma} = 3 \cdot \frac{\beta\gamma}{2\gamma+\beta} = 3 \cdot \frac{\gamma}{2\gamma/\beta + 1} < 3 \cdot \frac{\gamma}{2\gamma/\left(\frac{2\gamma}{(\gamma-1)}\right) + 1} = 3 \cdot \frac{\gamma}{(\gamma-1)+1} = 3$$

where we used the assumption $\beta < \frac{2\cdot\gamma}{\gamma-1}$ for the inequality. Consequently, we can solve Triangle Collection in $O(n^{3-\epsilon})$ time for some $\epsilon > 0$. ◀

As an $O(n\ell^2)$ time algorithm for Triangle Collection is trivial, it follows from Observation 1.1 that the running time lower bound is tight.

5 Conclusion

We introduced a framework for extending conditional running time lower bounds to parameterized running time lower bounds and applied it to various problems. Beyond the clear task to apply the framework to further problems there are further challenges for future work. For example, can we get "AND-hard" problems so that we can use AND-cross compositions similar to the ones used to exclude compression [21]? Moreover, can the framework be adapted to cope with dynamic, counting, or enumerating problems?

References

1. Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS 2015)*, pages 59–78. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.14.
2. Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. SETH-based lower bounds for subset sum and bicriteria path. *ACM Transactions on Algorithms*, 18(1):6:1–6:22, 2022. doi:10.1145/3450524.
3. Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016)*, pages 377–391. SIAM, 2016. doi:10.1137/1.9781611974331.ch28.

4 Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. *SIAM Journal on Computing*, 47(3):1098–1122, 2018. doi:10.1137/15M1050987.

5 Faisal N. Abu-Khzam, Sebastian Lamm, Matthias Mnich, Alexander Noe, Christian Schulz, and Darren Strash. Recent advances in practical data reduction. *CoRR*, abs/2012.12594, 2020. arXiv:2012.12594.

6 Pankaj K. Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. Computing the discrete Fréchet distance in subquadratic time. *SIAM Journal on Computing*, 43(2):429–449, 2014. doi:10.1137/130920526.

7 Anadi Agrawal and Pawel Gawrychowski. A faster subquadratic algorithm for the longest common increasing subsequence problem. In *Proceedings of the 31st International Symposium on Algorithms and Computation (ISAAC 2020)*, volume 181 of *LIPIcs*, pages 4:1–4:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ISAAC.2020.4.

8 Jérémy Barbay. Adaptive computation of the discrete fréchet distance. In *Proceeding of the 25th International Symposium on String Processing and Information Retrieval (SPIRE 2018)*, volume 11147 of *Lecture Notes in Computer Science*, pages 50–60. Springer, 2018. doi:10.1007/978-3-030-00479-8_5.

9 Matthias Bentert, Till Fluschnik, André Nichterlein, and Rolf Niedermeier. Parameterized aspects of triangle enumeration. *Journal of Computer and System Sciences*, 103:61–77, 2019. doi:10.1016/j.jcss.2019.02.004.

10 Aaron Bernstein. A nearly optimal algorithm for approximating replacement paths and k shortest simple paths in general graphs. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 742–755. SIAM, 2010. doi:10.1137/1.9781611973075.61.

11 Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009. doi:10.1016/j.jcss.2009.04.001.

12 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics*, 28(1):277–305, 2014. doi:10.1137/120880240.

13 Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *Proceedings of the 55th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2014)*, pages 661–670. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.76.

14 Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS 2015)*, pages 79–97. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.15.

15 Karl Bringmann and Marvin Künnemann. Multivariate fine-grained complexity of longest common subsequence. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*, pages 1216–1235. SIAM, 2018. doi:10.1137/1.9781611975031.79.

16 Timothy M. Chan and R. Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing razborov-smolensky. *ACM Transactions on Algorithms*, 17(1):2:1–2:14, 2021. doi:10.1145/3402926.

17 Yijia Chen, Jörg Flum, and Moritz Müller. Lower bounds for kernelizations and other preprocessing procedures. *Theory of Computing Systems*, 48(4):803–839, 2011. doi:10.1007/s00224-010-9270-y.

18 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009. URL: http://mitpress.mit.edu/books/introduction-algorithms.

19 David Coudert, Guillaume Ducoffe, and Alexandru Popa. Fully polynomial FPT algorithms for some classes of bounded clique-width graphs. *ACM Transactions on Algorithms*, 15(3):33:1–33:57, 2019. doi:10.1145/3310228.

20 Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *Journal of the ACM*, 61(4):23:1–23:27, 2014. doi:10.1145/2629620.

21 Andrew Drucker. New limits to classical and quantum instance compression. *SIAM Journal on Computing*, 44(5):1443–1479, 2015. doi:10.1137/130927115.

22 Guillaume Ducoffe. Maximum matching in almost linear time on graphs of bounded clique-width. In *Proceedings of the 16th International Symposium on Parameterized and Exact Computation (IPEC 2021)*, volume 214 of *LIPIcs*, pages 15:1–15:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.IPEC.2021.15.

23 Lech Duraj. A sub-quadratic algorithm for the longest common increasing subsequence problem. In *Proceedings of the 37th International Symposium on Theoretical Aspects of Computer Science (STACS 2020)*, volume 154 of *LIPIcs*, pages 41:1–41:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.STACS.2020.41.

24 Lech Duraj, Marvin Künnemann, and Adam Polak. Tight conditional lower bounds for longest common increasing subsequence. *Algorithmica*, 81(10):3968–3992, 2019. doi:10.1007/s00453-018-0485-7.

25 Thomas Eiter and Heikki Mannila. Computing discrete Fréchet distance. Technical report, Technische Universität Wien, 1994.

26 David Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673, 1998. doi:10.1137/S0097539795290477.

27 Henning Fernau, Till Fluschnik, Danny Hermelin, Andreas Krebs, Hendrik Molter, and Rolf Niedermeier. Diminishable parameterized problems and strict polynomial kernelization. *Computability*, 9(1):1–24, 2020. doi:10.3233/COM-180220.

28 Till Fluschnik, Christian Komusiewicz, George B. Mertzios, André Nichterlein, Rolf Niedermeier, and Nimrod Talmon. When can graph hyperbolicity be computed in linear time? *Algorithmica*, 81(5):2016–2045, 2019. doi:10.1007/s00453-018-0522-6.

29 Till Fluschnik, George B. Mertzios, and André Nichterlein. Kernelization lower bounds for finding constant-size subgraphs. In *Proceedings of the 14th Conference on Computability in Europe (CiE 2018)*, volume 10936 of *Lecture Notes in Computer Science*, pages 183–193. Springer, 2018. doi:10.1007/978-3-319-94418-0_19.

30 Fedor V. Fomin, Daniel Lokshtanov, Michal Pilipczuk, Saket Saurabh, and Marcin Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. *ACM Transactions on Algorithms*, 14(3):34:1–34:45, 2018. doi:10.1145/3186898.

31 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019. doi:10.1017/9781107415157.

32 Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct pcps for NP. *Journal of Computer and System Sciences*, 77(1):91–106, 2011.

33 Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry*, 5:165–185, 1995. doi:10.1016/0925-7721(95)00022-2.

34 Archontia C. Giannopoulou, George B. Mertzios, and Rolf Niedermeier. Polynomial fixed-parameter algorithms: A case study for longest path on interval graphs. *Theoretical Computer Science*, 689:67–95, 2017. doi:10.1016/j.tcs.2017.05.017.

35 Szymon Grabowski. New tabulation and sparse dynamic programming based techniques for sequence similarity problems. *Discrete Applied Mathematics*, 212:96–103, 2016. doi:10.1016/j.dam.2015.10.040.

36 Fabrizio Grandoni and Virginia Vassilevska Williams. Faster replacement paths and distance sensitivity oracles. *ACM Transactions on Algorithms*, 16(1):15:1–15:25, 2020. doi:10.1145/3365835.

37 Klaus Heeger, André Nichterlein, and Rolf Niedermeier. Parameterized lower bounds for problems in p via fine-grained cross-compositions. *CoRR*, abs/10.48550, 2023.

38 Falko Hegerfeld and Stefan Kratsch. On adaptive algorithms for maximum matching. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *LIPIcs*, pages 71:1–71:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. `doi:10.4230/LIPIcs.ICALP.2019.71`.

39 Monika Henzinger, Alexander Noe, Christian Schulz, and Darren Strash. Finding all global minimum cuts in practice. In *Proceedings of the 28th Annual European Symposium on Algorithms (ESA 2020)*, volume 173 of *LIPIcs*, pages 59:1–59:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.ESA.2020.59`.

40 Daniel S. Hirschberg. Algorithms for the longest common subsequence problem. *Journal of the ACM*, 24(4):664–675, 1977. `doi:10.1145/322033.322044`.

41 Yoichi Iwata, Tomoaki Ogasawara, and Naoto Ohsaka. On the power of tree-depth for fully polynomial FPT algorithms. In *Proceedings of the 35th Symposium on Theoretical Aspects of Computer Science (STACS 2018)*, volume 96 of *LIPIcs*, pages 41:1–41:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. `doi:10.4230/LIPIcs.STACS.2018.41`.

42 Naoki Katoh, Toshihide Ibaraki, and Hisashi Mine. An efficient algorithm for K shortest simple paths. *Networks*, 12(4):411–427, 1982. `doi:10.1002/net.3230120406`.

43 Tomohiro Koana, Viatcheslav Korenwein, André Nichterlein, Rolf Niedermeier, and Philipp Zschoche. Data reduction for maximum matching on real-world graphs: Theory and experiments. *ACM Journal of Experimental Algorithmics*, 26:1.3:1–1.3:30, 2021. `doi:10.1145/3439801`.

44 Stefan Kratsch and Florian Nelles. Efficient and adaptive parameterized algorithms on modular decompositions. In *Proceedings of the 26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112 of *LIPIcs*, pages 55:1–55:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. `doi:10.4230/LIPIcs.ESA.2018.55`.

45 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. `doi:10.1016/j.jda.2011.03.013`.

46 Eugene L. Lawler. A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem. *Management Sci.*, 18:401–405, 1971/72. `doi:10.1287/mnsc.18.7.401`.

47 George B. Mertzios, André Nichterlein, and Rolf Niedermeier. The power of linear-time data reduction for maximum matching. *Algorithmica*, 82(12):3521–3565, 2020. `doi:10.1007/s00453-020-00736-0`.

48 Liam Roditty and Uri Zwick. Replacement paths and k simple shortest paths in unweighted directed graphs. *ACM Transactions on Algorithms*, 8(4):33:1–33:11, 2012. `doi:10.1145/2344422.2344423`.

49 Shmuel Sifrony and Micha Sharir. A new efficient motion-planning algorithm for a rod in two-dimensional polygonal space. *Algorithmica*, 2:367–402, 1987. `doi:10.1007/BF01840368`.

50 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018*, pages 3447–3487. World Scientific, 2018.

51 Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *Journal of the ACM*, 65(5):27:1–27:38, 2018. `doi:10.1145/3186893`.

52 Gert Vegter. The visibility diagram: a data structure for visibility problems and motion planning. In *Proceedings of the 2nd Scandinavian Workshop on Algorithm Theory (Swat 1990)*, volume 447 of *Lecture Notes in Computer Science*, pages 97–110. Springer, 1990. `doi:10.1007/3-540-52846-6_81`.

53 R. Ryan Williams. Faster all-pairs shortest paths via circuit complexity. *SIAM Journal on Computing*, 47(5):1965–1985, 2018. `doi:10.1137/15M1024524`.