Finding and Counting Permutations via CSPs

Benjamin Aram Berendsohn
Institut für Informatik, Freie Universität Berlin, Germany
beab@zedat.fu-berlin.de

László Kozma
Institut für Informatik, Freie Universität Berlin, Germany
laszlo.kozma@fu-berlin.de

Dániel Marx
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
dmarx@mpi-inf.mpg.de

Abstract

Permutation patterns and pattern avoidance have been intensively studied in combinatorics and computer science, going back at least to the seminal work of Knuth on stack-sorting (1968). Perhaps the most natural algorithmic question in this area is deciding whether a given permutation of length n contains a given pattern of length k.

In this work we give two new algorithms for this well-studied problem, one whose running time is $n^{k/4+o(k)}$, and a polynomial-space algorithm whose running time is the better of $O(1.6181^n)$ and $O(n^{k/2+1})$. These results improve the earlier best bounds of $n^{0.47k+o(k)}$ and $O(1.79^n)$ due to Ahal and Rabinovich (2000) resp. Bruner and Lackner (2012) and are the fastest algorithms for the problem when $k \in \Omega(\log n)$. We show that both our new algorithms and the previous exponential-time algorithms in the literature can be viewed through the unifying lens of *constraint-satisfaction*.

Our algorithms can also *count*, within the same running time, the number of occurrences of a pattern. We show that this result is close to optimal: solving the counting problem in time $f(k) \cdot n^{o(k/\log k)}$ would contradict the *exponential-time hypothesis* (ETH). For some special classes of patterns we obtain improved running times. We further prove that 3-*increasing* and 3-*decreasing* permutations can, in some sense, *embed* arbitrary permutations of almost linear length, which indicates that an algorithm with sub-exponential running time is unlikely, even for patterns from these restricted classes.

2012 ACM Subject Classification Theory of computation → Data structures design and analysis; Theory of computation → Pattern matching

Keywords and phrases permutations, pattern matching, constraint satisfaction, exponential time

Digital Object Identifier 10.4230/LIPIcs.IPEC.2019.1

Related Version A full version of the paper is available at `https://arxiv.org/abs/1908.04673`.

Funding *Dániel Marx*: Supported by the European Research Council (ERC) Consolidator Grant No. 725978 SYSTEMATICGRAPH.

Acknowledgements An earlier version of the paper contained a mistake in the analysis of the algorithm for Theorem 2. We thank Günter Rote for pointing out the error.
This work was prompted by the Dagstuhl Seminar 18451 "Genomics, Pattern Avoidance, and Statistical Mechanics". The second author thanks the organizers for the invitation and the participants for interesting discussions.

1 Introduction

Let $[n] = \{1, \ldots, n\}$. Given two permutations $\tau : [n] \to [n]$, and $\pi : [k] \to [k]$, we say that τ *contains* π, if there are indices $1 \leq i_1 < \cdots < i_k \leq n$ such that $\tau(i_j) < \tau(i_\ell)$ if and only if $\pi(j) < \pi(\ell)$, for all $1 \leq j, \ell \leq k$. In other words, τ contains π, if the sequence $(\tau(1), \ldots, \tau(n))$

has a (possibly non-contiguous) subsequence with the same ordering as $(\pi(1), \ldots, \pi(k))$, otherwise τ *avoids* π. For example, $\tau = (1, 5, 4, 6, 3, 7, 8, 2)$ contains $(2, 3, 1)$, because its subsequence $(5, 6, 3)$ has the same ordering as $(2, 3, 1)$; on the other hand, τ avoids $(3, 1, 2)$.

Knuth showed in 1968 [40, § 2.2.1], that permutations sortable by a single stack are exactly those that avoid $(2, 3, 1)$. Sorting by restricted devices has remained an active research topic [53, 46, 48, 12, 3, 5], but permutation pattern avoidance has also taken on a life of its own (especially after the influential work of Simion and Schmidt [50]), becoming an important subfield of combinatorics. For more background on permutation patterns and pattern avoidance we refer to the extensive survey [55] and relevant textbooks [13, 14, 39].

Perhaps the most important enumerative result related to permutation patterns is the theorem of Marcus and Tardos [41] from 2004, stating that the number of length-n permutations that avoid a fixed pattern π is bounded by $c(\pi)^n$, where $c(\pi)$ is a quantity independent of n. (This was conjectured by Stanley and Wilf in the late 1980s.)

A fundamental algorithmic problem in this context is *Permutation Pattern Matching* (PPM): Given a length-n permutation τ ("text") and a length-k permutation π ("pattern"), decide whether τ contains π.

Solving PPM is a bottleneck in experimental work on permutation patterns [4]. The problem and its variants also arise in practical applications, e.g. in computational biology [39, § 2.4] and time-series analysis [38, 10, 45]. Unfortunately PPM is, in general, NP-complete, as shown by Bose, Buss, and Lubiw [15] in 1998. For small (e.g. constant-sized) patterns, the problem is solvable in polynomial (in fact, linear) time, as shown by Guillemot and Marx [33] in 2013. Their algorithm has running time $n \cdot 2^{O(k^2 \log k)}$, establishing the *fixed-parameter tractability* of the PPM problem in terms of the pattern length. The algorithm builds upon the Marcus-Tardos proof of the Stanley-Wilf conjecture and introduces a novel decomposition of permutations. Subsequently, Fox [30] refined the Marcus-Tardos result, thereby removing a factor $\log k$ from the exponent of the Guillemot-Marx bound. (Due to the large constants involved, it is however, not clear whether the algorithm can be efficient in practice.)

For longer patterns, e.g. for $k \in \Omega(\log n)$, the complexity of the PPM problem is less understood. An obvious algorithm with running time $O(n^{k+1})$ is to enumerate all $\binom{n}{k}$ length-k subsequences of τ, checking whether any of them has the same ordering as π. The first result to break this "triviality barrier" was the $O(n^{2k/3+1})$-time algorithm of Albert, Aldred, Atkinson, and Holton [4]. Shortly thereafter, Ahal and Rabinovich [1] obtained the running time $n^{0.47k+o(k)}$.

The two algorithms are based on a similar dynamic programming approach: they embed the entries of the pattern π one-by-one into the text τ, while observing the restrictions imposed by the current partial embedding. The order of embedding (implicitly) defines a *path-decomposition* of a certain graph derived from the pattern π, called the *incidence graph*. The running time obtainable in this framework is of the form $O(n^{\mathrm{pw}(\pi)+1})$, where $\mathrm{pw}(\pi)$ is the *pathwidth* of the incidence graph of π.

Ahal and Rabinovich also describe a different, *tree-based* dynamic programming algorithm that solves PPM in time $O(n^{2 \cdot \mathrm{tw}(\pi)+1})$, where $\mathrm{tw}(\pi)$ is the *treewidth* of the incidence graph of π. Using known bounds on the treewidth, however, this running time does not improve the previous one.

Our first result is based on the observation that PPM can be formulated as a *constraint satisfaction problem* (CSP) with binary constraints. In this view, the path-based dynamic programming of previous works has a natural interpretation not observed earlier: it amounts to solving the CSP instance by *Seidel's invasion algorithm*, a popular heuristic [49],[54, § 9.3].

It is well-known that binary CSP instances can be solved in time $O(n^{t+1})$, where n is the *domain size*, and t is the *treewidth* of the *constraint graph* [32, 24]. In our reduction, the domain size is the length n of the text τ, and the constraint graph is the incidence graph of the pattern π; we thus obtain a running time of $O(n^{\mathrm{tw}(\pi)+1})$, improving upon the earlier $O(n^{2\cdot\mathrm{tw}(\pi)+1})$. Second, making use of a bound known for low-degree graphs [28], we prove that the treewidth of the incidence graph of π is at most $k/3 + o(k)$. The final improvement from $k/3$ to $k/4$ is achieved via a technique inspired by recent work of Cygan, Kowalik, and Socała [23] on the k-OPT heuristic for the *traveling salesman problem* (TSP).

In summary, we obtain the following result, proved in § 3.

▶ **Theorem 1.** *Permutation Pattern Matching can be solved in time $n^{k/4+o(k)}$.*

Expressed in terms of n only, none of the mentioned running times improve, in the worst case, upon the trivial 2^n; consider the case of a pattern of length $k \geq n/\log n$. The first improvement in this parameter range was obtained by Bruner and Lackner [18]; their algorithm runs in time $O(1.79^n)$.

The algorithm of Bruner and Lackner works by decomposing both the text and the pattern into *alternating runs* (consecutive sequences of increasing or decreasing elements), and using this decomposition to restrict the space of admissible matchings. The exponent in the running time is, in fact, the *number of runs* of T, which can be as large as n. The approach is compelling and intuitive, the details, however, are intricate (the description of the algorithm and its analysis in [18] take over 24 pages).

Our second result improves this running time to $O(1.618^n)$, with an exceedingly simple approach. A different analysis of our algorithm yields the bound $O(n^{k/2+1})$, i.e. slightly above the Ahal-Rabinovich bound [1], but with polynomial space. The latter bound also matches an earlier result of Guillemot and Marx [33, § 7], obtained via involved techniques.

▶ **Theorem 2.** *Permutation Pattern Matching can be solved using polynomial space, in time $O(1.6181^n)$ or $O(n^{k/2+1})$.*

At the heart of this algorithm is the following observation: if all *even-index* entries of the pattern π are matched to entries of the text τ, then verifying whether the remaining *odd-index* entries of π can be correctly matched takes only a linear-time sweep through both π and τ. This algorithm can be explained very simply in the CSP framework: after substituting a value to every *even-index* variable, the graph of the remaining constraints is a union of paths, and hence can be handled very easily.

Counting patterns. We also consider the closely related problem of *counting* the number of occurrences of π in τ, i.e. finding the number of subsequences of τ that have the same ordering as π. Easy modifications of our algorithms solve this problem within the bounds of Theorems 1 and 2.

▶ **Theorem 3.** *The number of solutions for Permutation Pattern Matching can be computed*
- *in time $n^{k/4+o(k)}$,*
- *in time $O(n^{k/2+2})$ and polynomial space, and*
- *in time $O(1.6181^n)$ and polynomial space.*

Note that the FPT algorithm of Guillemot and Marx [33] cannot be adapted for the counting version. In fact, we argue (§ 5) that a running time of the form $n^{O(k)}$ is almost best possible and a significant improvement in running time for the counting problem is unlikely.

▶ **Theorem 4.** *Assuming the exponential-time hypothesis (ETH), there is no algorithm that counts the number of occurrences of π in τ in time $f(k) \cdot n^{o(k/\log k)}$, for any function f.*

Special patterns. It is possible that PPM is easier if the pattern π comes from some restricted family of permutations, e.g. if it avoids some smaller fixed pattern σ. Several such examples have been studied in the literature, and recently Jelínek and Kynčl [37] obtained the following characterization: PPM is polynomial-time solvable for σ-avoiding patterns π, if σ is one of (1), $(1,2)$, $(1,3,2)$, $(2,1,3)$ or their reverses, and NP-complete for all other σ. (All tractable cases are such that π is a *separable* permutation [15, 36, 56, 4].)

In particular, Jelínek and Kynčl show that PPM is NP-complete even if π avoids $(1,2,3)$ or $(3,2,1)$, but polynomial-time solvable for any proper subclass of these families. For $(1,2,3)$-avoiding and $(3,2,1)$-avoiding patterns, it is known however, that PPM can be solved in time $n^{O(\sqrt{k})}$, i.e. faster than the general case (Guillemot and Vialette [34]).

These results motivate the following general and natural question.

▶ **Question.** *What makes a permutation pattern easier to find than others?*

A permutation is *t-monotone*, if it can be obtained by interleaving t monotone sequences. When all t sequences are increasing (resp. decreasing), we call the resulting permutation t-increasing (resp. t-decreasing). It is well-known that t-increasing (resp. t-decreasing) permutations are exactly those that avoid $(t+1,\dots,1)$, resp. $(1,\dots,t+1)$, see e.g. [7].

We prove that if π is 2-monotone, then the running time of the algorithm of Theorem 1 is $n^{O(\sqrt{k})}$. This result follows from bounding the treewidth of the incidence graph of π, by observing that this graph is *almost planar*. For 2-increasing or 2-decreasing patterns we thus match the bound of Guillemot and Vialette by a significantly simpler argument. (In these special cases the incidence graph is, in fact, planar.)

Jordan-permutations are a natural family of geometrically-defined permutations with applications in computational geometry [47]. They were studied by Hoffmann, Mehlhorn, Rosenstiehl, and Tarjan [35], who showed that they can be sorted with a linear number of comparisons (see also [2] for related enumerative results). A Jordan permutation is generated by the intersection-pattern of two simple curves in the plane: label the intersection points between the curves in increasing order along the first curve, and read out the labels along the second curve; the obtained sequence is a Jordan-permutation (Figure 1). As the incidence graph of the pattern π is planar whenever π is a Jordan-permutation, in this case too an $n^{O(\sqrt{k})}$ bound on the running time follows.

▶ **Theorem 5.** *The treewidth of the incidence graph of π is $O(\sqrt{k})$,*
(i) if π is 2-monotone, or (ii) if π is a Jordan-permutation.

We show that both 2-monotone (and even 2-increasing or 2-decreasing) and Jordan-permutations of length $O(k)$ may contain grids of size $\sqrt{k} \times \sqrt{k}$ in their incidence-graphs, both statements of Theorem 5 are therefore tight, via known lower bounds on the treewidth of grids [11].

In light of these results, one may try to obtain further treewidth-bounds for families of patterns, in order to solve PPM in sub-exponential time. In this direction we show a (somewhat surprising) negative result.

▶ **Theorem 6.** *There are 3-increasing permutations of length k whose incidence graph has treewidth $\Omega(k/\log k)$.*

The same bound applies, by symmetry, to 3-decreasing permutations. The result is obtained by embedding the incidence graph of an *arbitrary* permutation of length $O(k/\log k)$ as a *minor* of the incidence graph of a 3-increasing permutation of length k.

Theorems 5 and 6 (proved in § 4) lead to an almost complete characterization of the treewidth of σ-avoiding patterns. By the Erdős-Szekeres theorem [27] every k-permutation

contains a monotone pattern of length $\lceil\sqrt{k}\rceil$. Thus, for all permutations σ of length at least 10, the class of σ-avoiding permutations contains all 3-increasing or all 3-decreasing permutations, hence by Theorem 6 there exist σ-avoiding patterns π with $\mathrm{tw}(\pi) \in \Omega(k/\log k)$. Addressing a few additional small cases by similar arguments (details given in the thesis of the first author), the threshold 10 can be further reduced. We remark that no algorithm is known to solve PPM in time $n^{o(\mathrm{tw}(\pi))}$; see the discussion in [1, 37].

With a weaker bound we obtain a full characterisation that strengthens the dichotomy-result of Jelínek and Kynčl: in the worst case, the only σ-avoiding patterns π for which $\mathrm{tw}(\pi) \in o(\sqrt{k})$ are those for which PPM is known to be polynomial-time solvable.

Further related work. The complexity of the PPM problem has also been studied under the stronger restriction that the text τ is pattern-avoiding. The problem is polynomial-time solvable if τ is monotone [21] or 2-monotone [19, 34, 6, 4, 43], but NP-hard if τ is 3-monotone [37]. A broader characterization is missing.

Only *classical patterns* are considered in this paper; variants in the literature include *vincular*, *bivincular*, *consecutive*, and *mesh* patterns; we refer to [17] for a survey of related computational questions.

Newman et al. [44] study pattern matching in a *property-testing* framework (aiming to distinguish pattern-avoiding sequences from those that contain *many copies* of the pattern). In this setting, the focus is on the *query complexity* of different approaches, and sampling techniques are often used; see also [9, 31].

A different line of work investigates whether standard algorithmic problems on permutations (e.g. sorting, selection) become easier if the input can be assumed to be pattern-avoiding [8, 20].

2 Preliminaries

A length-n permutation σ is a bijective function $\sigma : [n] \to [n]$, alternatively viewed as the sequence $(\sigma(1), \ldots, \sigma(n))$. Given a length-$n$ permutation σ, we denote as $S_\sigma = \{(i, \sigma(i)) \mid 1 \le i \le n\}$ the *set of points* corresponding to permutation σ.

For a point $p \in S_\sigma$ we denote its first entry as $p.x$, and its second entry as $p.y$, referring to these values as the *index*, respectively, the *value* of p. Observe that for every $i \in [n]$, we have $|\{p \in S_\sigma \mid p.x = i\}| = |\{p \in S_\sigma \mid p.y = i\}| = 1$.

We define four neighbors of a point $(x, y) \in S_\sigma$ as follows.

$$\begin{aligned} N^R((x,y)) &= (x+1,\ \sigma(x+1)),\\ N^L((x,y)) &= (x-1,\ \sigma(x-1)),\\ N^U((x,y)) &= (\sigma^{-1}(y+1),\ y+1),\\ N^D((x,y)) &= (\sigma^{-1}(y-1),\ y-1). \end{aligned}$$

The superscripts R, L, U, D are meant to evoke the directions *right*, *left*, *up*, *down*, when plotting S_σ in the plane. Some neighbors of a point may coincide. When some index is out of bounds, we let the offending neighbor be a "virtual point" as follows: $N^R(n, i) = N^U(i, n) = (\infty, \infty)$, and $N^L(1, i) = N^D(i, 1) = (0, 0)$, for all $i \in [n]$. The virtual points are not contained in S_σ, we only define them to simplify some of the statements.

The *incidence graph* of a permutation σ is $G_\sigma = (S_\sigma, E_\sigma)$, where

$$E_\sigma = \{(p, N^\alpha(p)) \mid \alpha \in \{R, L, U, D\} \text{ and } p, N^\alpha(p) \in S_\sigma\}.$$

In words, each point is connected to its (at most) four neighbors: its successor and predecessor by index, and its successor and predecessor by value. It is easy to see that G_σ is a union of two Hamiltonian paths on the same set of vertices and that this is an exact characterization of permutation incidence-graphs. (See Figure 1 for an illustration.)

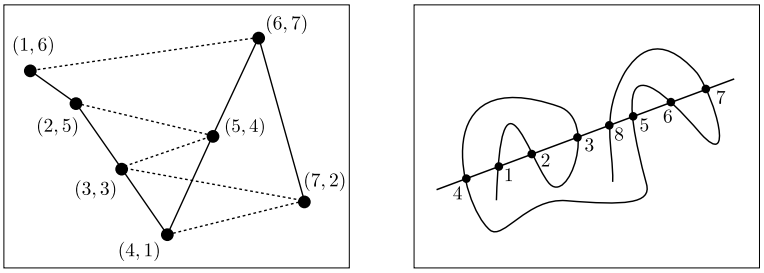

Figure 1 (*left*) Permutation $\pi = (6, 5, 3, 1, 4, 7, 2)$ and its incidence graph G_π. Solid lines indicate neighbors by index, dashed lines indicate neighbors by value (lines may overlap). Indices plotted on x-coordinate, values plotted on y-coordinate. (*right*) Jordan-permutation $(4, 1, 2, 3, 8, 5, 6, 7)$.

Throughout the paper we consider a text permutation $\tau : [n] \to [n]$, and a pattern permutation $\pi : [k] \to [k]$, where $n \geq k$. We give an alternative definition of the Permutation Pattern Matching (PPM) problem in terms of embedding S_π into S_τ.

Consider a function $f : S_\pi \to S_\tau$. We say that f is a *valid embedding* of S_π into S_τ if for all $p \in S_\pi$ the following hold:

$$f(N^L(p)).x \quad < \quad f(p).x \quad < \quad f(N^R(p)).x, \quad \text{and} \tag{1}$$

$$f(N^D(p)).y \quad < \quad f(p).y \quad < \quad f(N^U(p)).y, \tag{2}$$

whenever the corresponding neighbor $N^\alpha(p)$ is also in S_π, i.e. not a virtual point. In words, valid embeddings preserve the relative positions of neighbors in the incidence graph.

▶ **Lemma 7.** *Permutation τ contains permutation π if and only if there exists a valid embedding $f : S_\pi \to S_\tau$.*

For sets $A \subseteq B \subseteq S_\pi$ and functions $g : A \to S_\tau$ and $f : B \to S_\tau$ we say that g is the *restriction* of f to A, denoted $g = f|_A$, if $g(i) = f(i)$ for all $i \in A$. In this case, we also say that f is the *extension* of g to B. Restrictions of valid embeddings will be called *partial embeddings*. We observe that if $f : B \to S_\tau$ is a partial embedding, then it satisfies conditions (1) and (2) with respect to all edges in the induced graph $G_\pi[B]$, i.e. the corresponding inequality holds whenever $p, N^\alpha(p) \in B$.

3 Pattern matching as constraint satisfaction

Readers familiar with the terminology of CSPs should immediately recognize that the definition of valid embedding and Lemma 7 allow us to formulate PPM as a CSP instance with binary constraints. Then known techniques can be applied to solve the problem. A (somewhat different) connection of PPM to CSPs was previously observed by Guillemot and Marx [33]. We first review briefly the CSP problem, referring to [54, 49, 22] for more background.

A *binary CSP* instance is a triplet (V, D, C), where V is a set of variables, D is a set of admissible values (the *domain*), and C is a set of constraints $C = \{c_1, \ldots, c_m\}$, where each constraint c_i is of the form $((x, y), R)$, where $x, y \in V$, and $R \subseteq D^2$ is a binary relation.

A solution of the CSP instance is a function $f : V \to D$ (i.e. an assignment of admissible values to the variables), such that for each constraint $c_i = ((x_i, y_i), R_i)$, the pair of assigned values $(f(x_i), f(y_i))$ is contained in R_i.

The reduction from PPM to CSP is straightforward. Given a PPM instance with text τ and pattern π, of lengths n and k respectively, let $V = \{x_1, \ldots, x_k\}$, and $D = \{1, \ldots, n\}$. The fact that variable x_i takes value j signifies that $\pi(i)$ is matched (embedded) to $\tau(j)$. For the embedding to be valid, by Lemma 7, the relative ordering of entries must be respected, in accordance with conditions (1) and (2). These conditions can readily be described by binary relations for all pairs of variables whose corresponding entries are neighbors in the incidence graph G_π.

More precisely, for $p, N^\alpha(p) \in S_\pi$, for $\alpha \in \{R, L, U, D\}$, we add constraints of the form $((x_i, x_j), R)$, where $i = p.x$, $j = N^\alpha(p).x$ and R contains those pairs $(a, b) \in [n]^2$, for which the relative position of $(a, \tau(a))$ and $(b, \tau(b))$ matches the relative position of p and $N^\alpha(p)$.

The *constraint graph* of the binary CSP instance (also known as *primal graph* or *Gaifman graph*) is a graph whose vertices are the variables V and whose edges connect all pairs of variables that occur together in a constraint. Observe that for instances obtained via our reduction, the constraint graph is exactly the incidence graph G_π. We make use of the following well-known result.

▶ **Lemma 8** ([32, 24]). *A binary CSP instance (V, D, C) can be solved in time $O(|D|^{t+1})$ where t is the treewidth of the constraint graph.*

As discussed in § 2, the incidence graph G_π consists of two Hamiltonian-paths. Accordingly, its vertices have degree at most 4, and the following structural result is applicable.

▶ **Lemma 9** ([28, 29]). *If G is an order-k graph with vertices of degree at most 4, then the pathwidth (and consequently, the treewidth) of G is at most $k/3 + o(k)$. A corresponding tree-(path-)decomposition can be found in polynomial time.*

Algorithms. Our first algorithm amounts to reducing the PPM instance to a binary CSP instance, and using the algorithm of Lemma 8 with a tree-decomposition obtained via Lemma 9. To reach the bound given in Theorem 1, it remains to improve the $k/3$ term in the exponent to $k/4$. We achieve this with a recent technique of Cygan et al. [23], developed in the context of the k-OPT heuristic for TSP.

In our setting, the technique works as follows. We split $[n]$ into $n^{1/4}$ contiguous intervals of equal widths, $n^{3/4}$ each. (For simplicity, we ignore issues of rounding and divisibility.) The intervals induce vertical *strips* in the text τ. For each pattern-index $i \in [k]$ we *guess* the vertical strip of τ into which i is mapped in the sought-for embedding of π into τ. It is sufficient to do this for a subset of the entries in π, namely those that become the *leftmost* in their respective strips in τ. Let $X \subseteq [k]$ be the set of indices of such entries in π.

Guessing X and the strips of τ into which entries of X are mapped increases the running time by a factor of $\sum_{X \subseteq [k]} \binom{n^{1/4}}{|X|} \leq \sum_{X \subseteq [k]} n^{|X|/4}$. Assuming that we guessed correctly, the problem simplifies. First, each pattern-entry can now be embedded into at most $n^{3/4}$ possible locations, hence the domain of each variable will be of size at most $n^{3/4}$. Second, the horizontal constraints that go across strip-boundaries can now be removed as they are implicitly enforced by the distribution of entries into strips (the L-constraint of every X-entry is removed). We have thus reduced the number of edges in the constraint-graph by $|X| - 1$ and can use a stronger upper bound of $(k - |X|)/3 + o(k)$ on the treewidth (see e.g. [28, 23]).

The overall running time becomes

$$\sum_{X \subseteq [k]} n^{\frac{|X|}{4}} \cdot n^{\frac{3}{4} \cdot (\frac{k}{3} - \frac{|X|}{3}) + o(k)} = 2^k \cdot n^{k/4 + o(k)} = n^{k/4 + o(k)}.$$

We remark that our use of this technique is essentially the same as in Cygan et al. [23], but the CSP-formalism makes its application more transparent. We suspect that further classes of CSPs could be handled with a similar approach.

The even-odd method. The algorithm for Theorem 2 can be obtained as follows. Let (Q^E, Q^O) be the partition of S_π into points with even and odd indices. Formally, $Q^E = \{(2k, \pi(2k)) \mid 1 \leq k \leq \lfloor k/2 \rfloor\}$, and $Q^O = \{(2k-1, \pi(2k-1)) \mid 1 \leq k \leq \lceil k/2 \rceil\}$. Construct the CSP instance corresponding to the problem as above. A solution is now found by trying first every possible combination of values for the variables representing Q^E. Clearly, there are $n^{|Q^E|} = n^{\lfloor k/2 \rfloor}$ possible combinations. If the value of a variable x_i is fixed to $a \in [n]$, then x_i is removed from the problem and every neighbor of x_i is restricted by a new unary constraint in an appropriate way, i.e. if there is a constraint $((x_i, x_j), R)$, then x_j should be restricted to values b for which $(a, b) \in R$.

How does the constraint graph look like if we remove every variable (and its incident edges) corresponding to Q^E? It is easy to see that this destroys every constraint corresponding to L-R neighbors and all the remaining binary constraints represent U-D neighbors. As these constraints form a Hamiltonian path, the remaining constraint graph consists of a union of disjoint paths. Such graphs have treewidth 1, hence the resulting CSP instance can be solved efficiently using Lemma 8, resulting in the running time $O(n^{k/2+2})$. A more careful argument improves this bound to $O(n^{k/2+1})$; we defer the details to the full version of the paper.

We can refine the analysis, noting that when we are assigning values $a_2 < a_4 < a_6 < \ldots$ to the variables x_2, x_4, x_6, ... representing Q^E, then we need to consider only increasing sequences where there is a gap of at least one between each successive entry (e.g. $a_4 > a_2 + 1$) to allow a value for the odd-indexed variables. The number of such subsequences is $\binom{n - \lceil k/2 \rceil}{\lfloor k/2 \rfloor}$: consider a sequence with a minimum required gap of one between consecutive entries, then distribute the remaining total gap of $n - k$ among the $\lfloor k/2 \rfloor + 1$ slots. As $\max_k \binom{n-k}{k} = O(1.6181^n)$, see e.g. [52, 51], we obtain an upper bound of this form (independent of k) on the running time of the algorithm.

Counting solutions. The algorithms described above can be made to work for the counting version of the problem. This has to be contrasted with the FPT algorithm of Guillemot and Marx [33], which cannot be adapted for the counting version: a crucial step in that algorithm is to say that if the text is sufficiently complicated, then it contains every pattern of length k, hence we can stop. Indeed, as we show in § 5, we cannot expect an FPT algorithm for the counting problem.

To solve the counting problem, we modify the dynamic programming algorithm behind Lemma 8 in a straightforward way. Even if not stated in exactly the following form, results of this type are implicitly used in the counting literature.

▶ **Lemma 10.** *The number of solutions of a binary CSP instance (V, D, C) can be computed in time $O(|D|^{t+1})$ where t is the treewidth of the constraint graph.*

It is not difficult to see that by replacing the use of Lemma 8 with Lemma 10 in the algorithms of Theorems 1 and 2, the counting algorithms stated in Theorem 3 follow.

4 Special patterns

In this section we prove Theorems 5 and 6. We define a *k-track graph* $G = (V, E)$ to be the union of two Hamiltonian paths H_1 and H_2, where V can be partitioned into *sequences* $S_1, S_2, \ldots, S_k$, the *tracks* of G, such that both H_1 and H_2 visit the vertices of S_i in the given order, for all $i \in [k]$. Observe that k-track graphs are exactly the incidence graphs of permutations that are either k-increasing or k-decreasing.

2-monotone patterns. We prove Theorem 5(i). As a special case, we first look at patterns that are 2-increasing. Let G be a 2-track graph. Arrange the vertices of the two tracks on a line ℓ, the first track in reverse order, followed by the second track in sorted order. Any Hamiltonian path that respects the order of the two tracks can be drawn (without crossings) on one side of ℓ. This means that the two Hamiltonian paths of G can be drawn on different sides of ℓ, and therefore G is planar. See Figure 2 (left) for an example.

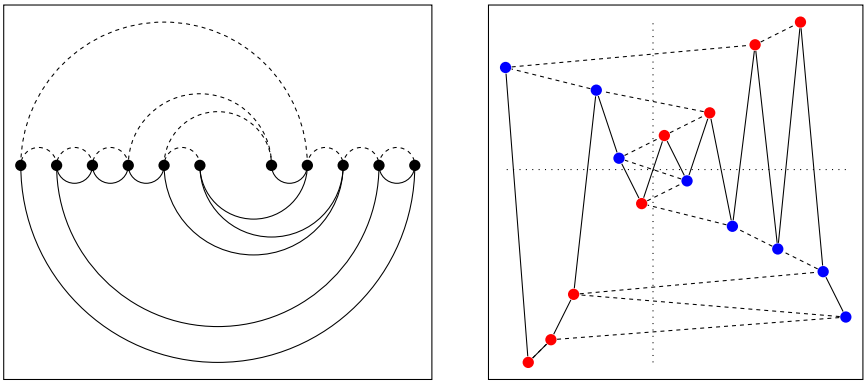

Figure 2 (*left*) A planar drawing of a 2-track graph, with one of the two Hamiltonian paths drawn with dashed arcs. Note that edges contained in both Hamiltonian paths are drawn twice for clarity. (*right*) A drawing of the incidence graph of a 2-monotone permutation. Red and blue dots indicate an increasing (resp. decreasing) subsequence.

The treewidth of a k-vertex planar graph is known to be $O(\sqrt{k})$ [11, 25]. A corresponding path-decomposition can be obtained by a recursive use of planar separators. For the case of a pattern π that consists of an increasing and a decreasing subsequence (i.e. 2-monotone patterns), we show that the straight-line drawing of G_π (with points S_π as vertices) has at most one intersection. An $O(\sqrt{k})$ bound on the treewidth follows via known results [26].

Divide S_π by one horizontal and one vertical line, such that each of the resulting four sectors contains a monotone sequence. More precisely, the top left and bottom right sectors contain decreasing subsequences, and the other two sectors contain increasing subsequences. Let $e = \{u, v\}$ be an edge of the horizontal Hamiltonian path such that $u.x = v.x - 1$, and let $f = \{s, t\}$ be an edge that intersects e, such that $s.x < t.x$. Edge f must come from the vertical Hamiltonian path, i.e. $|s.y - t.y| = 1$. As u and v are horizontal neighbors, $s.x < u.x < v.x < t.x$ holds. Assume that $u.y < v.y$ (otherwise flip G_π vertically before the argument, without affecting the graph structure). We claim that $u.y < t.y < s.y < v.y$ must hold, as otherwise π contains the pattern $(2, 1, 4, 3)$ and cannot decompose into an increasing and a decreasing subsequence.

Thus (s, u, v, t) must form the pattern $(3, 1, 4, 2)$, and therefore s and t belong to the decreasing and u and v to the increasing subsequence. It is easy to see now that s, u, v, t must be in pairwise distinct sectors, and u (s, t, v) is the unique rightmost (bottommost,

topmost, leftmost) point of the bottom left (top left, top right, bottom right) sector, and due to the monotonicity of all four sectors no more intersections can happen; see Figure 2 (right). This concludes the proof of Theorem 5(i).

We show that the previous result is tight, by constructing a 2-track graph $G = (V, E)$ with $n = 2k^2$ vertices, for some even k, that contains a $k \times 2k$ grid graph.

Let $x_1, x_2, \ldots, x_{k^2}$ and $y_1, y_2, \ldots, y_{k^2}$ be the two tracks of G. We obtain G as the union of the following two Hamiltonian paths:

$$x_1, y_1, y_2, x_2, x_3, y_3, y_4, x_4, \ldots, x_{k^2-1}, y_{k^2-1}, y_{k^2}, x_{k^2}; \quad \text{and}$$
$$x_1, x_2, \ldots, x_k,$$
$$y_1, x_{k+1}, x_{k+2}, y_2, y_3, x_{k+3}, x_{k+4}, y_4, \ldots y_{k^2-k-1} x_{k^2-1}, x_{k^2}, y_{k^2-k},$$
$$y_{k^2-k+1}, y_{k^2-k+2}, \ldots, y_{k^2}.$$

The two Hamiltonian paths respect the order of the two tracks.

We now relabel the vertices to show the contained grid. For $i \in [k]$, $j \in [2k]$, let $z_{i,j} = x_{\lfloor j/2 \rfloor k+i}$ if j is odd, and $z_{i,j} = y_{(j/2-1)k+i}$ if j is even. It is easy to see that $z_{i,j}$ is adjacent to $z_{i+1,j}$ and $z_{i,j+1}$ for $i \in [k-1]$ and $j \in [2k-1]$. For an illustration of the obtained permutation and the contained grid graph, see Figure 3.

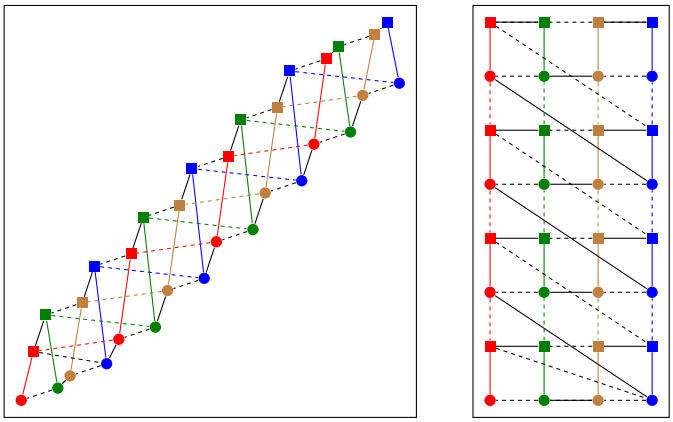

Figure 3 A 2-increasing permutation of length 32 whose incidence graph contains a 4×8 grid. Vertex shape indicates the track, colors are for emphasis of the grid structure.

The case of Jordan patterns (Theorem 5(ii)) is immediate, as the incidence graph of Jordan permutations is by definition planar. We defer the details to the full version of the paper.

3-monotone patterns. We now prove Theorem 6. Due to space constraints we omit some figures that illustrate the proof; these can be found in the full version of the paper. We start with some definitions and observations. For a set Π of length-n permutations, define the graph $G_H(\Pi) = ([n], E)$, where E is the union of the Hamiltonian paths corresponding to all $\pi \in \Pi$. For an arbitrary length-n permutation π, the graph G_π is isomorphic to $G_H(\{\mathrm{id}_n, \pi\})$, where id_n is the length-n identity permutation.

Permutation π' is a *split* of a permutation π if π' arises from π by moving a subsequence of π to the front. For example, $(1, 3, 5, 2, 4)$ is a split of id_5, obtained by moving $(1, 3, 5)$ to the front. We call a permutation *split permutation* if it is a split of the identity permutation. Observe that for a length-n split permutation $\sigma \neq \mathrm{id}_n$, there is a unique integer $p(\sigma) \in [n]$

such that both $\sigma(1), \sigma(2), \ldots, \sigma(p(\sigma))$ and $\sigma(p(\sigma)+1), \sigma(p(\sigma)+2), \ldots, \sigma(n)$ are increasing. Furthermore, σ^{-1} is a merge of the two subsequences $1, 2, \ldots, p(\sigma)$ and $p(\sigma)+1, p(\sigma)+2, \ldots, n$.

If π' is a split of π, then $\pi' = \pi \circ \sigma$ for some split permutation σ. Ahal and Rabinovich [1] mention that every n-permutation can be obtained from id_n by at most $\lceil \log n \rceil$ splits. Let π be an *arbitrary* n-permutation and consider the sequence $\mathrm{id}_n = \pi_1, \ldots, \pi_m = \pi$, where for each $i \in [m-1]$ we have $\pi_{i+1} = \pi_i \circ \sigma_i$ for some split permutation $\sigma_i \neq \mathrm{id}_n$, and $m \leq \lceil \log n \rceil$. Let $\Pi = \{\pi_1, \ldots, \pi_m\}$.

To prove Theorem 6, we show that the graph $G_H(\Pi)$ can be embedded (as a minor) in the incidence graph G of some permutation of length at most $2mn$. We further show that G is a 3-track graph (and thus, its underlying permutation can be assumed 3-increasing). The lower bound on the treewidth of G then follows by (i) choosing π to be a permutation whose incidence graph has treewidth $\Omega(n)$, (ii) the fact that G_π is a subgraph of $G_H(\Pi)$, and thus, a minor of G, and (iii) the observation that the treewidth of a graph is not less than the treewidth of its minor.

We first define the vertex sets corresponding to the three tracks of G. Let

$$V_x = \{x_{i,j} \mid i \in [m], j \in [n]\},$$
$$V_y = \{y_{i,j} \mid i \in [m-1], j \in [p(\sigma_i)]\}, \quad \text{and}$$
$$V_z = \{z_{i,j} \mid i \in [m-1], j \in [n] \setminus [p(\sigma_i)]\}.$$

Let $V = V_x \cup V_y \cup V_z$ be the vertex set of G, and observe that $|V| = mn + (m-1)n \leq 2mn$.

To later show that G is a 3-track graph, we fix a total order $\prec$ on each track, namely, the lexicographic order of the vertex-indices, i.e. $x_{i,j} \prec x_{i',j'}$ if and only if $i < i'$ or $(i = i') \wedge (j < j')$, and analogously for V_y and V_z. Before proceeding, we define the following functions:

$$s_x \colon V_x \setminus \{x_{m,n}\} \to V_x \setminus \{x_{1,1}\},$$
$$s_x(x_{i,j}) = \begin{cases} x_{i,j+1}, & \text{if } j < n, \\ x_{i+1,1}, & \text{if } j = n. \end{cases}$$
$$s_c \colon V \setminus \{x_{m,j} \mid j \in [n]\} \to V \setminus \{x_{1,j} \mid j \in [n]\},$$
$$s_c(x_{i,j}) = \begin{cases} y_{i,\sigma_i^{-1}(j)}, & \text{if } \sigma_i^{-1}(j) \leq p(\sigma_i), \\ z_{i,\sigma_i^{-1}(j)}, & \text{if } \sigma_i^{-1}(j) > p(\sigma_i). \end{cases}$$
$$s_c(y_{i,j}) = x_{i+1,j},$$
$$s_c(z_{i,j}) = x_{i+1,j}.$$

Note that s_x is just the successor with respect to the total order $\prec$ on V_x, and that s_c is a bijection.

Now we define the two Hamiltonian paths whose union is G. The first path P_1 goes as follows: start at $x_{1,1}$, then, from every $x_{i,j}$ with $i < m$, go to $s_c(x_{i,j})$, and then to $s_x(x_{i,j})$. For $x_{m,j}$ with $j < n$, go directly to $s_x(x_{m,j})$. Path P_1 contains all vertices of V_x in the correct order. The same holds for V_y and V_z, by the definition of s_c.

The second path P_2 also starts at $x_{1,1}$, but first goes along V_x until it reaches $x_{2,1}$, i.e. the first part of P_2 is $x_{1,1}, x_{1,2}, \ldots, x_{1,n}, x_{2,1}$. Then, from every $x_{i,j}$ with $i \geq 2$, it first moves to $s_c^{-1}(x_{i,j})$ and then to $s_x(x_{i,j})$. Again, P_2 contains all vertices of V_x in the correct order. As $s_c^{-1}(x_{i,j})$ is either $y_{i-1,j}$ or $z_{i-1,j}$, this is also true for V_y and V_z.

To obtain $G_H(\Pi) = ([n], E)$, color the vertices of the graph with n colors, where color k induces a path C_k of length m in G. We then prove that for each $\{k_1, k_2\} \in E$, the graph G

contains adjacent vertices of the colors k_1 and k_2. Then, by contracting C_k for $k \in [n]$, we obtain a supergraph of $G_H(\Pi)$.

For $k \in [n]$, define the path $C_k = (x_{1,k}, s_c(x_{1,k}), s_c^2(x_{1,k}), \ldots, s_c^{2m-2}(x_{1,k}))$. As s_c is a bijection, these paths are disjoint. Note that for each $x_{i,j} \in V_x \setminus \{x_{m,n}\}$,

$$s_c^2(x_{i,j}) = x_{i+1,\sigma_i^{-1}(j)}.$$

We claim that the color of $x_{i,j}$ is $\pi_i(j)$. This is because:

$$\begin{aligned} s_c^{2i-2}(x_{1,\pi_i(j)}) &= s_c^{2i-2}(x_{1,\sigma_1\sigma_2\ldots\sigma_{i-1}(j)}) \\ &= s_c^{2i-4}(x_{2,\sigma_1^{-1}\sigma_1\sigma_2\ldots\sigma_i(j)}) = s_c^{2i-4}(x_{2,\sigma_2\sigma_3\ldots\sigma_{i-1}(j)}) \\ &= \ldots = s_c^{2i-2\ell}(x_{\ell,\sigma_\ell\sigma_{\ell+1}\ldots\sigma_{i-1}(j)}) \\ &= \ldots = x_{i,j}. \end{aligned}$$

Now let k_1 and k_2 be adjacent in $G_H(\Pi)$. Then, there exist i, j such that $\pi_i(j) = k_1$ and $\pi_i(j+1) = k_2$ and, as discussed above, $x_{i,j} \in C_{k_1}$ and $x_{i,j+1} \in C_{k_2}$. By definition $x_{i,j} \in C_{k_1}$ implies $s_c^{-1}(x_{i,j}) \in C_{k_1}$. Finally, P_2 has an edge from $s_c^{-1}(x_{i,j})$ to $s_x(x_{i,j}) = x_{i,j+1}$. This concludes the proof.

The construction can be extended to embed the union of k arbitrary Hamiltonian paths on n vertices as a minor of a 3-track graph with $O(kn \log n)$ vertices. As every order-n graph of maximum degree d is edge-colorable with $d+1$ colors (by Vizing's theorem), such graphs are in the union of at most $d+1$ Hamiltonian paths, can thus be embedded in 3-track graphs of order $O(dn \log n)$.

5 Hardness result

In this section we prove Theorem 4. The hardness proof proceeds in two steps. First, we reduce the *partitioned subgraph isomorphism* (PSI) problem to the *partitioned permutation pattern matching* (PPPM) problem. Then, we reduce from the more difficult, counting variant of PPPM to the regular counting PPM (the subject of Theorem 4), using a (by now standard) technique based on inclusion-exclusion.

PSI to PPPM. The input to the PSI problem (introduced in [42]) consists of a graph G, a graph H, and a coloring ϕ of $V(G)$ with colors $V(H)$. The task is to decide whether there is a mapping $g : V(H) \to V(G)$ such that $\{u, v\} \in E(H)$ if and only if $\{g(u), g(v)\} \in E(G)$, and $\phi(g(u)) = u$ for all $u \in V(H)$. In words, we look for a subgraph of G that is isomorphic to H, with the restriction that each vertex of H can only correspond to a vertex of G from a prescribed set, moreover, these sets are disjoint.

Let n denote the number of vertices of G, and let k denote the number of *edges* of H. It is known [42, Corr. 6.3], that PSI cannot be solved in time $f(k) \cdot n^{o(k/\log k)}$, unless ETH fails, moreover, this holds even if $|E(H)| = |V(H)|$ (see e.g. [16]).

The input to the PPPM problem (introduced in [33]) consists of permutations τ and π of lengths n and k respectively, and a coloring $\phi : [n] \to [k]$ of the entries of τ. The task is to decide whether there is an embedding $g : [k] \to [n]$ of π into τ in the sense of the standard PPM problem, with the additional restriction that $\phi(g(i)) = i$, for all $i \in [k]$.

Guillemot and Marx show [33, Thm. 6.1], through a reduction from *partitioned clique*, that PPPM is $W[1]$-hard. Due to the density of a clique, the same reduction would, at best, yield a lower bound with exponent $\sqrt{k}$. We strengthen (and somewhat simplify) this reduction, to show that PPPM is at least hard as PSI, obtaining the following.

▶ **Lemma 11.** *PPPM cannot be solved in time $f(k) \cdot n^{o(k/\log k)}$, unless ETH fails.*

#PPPM to #PPM. The counting variant of PPPM (denoted #PPPM) is clearly at least as hard as PPPM. We now show that the counting variant of PPM (denoted #PPM) is at least as hard as #PPPM, thereby proving Theorem 4.

We use oracle-calls to #PPM for all subsets $X \subseteq [k]$, to count the number of embeddings of π into τ using entries of τ with colors from the set X, but ignoring colors for the purpose of the embedding. (We can achieve this by deleting the entries of τ with color in $[k] \setminus X$ before each oracle-call.) Then, using the inclusion-exclusion formula, we obtain the number C of embeddings that use *all* colors in $[k]$ as follows:

$$C = \sum_{X \subseteq [k]} (-1)^{k-|X|} C_X,$$

where C_X denotes the number of embeddings that use colors from the set X (obtained by oracle calls). Since π is of length k, the quantity C counts exactly the number of embeddings that use each color once.

It remains to show that embeddings that use every color in $[k]$ are such that π_i is matched to an entry of τ of color i, for all $i \in [k]$, i.e. the colors are not permuted. This is indeed the case for the hard instance constructed in the proof of Lemma 11. Towards this claim (referring to the details of the reduction in the full version of the paper) observe that all points that are unique in their respective pattern-cell (i, j) can only be matched to a point in the corresponding text-cell (i, j), which is of the correct color. In each diagonal cell (i, i), for $i > 0$, there is a matched point, and the pattern has two bracketing points in decreasing order in pattern-cell $(i, 0)$, and two bracketing points in increasing order in pattern-cell $(0, i)$. By construction, the only two points in the correct order are the nearest bracketing points in text-cell $(i, 0)$, resp. $(0, i)$, which are indeed of the correct color.

The number of oracle calls and additional overhead amounts to a factor 2^k in the running time, absorbed in the quantity $f(k) \cdot n^{o(k/\log k)}$. This concludes the proof.

References

1. Shlomo Ahal and Yuri Rabinovich. On Complexity of the Subpattern Problem. *SIAM J. Discrete Math.*, 22(2):629–649, 2008. doi:10.1137/S0895480104444776.
2. M.H. Albert and M.S. Paterson. Bounds for the growth rate of meander numbers. *Journal of Combinatorial Theory, Series A*, 112(2):250–262, 2005. doi:10.1016/j.jcta.2005.02.006.
3. Michael Albert and Mireille Bousquet-Mélou. Permutations sortable by two stacks in parallel and quarter plane walks. *Eur. J. Comb.*, 43:131–164, 2015. doi:10.1016/j.ejc.2014.08.024.
4. Michael H. Albert, Robert E. L. Aldred, Mike D. Atkinson, and Derek A. Holton. Algorithms for Pattern Involvement in Permutations. In *Proceedings of the 12th International Symposium on Algorithms and Computation*, ISAAC '01, pages 355–366, London, UK, 2001. Springer-Verlag. URL: http://dl.acm.org/citation.cfm?id=646344.689586.
5. Michael H. Albert, Cheyne Homberger, Jay Pantone, Nathaniel Shar, and Vincent Vatter. Generating permutations with restricted containers. *J. Comb. Theory, Ser. A*, 157:205–232, 2018. doi:10.1016/j.jcta.2018.02.006.
6. Michael H. Albert, Marie-Louise Lackner, Martin Lackner, and Vincent Vatter. The Complexity of Pattern Matching for 321-Avoiding and Skew-Merged Permutations. *Discrete Mathematics & Theoretical Computer Science*, 18(2), 2016. URL: http://dmtcs.episciences.org/2607.
7. David Aldous and Persi Diaconis. Longest Increasing Subsequences: From Patience Sorting to the Baik-Deift-Johansson Theorem. *Bull. Amer. Math. Soc*, 36:413–432, 1999.

8 David Arthur. Fast Sorting and Pattern-avoiding Permutations. In *Proceedings of the Fourth Workshop on Analytic Algorithmics and Combinatorics, ANALCO 2007*, pages 169–174, 2007. doi:10.1137/1.9781611972979.1.

9 Omri Ben-Eliezer and Clément L. Canonne. Improved Bounds for Testing Forbidden Order Patterns. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 2093–2112, 2018. doi:10.1137/1.9781611975031.137.

10 Donald J. Berndt and James Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, pages 359–370. AAAI Press, 1994. URL: http://dl.acm.org/citation.cfm?id=3000850.3000887.

11 Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.

12 Miklós Bóna. A survey of stack-sorting disciplines. *The Electronic Journal of Combinatorics*, 9(2):1, 2003.

13 Miklós Bóna. *Combinatorics of Permutations*. CRC Press, Inc., Boca Raton, FL, USA, 2004.

14 Miklós Bóna. *A Walk Through Combinatorics: An Introduction to Enumeration and Graph Theory*. World Scientific, 2011. URL: https://books.google.de/books?id=TzJ2L9ZmlQUC.

15 Prosenjit Bose, Jonathan F. Buss, and Anna Lubiw. Pattern Matching for Permutations. *Inf. Process. Lett.*, 65(5):277–283, 1998. doi:10.1016/S0020-0190(97)00209-3.

16 Karl Bringmann, László Kozma, Shay Moran, and N. S. Narayanaswamy. Hitting Set for Hypergraphs of Low VC-dimension. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 23:1–23:18, 2016. doi:10.4230/LIPIcs.ESA.2016.23.

17 Marie-Louise Bruner and Martin Lackner. The computational landscape of permutation patterns. *Pure Mathematics and Applications*, 24(2):83–101, 2013.

18 Marie-Louise Bruner and Martin Lackner. A Fast Algorithm for Permutation Pattern Matching Based on Alternating Runs. *Algorithmica*, 75(1):84–117, 2016. doi:10.1007/s00453-015-0013-y.

19 Laurent Bulteau, Romeo Rizzi, and Stéphane Vialette. Pattern Matching for k-Track Permutations. In Costas Iliopoulos, Hon Wai Leong, and Wing-Kin Sung, editors, *Combinatorial Algorithms*, pages 102–114, Cham, 2018. Springer International Publishing.

20 Parinya Chalermsook, Mayank Goswami, László Kozma, Kurt Mehlhorn, and Thatchaphol Saranurak. Pattern-Avoiding Access in Binary Search Trees. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015*, pages 410–423, 2015. doi:10.1109/FOCS.2015.32.

21 Maw-Shang Chang and Fu-Hsing Wang. Efficient algorithms for the maximum weight clique and maximum weight independent set problems on permutation graphs. *Information Processing Letters*, 43(6):293–295, 1992. doi:10.1016/0020-0190(92)90114-B.

22 Hubie Chen. A rendezvous of logic, complexity, and algebra. *ACM Comput. Surv.*, 42(1):2:1–2:32, 2009. doi:10.1145/1592451.1592453.

23 Marek Cygan, Lukasz Kowalik, and Arkadiusz Socala. Improving TSP Tours Using Dynamic Programming over Tree Decompositions. In *25th Annual European Symposium on Algorithms, ESA 2017*, pages 30:1–30:14, 2017. doi:10.4230/LIPIcs.ESA.2017.30.

24 Rina Dechter and Judea Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38(3):353–366, 1989. doi:10.1016/0004-3702(89)90037-4.

25 Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999. doi:10.1007/978-1-4612-0515-9.

26 Vida Dujmovic, David Eppstein, and David R. Wood. Structure of Graphs with Locally Restricted Crossings. *SIAM J. Discrete Math.*, 31(2):805–824, 2017. doi:10.1137/16M1062879.

27 Pál Erdős and G. Szekeres. A combinatorial problem in geometry. *Compositio Mathematica*, 2:463–470, 1935. URL: http://www.numdam.org/item/CM_1935__2__463_0.

28 Fedor V. Fomin, Serge Gaspers, Saket Saurabh, and Alexey A. Stepanov. On Two Techniques of Combining Branching and Treewidth. *Algorithmica*, 54(2):181–207, 2009. doi:10.1007/s00453-007-9133-3.

29 Fedor V. Fomin and Kjartan Høie. Pathwidth of cubic graphs and exact algorithms. *Information Processing Letters*, 97(5):191–196, 2006. doi:10.1016/j.ipl.2005.10.012.

30 Jacob Fox. Stanley-Wilf limits are typically exponential. *CoRR*, abs/1310.8378, 2013. arXiv:1310.8378.

31 Jacob Fox and Fan Wei. Fast property testing and metrics for permutations. *Combinatorics, Probability and Computing*, pages 1–41, 2018.

32 Eugene C. Freuder. Complexity of K-tree Structured Constraint Satisfaction Problems. In *Proceedings of the Eighth National Conference on Artificial Intelligence - Volume 1*, AAAI'90, pages 4–9. AAAI Press, 1990. URL: http://dl.acm.org/citation.cfm?id=1865499.1865500.

33 Sylvain Guillemot and Dániel Marx. Finding small patterns in permutations in linear time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014*, pages 82–101, 2014. doi:10.1137/1.9781611973402.7.

34 Sylvain Guillemot and Stéphane Vialette. Pattern Matching for 321-Avoiding Permutations. In Yingfei Dong, Ding-Zhu Du, and Oscar Ibarra, editors, *Algorithms and Computation*, pages 1064–1073. Springer Berlin Heidelberg, 2009.

35 Kurt Hoffmann, Kurt Mehlhorn, Pierre Rosenstiehl, and Robert E. Tarjan. Sorting Jordan sequences in linear time using level-linked search trees. *Information and Control*, 68(1-3):170–184, 1986.

36 Louis Ibarra. Finding pattern matchings for permutations. *Information Processing Letters*, 61(6):293–295, 1997. doi:10.1016/S0020-0190(97)00029-X.

37 Vít Jelínek and Jan Kyncl. Hardness of Permutation Pattern Matching. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 378–396, 2017. doi:10.1137/1.9781611974782.24.

38 Eamonn J. Keogh, Stefano Lonardi, and Bill Yuan-chi Chiu. Finding surprising patterns in a time series database in linear time and space. In *Proceedings of the Eighth ACM SIGKDD 2002 International Conference on Knowledge Discovery and Data Mining*, pages 550–556, 2002. doi:10.1145/775047.775128.

39 Sergey Kitaev. *Patterns in Permutations and Words*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2011. doi:10.1007/978-3-642-17333-2.

40 Donald E. Knuth. *The Art of Computer Programming, Volume I: Fundamental Algorithms*. Addison-Wesley, 1968.

41 Adam Marcus and Gábor Tardos. Excluded permutation matrices and the Stanley-Wilf conjecture. *J. Comb. Theory, Ser. A*, 107(1):153–160, 2004. doi:10.1016/j.jcta.2004.04.002.

42 Dániel Marx. Can You Beat Treewidth? *Theory of Computing*, 6(1):85–112, 2010. doi:10.4086/toc.2010.v006a005.

43 Both Neou, Romeo Rizzi, and Stéphane Vialette. Permutation Pattern matching in (213, 231)-avoiding permutations. *Discrete Mathematics & Theoretical Computer Science*, Vol. 18 no. 2, Permutation Patterns 2015, 2017. URL: https://dmtcs.episciences.org/3199.

44 Ilan Newman, Yuri Rabinovich, Deepak Rajendraprasad, and Christian Sohler. Testing for Forbidden Order Patterns in an Array. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 1582–1597, 2017. doi:10.1137/1.9781611974782.104.

45 Pranav Patel, Eamonn Keogh, Jessica Lin, and Stefano Lonardi. Mining Motifs in Massive Time Series Databases. In *In Proceedings of IEEE International Conference on Data Mining ICDM'02*, pages 370–377, 2002.

46 Vaughan R. Pratt. Computing Permutations with Double-ended Queues, Parallel Stacks and Parallel Queues. In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, STOC '73, pages 268–277. ACM, 1973. doi:10.1145/800125.804058.

47 Pierre Rosenstiehl. Planar permutations defined by two intersecting Jordan curves. In *Graph Theory and Combinatorics*, pages 259–271. London, Academic Press, 1984. URL: https://hal.archives-ouvertes.fr/hal-00259765.

48 Pierre Rosenstiehl and Robert E. Tarjan. Gauss codes, planar hamiltonian graphs, and stack-sortable permutations. *Journal of Algorithms*, 5(3):375–390, 1984. doi:10.1016/0196-6774(84)90018-X.

49 Raimund Seidel. A New Method for Solving Constraint Satisfaction Problems. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence, IJCAI 1981*, pages 338–342, 1981. URL: http://ijcai.org/Proceedings/81-1/Papers/062.pdf.

50 Rodica Simion and Frank W. Schmidt. Restricted Permutations. *European Journal of Combinatorics*, 6(4):383–406, 1985. doi:10.1016/S0195-6698(85)80052-4.

51 N. J. A. Sloane. The Encyclopedia of Integer Sequences, http://oeis.org. Sequence A073028.

52 S. M. Tanny and M. Zuker. On a unimodal sequence of binomial coefficients. *Discrete Mathematics*, 9(1):79–89, 1974. doi:10.1016/0012-365X(74)90073-9.

53 Robert E. Tarjan. Sorting Using Networks of Queues and Stacks. *J. ACM*, 19(2):341–346, April 1972. doi:10.1145/321694.321704.

54 Edward P. K. Tsang. *Foundations of constraint satisfaction*. Computation in cognitive science. Academic Press, 1993.

55 Vincent Vatter. Permutation Classes. In Miklós Bóna, editor, *Handbook of Enumerative Combinatorics*, chapter 12. Chapman and Hall/CRC, New York, 2015. Preprint at https://arxiv.org/abs/1409.5159.

56 V. Yugandhar and Sanjeev Saxena. Parallel algorithms for separable permutations. *Discrete Applied Mathematics*, 146(3):343–364, 2005. doi:10.1016/j.dam.2004.10.004.