Algorithmica
https://doi.org/10.1007/s00453-021-00812-z

# Finding and Counting Permutations via CSPs

**Benjamin Aram Berendsohn[1] · László Kozma[1] · Dániel Marx[2]**

Received: 30 December 2019 / Accepted: 3 February 2021
© The Author(s) 2021

**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* (4321-avoiding) and 3-*decreasing* (1234-avoiding) permutations can, in some sense, *embed* arbitrary permutations of almost linear length, which indicates that a sub-exponential running time is unlikely with the current techniques, even for patterns from these restricted classes.

B.A. Berendsohn and L. Kozma supported by DFG grant KO 6140/1-1. D. Marx supported by the European Research Council (ERC) Consolidator Grant No. 725978 SYSTEMATICGRAPH.

✉ László Kozma
laszlo.kozma@fu-berlin.de

Benjamin Aram Berendsohn
beab@zedat.fu-berlin.de

Dániel Marx
dmarx@mpi-inf.mpg.de

[1] Institut für Informatik, Freie Universität Berlin, Berlin, Germany

[2] Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Published online: 01 March 2021

## 1 Introduction

Let $[n] = \{1, \ldots, n\}$. Given two permutations $\tau : [n] \to [n]$, and $\pi : [k] \to [k]$, we say that $\tau$ *contains* $\pi$, if there are indices $1 \le i_1 < \cdots < i_k \le n$ such that $\tau(i_j) < \tau(i_\ell)$ if and only if $\pi(j) < \pi(\ell)$, for all $1 \le j, \ell \le k$. In other words, $\tau$ contains $\pi$, 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 $\tau$ *avoids* $\pi$. 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, $\tau$ avoids (3, 1, 2).

Knuth showed in 1968 [41, § 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 [3, 5, 13, 50, 52, 57], but permutation pattern avoidance has also taken on a life of its own (especially after the influential work of Simion and Schmidt [54]), becoming an important subfield of combinatorics. For more background on permutation patterns and pattern avoidance we refer to the extensive survey [59] and relevant textbooks [14, 15, 40].

Perhaps the most important enumerative result related to permutation patterns is the theorem of Marcus and Tardos [44] from 2004, implying that the number of length-$n$ permutations that avoid a fixed pattern $\pi$ 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 $\tau$ ("text") and a length-$k$ permutation $\pi$ ("pattern"), decide whether $\tau$ contains $\pi$.

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 [40, § 2.4] and time-series analysis [11, 39, 49]. Unfortunately PPM is, in general, NP-complete, as shown by Bose, Buss, and Lubiw [16] 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 [34] 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 [31] 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 $\tau$, checking whether any of them has the same ordering as $\pi$. 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 $\pi$ one-by-one into the text $\tau$, 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 $\pi$, 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 $\pi$.

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 $\pi$. 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 [53, 58, § 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* [25, 33]. In our reduction, the domain size is the length $n$ of the text $\tau$, and the constraint graph is the incidence graph of the pattern $\pi$; 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 [29], we prove that the treewidth of the incidence graph of $\pi$ 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 [24] 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 [19]; 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 [19] 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 [34, § 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 $\pi$ are matched to entries of the text $\tau$, then verifying whether the remaining *odd-index* entries of $\pi$ can be correctly matched takes only a linear-time sweep through both $\pi$ and $\tau$. 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 $\pi$ in $\tau$, i.e. finding the number of subsequences of $\tau$ that have the same ordering as $\pi$. 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 [34] 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* $\pi$ *in* $\tau$ *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 $\pi$ comes from some restricted family of permutations, e.g. if it avoids some smaller fixed pattern $\sigma$. Several such examples have been studied in the literature, and recently Jelínek and Kynčl [38] obtained the following characterization: PPM is polynomial-time solvable for $\sigma$-avoiding patterns $\pi$, if $\sigma$ is one of (1), (1, 2), (1, 3, 2), (2, 1, 3) or their reverses, and NP-complete for all other $\sigma$. (All tractable cases are such that $\pi$ is a *separable* permutation [4, 16, 37, 60].)

In particular, Jelínek and Kynčl show that PPM is NP-complete even if $\pi$ avoids (1, 2, 3) or (3, 2, 1), but polynomial-time solvable for any proper subclass of either 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 [35]).

These results motivate the following general and natural 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 $\pi$ 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 $\pi$, 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 [51]. They were studied by Hoffmann, Mehlhorn, Rosenstiehl, and Tarjan [36], 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 (Fig. 1). As the incidence graph of the pattern $\pi$ is planar whenever $\pi$ 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 $\pi$ is $O(\sqrt{k})$,*

*(i) if $\pi$ is 2-monotone, or (ii) if $\pi$ 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 [12].

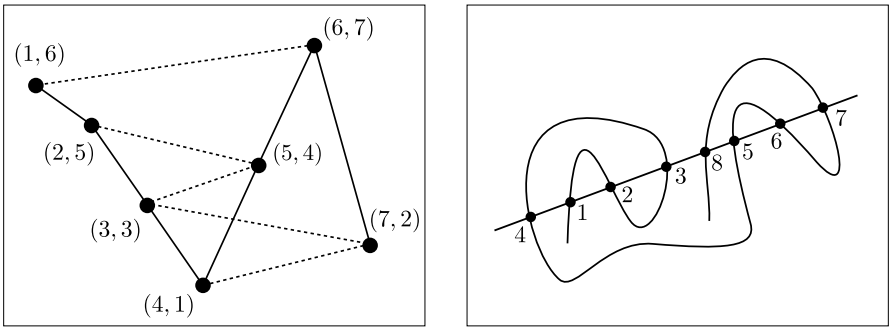


**Fig. 1** (*left*) Permutation $\pi = (6, 5, 3, 1, 4, 7, 2)$ and its incidence graph $G_\pi$. Solid lines indicate neighbors by index, dashed lines indicate neighbors by value (lines may overlap). Indices plotted on $x$-coordinate, values plotted on $y$-coordinate. (*right*) Jordan-permutation (4, 1,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

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 $\sigma$-avoiding patterns. By the Erdős-Szekeres theorem [28] every $k$-permutation contains a monotone pattern of length $\lceil\sqrt{k}\rceil$. Thus, for all permutations $\sigma$ of length at least 10, the class of $\sigma$-avoiding permutations contains all 3-increasing or all 3-decreasing permutations, hence by Theorem 6 there exist $\sigma$ -avoiding patterns $\pi$ 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 [10]), 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, 38].

With a weaker bound we obtain a full characterisation that strengthens the dichotomy-result of Jelínek and Kynčl: in the worst case, the only $\sigma$-avoiding patterns $\pi$ 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 $\tau$ is pattern-avoiding. The problem is polynomial-time solvable if $\tau$ is monotone [22] or 2-monotone [4, 6, 20, 35, 46], but NP-hard if $\tau$ is 3-monotone [38]. 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 [18] for a survey of related computational questions.

Newman et al. [47] 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, 32].

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, 21].

## 2 Preliminaries

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

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 $S_\sigma$ 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 $S_\sigma$, we only define them to simplify some of the statements.

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

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

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 $G_\sigma$ 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 Fig. 1 for an illustration.)

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 $S_\pi$ into $S_\tau$.

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

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

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

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

**Lemma 1** *Permutation $\tau$ contains permutation $\pi$ if and only if there exists a valid embedding $f : S_\pi \to S_\tau$.*

***Proof*** Suppose $\tau$ contains $\pi$, and let $(\tau(i_1), \ldots, \tau(i_k))$ be the subsequence witnessing this fact. Let $p_j$ denote the point $(j, \pi(j))$, and set $f(p_j) = (i_j, \tau(i_j))$ for all $j \in [k]$. Observe that $f(N^L(p_j)).x = i_{j-1}$, and $f(N^R(p_j)).x = i_{j+1}$, condition (1) thus holds since $i_{j-1} < i_j < i_{j+1}$.

Let $\pi(j') = N^D(p_j).y$, and $\pi(j'') = N^U(p_j).y$. By definition, $\pi(j') < \pi(j) < \pi(j'')$. Condition (2) now becomes $\tau(i_{j'}) < \tau(i_j) < \tau(i_{j''})$, which holds since $\tau$ contains $\pi$.

In the other direction, let $f : S_\pi \to S_\tau$ be a valid embedding. Define $i_j = f(p_j).x$, for all $j \in [k]$. Since $f(N^L(p_j)).x < f(p_j).x < f(N^R(p_j)).x$ for all $j$, we have $i_1 < \cdots < i_k$.

Let $j', j'' \in [k]$, such that $j' < j''$. Suppose $\pi(j') < \pi(j'')$ (the other case is symmetric). Then $p_{j''} = N^U(\ldots N^U(p_{j'})\ldots)$, where the $N^U(\cdot)$ operator is applied $\pi(j'') - \pi(j')$ times. By applying (2) the same number of times we get $f(p_{j'}).y < f(N^U(\ldots N^U(p_{j'})\ldots)).y$. Since $f(p_{j'}).y = \tau(i_{j'})$ and $f(N^U(\ldots N^U(p_{j'})\ldots)).y = f(p_{j''}).y = \tau(i_{j''})$, we get that $\tau(i_{j'}) < \tau(i_{j''})$, as needed. □

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 1 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 [34]. We first review briefly the CSP problem, referring to [23, 53, 58] 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 $(f(x_i), f(y_i))$ belongs to $R_i$.

The reduction from PPM to CSP is straightforward. Given a PPM instance with text $\tau$ and pattern $\pi$, 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 $j$ means that $(i, \pi(i))$ is matched (embedded) to $(j, \tau(j))$, in the sense of a valid embedding, as defined in § 2. By Lemma 1, 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 $G_\pi$.

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 $N^\alpha(p)$. That is, we require $a \neq b$ and $a < b \Leftrightarrow p.x < N^\alpha(p).x$ and $\tau(a) < \tau(b) \Leftrightarrow p.y < N^\alpha(p).y$.

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 $G_\pi$. We make use of the following well-known result.

**Lemma 2** ([25, 33]) *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, if a width-$t$ tree decomposition of the constraint graph is given.*

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

**Lemma 3** ([29, 30]) *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 2 with a tree-decomposition obtained via Lemma 3. 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. [24], 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 $\tau$. For each pattern-index $i \in [k]$ we *guess* the vertical strip of $\tau$ into which $i$ is mapped in the sought-for embedding of $\pi$ into $\tau$. It is sufficient to do this for a subset of the entries in $\pi$, namely those that become the *leftmost* in their respective strips in $\tau$. Let $X \subseteq [k]$ be the set of indices of such entries in $\pi$.

Guessing $X$ and the strips of $\tau$ 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. [24, 29]). 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. [24], 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 $S_\pi$ 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 2, resulting in the running time $O(n^{k/2+2})$. A more careful argument improves this bound to $O(n^{k/2+1})$; we describe the details in § 6.

We can refine the analysis, noting that when we are assigning values $a_2 < a_4 < a_6 < \ldots$ to the variables $x_2, x_4, x_6, \cdots$ representing $Q^E$, we need to consider only those increasing sequences that leave a gap of at least one for each odd-indexed variable (e.g. $a_2 \geq 2$, $a_4 > a_2 + 1$, etc.). The number of such sequences is $\binom{n-\lceil k/2 \rceil}{\lfloor k/2 \rfloor}$. To see this, start with a sequence that has a minimum required gap of one for each odd-indexed entry, then distribute the remaining total gap of $n-k$ among the $\lfloor k/2 \rfloor + 1$ gaps between consecutive even-indexed entries, as well as before the first, and after the last even-indexed entry. Recall that $a$ indistinguishable balls can be placed into $b$ bins in $\binom{a+b-1}{b-1}$ ways. As $\max_k \binom{n-k}{k} = O(1.6181^n)$, see e.g. [55, 56], 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 [34], 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 2 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 4** *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 2 with Lemma 4 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 $G$, so that both $H_1$ and $H_2$ visit the vertices of $S_i$ in the given order, for all $i \in [k]$. The following observation is immediate.

**Lemma 5** *The incidence graph $G_\pi$ of $\pi$ is k-track if and only if $\pi$ is either k-increasing or k-decreasing.*

2-*monotone patterns* We now 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 $\ell$, 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 $\ell$. This means that the two Hamiltonian paths of $G$ can be drawn on different sides of $\ell$, and therefore $G$ is planar. See Fig. 2 (left) for an example.

The treewidth of a $k$-vertex planar graph is known to be $O(\sqrt{k})$ [12, 26]. A corresponding path-decomposition can be obtained by a recursive use of planar separators. For the case of a pattern $\pi$ that consists of an increasing and a decreasing subsequence (i.e. 2-monotone patterns), we show that the straight-line drawing of

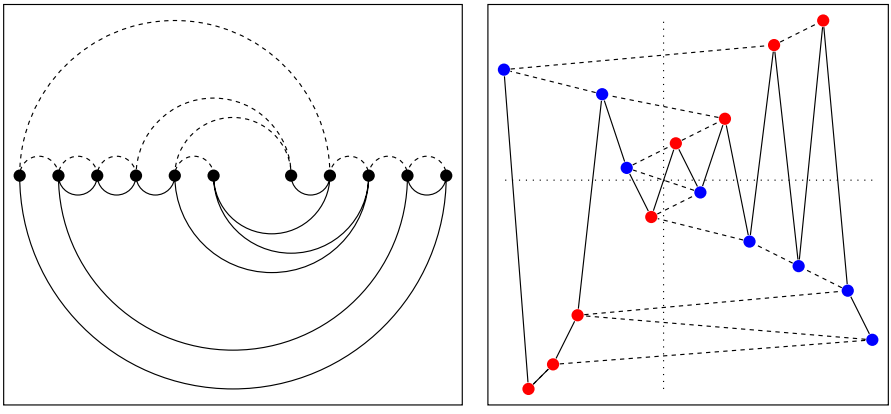

**Fig. 2** (*left*) A planar drawing of a 2-track graph, with one of the two Hamiltonian paths drawn with dashed arcs. Note that edges contained in both Hamiltonian paths are drawn twice for clarity. (*right*) A drawing of the incidence graph of a 2-monotone permutation. Red and blue dots indicate an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

$G_\pi$ (with points $S_\pi$ as vertices) has at most one intersection. An $O(\sqrt{k})$ bound on the treewidth follows via known results [27].

Divide $S_\pi$ by one horizontal and one vertical line, so 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. We argue next that an intersection of two edges must involve the four points closest to the intersection of the horizontal and vertical line, as in Fig. 2 (right). This will immediately imply that there is at most one intersection.

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 $G_\pi$ vertically before the argument, without affecting the graph structure). Observe that $u.y < t.y < s.y < v.y$ must hold. Otherwise, $\pi$ contains the pattern (2, 1, 4, 3), which cannot decompose into an increasing and a decreasing subsequence, so $\pi$ cannot decompose into an increasing and a decreasing subsequence either.

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. This concludes the proof of Theorem 5(i).

We show that the bound in Theorem 5(i) 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:

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

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

We 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 Fig. 3.

*Jordan patterns* The proof of Theorem 5(ii) is immediate, as the incidence graph of Jordan permutations is by definition planar. To see this, recall that a Jordan permutation is defined by the intersection pattern of two curves. We view the curves as the planar embedding of $G_\pi$. The portions of the curves between intersection points correspond to edges (we trim away the loose ends of both curves), and the curves connect the points in the order of their index, resp. value. This turns out to be an exact characterization: $G_\pi$ is planar if and only if $\pi$ is a Jordan permutation.

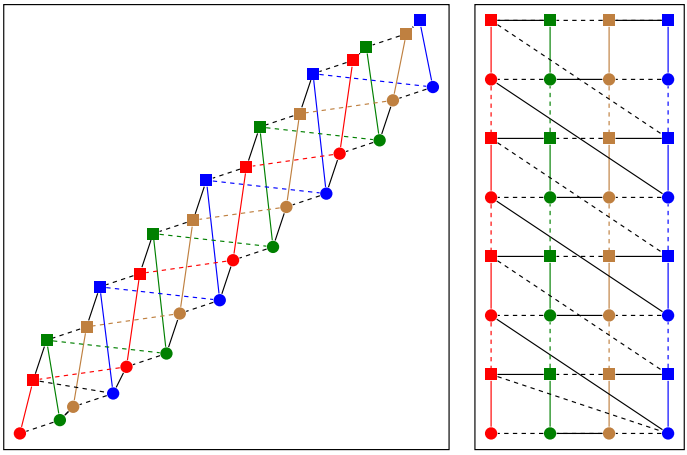

**Fig. 3** A 2-increasing permutation of length 32 whose incidence graph contains a $4 \times 8$ grid. Vertex shape indicates the track, colors are for emphasis of the grid structure (Color figure online)

We remark that for the "only if" direction to hold, *touching points* between the two curves must also be allowed. Consider any noncrossing embedding of $G_\pi$, and construct the two curves as the Hamiltonian paths of $G_\pi$ that connect the vertices by increasing index, resp. value. Whenever the two curves overlap over an edge of $G_\pi$, bend the corresponding part of one of the curves, such as to create two intersection points at the two endpoints of the edge (one of the two intersection points may need to be a touching point).

The tightness of the result follows by the previous example (Fig. 3). As 2-track graphs are planar, the given 2-increasing permutation (whose incidence-graph contains a large grid) is also a Jordan-permutation.

3-*monotone patterns* We now prove Theorem 6. Our rough strategy is to show that for an *arbitrary* lenght-$n$ permutation $\pi$, we can construct a 3-track graph $G$ of order $O(n \log n)$ that contains $G_\pi$ as a minor. Ahal and Rabinovich [1, Theorem 3.4] show that there exist permutations $\pi$, such that the treewidth of $G_\pi$ is $\Omega(n)$. Together with the observation that the treewidth of a graph is not less than the treewidth of its minor, this proves Theorem 6.

Instead of constructing $G$ directly from $\pi$, we construct one intermediate graph $G'$, so that $G'$ contains $G_\pi$ as a subgraph and $G'$ is a minor of $G$. We first show how to construct $G'$ from $\pi$. For this, we need some definitions and observations.

Permutation $\rho'$ is a *split* of a permutation $\rho$ if $\rho'$ arises from $\rho$ by moving a subsequence of $\rho$ to the front. For example, (1, 3, 5, 2, 4) is a split of the identity permutation $\mathrm{id}_5$, 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, $\sigma^{-1}$ is a merge of the two subsequences $1, 2, \ldots, p(\sigma)$ and $p(\sigma)+1, p(\sigma)+2, \ldots, n$.

If $\rho'$ is a split of $\rho$, then $\rho' = \rho \circ \sigma$ for some split permutation $\sigma$. Ahal and Rabinovich [1] mention that every $n$-permutation can be obtained from $\mathrm{id}_n$ by at most $\lceil \log n \rceil$ splits. That means that we can write each permutation $\rho$ as a composition $\sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_{\lceil \log n \rceil}$ of $\lceil \log n \rceil$ split permutations.

For the remainder of the proof, we fix some permutation $\pi$ of length $n$. Let $m \leq \lceil \log n \rceil + 1$, and let $\sigma_1, \sigma_2, \ldots, \sigma_{m-1}$ be split permutations such that $\pi = \sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_{m-1}$ and $\sigma_i \neq \mathrm{id}_n$ for all $i \in [m-1]$. Moreover, let $\pi_1 = \mathrm{id}_n$ and $\pi_{i+1} = \pi_i \circ \sigma_i$ for $i \in [m-1]$. Observe that $\pi_m = \pi$.

We are now ready to define the intermediate graph. Let $G' = ([n], E)$, where $E$ is the union of the Hamiltonian paths corresponding to $\pi_1, \pi_2, \ldots, \pi_m$. The fact that $\pi_1 = \mathrm{id}_n$ and $\pi_m = \pi$ implies that $G_\pi$ a subgraph of $G'$.

It remains to construct the 3-track graph $G$ that contains $G'$ as a minor. We first define the vertex sets corresponding to the three tracks of $G$. Let

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

Let $V = V_x \cup V_y \cup V_z$ be the vertex set of $G$, and observe that $|V| = mn + (m-1)n \leq 2mn \in O(n \log n)$. Figure 4 (left) shows the vertices in an arrangement useful for the rest of the construction.

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:

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

Note that $s_x$ is simply the successor with respect to the total order $\prec$ on $V_x$, and that $s_c$ is a bijection. Figure 4 (middle) illustrates the two functions.

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 $s_x(x_{m,j}) = x_{m,j+1}$. 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

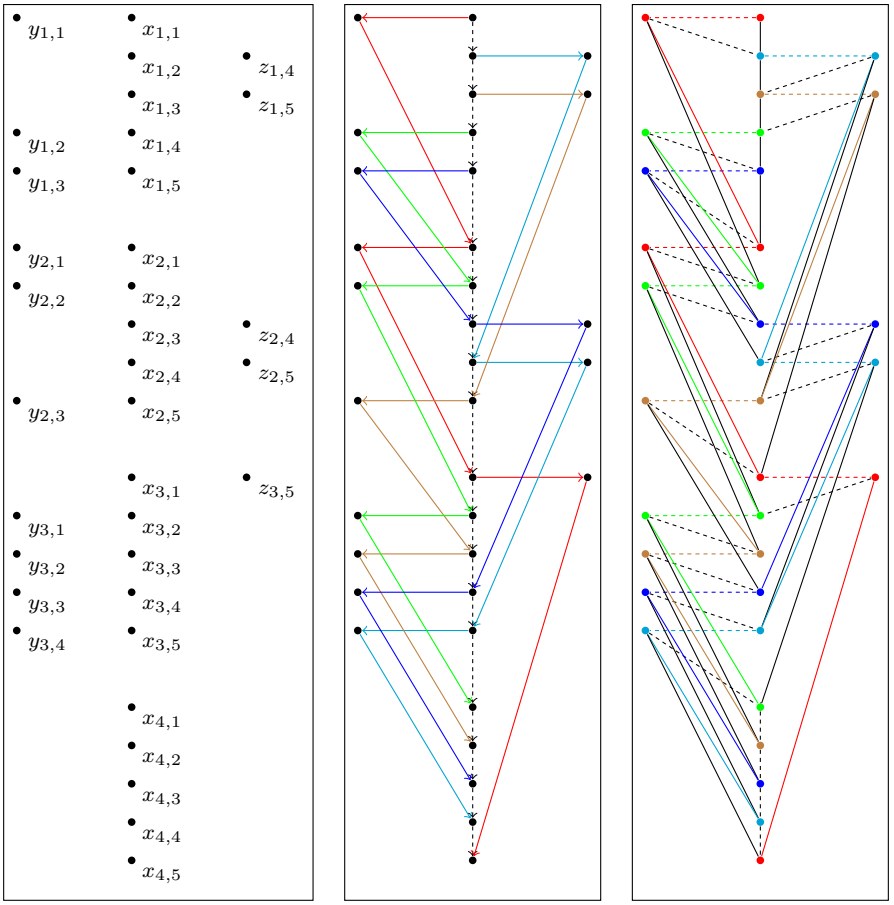


**Fig. 4** Construction of $G$ for $\pi = 43521$. We have $\sigma_1 = 14523$, $\sigma_2 = 12534$, $\sigma_3 = 23451$, $\sigma_1^{-1} = 14523$, $\sigma_2^{-1} = 12453$, $\sigma_3^{-1} = 51234$, and $\pi_1 = 12345$, $\pi_2 = 14523$, $\pi_3 = 14352$, $\pi_4 = 43521$. (All permutations are of length 5, we omitted commas and parentheses.) (*left*) Vertices of $G$ with labels. (*middle*) Illustration of the functions $s_x$ (dashed) and $s_c$ (colored by connected component). (*right*) The actual graph $G$. Dashed edges belong to $P_1$, solid edges to $P_2$

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 show that $G' = ([n], E)$ is a minor of $G$, color the vertices of $G$ with $n$ colors, where color $k$ induces a path $C_k$ of length $2m - 1$ 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 $G'$. See Fig. 4 (right) for an illustration.

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 $G'$. 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 of Theorem 6.

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 $O(dn \log n)$.

*$\sigma$-avoiding patterns* In the proof of Theorem 5(i), we constructed length-$k$ permutations $\pi$ that avoid (1, 2, 3) or (3, 2, 1) with the property that $\mathrm{tw}(\pi) = \Omega(\sqrt{k})$. The same construction works for length-$k$ permutations that avoid an arbitrary $\sigma$, for $|\sigma| \geq 5$: by the Erdős-Szekeres theorem all permutations of length 5 or more contain (1, 2, 3) or (3, 2, 1), therefore, avoiding (1, 2, 3) or (3, 2, 1) implies avoiding $\sigma$.

The only length-4 permutations $\sigma$ that avoid both (1, 2, 3) and (3, 2, 1) are (2, 1, 4, 3), (3, 1, 4, 2), and their reverses. In these cases we can construct a large-treewidth $\sigma$-avoiding permutation using a structural observation of Jelínek and Kynčl [38]. They show that permutations that avoid *both* (2, 1, 4, 3) and (3, 1, 4, 2) have a certain spiraling block-decomposition (see [38, Fig. 13]). Given this structure, the embedding of a grid is a straightforward adaptation of the technique in § 4 (we omit the details).

It follows that there exist $\sigma$-avoiding length-$k$ patterns $\pi$ with treewidth $\Omega(\sqrt{k})$ for all patterns $\sigma$, except when

$\sigma \in \{(1), (1,2), (2,1), (1,3,2), (2,3,1), (2,1,3), (3,1,2)\}$, i.e. exactly the cases when PPM is polynomial-time solvable [38].

## 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. Note that in the second reduction we only need to concern ourselves with the hard instances resulting from the first, PSI-to-PPPM reduction.

*PSI to PPPM* The input to the PSI problem (introduced in [45]) consists of a graph $G$, a graph $H$, and a coloring $\phi$ 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 $\{u, v\} \in E(H)$ implies $\{g(u), g(v)\} \in E(G)$, 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 [45, 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. [17]).

The input to the PPPM problem (introduced in [34]) consists of permutations $\tau$ and $\pi$ of lengths $n$ and $k$ respectively, and a coloring $\phi : [n] \to [k]$ of the entries of $\tau$. The task is to decide whether there is an embedding $g : [k] \to [n]$ of $\pi$ into $\tau$ 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 [34, 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 as hard as PSI, obtaining the following.

**Lemma 6** *PPPM cannot be solved in time $f(k) \cdot n^{o(k/\log k)}$, unless ETH fails.*

***Proof*** Let $G, H, \phi$ be an instance of PSI, as described above, with $n = |V(G)|$, $m = |E(G)|$ and $k = |V(H)| = |E(H)|$. Let $V(H) = \{v_1, \ldots, v_k\}$. Let $V_1, \ldots, V_k$ be a partitioning of $V(G)$, according to $\phi$, such that $u \in V_i$ if and only if $\phi(u) = v_i$. Consider a canonical ordering $(u_1, \ldots, u_n)$ of the vertices of $G$, in which vertices in $V_i$ appear before vertices in $V_j$ for all $i < j$. Assume a preprocessing of $G$, where all edges with endpoints in the same class $V_i$ are deleted, for all $i \in [k]$, as these cannot be part of a subgraph of the required form.

A PPPM instance is created, consisting of a text permutation $\tau$ of length $O(n^2)$, a pattern $\pi$ of length $O(k)$, and a coloring of $\tau$ with $|\pi|$ colors. Intuitively, $\tau$ and $\pi$ represent the adjacency matrices of $G$ and $H$, as well as the coloring $\phi$ of $V(G)$, with minor additional gadgets that enforce that valid embeddings of $\pi$ into $\tau$ correspond exactly to selections of valid $H$-isomorphic subgraphs in $G$. Both $\tau$ and $\pi$ are based on a *tilted grid* permutation, i.e. a permutation corresponding to a set of points obtained by rotating a square grid by a sufficiently small angle (Fig. 5).

We first describe the construction of $\pi$. Consider a two-dimensional grid of size $(k+1) \times (k+1)$, with grid cells indexed from (0, 0) (top left) to $(k, k)$ (bottom right). We place points inside the grid cells, and the resulting point set $S_\pi$ will define the permutation $\pi$, as before. Figure 5 illustrates the construction.

In the first row, excluding cell (0, 0), place points forming the permutation $(2, 1, 4, 3, 6, 5, \ldots, 2k, 2k-1)$, with two consecutive points in each cell, left to right. In the first column, excluding cell (0, 0), place points forming the permutation $(2k-1, 2k, 2k-3, 2k-2, 2k-5, 2k-4, \ldots, 1, 2)$, with two consecutive points

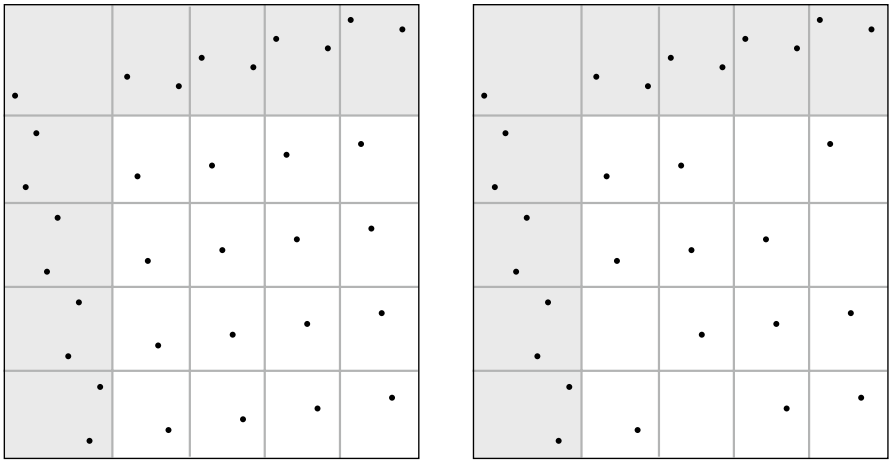

**Fig. 5** (*left*) Pattern $\pi$ corresponding to clique $K_4$. (*right*) Pattern $\pi$ corresponding to cycle $C_4$. First row and column highlighted for clarity. Note that the graph $K_4$ does not actually arise in the proof of Lemma 6, as we require $|V(H)| = |E(H)|$

in each cell, top to bottom. Place one point in cell (0, 0) below all other points in its row, and to the left of all other points in its column. In each grid cell $(i, j)$ for $i,j \in [k]$ place one point if and only if $i = j$ or $\{v_i, v_j\} \in E(H)$. The placement of a point in a cell $(i, j)$ for $i,j \in [k]$ is such that it falls horizontally between the two points in cell $(i, 0)$ and vertically between the two points in cell $(0, j)$. Furthermore the points in cells $(i, j)$ for $i,j \in [k]$ are in each row increasing by $y$-coordinate left-to-right, and are in each column increasing by $x$-coordinate top-to-bottom. Let $\pi$ denote the permutation corresponding to the constructed point set $S_\pi$. It is easy to verify that $\pi$ is of length $7k + 1$.

We next describe the construction of $\tau$. Consider again a grid of size $(k + 1) \times (k + 1)$, with cells indexed from (0, 0) (top left) to $(k, k)$ (bottom right). We place points inside the grid cells, and the resulting point set $S_\tau$ will define the permutation $\tau$. Figure 6 illustrates the construction.

In the first row, excluding cell (0, 0), place points forming the permutation $(2, 1, 4, 3, 6, 5, \ldots, 2n, 2n - 1)$, with $2|V_i|$ consecutive points in cell $(i, 0)$, left to right. In the first column, excluding cell (0, 0), place points forming the permutation $(2n - 1, 2n, 2n - 3, 2n - 2, 2n - 5, 2n - 4, \ldots, 1, 2)$, with $2|V_i|$ consecutive points in cell $(0, i)$, top to bottom. Place one point in cell (0, 0) below all other points in its row, and to the left of all other points in its column. Place additional points, horizontally between the $(2i)$-th and $(2i + 1)$-th points of the first row and vertically between the $(2j)$-th and $(2j + 1)$-th points of the first column, for all $i,j \in [n]$ such that $i = j$ or $\{u_i, u_j\} \in E(G)$. The placement of the points in cells $(\ell, h)$ for $\ell, h \in [k]$ is such that those vertically between the $(2j)$-th and $(2j + 1)$-th points of the first column are increasing by $y$-coordinate left-to-right, and those horizontally between the $(2i)$-th and $(2i + 1)$-th points of the first row are increasing by $x$-coordinate top-to-bottom. Let $\tau$ denote the permutation corresponding to the constructed point set $S_\tau$. It is easy to verify that $\tau$ is of length $5n + 2m + 1 \in O(n^2)$.

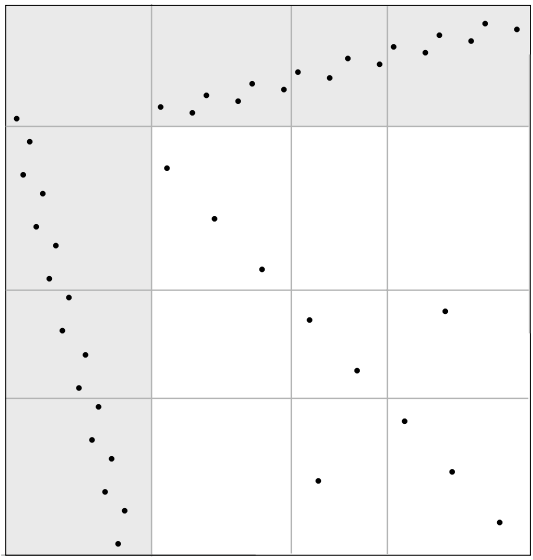

**Fig. 6** Text $\tau$ corresponding to a graph $G$ with $V(G) = \{u_1, \ldots, u_8\}$ partitioned into $V_1, V_2, V_3$, with $|V_1| = |V_3| = 3$ and $|V_2| = 2$, and $E(G) = \{\{u_4, u_7\}\}$

It remains to specify the coloring of $\tau$ with $O(k)$ colors. Color all points in cell $(i, j)$ of the text-grid, for $i, j \in [k]$, with the index of the unique point in cell $(i, j)$ of the pattern-grid. Cells $(i, 0)$ and $(0, i)$ of the pattern-grid, for $i \in [k]$ have two points each. Denote the indices of these as $a_{(i,0)}$ and $b_{(i,0)}$, resp. $a_{(0,i)}$ and $b_{(0,i)}$, left-to-right, resp. top-to-bottom. Color points in cell $(i, 0)$ of the text-grid for $i \in [k]$ with $a_{(i,0)}$ and $b_{(i,0)}$, alternatingly left-to-right. Color points in cell $(0, i)$ of the text-grid for $i \in [k]$ with $a_{(0,i)}$ and $b_{(0,i)}$, alternatingly top-to-bottom. Finally, color the unique point in the text-cell $(0, 0)$ with the index of the unique point in the pattern-cell $(0, 0)$. This concludes the construction of the PPPM instance.

Suppose there is a subgraph of $G$ that is isomorphic to $H$, with an embedding of the vertices that respects the coloring of $V(G)$. Let $u_{i_1}, \ldots, u_{i_k} \in V(G)$ be the vertices matched, respectively, to $v_1, \ldots, v_k \in V(H)$. Observe that $u_{i_j} \in V_j$ for all $j \in [k]$. We argue that there is a valid embedding of $\pi$ into $\tau$ that respects the coloring of $\tau$. Indeed, match the $(2j)$-th and $(2j + 1)$-th points in the first row of the pattern-cell to the $(2i_j)$-th and $(2i_j + 1)$-th points in the first row of the text-cell. Assuming the canonical ordering of vertices, the coloring of these points is in accordance with the matching. Similarly, match the $(2j)$-th and $(2j + 1)$-th points in the first column of the pattern-cell to the $(2i_j)$-th and $(2i_j + 1)$-th points in the first column of the text-cell. Assuming the canonical ordering of vertices, the coloring of these points is in accordance with the matching.

If there is a point $p$ in the pattern-cell $(\ell, h)$ for $\ell, h \in [k]$ and $\ell \neq h$, then $\{v_\ell, v_h\} \in E(H)$ and then, by assumption, $\{u_{i_\ell}, u_{i_h}\} \in E(G)$. Consequently, there is a unique point $q$ in the text-cell $(\ell, h)$ that is horizontally between the matched points in cell $(\ell, 0)$ and vertically between the matched points in cell $(0, h)$, and that, by construction, has the correct color. Thus, $p$ can be matched to $q$. The unique point in the diagonal pattern-cell $(\ell, \ell)$ for $\ell \in [k]$ can be matched to the unique point in the diagonal text-cell $(\ell, \ell)$ that is horizontally between the matched points in cell $(\ell, 0)$

and vertically between the matched points in cell $(0, \ell)$. Finally, match the unique point in the pattern-cell (0, 0) to the unique point in the text-cell (0, 0). The fact that the text-points used in the embedding have the correct relative positions is easy to verify.

Conversely, suppose there is a valid, color-respecting embedding of $\pi$ into $\tau$. Then, the unique points in each diagonal pattern-cell are matched to points in the corresponding diagonal text-cells, and this matching determines a selection of vertices $u_{i_1}, \ldots, u_{i_k}$ from the classes $V_1, \ldots, V_k$, respectively. Pairs of points in the pattern-cells $(1, 0), \ldots, (k, 0)$ and $(0, 1), \ldots, (0, k)$ must be matched to consecutive points in the corresponding text-cells of colors $a_{(*,*)}$ and $b_{(*,*)}$ that bracket the matched diagonal points (since only such pairs are in the correct relative position). The fact that pairs of points in the first row and pairs of points in the first column must bracket the same diagonal points ensures that the selection of the vertices is consistent. The existence of a point $p$ in a pattern-cell $(\ell, h)$ for $\ell, h \in [k]$ and $\ell \neq h$ means that $\{v_\ell, v_h\} \in E(H)$. By the coloring constraint, $p$ can only be matched to a point $q$ in the text-cell $(\ell, h)$, and since the embedding has the valid ordering, $q$ must be bracketed by the pairs of points in the first row and first column that correspond to the vertices $u_{i_\ell}$, resp. $u_{i_h}$. The existence of the point $q$ in the text-pattern means that the corresponding edge $\{u_{i_\ell}, u_{i_h}\}$ exists in $G$. □

*#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 (when restricted to the hard instances resulting from the reduction in the proof of Lemma 6), thereby proving Theorem 4.

Let $k$ now denote the size of the pattern $\pi$ (as well as the number of colors) in an instance of #PPPM. We use oracle-calls to #PPM for all subsets $X \subseteq [k]$, to count the number of embeddings of $\pi$ into $\tau$ using entries of $\tau$ with colors from the set $X$, but ignoring colors for the purpose of the embedding. (We can achieve this by deleting the entries of $\tau$ 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:

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

Here, $C_X$ denotes the number of embeddings that use colors from the set $X$ (obtained by oracle calls). Since $\pi$ 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 $\pi_i$ is matched to an entry of $\tau$ of color $i$, for all $i \in [k]$, i.e. the colors are not permuted. We show this for the hard instances constructed in the proof of Lemma 6.

Consider first the unique point $p$ in the text cell (0, 0). As $p$ is unique in its color-class and all color-classes must be used in the embedding, $p$ must be matched to some pattern-point. As $p$ is the leftmost point of $S_\tau$, all other color-classes are entirely to the right of $p$, therefore, $p$ can only be matched to the leftmost point of the pattern, which is the unique point in pattern-cell (0, 0).

Suppose that the size of the grid used in the construction of $\pi$ is $k' \times k'$. Recall that $k' \in \Theta(k)$. Consider now the $2k'$ pattern points in the first row, i.e. cells $(1,0), \ldots, (k',0)$. As these points must be matched to points above $p$, they can only be matched to text points in cells $(1,0), \ldots, (k',0)$. Pairs of pattern points from the same cell are in decreasing order, can thus only be matched to consecutive pairs of text points. As the pairs form an increasing order in both the pattern and the text, the matching must preserve this order. Finally, each pair of points from one pattern-cell must be matched to a pair of points in a unique text-cell, as otherwise some color would go unused. This shows that the matching of points in the first row respects the coloring. A very similar argument shows that the matching of points in the first column, i.e. cells $(0,1), \ldots, (0,k')$ is also according to the correct color. Finally, any pattern point $p$ from some cell $(i,j)$ for $i,j \in [k']$ is constrained by the already correctly matched pair of points in the first row and first column, that bracket it and constrain it to be matched to a text point in cell $(i,j)$, by construction, 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.

## 6 The Even-Odd Method Revisited

In this section we describe the algorithm of Theorem 2 in a self-contained manner, *not using* the formalism of CSPs.

Let $(Q^E, Q^O)$ be the partition of $S_\pi$ into points with even and odd indices. Formally, $Q^E = \{(2k, \pi(2k)) \mid 1 \le k \le \lfloor k/2 \rfloor\}$, and $Q^O = \{(2k-1, \pi(2k-1)) \mid 1 \le k \le \lceil k/2 \rceil\}$.

Suppose $\tau$ contains $\pi$. Then, by Lemma 1, there exists a valid embedding $f : S_\pi \to S_\tau$. We start by *guessing* a partial embedding $g_0 : Q^E \to S_\tau$. (For example, $g_0 = f|_{Q^E}$ is such a partial embedding.) We then extend $g_0$ step-by-step, adding points to its domain, until it becomes a valid embedding $S_\pi \to S_\tau$.

Let $p_1, \ldots, p_{\lceil k/2 \rceil}$ be the elements of $Q^O$ in increasing order of *value*, i.e. $1 \le p_1.y < \cdots < p_{\lceil k/2 \rceil}.y \le n$, and let $P_0 = \emptyset$ and $P_i = P_{i-1} \cup \{p_i\}$, for $1 \le i \le \lceil k/2 \rceil$. For all $i$, we maintain the invariant that $g_i$ is a restriction of some valid embedding to $Q^E \cup P_i$. By our choice of $g_0$, this is true initially for $i = 0$.

In the $i$-th step (for $i = 0, \ldots, \lceil k/2 \rceil - 1$), we extend $g_i$ to $g_{i+1}$ by mapping the next point $p_{i+1}$ onto a suitable point in $S_\tau$. For $g_{i+1}$ to be a restriction of a valid embedding, it must satisfy conditions (1) and (2) on the relative position of neighbors. Observe that all, except possibly one, of the neighbors of $p_{i+1}$ are already embedded by $g_i$. This is because $N^L(p_{i+1})$ and $N^R(p_{i+1})$ have even index, are thus in $Q^E$, unless they are virtual points and thus implicitly embedded. The point $N^D(p_{i+1})$ is either an even-index point, and thus in $Q^E$, or the virtual point (0, 0) and thus implicitly embedded, or an odd-index point, in which case, by our ordering, it must be $p_i$, and thus, contained in $P_i$. The only neighbor of $p_{i+1}$ possibly not embedded is $N^U(p_{i+1})$.

If we map $p_{i+1}$ to a point $q \in S_\tau$, we have to observe the constraints $g_i(N^L(p_{i+1})).x < q.x < g_i(N^R(p_{i+1})).x$, and $g_i(N^D(p_{i+1})).y < q.y$. If $N^U(p_{i+1})$ is also in the domain of $g_i$, then we have the additional constraint $q.y < g_i(N^U(p_{i+1})).y$.

These constraints determine an (open) axis-parallel box, possibly extending upwards infinitely (in case only three of the four neighbors of $p_{i+1}$ are embedded so far). Assuming $g_i$ is a restriction of a valid embedding $f$, the point $f(p_{i+1})$ must satisfy all constraints, it is thus contained in this box. We extend $g_i$ to obtain $g_{i+1}$ by mapping $p_{i+1}$ to a point $q \in S_\tau$ in the constraint-box, and if there are multiple such points, we pick the one that is *lowest*, i.e. the one with smallest value $q.y$.

The crucial observation is that if $g_i$ is a partial embedding, then $g_{i+1}$ is also a partial embedding, and the correctness of the procedure follows by induction.

Indeed, some valid embedding $f' : S_\pi \to S_\tau$ must be the extension of $g_{i+1}$. If $q = f(p_{i+1})$, then $f'$ is $f$ itself. Otherwise, let $f'$ be identical with $f$, except for mapping $p_{i+1} \to q$ (instead of mapping $p_{i+1} \to f(p_{i+1})$). The only conditions of a valid embedding that may become violated are those involving $p_{i+1}$. The conditions $f'(N^L(p_{i+1})).x < f'(p_{i+1}).x < f'(N^R(p_{i+1})).x$ and $f'(N^D(p_{i+1})).y < f'(p_{i+1}).y$ hold by our choice of $q$.

The condition $f'(p_{i+1}).y < f'(N^U(p_{i+1})).y$ holds a fortiori since we picked the *lowest point* in a box that also contained $f(p_{i+1})$, in other words, $f'(p_{i+1}).y = q.y \le f(p_{i+1}).y < f(N^U(p_{i+1})).y = f'(N^U(p_{i+1})).y$. Thus, $g_{i+1}$ is a partial embedding, which concludes the argument. See Fig. 7 for illustration.

Assuming that our initial guess $g_0$ was correct, we succeed in constructing a valid embedding that certifies the fact that $\tau$ contains $\pi$. We remark that guessing $g_0$ should be understood as trying all possible embeddings of $Q^E$. If our choice of $g_0$ is incorrect, i.e. not a partial embedding, then we reach a situation where we cannot extend $g_i$, and we abandon the choice of $g_0$. If extending $g_0$ to a valid embedding fails for all initial choices, we conclude that $\tau$ does not contain $\pi$. The resulting algorithm is described in Fig. 8.

The space requirement is linear in the input size; apart from minor bookkeeping, only a single embedding must be stored at all times. To analyse the running time, observe first, that $g_0$ must map points in $Q^E$ to points in $S_\tau$, preserving their

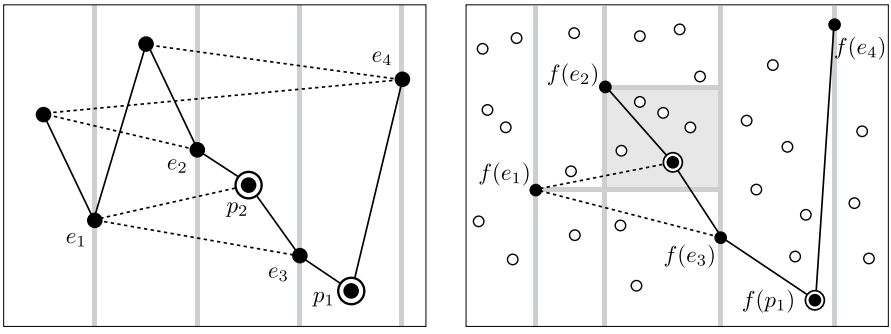


**Fig. 7** (*left*) Pattern $\pi = (6, 3, 8, 5, 4, 2, 1, 7)$ and its incidence graph $G_\pi$. Solid lines indicate neighbors by index, dashed lines indicate neighbors by value (lines may overlap). (*right*) Text permutation $\tau$, points shown as circles. Partial embedding of $\pi$ shown with filled circles. Vertical bars mark even-index points $e_1$, $e_2$, $e_3$, $e_4$. Double circles mark the first two odd-index points $p_1$, $p_2$. Shaded box indicates constraints for embedding $p_2$, determined by $N^U(p_2) = N^L(p_2) = e_2$, $N^D(p_2) = e_1$, and $N^R(p_2) = e_3$. Observe that $p_2$ is mapped to lowest point (by value) that satisfies constraints. Revealed edges of $G_\pi$ are shown

Polynomial-space algorithm for PPM:

```
for all g_0 : Q^E → S_t do
    if g_0 not valid, then next g_0
    for i ← 0 to ⌈k/2⌉ − 1 do
        let q ∈ S_τ with minimum q.y such that:
            g_i(N^L(p_{i+1})).x < q.x < g_i(N^R(p_{i+1})).x
            g_i(N^D(p_{i+1})).y < q.y
            g_i(N^U(p_{i+1})).y > q.y    (in case N^U(p_{i+1}) ∈ Q^E)
        if no such q, then next g_0
        extend g_i to g_{i+1} by mapping p_{i+1} → q
    return g_{⌈k/2⌉}
return "τ avoids π"
```

**Fig. 8** Finding a valid embedding of $S_\pi$ into $S_t$, or reporting that $t$ avoids $\pi$, with precomputed $Q^E$ (even-index points of $S_\pi$) and $(p_1, \ldots, p_{\lceil k/2 \rceil})$ (odd-index points of $S_\pi$ sorted by value)

left-to-right order (by index), and their bottom-to-top order (by value). The first condition can be enforced directly, by considering only *subsequences* of $\tau$. This amounts to $\binom{n}{\lfloor k/2 \rfloor}$ choices for $g_0$ in the outer loop. The second condition can be verified in a linear time traversal of $G_\pi$ (this is the second line of the algorithm in Fig. 8).

All remaining steps can be performed using straightforward data structuring: we need to traverse to neighbors in the incidence graph, to go from $x$ to $g_i(x)$ and back, and to answer rectangle-minimum queries; all can be achieved in constant time, with a polynomial time preprocessing. We can in fact do away with rectangle queries, since candidate points of $\tau$ are considered in increasing order of value—the inner loop thus consists of a single sweep through both $\pi$ and $\tau$, which can be implemented in $O(n)$ time. By a standard bound on the binomial coefficient, the claimed running time of $O(n^{k/2+1})$ follows.

The efficient enumeration of initial embeddings with the required property can be achieved with standard techniques, see e.g. [48]. Finally, we remark that instead of trying all embeddings $g_0$, in practice it may be preferable to build such an embedding incrementally, using backtracking. This allows the process to "fail early" if a certain embedding can not be extended to any partial embedding of $Q^E$. The order in which points of $Q^E$ are considered in the backtracking process can affect the performance significantly, see [42, 43] for consideration of similar issues. Alternatively, a modification of the algorithm of Theorem 1 may also be used to enumerate all valid initial $g_0$.

**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.

**Funding** Open Access funding enabled and organized by Projekt DEAL.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article

are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

## References

1. Ahal, S., Rabinovich, Y.: On complexity of the subpattern problem. SIAM J. Discrete Math. **22**(2), 629–649 (2008)
2. Albert, M.H., Paterson, M.S.: Bounds for the growth rate of meander numbers. J. Comb. Theory Ser. A **112**(2), 250–262 (2005)
3. Albert, M., Bousquet-Mélou, M.: Permutations sortable by two stacks in parallel and quarter plane walks. Eur. J. Comb. **43**, 131–164 (2015)
4. Albert, M.H., Aldred, R.E.L., Atkinson, M.D., Holton, D.A.: Algorithms for pattern involvement in permutations. In: Proceedings of the 12th International Symposium on Algorithms and Computation, ISAAC '01, pp. 355–366, Springer-Verlag, London, UK (2001)
5. Albert, M.H., Homberger, C., Pantone, J., Shar, N., Vatter, V.: Generating permutations with restricted containers. J. Comb. Theory Ser. A **157**, 205–232 (2018)
6. Albert, M.H., Lackner, M.-L., Lackner, M., Vatter, V.: The complexity of pattern matching for 321-avoiding and skew-merged permutations. Discrete Math. Theor. Comput. Sci. **18**(2), 1–17 (2016)
7. Aldous, D., Diaconis, P.: Longest increasing subsequences: from patience sorting to the baik-deift-johansson theorem. Bull. Am. Math. Soc **36**, 413–432 (1999)
8. Arthur, D.: Fast sorting and pattern-avoiding permutations. In: Proceedings of the Fourth Workshop on Analytic Algorithmics and Combinatorics, ANALCO 2007, pp. 169–174 (2007)
9. Ben-Eliezer, O., Canonne, C.L.: Improved bounds for testing forbidden order patterns. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, pp. 2093–2112 (2018)
10. Berendsohn, B.A.: Complexity of permutation pattern matching. Master's thesis, Freie Universität Berlin, (2019)
11. Berndt, D.J., Clifford, J.: 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, pp. 359–370. AAAI Press, (1994)
12. Bodlaender, H.L.: A partial k-arboretum of graphs with bounded treewidth. Theor. Comput. Sci. **209**(1), 1–45 (1998)
13. Bóna, M.: A survey of stack-sorting disciplines. Electron. J. Comb. **9**(2), 1 (2003)
14. Bóna, M.: Combinatorics of Permutations. CRC Press Inc, Boca Raton, FL, USA (2004)
15. Bóna, M.: A Walk Through Combinatorics: An Introduction to Enumeration and Graph Theory. World Scientific, Singapore (2011)
16. Bose, Pt, Buss, J.F., Lubiw, A.: Pattern matching for permutations. Inf. Process. Lett. **65**(5), 277–283 (1998)
17. Bringmann, K., Kozma, L., Moran, S., Narayanaswamy, N.S.: Hitting set for hypergraphs of low vc-dimension. In: 24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, pp. 23:1–23:18, Aarhus, Denmark (2016)
18. Bruner, M.-L., Lackner, M.: The computational landscape of permutation patterns. Pure Math. Appl. **24**(2), 83–101 (2013)
19. Bruner, M.-L., Lackner, M.: A fast algorithm for permutation pattern matching based on alternating runs. Algorithmica **75**(1), 84–117 (2016)
20. Bulteau, L., Rizzi, R., Vialette, S.: Pattern matching for k-track permutations. In: Iliopoulos, C., Leong, H.W., Sung, W.-K. (eds.) Combinatorial Algorithms, pp. 102–114. Springer International Publishing, Cham (2018)
21. Chalermsook, P., Goswami, M., Kozma, L., Mehlhorn, K., Saranurak, T.: Pattern-avoiding access in binary search trees. In: IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, pp. 410–423, 2015

22. Chang, M.-S., Wang, F.-H.: Efficient algorithms for the maximum weight clique and maximum weight independent set problems on permutation graphs. Inf. Process. Lett. **43**(6), 293–295 (1992)
23. Chen, Hubie: A rendezvous of logic, complexity, and algebra. ACM Comput. Surv. **42**(1), 2:1–2:32 (2009)
24. Cygan, M., Kowalik, L., Socala, A.: Improving TSP tours using dynamic programming over tree decompositions. In: 25th Annual European Symposium on Algorithms, ESA 2017, pp. 30:1–30:14, (2017)
25. Dechter, R., Pearl, J.: Tree clustering for constraint networks. Artif. Intell. **38**(3), 353–366 (1989)
26. Downey, R.G., Fellows, M.R.: Parameterized Complexity Monographs in Computer Science. Springer, Berlin(999)
27. Dujmovic, V., Eppstein, D., Wood, D.R.: Structure of graphs with locally restricted crossings. SIAM J. Discrete Math. **31**(2), 805–824 (2017)
28. Erdős, P., Szekeres, G.: A combinatorial problem in geometry. Compos. Math. **2**, 463–470 (1935)
29. Fomin, F.V., Gaspers, S., Saurabh, S., Stepanov, A.A.: On two techniques of combining branching and treewidth. Algorithmica **54**(2), 181–207 (2009)
30. Fomin, F.V., Høie, K.: Pathwidth of cubic graphs and exact algorithms. Inf. Process. Lett. **97**(5), 191–196 (2006)
31. Fox, J.: Stanley-wilf limits are typically exponential. *CoRR*, arxiv: abs/1310.8378, 2013
32. Fox, J., Wei, F.: Fast property testing and metrics for permutations. Combinatorics, Probability and Computing, pp. 1–41 (2018)
33. Freuder, E.C.: Complexity of k-tree structured constraint satisfaction problems. In: Proceedings of the Eighth National Conference on Artificial Intelligence - Volume 1, AAAI'90, pp 4–9. AAAI Press, (1990)
34. Guillemot, S., Marx, D.: Finding small patterns in permutations in linear time. In: Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, pp. 82–101 (2014)
35. Guillemot, S., Vialette, S.: Pattern matching for 321-avoiding permutations. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) Algorithms and Computation, pp. 1064–1073. Springer, Berlin Heidelberg (2009)
36. Hoffmann, K., Mehlhorn, K., Rosenstiehl, P., Tarjan, R.E.: Sorting jordan sequences in linear time using level-linked search trees. Inf. Control **68**(1–3), 170–184 (1986)
37. Ibarra, L.: Finding pattern matchings for permutations. Inf. Process. Lett. **61**(6), 293–295 (1997)
38. Jelínek, V., Kyncl, J.: Hardness of permutation pattern matching. Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, pp. 378–396 (2017)
39. Keogh, E.J., Lonardi, S., Yuan-chi Chiu, B.: 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, pp. 550–556, (2002)
40. Kitaev, S.: Patterns in Permutations and Words. Monographs in Theoretical Computer Science. An EATCS Series. Springer, Berlin (2011)
41. Knuth, D.E.: The Art of Computer Programming, Volume I: Fundamental Algorithms. Addison-Wesley, Boston (1968)
42. Knuth, Donald E.: Estimating the efficiency of backtrack programs. Math. Comput. **29**(129), 122–136 (1975)
43. Knuth, D.E.: Dancing links. arXiv preprint cs/0011047, (2000)
44. Marcus, A., Tardos, G.: Excluded permutation matrices and the stanley-wilf conjecture. J. Comb. Theory Ser. A **107**(1), 153–160 (2004)
45. Marx, D.: Can you beat treewidth? Theory Comput. **6**(1), 85–112 (2010)
46. Neou, B., Rizzi, R., Vialette, S.: Permutation Pattern matching in (213, 231)-avoiding permutations. Discrete Mathematics & Theoretical Computer Science, Vol. 18 no. 2, Permutation Patterns 2015, March 2017
47. Newman, I., Rabinovich, Y., Rajendraprasad, D., Sohler, C.: Testing for forbidden order patterns in an array. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, pp. 1582–1597 (2017)
48. Nijenhuis, A., Wilf, H.S.: Combinatorial Algorithms, 2nd edn. Academic Press Inc., Harcourt Brace Jovanovich, New York-London (1978)
49. Patel, P., Keogh, E., Lin, J., Lonardi, S.: Mining motifs in massive time series databases. In: In Proceedings of IEEE International Conference on Data Mining ICDM'02, pp. 370–377, (2002)

50. Pratt, V.R.: 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, pp 268–277. ACM, (1973)
51. Rosenstiehl, P.: Planar permutations defined by two intersecting Jordan curves. In: Bollobás, B., E Erdős,Refer Reference 28 P. (eds.) Graph Theory and Combinatorics, pp. 259–271. Academic Press, London (1984)
52. Rosenstiehl, P., Tarjan, R.E.: Gauss codes, planar hamiltonian graphs, and stack-sortable permutations. J. Algorithms **5**(3), 375–390 (1984)
53. Seidel, R.: A new method for solving constraint satisfaction problems. In: Proceedings of the 7th International Joint Conference on Artificial Intelligence, IJCAI 1981, pp. 338–342, (1981)
54. Simion, R., Schmidt, F.W.: Restricted permutations. Eur. J. Comb. **6**(4), 383–406 (1985)
55. Sloane, N.J.A.: The encyclopedia of integer sequences, http://oeis.org. Sequence A073028
56. Tanny, S.M., Zuker, M.: On a unimodal sequence of binomial coefficients. Discrete Math. **9**(1), 79–89 (1974)
57. Tarjan, R.E.: Sorting using networks of queues and stacks. J. ACM **19**(2), 341–346 (1972)
58. Tsang, E.P.K.: Foundations of Constraint Satisfaction. Computation in Cognitive Science. Academic Press, Cambridge (1993)
59. Vatter, V.: Permutation classes. In Miklós Bóna, editor, Handbook of Enumerative Combinatorics, chapter 12. Chapman and Hall/CRC, New York, 2015. Preprint at arxiv: abs/1409.5159
60. Yugandhar, V.: Saxena, Sanjeev: Parallel algorithms for separable permutations. Discrete Appl. Math. **146**(3), 343–364 (2005)


**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.