Discrete Mathematics and Theoretical Computer Science DMTCS vol. **18**:2, 2016, #11

The Complexity of Pattern Matching for 321*-Avoiding and Skew-Merged Permutations*

Michael Albert[1] Marie-Louise Lackner[2*]
Martin Lackner[3†] Vincent Vatter[4‡]

[1] *Department of Computer Science, University of Otago, Dunedin, New Zealand*
[2] *Institute of Discrete Mathematics and Geometry, TU Wien, Vienna, Austria*
[3] *Department of Computer Science, University of Oxford, Oxford, UK*
[4] *Department of Mathematics, University of Florida, Gainesville, Florida, USA*

received 22nd Oct. 2015, revised 5th Dec. 2016, accepted 20th Dec. 2016.

The PERMUTATION PATTERN MATCHING problem, asking whether a pattern permutation π is contained in a text permutation τ, is known to be NP-complete. We present two polynomial time algorithms for special cases. The first is applicable if both π and τ are 321-avoiding while the second is applicable if both permutations are skew-merged. Both algorithms have a runtime of $O(kn)$, where k is the length of π and n the length of τ.

Keywords: pattern matching, permutation class, permutation pattern

1 Introduction

In this paper, a permutation is a bijective function from $[n]$ to itself, where n is a positive integer and $[n] = \{1, 2, \ldots, n\}$. Therefore, a permutation, $\pi : [n] \to [n]$ *is* the set of ordered pairs $(i, \pi(i))$. We occasionally write specific permutations in the usual one line notation, e.g., 321 represents the permutation of $[3]$ equal to $\{(1, 3), (2, 2), (3, 1)\}$. The *size* of π is just the cardinality of this set, and we denote the elements, also called points, of a permutation by variables such as x and y. We adopt the usual conventions with respect to order of such points, i.e., $(i, \pi(i))$ lies to the left of $(j, \pi(j))$ if $i < j$ and above $(k, \pi(k))$ if $\pi(i) > \pi(k)$, with corresponding definitions for 'to the right of' and 'below'. Given an element x in a permutation π, we define (wherever possible):

- $x^{\blacktriangleleft}$ the element immediately to its left,
- $x^{\blacktriangleright}$ the element immediately to its right,
- $x^{\blacktriangle}$ the element immediately above it, and
- $x^{\blacktriangledown}$ the element immediately below it.

*M.-L. Lackner's research was supported by the Austrian Science Foundation FWF, grant P25337-N23.

†M. Lackner's research was supported by the Austrian Science Foundation FWF, grants P25518-N23 and Y698, and by the European Research Council (ERC) under grant number 639945 (ACCORD).

‡Vatter's research was partially supported by the National Science Foundation under Grant Number DMS-1301692.

ISSN 1365–8050

We use the symbol $\perp$ to represent 'undefined'. Any operator applied to $\perp$ also yields $\perp$. For example, in the permutation 31254 and for $x = 3$ we have: $x^{\blacktriangleleft} = \perp$, $x^{\blacktriangleright} = 1$, $x^{\blacktriangle} = 4$ and $x^{\blacktriangledown} = 2$.

Let π and τ be two permutations. An injective function, f, from π into τ is an *embedding* if, for all elements x and y of π, the elements $f(x)$ and $f(y)$ of τ are in the same relative order as x and y (e.g., if x lies below and to the right of y, then $f(x)$ also lies below and to the right of $f(y)$). If there is an embedding from π into τ then we say that τ *contains* π. If not, then we say that τ *avoids* π. The following problem, analogous to problems related to detecting occurrences of patterns in words, is central with respect to these concepts:

PERMUTATION PATTERN MATCHING (PPM)	
Input:	A text permutation τ of size n and a pattern π of size k.
Question:	Does τ contain π?

Bose, Buss, and Lubiw [5] showed in 1998 that the PPM problem is NP-complete[(i)].

The recent work of Guillemot and Marx [9] shows that the PPM problem can be solved in time $2^{O(k^2 \log k)}n$, i.e., linear-time in n when k is viewed as a constant. In particular, this implies that the PPM problem is *fixed-parameter tractable (fpt)* with respect to the size of the pattern k. Work prior to this breakthrough result achieved runtimes of $O(n^{1+2k/3} \cdot \log n)$ [2] and $O(n^{0.47k+o(k)})$ [1]. Using $\mathsf{run}(\tau)$ as parameter, i.e., the number of alternating runs of τ, Bruner and Lackner [6] present an fpt algorithm with runtime $O(1.79^{\mathsf{run}(\tau)} \cdot kn)$.

A *permutation class*, $\mathcal{C}$, is a set of permutations with the property that if $\tau \in \mathcal{C}$ and τ contains π then $\pi \in \mathcal{C}$. In other words, $\mathcal{C}$ is closed downwards with respect to the partial ordering of permutations given by the relation "is contained in". A permutation class is proper if it does not contain every permutation. Permutation classes are frequently defined in terms of avoidance conditions, namely, for any set B of permutations, the set $\mathrm{Av}(B)$ consisting of those permutations which avoid every element of B is a permutation class (and it is always proper if B is non-empty). Conversely, every permutation class is equal to the class of permutations avoiding its complement, or even avoiding the minimal elements (with respect to the partial ordering mentioned above) of its complement. For an overview of results on permutation classes, we refer to the corresponding chapter in the *Handbook of Enumerative Combinatorics* [15].

This leads to two natural ways in which one might restrict the PPM problem. One is to impose additional structure on the pattern, most naturally, to insist that the pattern belongs to a particular (proper) permutation class. One example which has been studied is the class of *separable* permutations, which are those avoiding both 3142 and 2413. If the pattern is separable, PPM can be solved in polynomial time [2, 5, 11, 13, 16]. The fastest algorithm for this case is by Ibarra [11] with a runtime of $O(kn^4)$. Formally, we define this class of problems as follows:

$\mathcal{C}$-PATTERN PERMUTATION PATTERN MATCHING ($\mathcal{C}$-PATTERN PPM)	
Input:	A text permutation τ of size n and a pattern π of size k, where π belongs to a fixed proper permutation class $\mathcal{C}$.
Question:	Does τ contain π?

A second and more restrictive specialisation of the PPM problem is to insist that both the pattern and text belong to a (proper) permutation class. This is the version of the problem that we study.

[(i)] Note that because both the pattern and text are regarded as input, the size of the input is $n + k$. Were we to regard the size of the pattern as fixed, then the trivial $\mathcal{O}\left(\binom{n}{k} \cdot k\right)$ algorithm would be polynomial time.

$\mathcal{C}$ PERMUTATION PATTERN MATCHING ($\mathcal{C}$-PPM)	
Input:	A text permutation τ of size n and a pattern π of size k, both belonging to a fixed proper permutation class $\mathcal{C}$.
Question:	Does τ contain π?

Clearly, for a fixed $\mathcal{C}$, polynomial time algorithms for $\mathcal{C}$-PATTERN PPM apply to $\mathcal{C}$-PPM as well. Consequently, the separable case, i.e., $\mathrm{Av}(3142, 2413)$-PPM, can be solved in $O(kn^4)$ time [11]. Note that if the pattern avoids 132, 231, 213 or 312 then it is automatically separable and thus the $\mathcal{C}$-PATTERN PPM problem for all four classes $\mathrm{Av}(132)$, $\mathrm{Av}(231)$, $\mathrm{Av}(213)$ or $\mathrm{Av}(312)$ can be solved in polynomial time. Most relevant to our work is a result by Guillemot and Vialette [10] that establishes an $O(k^2n^6)$-time algorithm for $\mathrm{Av}(321)$-PPM. In Sections 2 and 3, we improve their approach to give the following.

Theorem 1.1. *Given* 321*-avoiding permutations* τ *of size* n *and* π *of size* k*, there is an* $O(kn)$*-time algorithm which determines whether* τ *contains* π*.*

In Section 4 we show how to adapt this approach to the class of skew-merged permutations, which are those permutations whose elements can be partitioned into an increasing subsequence and a decreasing subsequence. Skew-merged permutations can also be characterised as those permutations that avoid both 3412 and 2143 [14].

Theorem 1.2. *Given skew-merged permutations* τ *of size* n *and* π *of size* k*, there is an* $O(kn)$*-time algorithm which determines whether* τ *contains* π*.*

The following elementary observation will be used repeatedly.

Lemma 1.3. *Let* π *and* τ *be permutations and* $f : \pi \to \tau$*. Then* f *is an embedding of* π *into* τ *if and only if for every element* x *of* π*:*

- *if* $x^{\blacktriangleleft} \neq \perp$ *then* $f(x)$ *lies strictly to the right of* $f(x^{\blacktriangleleft})$ *and*
- *if* $x^{\blacktriangledown} \neq \perp$ *then* $f(x)$ *lies strictly above* $f(x^{\blacktriangledown})$*.*

Proof: Suppose that x and y are points of π and that, without loss of generality, y lies strictly to the left of x. Then y occurs in the sequence $x^{\blacktriangleleft}, x^{\blacktriangleleft\blacktriangleleft}, x^{\blacktriangleleft\blacktriangleleft\blacktriangleleft}, \ldots$. So, by inductive use of the first property, $f(x)$ lies strictly to the right of $f(y)$. Similarly, inductive use of the second property establishes that the vertical relationship between $f(x)$ and $f(y)$ is the same as that between x and y, and the result follows.

The other direction follows directly from the definition of embeddings given on page 2: $x^{\blacktriangleleft}$ is an element strictly to the left of x and thus $f(x)$ lies strictly to the right of $f(x^{\blacktriangleleft})$ for an embedding f. In the same way, $x^{\blacktriangledown}$ is an element strictly below x and thus $f(x)$ lies strictly above $f(x^{\blacktriangledown})$. □

2 The Lattice of Rigid Embeddings of 321-Avoiding Permutations

It is easy to see that the elements of any 321-avoiding permutation π can be partitioned into two increasing subsequences. This partition is in general not unique but in any such partition, one of these subsequences will contain all those elements which participate as the '2' in a copy of 21—called the *upper elements* of π and denoted U_π— and the other will contain all those elements which participate as the '1' in a copy of 21—called the *lower elements* of π and denoted L_π. Elements that are neither upper nor lower elements, i.e., those that are not involved in a copy of 21, can be part of either of the two subsequences. Let us formalise these definitions: An element x of π is an upper element if there is some embedding of

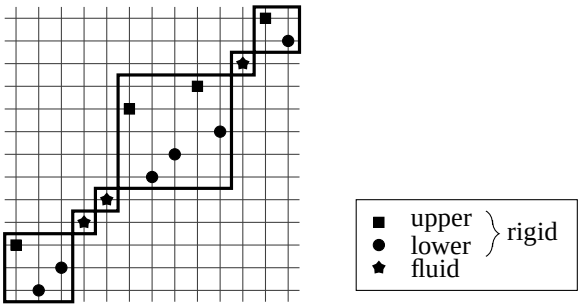

Fig. 1: The decomposition of the 321-avoiding permutation $\pi = 3\,1\,2\,4\,5\,9\,6\,7\,10\,8\,11\,13\,12$ into rigid and fluid elements.

$21 = \{(1,2),(2,1)\}$ into π such that x is the image of $(1,2)$ and a lower element if there is an embedding of 21 such that x is the image of $(2,1)$.

Following Albert, Atkinson, Brignall, Ruškuc, Smith, and West [3], elements which are either upper or lower elements of π are referred to as *rigid* elements, and π is called a *rigid permutation* if all of its elements are rigid (i.e., if $\pi = U_\pi \cup L_\pi$). The remaining elements will be called *fluid* elements. For an example of a 321-avoiding permutation and its decomposition into rigid and fluid elements, see Figure 1.

Note that it can be determined in linear time which elements are upper, lower and fluid in a permutation. For this purpose one simply needs to scan the permutation from left to right and record the largest element encountered so far, denoted by ℓ, and the smallest element yet to come at the right, denoted by s. When we read an element x, three cases can occur:

- $x > s$: In this case $x\,s$ forms a 21-pattern and thus x is an upper element.
- $x < \ell$: In this case $\ell\,x$ forms a 21-pattern and thus x is a lower element
- $x \leq s$ and $x \geq \ell$ (which implies that $x = s$ and $s > \ell$): In this case x does not occur in a 21-pattern and is thus a fluid element.

The existence of fluid elements in a pattern will be the source of some difficulty in solving the $\mathrm{Av}(321)$-PPM problem, and will be addressed in the next section. For the remainder of this section we consider a rigid pattern π of size k and a 321-avoiding text τ of size n. Since an embedding preserves relative locations of points, the image of any rigid element must be rigid. More precisely, we have the following:

Observation 2.1. *Let π be a rigid pattern and τ be an arbitrary 321-avoiding permutation. If there exists an embedding of π into τ, then it must map upper (resp., lower) elements of π to upper (resp., lower) elements of τ and the fluid elements of τ will never occur in an embedding.*

In order to look for such embeddings we must widen our search space. A map $f : \pi \to \tau$ is called a *rigid mapping* if f maps upper (resp., lower) elements of π to upper (resp., lower) elements of τ. As noted above, because π is rigid, every embedding of π into τ is a rigid mapping, but the converse is far from true since, among other reasons, rigid mappings need not be injective.

Given two points, x and y, in U_π, we say $x \leq y$ if y lies above and to the right of x. This is a linear order on U_π, and we have similar linear orders (all denoted $\leq$) on L_π, U_τ and L_τ. This makes the set of all rigid mappings of π into τ into a partially ordered set using point-wise comparison; that is, given rigid mappings $f, g : \pi \to \tau$, we write $f \leq g$ if $f(x) \leq g(x)$ for all elements x of π. In fact, it is easy to see that this partially ordered set is a distributive lattice; given two rigid mappings $f, g : \pi \to \tau$ their *meet*

and *join* can be defined, respectively, by

$$\begin{aligned}(f \wedge g)(x) &= \min\{f(x), g(x)\},\\ (f \vee g)(x) &= \max\{f(x), g(x)\}\end{aligned}$$

for all elements x of π.

It is notable that these observations also hold for embeddings. That is, the set of embeddings from π into τ is a sublattice of the lattice of rigid mappings:

Theorem 2.2. *bruner [[3, Theorem 2]] Given a rigid pattern π and a 321-avoiding text τ, the set of embeddings of π into τ forms a distributive lattice under the operations of meet and join defined above.*

It follows from Theorem 2.2 that if π is contained in τ then there is a *minimum embedding* of π into τ which we denote by $e_{\min}$.

Given an element x of some 321-avoiding permutation σ we define $x^{\blacktriangleleft}_U$ to be the rightmost element of U_σ that is to the left of x. Of course, we have corresponding notations such as $x^{\blacktriangledown}_L$, $x^{\blacktriangleright}_U$ and so on. In all cases, if no such element exists we get $\perp$ as usual. We also define the *type* of x, $T(x)$ to be U if $x \in U_\sigma$ and L if $x \in L_\sigma$. The following result forms the core of our algorithm for determining whether there is an embedding of π into τ, at least for the case where π is rigid. It will allow us to turn an arbitrary rigid mapping into an embedding, if possible.

Proposition 2.3. *Suppose that $e : \pi \to \tau$ is an embedding, $f : \pi \to \tau$ is a rigid mapping, and, for all $x \in \pi$, $f(x) \leq e(x)$. Then, for all $x \in \pi$:*

$$\max\{f(x^{\blacktriangleleft})^{\blacktriangleright}_{T(x)}, f(x^{\blacktriangledown})^{\blacktriangle}_{T(x)}\} \leq e(x),$$

where we define $\max\{y, \perp\} = y$ and $\max\{\perp\} = 0$.

Proof: We first establish that $f(x^{\blacktriangleleft})^{\blacktriangleright}_{T(x)} \leq e(x)$. Since x lies strictly to the right of $x^{\blacktriangleleft}$ (and e is an embedding), $e(x)$ lies strictly to the right of $e(x^{\blacktriangleleft})$ and is of the same type as x, so, it does not lie to the left of $e(x^{\blacktriangleleft})^{\blacktriangleright}_{T(x)}$. Consequently, $e(x^{\blacktriangleleft})^{\blacktriangleright}_{T(x)} \leq e(x)$. But $f(x^{\blacktriangleleft}) \leq e(x^{\blacktriangleleft})$ and so $f(x^{\blacktriangleleft})^{\blacktriangleright}_{T(x)} \leq e(x^{\blacktriangleleft})^{\blacktriangleright}_{T(x)} \leq e(x)$. The arguments for the other case are exactly the same. □

Applying the proposition above in the case where $f = e$, we see that for any embedding, e, from a rigid π into τ, and any $x \in \pi$:

$$\max\{e(x^{\blacktriangleleft})^{\blacktriangleright}_{T(x)}, e(x^{\blacktriangledown})^{\blacktriangle}_{T(x)}\} \leq e(x).$$

Now suppose that f is any rigid mapping from π to τ. We say that x *is a problem* if it violates the above condition, i.e., x is a problem if

$$f(x) < \max\{f(x^{\blacktriangleleft})^{\blacktriangleright}_{T(x)}, f(x^{\blacktriangledown})^{\blacktriangle}_{T(x)}\}. \tag{1}$$

Intuitively , x is a problem if $f(x)$ is too low compared with $f(x^{\blacktriangledown})$ or too far left compared with $f(x^{\blacktriangleleft})$. We let $P(f)$ be the set of problems for f, for which the following holds:

Corollary 2.4. *Let π be a rigid permutation and τ a 321-avoiding permutation. A rigid mapping f is an embedding of π into τ if and only if the set of problems $P(f)$ is empty.*

Proof: If f is an embedding, it follows from Proposition 2.3 that no element $x \in \pi$ fulfills condition (1). Thus $P(f)$ is empty.

For the other direction, assume that f is not an embedding. From Lemma 1.3 we know that there exists an $x \in \pi$ such that $f(x)$ is below or equal to $f(x^{\blacktriangledown})$ or such that $f(x)$ is left of or equal to $f(x^{\blacktriangleleft})$. First, if $f(x)$ is below or equal to $f(x^{\blacktriangledown})$ this implies that $f(x)$ is strictly below $f(x^{\blacktriangledown})^{\blacktriangle}_{T(x)}$ and hence $x \in P(f)$. Second, if $f(x)$ is left of or equal to $f(x^{\blacktriangleleft})$, we have that $f(x)$ is strictly left of $f(x^{\blacktriangleleft})^{\blacktriangleright}_{T(x)}$. Now note that for $f(x), f(y) \in \tau$ of the same type, $f(x)$ is left of $f(y)$ if and only if $f(x)$ is below $f(y)$. Moreover we know that f preserves types and thus $f(x)$ and $f(x^{\blacktriangleleft})^{\blacktriangleright}_{T(x)}$ have the same type. We conclude that $f(x) < f(x^{\blacktriangleleft})^{\blacktriangleright}_{T(x)}$ and thus $x \in P(f)$. □

We now describe an algorithm, displayed as Algorithm 1. Given as input a rigid permutation π and a 321-avoiding permutation τ, it returns the minimum embedding $e_{\min}$ of π into τ when it exists, and fails otherwise. The algorithm constructs and updates a rigid mapping f, ensuring that $f \leq e_{\min}$ at all times (if an embedding exists). Let f_0 be the map that sends all the elements of U_π to the least element of U_τ and all elements of L_π to the least element of L_τ.

Algorithm 1 Find a minimum embedding of π into τ, or demonstrate that no embeddings exist.

Initialise: $f \leftarrow f_0$.
Compute: $P(f)$.
while f is defined everywhere, and $P(f)$ is non-empty **do**
    Choose $x \in P(f)$.
    Update: $f(x) \leftarrow \max\{f(x^{\blacktriangleleft})^{\blacktriangleright}_{T(x)}, f(x^{\blacktriangledown})^{\blacktriangle}_{T(x)}\}$
    Recompute: $P(f)$.
end while
Return: f, which, if everywhere defined, equals $e_{\min}$.

The correctness of this algorithm is easy to establish. Within the while loop, if f is everywhere defined, $P(f)$ is non-empty, and x is chosen for the update step, then the updated version of f is strictly greater than the original at x, and has the same value elsewhere. Since the set of rigid maps is finite, the loop can be executed a bounded number of times, and the algorithm halts. In the case where $e_{\min}$ exists, we certainly have $f_0 \leq e_{\min}$. So, by Proposition 2.3, it is always the case that $f \leq e_{\min}$. Therefore, when the loop terminates, the algorithm returns an embedding that is less than or equal to $e_{\min}$, and hence must equal $e_{\min}$. Should $e_{\min}$ not exist, then termination can only occur because f is not everywhere defined, and so the algorithm fails as required in this case.

We can further combine the correctness analysis with a run-time analysis to obtain the following.

Proposition 2.5. *Given a rigid 321-avoiding permutation π of size k and a 321-avoiding permutation τ of size n there is an algorithm which determines an embedding of π into τ if one exists, and fails otherwise, whose run-time is $O(kn)$.*

Proof: The algorithm in question is Algorithm 1, and it remains to show that we can achieve the bound claimed for the run-time. As noted, each execution of the loop increases the value of $f(x)$ for at least one x (in the linear ordering, $\leq$, of either U_τ or L_τ). Since there are at most n possible values any $f(x)$ can take, and only k distinct x, the loop certainly executes not more than kn times. So, if we can establish that the computation in the loop can be carried out in constant time, the claim follows.

In an initialisation phase (not part of the algorithm proper) we can certainly compute tables of all the values x_b^a for x in both π and τ, $a \in \{\blacktriangleleft, \blacktriangleright, \blacktriangledown, \blacktriangle\}$ and b either absent or equal to one of L or U. For π this can be done in $O(k)$ time, and for τ in $O(n)$ time, so this initialisation can be absorbed into the claimed run-time. This ensures that the "Update" operations in the loop can be carried out in constant time. We can maintain $P(f)$ as a queue, and separately maintain an array of boolean values that indicate whether or not $x \in P(f)$. To start the loop, we dequeue some x. The update operation ensures that x is no longer a problem, so we can set its value in the array to `false`. Moreover, the update operation only changes the value of $f(x)$, and increases it. So it cannot "solve" any existing problem (other than that of x) and the only other way that it could change the problem set would be if $f(x)$ moved to the right of $f(x^{\blacktriangleright})$ or above $f(x^{\blacktriangle})$. Therefore, in the recompute phase we only need to check those two possibilities, and enqueue $x^{\blacktriangleright}$ and/or $x^{\blacktriangle}$ (setting their boolean values in the array to `true`) if necessary. By making reference to the array, we can ensure that we never have duplicate elements in the queue – so every iteration of the loop really does result in a proper update. □

Let us end this section by providing a simple example illustrating how the presented algorithm works.

Example 2.6. Let us consider the text permutation $\tau = 3\,1\,2\,4\,5\,9\,6\,7\,10\,8\,11\,13\,12$ represented in Figure 1 and the pattern $\pi = 2\,1\,4\,5\,3$. Note that π is indeed rigid, whereas τ is not; we can however ignore the fluid elements when looking for an embedding of π into τ as explained above. The upper elements in π are $2, 4$ and 5 and the lower elements are 1 and 3. We now describe a possible run of the algorithm (the order in which problems are resolved is not determined):

1. We start with the initial rigid mapping $f = f_0$ defined as follows: $f_0(1) = f_0(3) = 1$ and $f_0(2) = f_0(4) = f_0(5) = 3$. By checking the condition in equation (1) we see that all elements except 1 and 2 are problems: $P(f_0) = P_0 = \{3, 4, 5\}$.

2. We resolve the problem $x = 4$ for which we have $\max\{f(x^{\blacktriangleleft})^{\blacktriangleright}_U, f(x^{\blacktriangledown})^{\blacktriangle}_U\} = 9$ and update f such that $f(4) = 9$. In order to recompute $P(f)$, we only need to check $x^{\blacktriangleright} = x^{\blacktriangle} = 5$. We cannot possibly have resolved the problem 5 at the same time, so it remains in $P(f)$ and we have $P(f) = \{3, 5\}$.

3. We resolve the problem $x = 5$ for which we have $\max\{f(x^{\blacktriangleleft})^{\blacktriangleright}_U, f(x^{\blacktriangledown})^{\blacktriangle}_U\} = 10$ and update f such that $f(5) = 10$. In order to recompute $P(f)$, we only need to check $x^{\blacktriangleright} = 3$ ($5^{\blacktriangle}$ is not defined). We cannot possibly have resolved the problem 3 at the same time, so it remains in $P(f)$ and we have $P(f) = \{3\}$.

4. We resolve the last problem $x = 3$ for which we have $\max\{f(x^{\blacktriangleleft})^{\blacktriangleright}_L, f(x^{\blacktriangledown})^{\blacktriangle}_L\} = 8$ and update f such that $f(3) = 8$. In order to recompute $P(f)$, we only need to check $x^{\blacktriangle} = 4$ ($3^{\blacktriangleright}$ is not defined). The element 4 is no longer a problem since it is large enough and thus $P(f)$ is empty.

5. The algorithm terminates successfully since $P(f)$ is empty and has found the minimal embedding $e = e_{\min}$ of π into τ defined as follows: $e(2) = 3$, $e(1) = 1$, $e(4) = 9$, $e(5) = 10$ and $e(3) = 8$.

3 Fluid Elements and the $O(kn)$ Algorithm for 321-Avoiding Permutations

In this section we aim to complete the proof of Theorem 1.1 and to do so we must face the issue of fluid elements in the pattern π. Since a fluid element participates in no 21, each other element of π is either

below and left of it, or above and right of it. This is represented most easily using another notational convention. Suppose that σ and θ are two permutations of size m and n respectively. Then $\sigma \oplus \theta$ is the permutation whose points are:

$$\sigma \cup \{(i+m, \theta(i)+m) \,:\, i \in [n]\}.$$

Informally, to form $\sigma \oplus \theta$ we just place θ above and to the right of σ. Clearly $\oplus$ is associative, though of course not commutative.

For any 321-avoiding permutation π there is a unique decomposition:

$$\pi = \pi_1 \oplus \pi_2 \oplus \cdots \oplus \pi_t$$

where, for $1 \leq i \leq t$, π_i is either rigid or a singleton, and it is never the case that both π_i and π_{i+1} are rigid. The singleton elements of this representation correspond precisely to the fluid elements of π. For an example, consider again Figure 1 where the black squares correspond to the blocks π_i of this representation.

Given π of size k we can easily compute this representation in $O(k)$ time, simply by finding the fluid elements of π (which are those elements that are both left-to-right maxima and right-to-left minima). Henceforth, we assume that this representation is given.

In the algorithm to determine whether π embeds in τ we will construct, for each $1 \leq i \leq t$ at most two embeddings of $\pi_1 \oplus \cdots \oplus \pi_i$ into τ in such a way that, if any embedding of π into τ exists, then at least one of the two partial embeddings can be extended to a full embedding.

So we first consider the following question: given an embedding, e_i, of $\pi_1 \oplus \cdots \oplus \pi_i$ into τ that extends to an (unknown) embedding, e, of π into τ, how can we construct a pair of embeddings of $\pi_1 \oplus \cdots \oplus \pi_i \oplus \pi_{i+1}$ into τ, at least one of which extends to an embedding of π into τ?

We distinguish three cases for π_{i+1}. For this purpose, let T_i denote the set of elements that lie above and to the right of the image of e_i. Then, the image of e restricted to the elements corresponding to π_{i+1} is contained in T_i. Let us first consider the case where π_{i+1} is rigid. Then the image of e on the elements corresponding to π_{i+1} must be greater than or equal to (point by point), the image of π_{i+1} under its minimum embedding into T_i. Thus, if we choose the minimal embedding of π_{i+1} into T_i, the resulting embedding e_{i+1} extends to an embedding of π into τ. Though T_i is, strictly speaking, not a permutation all of its associated operators are the same as those of τ (except some are undefined, e.g., the leftmost element of T_i has no left neighbour in T_i but may well have one in τ). So, in this case we can use Algorithm 1 in order to find the minimal embedding of π_{i+1} into T_i and hereby obtain a single extension of e_i with the required property.

A similarly easy case is where π_{i+1} is a singleton, i.e., a fluid element and T_i begins with its least element (which is a fluid element as well). Then nothing can be lost by mapping π_{i+1} to that element.

The only remaining case is where π_{i+1} is a singleton and the first element of T_i is not its minimum. Since every element of T_i lies above its first element, or above and to the right of its minimum, we can extend e_i in two ways – one sending π_{i+1} to the leftmost element of T_i (which is an upper element) and one to its minimum (which is a lower element), and one of these must be extensible.

Now it seems that we might have a problem – given two partial embeddings of $\pi_1 \oplus \cdots \oplus \pi_i$ might they not extend to three or four candidate embeddings of $\pi_1 \oplus \cdots \oplus \pi_i \oplus \pi_{i+1}$? Indeed this is the case, but only if π_{i+1} is a singleton. If it has four possible images, two belong to U_τ and two to L_τ. Since all further elements of π lie above and to the right of this fluid element, we only need to retain the embeddings

where its image is the lesser of the two in each of these sets. Likewise, if it has three possible images (one of which might be fluid), at least one of them can be ignored. Another way to say this is that because $\pi_1 \oplus \cdots \oplus \pi_i \oplus \pi_{i+1}$ ends with its maximum element, so do its images under the embedding. Among three or more elements of a 321-avoiding permutation there are at least one and at most two elements that do not participate as the 2 in a 12 pattern. We need retain only those embeddings whose maximum is not such a 2, as otherwise they could be replaced by an embedding with a smaller maximum in forming a full extension.

Since the sum of the size of the rigid permutations in the representation of π is at most the total size of π, the parts of the algorithm where we construct minimal rigid embeddings still require at most $O(kn)$ time in total. Dealing with singletons (fluid elements) clearly requires only constant time since we can find the next (to the right) fluid/upper/lower element in τ in constant time. Also, filtering out non-optimal extensions can be done in constant time since only the maximal elements of these extensions have to be compared and at most four extensions exist at the same time. We conclude that the total cost of the algorithm is still $O(kn)$. If τ contains π the algorithm terminates successfully and returns one or possibly two embeddings.

What if no embedding exists? Then, following the plan above as if it did (beginning from an empty map, i.e., the case $i = 0$) we must at some point reach a failing case of Algorithm 1, or possibly encounter an empty T_i. In either case, we fail since we have demonstrated that no embedding can be possible.

This completes the proof of Theorem 1.1.

Again, let us provide an example demonstrating how the algorithm for arbitrary 321-avoiding patterns works.

Example 3.1. As in Example 2.6, we consider the text permutation $\tau = 3\,1\,2\,4\,5\,9\,6\,7\,10\,8\,11\,13\,12$ represented in Figure 1. The pattern is $\pi = 2\,1\,3\,4\,5\,7\,6\,8$. The upper elements in π are 2 and 7, the lower ones are 1 and 6, and the fluid elements are 3, 4, 5 and 8. The algorithm proceeds block by block in the decomposition of π.

1. It starts with the rigid block consisting of the elements 2 and 1. Algorithm 1 takes care of this block and, as in Example 2.6, $e(2) = 3$ and $e(1) = 1$.

2. The next block π_2 is the singleton element 3. T_1, the set of elements that lie above and to the right of the image of e starts with a fluid element and thus we can set $e(3) = 4$.

3. We have the same situation for π_3 which consists of the singleton element 4 and set $e(4) = 5$.

4. The block π_4 is again a singleton element. However, T_3 does not start with its minimal element and thus two choices are possible for $e(5)$: we can either send 5 to the leftmost upper element in T_3 or to the leftmost lower element. We store these two possibilities: $e_U(5) = 9$ and $e_L(5) = 6$.

5. The next block π_5 is rigid and we thus apply Algorithm 1 which is not detailed here. For the choice $e_U(5) = 9$ it leads to $e_U(7) = 13$ and $e_U(6) = 12$ whereas for $e_L(5) = 6$ it leads to $e_L(7) = 10$ and $e_L(6) = 8$. These two partial embeddings are rigid and thus comparable: $e_L \leq e_U$ and we can disregard e_U. This is a good choice, since e_U cannot be extended to an embedding of π into τ since the last element 8 cannot be mapped anywhere.

6. It remains to determine $e(8)$. Since T_5 starts with its minimal element we can choose this one and set $e(8) = 11$.

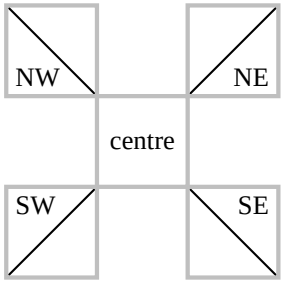

Fig. 2: The decomposition of a skew-merged permutation into its centre and four corners.

7. The algorithm terminates successfully and returns an embedding of π into τ: $e(2) = 3$, $e(1) = 1$, $e(3) = 4$, $e(4) = 5$, $e(5) = 6$, $e(7) = 10$, $e(6) = 8$ and $e(8) = 11$.

4 Skew-Merged Permutations

The permutations avoiding 321 can be partitioned into two monotone increasing sequences. Of course the permutations avoiding 123 can be similarly partitioned (into decreasing sequences) and the results of the previous section apply to them as well. However, the class of *skew-merged permutations*, those that can be partitioned into an increasing and a decreasing sequence, requires further analysis, though as we shall see the analogue of Theorem 1.1 is also true in this context.

Towards this goal, we first identify a set of *rigid* elements of a skew-merged permutation. In Figure 2 these are the elements lying in the corner regions. Specifically we say that an element of a skew-merged permutation is of type:

NE	if it participates as a 3 in a 213;
NW	if it participates as a 3 in a 312;
SW	if it participates as a 1 in a 132;
SE	if it participates as a 1 in a 231,

and we call any other element of a skew-merged permutation *central*. We first verify that the illustration of a skew-merged permutation shown in Figure 2 is correct. This is a result due to Atkinson [4], and so we only sketch part of the proof to give its flavour.

Proposition 4.1. *The elements of a skew-merged permutation decompose by type as shown in Figure 2. Moreover, the central elements form a monotone subsequence.*

Proof: Recall that another characterisation of skew-merged permutations is the following: they are those permutations that do not contain either 3412 or 2143.

Let a skew-merged permutation π be given, and suppose that $\pi = I \cup D$ is a partition of π into a monotone increasing and monotone decreasing sequence. Consider first elements of type NE (all other types can be handled by parallel arguments due to symmetry). Since any such participates as a 3 in a 213, it must belong to I (otherwise, the elements participating as the 2 and 1 would both belong to I which is of course impossible). So the elements of type NE form a monotone increasing sequence.

Suppose that C is of type NE, with BAC an occurrence of 213 and a is of type SW with acb an occurrence of 132. Then $a \in I$ for similar reasons to the preceding ones. If C preceded a (and hence was

also smaller than it) then, $BAcb$ would be an occurrence of 2143. So, all elements of type SW lie below and to the left of those of type NE. Now suppose that z is of type NW, with zxy an occurrence of 312. If C were to precede z we would have various cases: first if C lay below y then $BAzy$ would be 2143, if C lay above y but below z then $Czxy$ would be 3412, if C lay above z and B above y, then $BCxy$ would be 3412, but if B lay below y then $BAzy$ would be 2143. As all these cases lead to contradictions, C must follow z.

All other cases can be dealt with similarly. Finally, to see that the central elements form a monotone sequence observe that they must certainly avoid all of 132, 213, 231, and 312 lest some of them be non-central. But, only monotone permutations (of either type) avoid these four permutations. □

This decomposition can be computed in linear time:

Lemma 4.2. *Given a skew-merged permutation of size n, there is an algorithm that computes its partition into types in $O(n)$-time.*

Proof: Let θ be an arbitrary skew-merged permutation. Notice that the part of θ to the left of the leftmost element of type NE or SE avoids 231 and 213. Such permutations have a characteristic $>$ shape since any element must not be intermediate in value between two to its right. We are interested in finding the maximum prefix of θ which has this characteristic shape, or what amounts to the same thing, the leftmost element of θ such that the prefix ending at that element involves 231 or 213.

This can be accomplished in linear time: we scan θ from left to right and determine for every position i whether it is an ascent ($\theta(i) < \theta(i+1)$) or a descent ($\theta(i) > \theta(i+1)$). At any moment we only store the last encountered ascent a and descent d. The element $\theta(i)$ plays the role of a 1 in a 231 pattern, if $\theta(i) < \theta(a)$; it plays the role of a 3 in a 213, if $\theta(i) > \theta(d)$. If either of the two conditions apply to position i, we have identified the leftmost element of type NE or SE. That is, we have found the boundary line between the centre region and the Eastern region of θ.

In a similar manner we can find all of the boundary lines: by scanning θ from right to left we find the boundary between West and centre, by scanning from bottom to top we find the boundary between South and centre and by scanning from top to bottom we find the boundary between North and centre. We can thus compute the partition of θ into types by scanning θ four times. □

We will now describe an algorithm for skew-merged patterns and texts and provide the necessary theoretic background. This algorithm consists of two main parts: In the first part, the non central elements of the pattern π are embedded into τ using a similar approach as for rigid permutations and adapting Algorithm 1 which will deliver a minimal embedding of the non-central elements. In the second part, we will extend this minimal embedding to the central elements of π.

In this sense, the non-central elements of a skew-merged permutation correspond to the rigid elements of a 321-avoiding permutation. Since they are defined by the occurrence of certain patterns and since embeddings preserve such patterns it is immediately clear that if $e : \pi \to \tau$ is an embedding of one skew-merged permutation into another, then e must preserve the type of all non-central elements.

In order to be able to speak of minimal embeddings in the context of skew-merged permutations, we need to introduce some new notation. For two non-central elements of the same type we write $x \lhd y$ if x lies strictly further out from the center than y ($x \trianglelefteq y$ will mean that either $x \lhd y$ or $x = y$). The minimum with respect to this relation $\lhd$ is denoted by *outer* and the maximum by *inner*. For two embeddings, e_1 and e_2 of the non-central elements of π into τ that preserve types we can define their meet by

$$e_1 \wedge e_2(x) = \mathit{outer}\{e_1(x), e_2(x)\}$$

for all non-central $x \in \pi$. Then, just as in the 321-avoiding case, $e_1 \wedge e_2$ is also an embedding of the non-central elements of π into those of τ:

Lemma 4.3. *Let π be a skew-merged pattern with no central elements and τ be an arbitrary skew-merged permutation. Then the following holds: If e_1 and e_2 are embeddings of π into τ then their meet $f := e_1 \wedge e_2$ as defined above is an embedding as well.*

Proof: Let $x \neq y$ be two elements in π and let us assume that x lies to the left of y in π. We need to show that $f(x)$ lies to the left of $f(y)$ in τ and that the horizontal relation between x and y is preserved as well. The key argument is that taking the minimum of the elements in the above sense automatically translates into taking their actual minimum or maximum (equivalently, the leftmost or rightmost element), depending on the type of element. In order to give a formal proof, we distinguish between three cases.

- If x and y are of the same type. We detail the case of SW elements here, as the other ones are analogous (one simply needs to replace "minimum" by "maximum" and/or "left of" by "right of" depending on the type). In this case, taking the minimum of the elements in the sense defined earlier is nothing else than taking their actual minimum, which again is the same as taking the left-most element. Since we have that $f(x) \leq e_1(x) < e_1(y)$ (and $f(x)$ is to the left of $e_1(y)$) as well as $f(x) \leq e_2(x) < e_2(y)$ (and $f(x)$ is to the left of $e_2(y)$), it follows that $f(x) < \mathit{outer}(e_1(y), e_2(y)) = f(y)$ (and $f(x)$ is to the left of $f(y)$).

- If x and y lie in opposite corners of the diagram. In this case the statement follows immediately from the fact that an embedding preserves types. Indeed, all SW elements are to the left of and smaller than NE ones and all NW elements are to the left of and larger than SE ones. Thus both the vertical as well as the horizontal relation between x and y is preserved.

- The remaining cases, where x and y are not of the same type, but are both elements in the south, north, east or west. We detail the case of two elements in the north, i.e., x is a NW element and y a NE one. The other cases can be dealt with analogously (by interchanging minimum with maximum or vertical with horizontal positions). Without loss of generality, we further assume that $x < y$. First, it is clear that $f(x)$ lies strictly to the left of $f(y)$ since types are preserved. Second, regarding the horizontal relation between x and y, let us note that taking the element that is furthest away from the centre translates into taking the maximum. Thus, we have that $f(y) \geq e_1(y) > e_1(x)$ as well as $f(y) \geq e_2(y) > e_2(x)$ which implies that $f(y) > f(x)$.

The consideration of these cases completes the proof. □

Observe the following: if either e_1 or e_2 was the restriction of an actual embedding, e, of π into τ to the non-central elements then we can extend the mapping $e_1 \wedge e_2$ to central elements using e there, and thereby obtain an embedding. So, among all embeddings of π into τ there is one whose effect on the non-central elements is the minimum of all the embeddings of the non-central elements of π into those of τ. We will see later on how such an extension to the central elements of π can be found.

This minimum embedding of the non-central elements can be found by modifying the definition of the problem set and the update rule of Algorithm 1. The only thing we need to do in order to reflect the new notion of minimum/maximum in this definition, is to redefine the notation introduced in the Introduction. Given a non-central element x in a skew-merged permutation π, we denote by (wherever possible):

x^{oh} the next non-central element further out from the center in horizontal direction,
x^{ih} the next non-central element towards the center (inwards) in horizontal direction,
x^{iv} the next non-central element towards the center (inwards) in vertical direction, and
x^{ov} the next non-central element further out from the center in vertical direction.

For example, in the skew-merged pattern π depicted in Figure 3 and $x = 7$, we have: $x^{oh} = 1$, $x^{ih} = \perp$, $x^{iv} = 6$, and $x^{ov} = \perp$.

We also define the type of a non-central element x in a skew-merged permutation, $T(x)$, to be the corner in which x lies, i.e., $T(x)$ can be NW, SW, NE or SE. Moreover, we extend the notation introduced above as follows: For $a \in \{oh, ih, iv, ov\}$ and $b \in \{NE, SE, SW, NW\}$, we define x^a_b to be the next non-central element in π according to direction a that is of type b. In other words, x^a_b is the first element in the sequence $(x^a, (x^a)^a, \dots)$ of type b. If there is no such element, i.e., no element in $(x^a, (x^a)^a, \dots)$ is of type b, then we set $x^a_b = \perp$. For example, in the skew-merged pattern π depicted in Figure 3 and $x = 5$, we have $x^{ov}_{NW} = 7$, $x^{ov}_{NE} = 6$, whereas $x^{oh}_{SE} = \perp$ and $x^{iv}_{SE} = \perp$.

With this new notation, one can see that an analogue of Proposition 2.3 holds for skew-merged permutations:

Proposition 4.4. *Suppose that e is an embedding of the non-central elements of π into τ, f is a mapping of the non-central elements of π into τ that preserves types, and, for all non-central $x \in \pi$, $f(x) \trianglelefteq e(x)$. Then, for all non-central $x \in \pi$:*

$$\mathit{inner}\left\{f(x^{oh})^{ih}_{T(x)}, f(x^{ov})^{iv}_{T(x)}\right\} \trianglelefteq e(x).$$

The proof of this Proposition is analogous to the one of Proposition 2.3.

We thus say that a non-central element x of a skew-merged permutation is a *problem* if:

$$f(x) \triangleleft \mathit{inner}\left\{f(x^{oh})^{ih}_{T(x)}, f(x^{ov})^{iv}_{T(x)}\right\}, \tag{2}$$

for a mapping f of the non-central elements of π into τ that preserves types. Moreover, when we resolve the problem x by updating the value of $f(x)$ this is done analogously to the case of 321-avoiding permutations and we set $f(x) = \mathit{inner}\left\{f(x^{oh})^{ih}_{T(x)}, f(x^{ov})^{iv}_{T(x)}\right\}$.

This finishes the description of the necessary modifications of Algorithm 1. As for Algorithm 1 we assume that x^a and x^a_b for $a \in \{oh, ih, iv, ov\}$ and $b \in \{NE, SE, SW, NW\}$ is precomputed and thus can be found in constant time. Given the decomposition of π and τ into types, these precomputations can be done in linear time. Both the steps required for the update of f and the recomputation of the problem set $P(f)$ can be carried out in constant time.

To complete the proof of Theorem 1.2 we must show that, having found a minimum embedding of the non-central elements of π to those of τ, the existence of a full embedding can also be determined sufficiently quickly. We need to determine whether or not the central part of π can be embedded into the remainder of τ, i.e., the set of elements in τ which consists of central elements and all adjacent elements that have not yet been used in the minimum embedding. The central part of π is a monotone pattern of a certain size at most k, and the remaining part of τ is a skew-merged permutation of size at most n (whose endpoints we know).

In general, finding a longest increasing (or decreasing) subsequence of size k in a permutation of size n can be done in time $O(n \log\log k)$ [7]. Thus, checking whether the central part can be embedded into

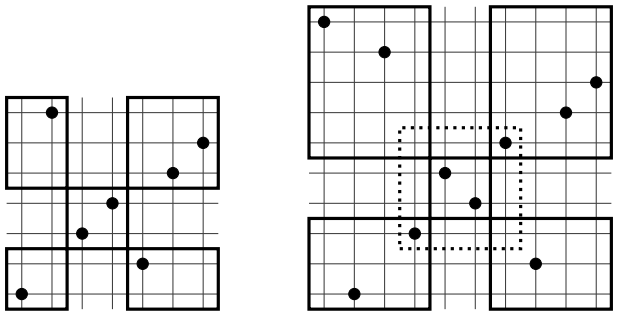

Fig. 3: Decomposition of the skew-merged permutations π and τ into their centres and four corners.

the remaining part of τ can be done within the $O(nk)$ runtime bound of our algorithm. In the special case of skew-merged permutations, finding a longest increasing (resp., decreasing) subsequence can even be done in $O(n)$ time. To be more precise, $O(n)$ time is only required for obtaining the partition into five types as represented in Figure 2 (which is already available in our case); the remaining steps require only constant time.

Indeed, for longest increasing subsequences the following observations can be made (the case of decreasing subsequences can be treated analogously): The elements of type SW and NE will always contribute to a longest increasing subsequence. Moreover, such a subsequence also contains as many elements as possible from the centre, i.e., if the center is increasing then all elements contribute to a longest increasing subsequence and if the center is decreasing we can arbitrarily pick one centre element. Note that it is never advantageous to include elements of type NW or SE. This can be seen as follows: At most one NW or SE element can be part of an increasing subsequence. Thus, if the centre is non-empty, it is certainly not advantageous to include a NW or SE element. Let us assume that the centre is empty. An element of type NW occurs as a 3 in a 312 pattern. Among the elements playing the role of the 1 and the 2, at least one element (and possibly both of them) is of type SW or NE. Thus, including an element of type NW would force us to exclude one or two elements of type SW or NE. In other words, we cannot increase the size of an increasing subsequence by adding an element of type NW. A similar argument holds for elements of type SE. We conclude that for the size of the longest increasing subsequence we only have to add the number of elements of type SW and NE as well as the size of the longest increasing subsequence in the central part. Let us end this section by providing a simple example illustrating how this modified version of Algorithm 1 works.

Example 4.5. Let us consider the text permutation $\tau = 10\,1\,9\,3\,5\,4\,6\,2\,7\,8$ and the pattern $\pi = 1\,7\,3\,4\,2\,5\,6$. Both permutations and their decomposition into types are shown in Figure 3. We start by describing a possible run of the algorithm (the order in which problems are resolved is not determined) finding the minimal embedding of the non-central elements of π into τ:

1. We start with the initial mapping $f = f_0$ that sends all non-central elements of one type in π to the minimal element of this type in τ (i.e., the element that is furthest out from the center). It is defined as follows: $f(1) = 1$, $f(7) = 10$, $f(5) = f(6) = 8$ and $f(2) = 2$. We compute the problem set

using the condition in equation (2) and obtain $P(f) = \{5, 7\}$.

2. We resolve the problem $x = 5$ for which we have *inner* $\{f(x^{oh})^{ih}_{NE}, f(x^{ov})^{iv}_{NE}\} = 7$ and update f such that $f(5) = 7$. In order to recompute $P(f)$, we only need to check $x^{ih} = 2$ since x^{iv} is not defined. The choice $f(2) = 2$ does not create a problem with this new choice for $f(5)$. We cannot possibly have resolved the problem 7 at the same time, so it remains in $P(f)$ and we have $P(f) = \{7\}$.

3. We resolve the problem $x = 7$ for which we have *inner* $\{f(x^{oh})^{ih}_{NW}, f(x^{ov})^{iv}_{NW}\} = 9$ and update f such that $f(7) = 9$. In order to recompute $P(f)$, we only need to check $x^{iv} = 6$ since x^{ih} is not defined. The choice $f(6) = 8$ does not create a problem with this new choice for $f(7)$.

4. The algorithm has found the minimal embedding $e = e_{\min}$ of the non-central elements of π into τ defined as follows: $f(1) = 1$, $f(7) = 9$, $f(2) = 2$, $f(5) = 7$ and $f(6) = 8$.

5. We need to map the central elements 3 and 4 of π into the remaining part of τ (marked by a dotted line in Figure 3). Since the central elements of π consist of an increasing subsequence of size two, we can choose any such subsequence within the dotted area in τ. We decide to set $f(3) = 3$ and $f(4) = 5$ which finally gives an embedding of π into τ.

5 Concluding Remarks

We conclude by mentioning some open problems related to this work. We have seen in Theorem 1.1 that $\mathrm{Av}(321)$-PPM can be solved in $O(kn)$ time. Guillemot and Vialette showed that the more general $\mathrm{Av}(321)$-PATTERN PPM problem can be solved in $\mathcal{O}(kn^{4\sqrt{k}+12})$ time. It is an open problem whether $\mathrm{Av}(321)$-PATTERN PPM can be solved in polynomial time. Note that if the pattern avoids 132, 231, 213 or 312 then it is automatically separable and thus the $\mathcal{C}$-PPM problem and the $\mathcal{C}$-PATTERN PPM problem for all four classes $\mathrm{Av}(132)$, $\mathrm{Av}(231)$, $\mathrm{Av}(213)$ or $\mathrm{Av}(312)$ can be solved in polynomial time. Consequently the $\mathrm{Av}(321)$-PATTERN PPM—which is equivalent to the $\mathrm{Av}(123)$-PATTERN PPM—is the only open case for $\mathrm{Av}(\beta)$-PATTERN PPM where β has size 3.

In case $\mathrm{Av}(321)$-PATTERN PPM turns out to be NP-complete, $\mathrm{Av}(\beta)$-PATTERN PPM will also be NP-complete if β is any permutation of size four other than 2143, 3142, 2413, or 3412. Interestingly, this list contains exactly those patterns that define the classes of skew-merged and of separable permutations. Moreover, NP-completeness of $\mathrm{Av}(321)$-PATTERN PPM would imply that $\mathrm{Av}(\beta)$-PATTERN PPM is NP-complete for β of size five or more, since by Erdős–Szekeres Theorem [8] every permutation of size at least five contains 123 or 321.

Looking at the big picture, Theorems 1.1 and 1.2 show that $\mathcal{C}$-PPM can be solved in polynomial time for $\mathrm{Av}(321)$ and $\mathrm{Av}(2143, 3412)$, respectively. It might be that $\mathcal{C}$-PPM is always polynomial-time solvable for a fixed, proper class $\mathcal{C}$. It would be of considerable interest to either establish this statement or to prove a dichotomy theorem that distinguishes permutation classes for which $\mathcal{C}$-PPM is polynomial-time solvable and those that yield hard $\mathcal{C}$-PPM instances. The same question can be asked for $\mathcal{C}$-PATTERN PPM, although it seems rather unlikely that this problem is polynomial time solvable for every fixed, proper class $\mathcal{C}$.

Note added in proof. After a draft of this paper was posted on the arXiv, Jelínek and Kynčl [12] established that the $\mathrm{Av}(\beta)$-PATTERN PPM problem is indeed NP-complete for every

$$\beta \notin \{1, 12, 21, 132, 213, 231, 312\}.$$

They further showed that the $\mathrm{Av}(4321)$-PPM problem is NP-complete, even when the pattern is restricted to be 321-avoiding.

References

[1] S. Ahal and Y. Rabinovich. On complexity of the subpattern problem. *SIAM J. Discrete Math.*, 22(2):629–649, 2008.

[2] M. H. Albert, R. E. L. Aldred, M. D. Atkinson, and D. A. Holton. Algorithms for pattern involvement in permutations. In P. Eades and T. Takaoka, editors, *12th International Symposium on Algorithms and Computation (ISAAC)*, volume 2223 of *Lecture Notes in Comput. Sci.*, pages 355–366. Springer, Berlin, Germany, 2001.

[3] M. H. Albert, M. D. Atkinson, R. Brignall, N. Ruškuc, R. Smith, and J. West. Growth rates for subclasses of $\mathrm{Av}(321)$. *Electron. J. Combin.*, 17:Paper 141, 16 pp., 2010.

[4] M. D. Atkinson. Permutations which are the union of an increasing and a decreasing subsequence. *Electron. J. Combin.*, 5:Paper 6, 13 pp., 1998.

[5] P. Bose, J. F. Buss, and A. Lubiw. Pattern matching for permutations. *Inform. Process. Lett.*, 65(5):277–283, 1998.

[6] M.-L. Bruner and M. Lackner. A fast algorithm for permutation pattern matching based on alternating runs. *Algorithmica*, 75(1):84–117, 2016.

[7] M. Crochemore and E. Porat. Fast computation of a longest increasing subsequence and application. *Inform. and Comput.*, 208(9):1054–1059, 2010.

[8] P. Erdős and G. Szekeres. A combinatorial problem in geometry. *Compos. Math.*, 2:463–470, 1935.

[9] S. Guillemot and D. Marx. Finding small patterns in permutations in linear time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 82–101. SIAM, Philadelphia, Pennsylvania, 2014.

[10] S. Guillemot and S. Vialette. Pattern matching for 321-avoiding permutations. In Y. Dong, D.-Z. Du, and O. Ibarra, editors, *Algorithms and Computation, Proceedings of the Twentieth International Symposium (ISAAC)*, volume 5878 of *Lecture Notes in Comput. Sci.*, pages 1064–1073. Springer, Berlin, Germany, 2009.

[11] L. Ibarra. Finding pattern matchings for permutations. *Inform. Process. Lett.*, 61(6):293–295, 1997.

[12] V. Jelínek and J. Kynčl. Hardness of permutation pattern matching. arXiv:1608.00529 [math.CO].

[13] B. E. Neou, R. Rizzi, and S. Vialette. Pattern matching for separable permutations. In S. Inenaga, K. Sadakane, and T. Sakai, editors, *String Processing and Information Retrieval: 23rd International Symposium, SPIRE 2016*, volume 9954 of *Lecture Notes in Comput. Sci.*, pages 260–272. Springer, Berlin, Germany, 2016.

[14] Z. E. Stankova. Forbidden subsequences. *Discrete Math.*, 132(1-3):291–316, 1994.

[15] V. Vatter. Permutation classes. In M. Bóna, editor, *Handbook of Enumerative Combinatorics*, pages 754–833. CRC Press, Boca Raton, Florida, 2015.

[16] V. Yugandhar and S. Saxena. Parallel algorithms for separable permutations. *Discrete Appl. Math.*, 146(3):343–364, 2005.