View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Dagstuhl Research Online Publication Server

Generalized Coloring of Permutations

Vít Jelínek
Computer Science Institute, Charles University
Malostranské náměstí 25, Praha 1, 11800, Czechia
jelinek@iuuk.mff.cuni.cz
https://orcid.org/0000-0003-4831-4079

Michal Opler
Computer Science Institute, Charles University
Malostranské náměstí 25, Praha 1, 11800, Czechia
opler@iuuk.mff.cuni.cz
https://orcid.org/0000-0002-4389-5807

Pavel Valtr
Department of Applied Mathematics, Charles University
Malostranské náměstí 25, Praha 1, 11800, Czechia
https://orcid.org/0000-0002-3102-4166

Abstract

A permutation π is a *merge* of a permutation σ and a permutation τ, if we can color the elements of π red and blue so that the red elements have the same relative order as σ and the blue ones as τ. We consider, for fixed hereditary permutation classes $\mathcal{C}$ and $\mathcal{D}$, the complexity of determining whether a given permutation π is a merge of an element of $\mathcal{C}$ with an element of $\mathcal{D}$.

We develop general algorithmic approaches for identifying polynomially tractable cases of merge recognition. Our tools include a version of nondeterministic logspace streaming recognizability of permutations, which we introduce, and a concept of bounded width decomposition, inspired by the work of Ahal and Rabinovich.

As a consequence of the general results, we can provide nontrivial examples of tractable permutation merges involving commonly studied permutation classes, such as the class of layered permutations, the class of separable permutations, or the class of permutations avoiding a decreasing sequence of a given length.

On the negative side, we obtain a general hardness result which implies, for example, that it is NP-complete to recognize the permutations that can be merged from two subpermutations avoiding the pattern 2413.

2012 ACM Subject Classification Mathematics of computing → Permutations and combinations, Mathematics of computing → Combinatorial algorithms, Theory of computation → Problems, reductions and completeness

Keywords and phrases Permutations, merge, generalized coloring

Digital Object Identifier 10.4230/LIPIcs.ESA.2018.50

Funding Supported by the GAUK project 1766318, by project Impuls of the Neuron Fund for Support of Science, and by project 18-19158S of the Czech Science Foundation.

© Vít Jelínek, Michal Opler, and Pavel Valtr;
licensed under Creative Commons License CC-BY

1 Introduction

Definitions and previous results

A *permutation* is a sequence $\pi = \pi_1, \pi_2, \ldots, \pi_n$ in which each number from the set $[n] = \{1, 2, \ldots, n\}$ appears exactly once. We then say that a permutation $\pi = \pi_1, \ldots, \pi_n$ *contains* a permutation $\sigma = \sigma_1, \ldots, \sigma_k$, if π has a subsequence of length k whose elements have the same relative order as the elements of σ (see Section 2 for a more formal definition). If π does not contain σ, we say that π *avoids* σ, or π is *σ-avoiding*.

A permutation π is a *merge* of a permutation σ and a permutation τ, if we can color the elements of π with colors red and blue so that the red elements have the same relative order as σ and the blue ones as τ. For two sets of permutations $\mathcal{C}$ and $\mathcal{D}$, we let $\mathcal{C} \odot \mathcal{D}$ denote the set of the permutations that can be obtained by merging a permutation $\tau \in \mathcal{C}$ with a permutation $\sigma \in \mathcal{D}$.

In this paper, we study the algorithmic complexity of determining whether a given permutation is a merge of a pair of permutations with a prescribed structure. More formally, for a fixed pair of hereditary permutation classes $\mathcal{C}$ and $\mathcal{D}$, we consider the complexity of determining whether a given permutation π belongs to $\mathcal{C} \odot \mathcal{D}$.

The notion of merge has been originally introduced as an approach for the enumeration of pattern-avoiding permutations [5, 4]. For instance, Claesson et al. [13] have shown that every 1324-avoiding permutation can be obtained by merging a 132-avoiding permutation with a 213-avoiding one, and this result, and its subsequent strengthenings by Bóna [8, 9] and Bevan et al. [7], are the basis of the best known upper bounds for the number of 1324-avoiding permutations.

Apart from enumeration questions, the research into permutation merges has also addressed structural issues, such as whether a given permutation class can be obtained by merging two of its proper subclasses [18, 17], or which classes can be obtained by merging a bounded number of permutations from a given class [3, 19, 21]. Our paper is, however, the first to address algorithmic aspects of permutation merges.

So far, most of the algorithmic research related to permutations has focused on the decision problem known as *Permutation Pattern Matching*, or PPM, where the goal is to determine whether a given permutation π (the 'pattern') is contained in a permutation τ (the 'text'). Bose et al. [11] have shown that PPM is NP-complete for general π and τ, but it is polynomial when π is restricted to the class of the so-called separable permutations. The latter result was generalized by Ahal and Rabinovich [2]. More precisely, Ahal and Rabinovich introduced a notion of tree decomposition for permutations and an associated width parameter, closely related to the concept of tree-width from graph theory; they then proved that PPM is polynomial when the pattern π is restricted to a class of bounded tree-width, of which separable permutations are a special case. We remark that a different width parameter for permutations was introduced by Guillemot and Marx [16], who used it to prove that PPM is in FPT with the length of the pattern π as the parameter. While the results on the complexity of PPM do not have any immediate consequences for the problems we consider in this paper, the tree-width concept of Ahal and Rabinovich is a crucial ingredient in our results.

The decision problem of recognizing permutations from $\mathcal{C} \odot \mathcal{D}$ can be viewed as a permutation analogue of the generalized graph coloring problem from graph theory. For a fixed k-tuple $\mathcal{G}_1, \ldots, \mathcal{G}_k$ of graph classes, a generalized coloring of a graph is an assignment of colors $1, 2, \ldots, k$ to its vertices so that the vertices of color i induce a subgraph from $\mathcal{G}_i$. In particular, if all the $\mathcal{G}_i$ are equal to the class of edgeless graphs, this notion reduces to the classical notion of k-coloring. The research into the complexity of generalized graph coloring was initiated by Rutenburg [20], who considered graph properties defined by a finite set of forbidden subgraphs. Later, Farrugia [15] has shown that if all the $\mathcal{G}_i$ are hereditary and additive (i.e., closed under taking induced subgraphs and forming disjoint unions) then the problem is NP-hard, except the trivially polynomial case when $k = 2$ and both $\mathcal{G}_1$ and $\mathcal{G}_2$ are equal to the class of edgeless graphs. Further results in this area were obtained, e.g., by Brown [12], Alexeev et al. [6], Achlioptas et al. [1], or Borowiecki [10].

As with generalized graph coloring, the recognition of permutation merges admits several cases which are trivially polynomial. For instance, let $\mathcal{I}_k$ be the class of permutations that can be merged from at most k increasing subsequences, or equivalently, of permutations that avoid the pattern $k+1, k, \dots, 1$. Similarly, let $\mathcal{D}_k$ be the permutations merged from at most k decreasing subsequences, which are exactly the avoiders of $1, 2, \dots, k+1$. One may easily see that $\mathcal{I}_k \odot \mathcal{I}_\ell = \mathcal{I}_{k+\ell}$ and $\mathcal{D}_k \odot \mathcal{D}_\ell = \mathcal{D}_{k+\ell}$, and in particular, these merges are trivially polynomially recognizable. Moreover, Kézdy et al. [19] have shown that for any $k, \ell \geq 1$, the class $\mathcal{I}_k \odot \mathcal{D}_\ell$ has only finitely many minimal excluded patterns, and therefore these classes are polynomially recognizable as well.

Ekim et al. [14] studied the complexity of generalized 2-colorings when the input graph is restricted to the class of the so-called permutation graphs. Their results, in our terminology, imply the polynomial recognition of $\mathcal{L} \odot \mathcal{I}_1$ and of $\mathcal{L} \odot \overline{\mathcal{L}}$, where $\mathcal{L}$ and $\overline{\mathcal{L}}$ denote the classes of layered and co-layered permutations (see Section 2 for definitions).

Our results

In this paper, we show that there are many more cases of polynomially tractable merges of permutation classes. This contrasts with Farrugia's above-mentioned result on generalized graph coloring. As our main results, we will present two general approaches to show that a permutation class of the form $\mathcal{C} \odot \mathcal{D}$ is polynomially recognizable.

Our first approach, which we present in Section 3.1, is based on the concept of non-deterministically logspace on-line recognizable (or NLOL-recognizable) permutation classes, which we introduce. We will show that an arbitrary merge of NLOL-recognizable classes is polynomially recognizable. While this approach is conceptually quite simple, it generalizes all the previously known examples of tractable merges following from the work of Kézdy et al. [19] and Ekim et al. [14].

For our second approach, presented in Section 4, we introduce the notion of grid decompositions, and the associated width parameter called grid-width. We combine the grid decomposition technique with a restricted version of NLOL, called 2D-NLOL, to prove that the merge of a 2D-NLOL-recognizable class with a class of bounded grid-width can be recognized in polynomial time. This approach allows us to handle further cases of natural permutation classes that are not NLOL-recognizable, such as the class of 213-avoiders, or the class of separable permutations.

To complement our tractability results, we also provide, in Section 5, an NP-hardness result. The result implies, among other examples, that the recognition of $\mathrm{Av}(2413) \odot \mathrm{Av}(2413)$ is NP-hard, where $\mathrm{Av}(2413)$ is the class of 2413-avoiding permutations.

2 Basic definitions

A *permutation of order* n is a sequence in which each element of the set $[n]$ appears exactly once. We let $\mathcal{S}_n$ denote the set of permutations of order n. When writing out short permutations explicitly, we shall omit all punctuation and write, e.g., 15342 for the

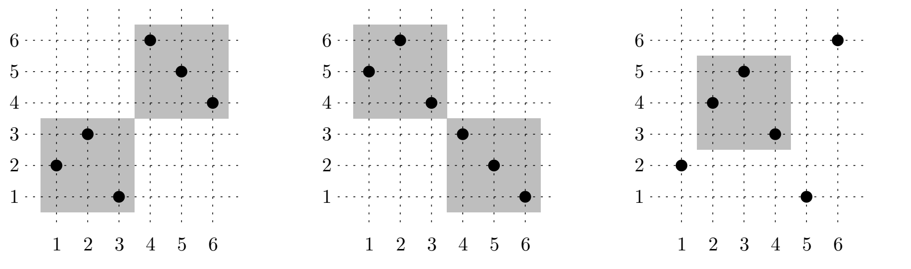
6
5
4
3
2
1
1 2 3 4 5 6
$231 \oplus 321 = 231654$
$231 \ominus 321 = 564321$
the '3' of 2314 inflated by 231

Figure 1 Example of direct sum (left), skew sum (center) and inflation (right).

permutation $1, 5, 3, 4, 2 \in \mathcal{S}_5$. We shall assume that there is a unique permutation of order 0, corresponding to the empty sequence.

To represent a permutation $\pi = \pi_1, \pi_2, \ldots, \pi_n$ graphically, we will use the *permutation diagram*, which is the set of points $\{(i, \pi_i);\ i \in [n]\}$ in the plane. See Figure 1 for an example. Note that we use Cartesian coordinates, that is, the first row of the diagram is at the bottom.

Let $x = x_1, x_2, \ldots, x_n$ and $y = y_1, y_2, \ldots, y_n$ be two sequences of numbers. We say that x and y are *order-isomorphic*, if, for every $1 \le i, j \le n$ we have $x_i < x_j \iff y_i < y_j$. A permutation $\pi \in \mathcal{S}_n$ *contains* a permutation $\sigma \in \mathcal{S}_k$, if π has a subsequence order-isomorphic to σ. Such a subsequence is then an *occurrence* (or a *copy*) of σ in π. If π does not contain σ, we say that π *avoids* σ.

A *hereditary permutation class* (or just *permutation class*, for short) is a set $\mathcal{C}$ of permutations with the property that if π is in $\mathcal{S}$, then all the permutations contained in π are in $\mathcal{C}$ as well. For a permutation σ, we let $\mathrm{Av}(\sigma)$ denote the set of σ-avoiding permutations. More generally, for a set F of permutations, we let $\mathrm{Av}(F)$ be the set of permutations that avoid all the elements of F. Clearly, $\mathrm{Av}(F)$ is a permutation class, and any permutation class is equal to $\mathrm{Av}(F)$ for a (possibly infinite) set F.

Consider a pair of permutations $\sigma = \sigma_1, \ldots, \sigma_k \in \mathcal{S}_k$ and $\tau = \tau_1, \ldots, \tau_\ell \in \mathcal{S}_\ell$. The *direct sum* of σ and τ, denoted $\sigma \oplus \tau$, is the permutation $\pi = \sigma_1, \ldots, \sigma_k, k + \tau_1, k + \tau_2, \ldots, k + \tau_\ell \in \mathcal{S}_{k+\ell}$. Similarly, their *skew sum*, denoted $\sigma \ominus \pi$, is the permutation $\ell + \sigma_1, \ldots, \ell + \sigma_k, \tau_1, \tau_2, \ldots, \tau_\ell \in \mathcal{S}_{k+\ell}$; see Figure 1.

For a pair of permutation classes $\mathcal{C}$ and $\mathcal{D}$, we let $\mathcal{C} \oplus \mathcal{D}$ be the set $\{\sigma \oplus \tau;\ \sigma \in \mathcal{C}, \tau \in \mathcal{D}\}$; note that this is again a permutation class. The class $\mathcal{C} \ominus \mathcal{D}$ is defined analogously. The *sum-closure* of a class $\mathcal{C}$, denoted $\mathcal{C}^{\oplus}$, is the class of all the permutations that can be obtained as a direct sum of finitely many members of $\mathcal{C}$; the *skew-closure* $\mathcal{C}^{\ominus}$ is defined analogously.

A permutation π is a *merge* of permutations σ and τ if we can color the elements of π with colors red and blue so that the red elements are order-isomorphic to σ and the blue ones to τ. The merge of a class $\mathcal{C}$ and a class $\mathcal{D}$ is the class $\mathcal{C} \odot \mathcal{D}$ of permutations that can be obtained by merging an element of $\mathcal{C}$ with an element of $\mathcal{D}$.

For integers i and j, we let $[i, j]$ denote the set $\{k \in \mathbb{Z};\ i \le k \le j\}$. A set of this form is an *integer interval.* We also use the notation $[i, j)$ for the interval $[i, j-1]$ and $(i, j]$ for $[i+1, j]$. A *box* is the Cartesian product of two integer intervals. For a box $B = I \times J \subseteq [n] \times [n]$ and a permutation $\pi = \pi_1, \ldots, \pi_n$, the *restriction* of π to B, denoted $\pi|_B$, is the subsequence of π formed by the entries π_i satisfying $i \in I$ and $\pi_i \in J$.

Let $\sigma = \sigma_1, \ldots, \sigma_k$ and $\tau = \tau_1, \ldots, \tau_\ell$ be again a pair of nonempty permutations. The *inflation* of an element σ_i of σ by τ, is an operation which produces a permutation

$$\pi = \sigma'_1, \sigma'_2, \ldots, \sigma'_{i-1}, \tau'_1, \tau'_2, \ldots, \tau'_\ell, \sigma'_{i+1}, \ldots, \sigma'_k,$$

where the subsequence $\tau'_1, \tau'_2, \dots, \tau'_\ell$ is a copy of τ, and for any $j \in \ell$, the subsequence $\sigma'_1, \sigma'_2, \dots, \sigma'_{i-1}, \tau'_j, \sigma'_{i+1}, \dots, \sigma'_k$ is a copy of σ; see again Figure 1. A permutation is *simple*, if it cannot be obtained from two strictly smaller permutations by an inflation. For instance, the permutation 25314 is simple, while 25341 is not, since it can be obtained, e.g., by inflating the element '3' in 2431 by the permutation 12.

For a permutation $\pi = \pi_1, \dots, \pi_n$ the *reverse* of π is the permutation $\pi_n, \pi_{n-1}, \dots, \pi_1$, the *complement* of π is the permutation $n+1-\pi_1, n+1-\pi_2, \dots, n+1-\pi_n$, and the *inverse* of π is the permutation $\sigma = \sigma_1, \dots, \sigma_n$ satisfying $\pi_i = j \iff \sigma_j = i$. We let π^r, π^c and π^{-1} denote the reverse, complement and inverse of π, respectively. Similarly, for a class of permutations $\mathcal{C}$, we let $\mathcal{C}^r$ denote the set $\{\pi^r;\ \pi \in \mathcal{C}\}$, and similarly for $\mathcal{C}^c$ and $\mathcal{C}^{-1}$. Note that $\mathcal{C}^r$, $\mathcal{C}^c$ and $\mathcal{C}^{-1}$ are again permutation classes.

Several commonly encountered permutation classes have standard names in the literature. The *increasing permutations* are the permutations from the class $\mathrm{Av}(21)$ and symmetrically, the elements of $\mathrm{Av}(12)$ are the *decreasing permutations*. The permutations avoiding both 231 and 312 are known as the *layered permutations*. Layered permutations can also be characterized as those permutations that can be written as a finite direct sum in which each summand is a decreasing permutation; that is, the class of layered permutations is the sum-closure of $\mathrm{Av}(12)$. The complements of layered permutations are known as the *co-layered permutations*; they form the class $\mathrm{Av}(\{132, 213\})$. Finally, the permutations from the class $\mathrm{Av}(\{2413, 3142\})$ are known as the *separable permutations*; it is known [11] that these are precisely the permutations that can be created from the permutation of size 1 by direct sums and skew sums.

3 Tractable merges

For a permutation class $\mathcal{C}$, *$\mathcal{C}$-recognition* is the decision problem to determine whether a given permutation belongs to $\mathcal{C}$. Our main goal is to identify pairs of classes $\mathcal{C}, \mathcal{D}$ for which the $(\mathcal{C} \odot \mathcal{D})$-recognition problem is tractable, i.e., solvable in polynomial time.

3.1 NLOL-recognizable classes

Our first nontrivial example of classes whose merges can be efficiently recognized are the so-called NLOL-recognizable permutation classes. Informally speaking, a permutation class is NLOL-recognizable if its members can be recognized by a single-pass nondeterministic streaming algorithm with logarithmic memory.

More formally, we say that a permutation class $\mathcal{C}$ is *nondeterministically logspace on-line* recognizable, or NLOL-recognizable for short, if there is a nondeterministic algorithm A that recognizes $\mathcal{C}$ in the following setting: as the first part of the input, the algorithm A receives a number n, which is an upper bound on the length and also on the largest value in the input sequence. The algorithm is then given access to $O(\log n)$ bits of memory, and it receives a sequence of distinct values $\pi_1, \dots, \pi_k$ from the set $[n]$, terminated by a special symbol `EOF`. Upon receiving the `EOF` symbol, A answers whether the input sequence is order-isomorphic to a permutation in $\mathcal{C}$. The algorithm can store arbitrary data of size $O(\log n)$ in its memory, but as soon as it reads the input value π_i, it can no longer access the previous values of the input. A is nondeterministically recognizing $\mathcal{C}$ in the sense that the input sequence is order-isomorphic to a permutation in $\mathcal{C}$ if and only if at least one computation of A accepts it. The algorithm A is then called an NLOL-recognizer of $\mathcal{C}$. Note that the input sequence is guaranteed to consist of distinct values, so the NLOL-recognizer itself does not need to verify this property. This also implies that the input sequence has length at most n.

We let NLOL denote the set of the NLOL-recognizable permutation classes. Clearly, for any permutation class $\mathcal{C} \in$ NLOL, the $\mathcal{C}$-recognition problem is tractable, since nondeterministic logspace computations can be simulated in polynomial time.

One may easily observe that NLOL contains any finite permutation class, as well as the classes Av(12) and Av(21). The key feature of NLOL is that it is closed under many important operations with permutation classes, including the merge operation.

▶ **Lemma 1.** *If $\mathcal{C}$ and $\mathcal{D}$ are NLOL-recognizable classes, then the following classes are NLOL-recognizable as well:*
(a) *The classes $\mathcal{C} \cap \mathcal{D}$ and $\mathcal{C} \cup \mathcal{D}$.*
(b) *The classes $\mathcal{C}^r$ and $\mathcal{C}^c$.*
(c) *The classes $\mathcal{C}^\oplus$ and $\mathcal{C}^\ominus$, i.e., the sum-closure and skew-closure of $\mathcal{C}$.*
(d) *The classes $\mathcal{C} \oplus \mathcal{D}$ and $\mathcal{C} \ominus \mathcal{D}$.*
(e) *The class $\mathcal{C} \odot \mathcal{D}$.*

▶ **Corollary 2.** *For any sequence of classes $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_k \in$ NLOL, the class $\mathcal{C}_1 \odot \mathcal{C}_2 \odot \cdots \odot \mathcal{C}_k$ is in NLOL, and therefore polynomially recognizable.*

Lemma 1 shows that NLOL contains many important permutation classes, including the classes of layered and co-layered permutations, as well as any class of the form $\mathrm{Av}(1, 2, 3 \dots, k)$ or $\mathrm{Av}(k, k-1, \dots, 1)$.

On the negative side, it can be shown that NLOL does not contain some other important classes, such as the class of separable permutations, or its subclasses Av(231), Av(213), Av(312) and Av(132). These five classes share a common feature: their elements have a simple recursive tree-like structure involving direct sums, skew sums and inflations. We shall soon formalize this notion of tree-like structure via the concept of bounded grid-width, and show that it leads to another general type of tractable merges. Before we get there, however, we first introduce a restricted form of NLOL that will play an important part in conjunction with bounded grid-width classes.

3.2 2D-NLOL-recognizable classes

Informally speaking, a permutation class is 2D-NLOL-recognizable, if its members can be recognized by a single-pass nondeterministic streaming algorithm over a sequence of index-value pairs in a left-to-right, bottom-to-top order.

Let $P = \{(x_1, y_1), \dots, (x_n, y_n)\}$ be a set of points in the plane. We say that P is *in general position* if no two of its points are on the same horizontal or vertical line, i.e., there is no $i \neq j$ with $x_i = x_j$ or $y_i = y_j$. We say that a permutation $\pi \in \mathcal{S}_n$ is *shape-isomorphic* to P if there is a bijection $f \colon [n] \to [n]$ such that for every i and j the following holds: $i < j$ if and only if $x_{f(i)} < x_{f(j)}$ and $\pi_i < \pi_j$ if and only if $y_{f(i)} < y_{f(j)}$. Note that a permutation $\pi \in \mathcal{S}_n$ is shape-isomorphic to its diagram $\{(i, \pi_i);\ i \in [n]\}$.

Let $(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)$ be a sequence of distinct points in general position. We say that the sequence is *top-right monotone* if for every $i \in [k]$ the point (x_i, y_i) is to the right or above all the previous points of the sequence; formally, for every $i \in [k]$, either for every $j < i$ we have $x_j < x_i$ or for every $j < i$ we have $y_j < y_i$. Note that there can be several top-right monotone sequences corresponding to a single point set. Note also, that a sequence $(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k) \subseteq [n] \times [n]$ in general position is top-right monotone if and only if for every $i \in [k]$ there is a box $B_i = [1, r_i] \times [1, t_i]$ which contains the points $(x_1, y_1), \dots, (x_i, y_i)$ but none of the points $(x_{i+1}, y_{i+1}), \dots, (x_k, y_k)$. A sequence of points is *admissible* if it is in general position and top-right monotone.

We will now consider nondeterministic logspace algorithms that receive an integer n, followed by an admissible sequence of points in $[n] \times [n]$ as their input. To describe the assumptions we make about the algorithms, we first introduce some terminology. Let A be such nondeterministic algorithm. A *position* of the algorithm A is a pair (p, m), where $p = (p_x, p_y)$ is a point in $[n+1] \times [n+1]$ and m is a memory state of A. Let S be an admissible sequence of points. We say that A *can reach the position* (p, m) *on input* S, if S is contained in the box $[1, p_x) \times [1, p_y)$ and there is a computation of A starting from its initial state and ending in state m after processing S. Let $p = (p_x, p_y)$ and $p' = (p'_x, p'_y)$ be two points with $1 \leq p_x \leq p'_x \leq n$ and $1 \leq p_y \leq p'_y \leq n$, and S be an admissible sequence. We say that A can *reach position* (p', m') *from position* (p, m) *on input* S, if S is contained in the box $[1, p'_x) \times [1, p'_y)$ but disjoint from the box $[1, p_x) \times [1, p_y)$, and the algorithm A has a computation starting in state m and ending in state m' after processing S.

We say that an algorithm A is *order-oblivious* if it has the following property: for any pair of positions (p, m) and (p', m') with $p \leq p'$, and for any pair of admissible sequences S and S' that correspond to two top-right monotone orderings of the same point set, A can reach (p', m') from (p, m) on input S if and only if it can reach (p', m') from (p, m) on input S'. Informally speaking, the state reached by an order-oblivious algorithm only depends on the set of points it has received as input, but not on their ordering. We may therefore say, e.g., that A reaches position (p, m) on a set of points P, without specifying the particular ordering of P, with the assumption that the ordering is top-right monotone.

We say that an order-oblivious algorithm A is *box-coherent* if it has the following property: for any indices $i \leq i'$ and $j \leq j'$, consider the four points $p^{\llcorner} = (i, j)$, $p^{\ulcorner} = (i, j')$, $p^{\lrcorner} = (i', j)$ and $p^{\urcorner} = (i', j')$ and four corresponding memory states $m^{\llcorner}$, $m^{\ulcorner}$, $m^{\lrcorner}$ and $m^{\urcorner}$. Suppose that A can reach the position $(p^{\lrcorner}, m^{\lrcorner})$ from $(p^{\llcorner}, m^{\llcorner})$ on an input $X \subseteq [i, i') \times [1, j)$, and that it can reach $(p^{\ulcorner}, m^{\ulcorner})$ from $(p^{\llcorner}, m^{\llcorner})$ on an input $Y \subseteq [1, i) \times [j, j')$. Let Z be a subset of $[i, i') \times [j, j')$ such that $X \cup Y \cup Z$ is in general position. Let $(p^{\urcorner}, m^{\urcorner})$ be a position reachable from $(p^{\lrcorner}, m^{\lrcorner})$ on input $Y \cup Z$. Then the reachability of $(p^{\urcorner}, m^{\urcorner})$ from $(p^{\ulcorner}, m^{\ulcorner})$ on input $X \cup Z$ only depends on the four states $m^{\llcorner}, m^{\lrcorner}, m^{\ulcorner}, m^{\urcorner}$ and the set Z; in particular, it does not depend on the the set X itself. Symmetrically, if we let $(p^{\urcorner}, m^{\urcorner})$ be a position reachable from $(p^{\ulcorner}, m^{\ulcorner})$ on input $X \cup Z$ then the reachability of $(p^{\urcorner}, m^{\urcorner})$ from $(p^{\lrcorner}, m^{\lrcorner})$ on input $Y \cup Z$ only depends on the four memory states and the set Z, but does not depend on Y. Informally, box-coherence means that the memory states $m^{\llcorner}$ and $m^{\lrcorner}$ retain enough information about X to determine the reachable states on inputs of the form $X \cup Z$.

We say that a permutation class $\mathcal{C}$ is *2D nondeterministically logspace on-line* recognizable, or 2D-NLOL-recognizable for short, if there is a nondeterministic order-oblivious box-coherent algorithm A that recognizes $\mathcal{C}$ in the following setting: as the first part of the input, the algorithm A receives a number n, which is an upper bound on the largest value in the input sequence. The algorithm is then given access to $O(\log n)$ bits of memory, and it receives a top-right monotone sequence of points $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$ from $[n] \times [n]$, terminated by a special symbol `EOF`. Upon receiving the `EOF` symbol, the algorithm answers whether the input sequence is shape-isomorphic to a permutation in $\mathcal{C}$. The algorithm can store arbitrary data of size $O(\log n)$ in its memory, but after it reads the value (x_i, y_i) from the input, it cannot access any of the previous values. A is nondeterministically recognizing $\mathcal{C}$ in the sense that the input sequence is shape-isomorphic to a permutation in $\mathcal{C}$ if and only if at least one computation of A accepts it. The algorithm A is then called a 2D-NLOL-recognizer of $\mathcal{C}$. Note that the algorithm A does not have to verify that the input is in general position; in other words, on inputs that fail this condition, the behavior of A can be arbitrary.

We let 2D-NLOL denote the set of the 2D-NLOL-recognizable permutation classes. Clearly, 2D-NLOL is contained in NLOL: the left-to-right ordering is a special case of a top-right monotone ordering, and any 2D-NLOL-recognizer can be trivially transformed into an NLOL-recognizer. Furthermore, we observe that 2D-NLOL contains any finite permutation class, as well as the classes Av(12) and Av(21). And like NLOL, 2D-NLOL is closed under many important operations with permutation classes, including the merge operation.

▶ **Lemma 3.** *If $\mathcal{C}$ and $\mathcal{D}$ are 2D-NLOL-recognizable classes, then the following classes are 2D-NLOL-recognizable as well:*

(a) *The classes $\mathcal{C} \cap \mathcal{D}$ and $\mathcal{C} \cup \mathcal{D}$.*
(b) *The class $\mathcal{C}^{-1}$, which contains the inverses of the permutations of $\mathcal{C}$.*
(c) *The classes $\mathcal{C}^{\oplus}$ and $\mathcal{C}^{\ominus}$, i.e., the sum-closure and skew-closure of $\mathcal{C}$.*
(d) *The classes $\mathcal{C} \oplus \mathcal{D}$ and $\mathcal{C} \ominus \mathcal{D}$.*
(e) *The class $\mathcal{C} \odot \mathcal{D}$.*

4 Grid-width

Let us introduce a decomposition of permutations and a corresponding width parameter, which are suited for describing various algorithms using dynamic programming.

An *interval family* I is a set of pairwise disjoint integer intervals with the natural ordering $I_1, \ldots, I_n$ such that for $j < k$, $I_j < I_k$. For two interval families $\mathcal{I}$ and $\mathcal{J}$, let $\mathcal{I} \times \mathcal{J}$ denote the naturally defined set of boxes in the plane. For a point set A in the plane, let $x(A)$ denote its projection on the x-axis and equivalently $y(A)$ its projection on the y-axis. The *intervalicity* of a set $A \subseteq [n]$, denoted by $I(A)$, is the size of the smallest interval family whose union is equal to A.

A *grid tree* of a permutation $\pi \in \mathcal{S}_n$ is a rooted binary tree T with n leaves, each leaf being labeled by a distinct point of the permutation diagram $\{(i, \pi_i);\ i \in [n]\}$. Let S_v^T denote the point set of the labels on the leaves in the subtree of T rooted in v. The *grid-width* of a vertex v in T is the maximum of the intervalicities $I(x(S_v^T))$ and $I(y(S_v^T))$, and the *grid-with* of T, denoted by $\mathrm{gw}^T(\pi)$, is the maximum grid-width of a vertex of T. Finally, the *grid-width* of a permutation π, denoted by $\mathrm{gw}(\pi)$, is the minimum of $\mathrm{gw}^T(\pi)$ over all grid trees T of π.

It can be shown, by using the ideas of Ahal and Rabinovich [2], that the grid-width of a permutation π corresponds, up to a multiplicative constant, to the tree-width of the so-called adjacency graph G_π associated with π. This also implies that grid-width admits an efficient constant-factor approximation.

4.1 GT-recognizable classes

We shall now define a type of class whose recognition problem is tractable on inputs of bounded grid-width. Informally speaking, a class is GT-recognizable if its members can be recognized by a dynamic programming algorithm over their grid tree.

First, let us define an efficient way to encode merging of two interval families. A *merge description of interval families $\mathcal{I}_1$ and $\mathcal{I}_2$ into an interval family $\mathcal{I}$* is a pair (f, g), where

- $f : [|\mathcal{I}_1| + |\mathcal{I}_2|] \to \{1, 2\}$ encodes the interleaving of the intervals of $\mathcal{I}_1$ and $\mathcal{I}_2$, and
- g is a monotone function $[|\mathcal{J}|] \to [|\mathcal{I}_1| + |\mathcal{I}_2|]$ that describes first interval of each consecutive sequence of intervals that merges to a single interval.

Observe that the knowledge of the merge description together with the interval families $\mathcal{I}_1$ and $\mathcal{I}_2$ uniquely determines the resulting interval family $\mathcal{I}$.

For the following definitions, fix a permutation π of length n with a grid tree T. For any vertex v, let $\mathcal{I}_v$ be the unique minimal interval decomposition of $x(S_v^T)$ and similarly let $\mathcal{J}_v$ be the unique minimal interval decomposition of $y(S_v^T)$. Let u be a vertex of T with children v and w. A *merge description of vertex* u is then a pair (M_1, M_2), where

- M_1 is a merge description of the interval families $\mathcal{I}_v$ and $\mathcal{I}_w$ into $\mathcal{I}_u$, and
- M_2 is a merge description of the interval families $\mathcal{J}_v$ and $\mathcal{J}_w$ into $\mathcal{J}_u$.

It is easy to see that the shape of T (omitting the labels on its leaves) together with merge descriptions of its inner vertices uniquely determines both the original T and π. For technical reasons, we now allow grid trees to have unlabeled (empty) leaves, which represent an empty subpermutation. For an empty leaf v, both interval families $\mathcal{I}_v$ and $\mathcal{J}_v$ are just empty sets. We say that T is a *merge-labeled tree* if every inner vertex is labeled with its merge description, every non-empty leaf has label ϵ_0 and every empty leaf has label ϵ_1.

We say that a permutation class $\mathcal{C}$ is *grid tree* recognizable, or GT-recognizable for short, if there is an algorithm A that receives the grid-width g and outputs a tree automaton that recognizes $\mathcal{C}$ over merge-labeled trees of grid-width at most g.

A *tree automaton* over merge-labeled trees is a tuple $\mathcal{A} = (Q, \Delta, F)$, where Q is a set of states, $F \subseteq Q$ is a set of final states, and Δ is a set of transition rules of the form $(M, q_1, q_2) \to q$, for merge description M and states $q_1, q_2 \in Q$, and of rules of the form $\epsilon_i \to q$ for $i \in \{0, 1\}$ and $q \in Q$. A *run* of $\mathcal{A}$ on a merge-labeled tree T is simply T labeled with states from Q such that all the states together with their transitions are consistent with the rules of Δ. A run is *accepting* if its state q in the root of T belongs to the set of final states F.

As with NLOL and 2D-NLOL, the GT-recognizable classes are closed with respect to many important operations.

▶ **Lemma 4.** *If $\mathcal{C}$ and $\mathcal{D}$ are GT-recognizable classes, then the following classes are GT-recognizable as well:*
(a) *The classes $\mathcal{C} \cap \mathcal{D}$ and $\mathcal{C} \cup \mathcal{D}$.*
(b) *The classes $\mathcal{C}^r$, $\mathcal{C}^c$ and $\mathcal{C}^{-1}$.*
(c) *The classes $\mathcal{C}^{\oplus}$ and $\mathcal{C}^{\ominus}$, i.e., the sum-closure and skew-closure of $\mathcal{C}$.*
(d) *The classes $\mathcal{C} \oplus \mathcal{D}$ and $\mathcal{C} \ominus \mathcal{D}$.*
(e) *The class $\mathcal{C} \odot \mathcal{D}$.*
Moreover, it can be shown that any class determined by a finite set of minimal forbidden patterns is GT-recognizable.

Fix an input permutation π. Let A be a 2D-NLOL-recognizer and M its set of memory states. We call a point set E a *grid set* if it can be expressed as $E_x \times E_y$ for some $E_x, E_y \subseteq [n]$. A tuple (E, g) is a *grid set of positions* if E is a grid point set and $g : E \to M$. We say that (E, g) is *consistent* if for any two points $p = (p_x, p_y), r = (r_x, r_y) \in E$ such that $p_x \leq r_x$ and $p_y \leq r_y$, A can reach position $(p, g(p))$ from position $(r, g(r))$. The first lemma claims that if we have a box with prescribed states in the lower left and upper right corner, which constitute a reachable pair, then we can extend it to consistent grid set for arbitrary subgridding of the box. The second simply states that for a consistent grid set, we can exchange contents of any box as long as we do not violate reachability locally.

▶ **Lemma 5.** *Let $E = E_x \times E_y$ be a grid set, $e_1 \leq e_2$ the minimal and maximal element of E_x and $f_1 \leq f_2$ the minimal and maximal element of E_y. Let $m^{\llcorner}, m^{\urcorner} \in A$ be a pair of states such that (e_1, f_1) and (e_2, f_2) are reachable through $m^{\llcorner}$ and $m^{\urcorner}$. Then there is a function $g : E \to M$ such that (E, g) is consistent and moreover $g(e_1, f_1) = m^{\llcorner}$ and $g(e_2, f_2) = m^{\urcorner}$.*

▶ **Lemma 6.** *Let (E, g) be a grid set consistent over some subpermutation π' of permutation π, and $p = (p_1, p_2)$ and $r = (r_1, r_2)$ two its points such that $E \cap [p_1, p_2] \times [r_1, r_2]$ contains only the four points $(p_1, p_2), (r_1, p_2), (p_1, r_2), (r_1, r_2)$. Then replacing the subpermutation $\pi'|_{[p_1,r_1)\times[p_2,r_2)}$ with a different subpermutation σ of π does not violate the consistency property as long as the reachability is preserved for all the pairs among the points $(p_1, p_2), (r_1, p_2), (p_1, r_2), (r_1, r_2)$.*

We may now state and prove our main result.

▶ **Theorem 7.** *If $\mathcal{C}$ is a 2D-NLOL-recognizable class and $\mathcal{D}$ is a GT-recognizable class such that every $\pi \in \mathcal{D}$ has grid-width bounded by g, then $\mathcal{C} \odot \mathcal{D}$ is polynomially recognizable.*

Proof. Let the input be a permutation π of length n, let A be the 2D-NLOL-recognizer of $\mathcal{C}$ and $\mathcal{B}$ be the tree automaton recognizing $\mathcal{D}$ over merge-labeled trees of grid-with at most g. The general outline of our approach is fairly simple, we want to efficiently emulate $\mathcal{B}$ on all the subpermutations of π with grid-width at most g while at the same time simulating A on the remaining elements. Throughout this proof we shall use the color red to color the part belonging to $\mathcal{C}$ and blue for the part belonging to $\mathcal{D}$. Let M denote the set of possible memory states of A during computation on permutation of length n, and let N denote the set of states of $\mathcal{B}$. Observe that the size of M is at most n^c for some constant c and the size of N is at most $f(g)$ for a computable function f.

We shall define a polynomially bounded number of problems that can be effectively solved by recursion. A *problem* is a tuple $(\mathcal{I}, \mathcal{J}, \mathcal{Q}, s)$, where

- $\mathcal{I}$ and $\mathcal{J}$ are interval families of integers in $[n]$ each of size at most g,
- $\mathcal{Q} : \mathcal{I} \times \mathcal{J} \to M^4$ assigns four memory states of A to each pair of the intervals, and
- $s \in N$ is a possible state of the automaton $\mathcal{B}$.

There are at most n^{4g} choices for the intervals, at most n^{4cg^2} choices for the memory states and finally at most $f(g)$ choices for the states of the tree automaton $\mathcal{B}$, which makes the total number of problems at most $f(g)n^{4g+4g^2c}$.

We then say that a problem $(\mathcal{I}, \mathcal{J}, \mathcal{Q}, s)$ is *feasible* if there is a red-blue coloring of the subset of π that lies in the union of $\mathcal{I} \times [n]$ and $[n] \times \mathcal{J}$ with the following properties:

- the permutation π_B corresponding to the blue elements is contained in $\mathcal{I} \times \mathcal{J}$ and moreover, for every $I = [i_1, i_2] \in \mathcal{I}$ it holds that both (i_1, π_{i_1}) and (i_2, π_{i_2}) are colored blue, and similarly for every $J = [j_1, j_2] \in \mathcal{J}$ we have that both $(\pi^{-1}_{j_1}, j_1)$ and $(\pi^{-1}_{j_2}, j_2)$ are colored blue,
- π_B belongs to $\mathcal{D}$ and there exists its grid tree T of grid-width at most g whose root has its minimal interval decompositions identical to $\mathcal{I}$ and $\mathcal{J}$, and moreover, there is a run of the automaton $\mathcal{B}$ over the tree T that assigns the state s to the root of T, and
- for any two intervals $I = [i_1, i_2] \in \mathcal{I}$ and $J = [j_1, j_2] \in \mathcal{J}$ such that $\mathcal{Q}(I, J) = (m^{\llcorner}, m^{\lrcorner}, m^{\ulcorner}, m^{\urcorner})$, the grid set (E, l) that contains the points $(i_1, j_1), (i_2 + 1, j_1), (i_1, j_2 + 1), (i_2 + 1, j_2 + 1)$ with their respective states $m^{\llcorner}, m^{\lrcorner}, m^{\ulcorner}, m^{\urcorner}$, is consistent over the elements of π_R.

Let $m_0 \in M$ be the initial memory state of A, $m_F \in M$ be a memory state corresponding to a permutation in $\mathcal{C}$ and $s \in N$ be a final state of $\mathcal{B}$. We say that a problem $(\mathcal{I}, \mathcal{J}, \mathcal{Q}, s)$ is *initial* if $\mathcal{I}$ and $\mathcal{J}$ contain only single intervals $I = [i_1, i_2]$ and $J = [j_1, j_2]$ and $\mathcal{Q}(I, J) = (m^{\llcorner}, m^{\lrcorner}, m^{\ulcorner}, m^{\urcorner})$ such that the positions $((i_1, j_1), m^{\llcorner})$, $((i_2, j_1), m^{\lrcorner})$ and $((i_1, j_2), m^{\ulcorner})$ are reachable from $((0, 0), m_0)$, and $((n + 1, n + 1), m_F)$ is reachable from $((i_2 + 1, j_2 + 1), m^{\urcorner})$. It follows from the definition that π belongs to $\mathcal{C} \odot \mathcal{D}$ if and only if one of

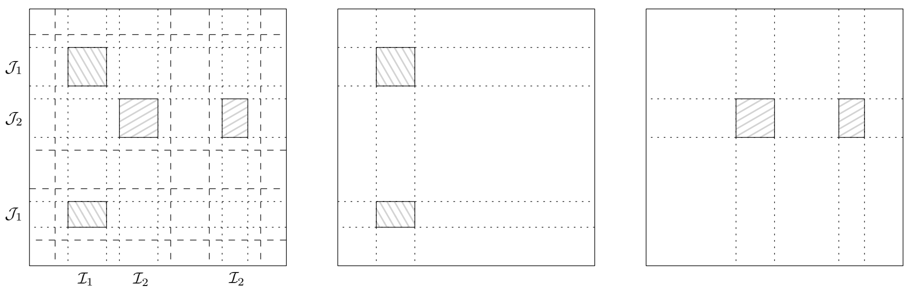

Figure 2 Decomposing problems into subproblems in Feasible. The original problem (left) and its possible subproblems (center and right).

the initial problems is feasible. Thus, we can decide membership if we compute the feasibility of all the initial problems since the additional conditions above are easily checkable.

We describe a recursive algorithm Feasible($\mathcal{I}$, $\mathcal{J}$, $\mathcal{Q}$, s) that takes a problem and either reports unsuccess or outputs some red-blue coloring of π restricted to $\mathcal{I} \times [n] \cup [n] \times \mathcal{J}$ that witnesses its feasibility. If we have a problem where both $\mathcal{I}$ and $\mathcal{J}$ contain only one interval, and the interval is in fact just a single point, then the feasibility of such problem is easily decidable. Otherwise, we recursively call Feasible on a pair of subproblems ($\mathcal{I}_1$, $\mathcal{J}_1$, $\mathcal{Q}_1$, s_1) and ($\mathcal{I}_2$, $\mathcal{J}_2$, $\mathcal{Q}_2$, s_2) with the following properties:

- $\mathcal{I}_1, \mathcal{I}_2$ are two disjoint non-empty interval families whose union is contained in $\mathcal{I}$, and $\mathcal{J}_1, \mathcal{J}_2$ two disjoint interval families whose union is contained in $\mathcal{J}$,
- $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are consistent with $\mathcal{Q}$, and
- $s_1, s_2 \in N$ are arbitrary.

See Figure 2. There are at most n^{4g} choices for the interval families. In order to bound the number of states, observe that we have $8g^2$ positions for the memory states and $f(g)$ states of $\mathcal{B}$, which gives us at most $f(g)^2 n^{8g^2}$ choices. This way, we defined at most $f(g)^2 n^{4g+8cg^2}$ pairs of strictly smaller subproblems and we call Feasible recursively on each of them.

We continue by describing the composition of outputs returned by Feasible on the subproblems $(\mathcal{I}_1, \mathcal{J}_1, \mathcal{Q}_1, s_1)$ and $(\mathcal{I}_2, \mathcal{J}_2, \mathcal{Q}_2, s_2)$ into a coloring returned by Feasible on the problem ($\mathcal{I}$, $\mathcal{J}$, $\mathcal{Q}$, s). If at least one of the recursive calls ends unsuccessfully we move to the next pair. Suppose that both of them are feasible and we have red-blue colorings of π restricted to the union of $\mathcal{I}_\alpha \times [n]$ and $[n] \times \mathcal{J}_\alpha$ for $\alpha \in \{0, 1\}$. Since we are trying to emulate the interval merging of a grid tree, we color all the remaining elements in the union of $\mathcal{I} \times [n]$ and $[n] \times \mathcal{J}$ red. Now we trivially check if the first condition of feasibility holds. In order to satisfy the second condition, it is sufficient to verify that $\mathcal{B}$ contains a transition $((M_1, M_2), s_1, s_2) \to s$ where M_1 and M_2 are the merge descriptions of the interval families $\mathcal{I}_1, \mathcal{I}_2$ into $\mathcal{I}$ and $\mathcal{J}_1, \mathcal{J}_2$ into $\mathcal{J}$. Note that in our case the union $\mathcal{I}_1$ and $\mathcal{I}_2$ might not be equal to $\mathcal{I}$ but we simply define the merge descriptions while forgetting the missing elements (and similarly for $\mathcal{J}$). This check takes at most $h(g)$ time for some computable function h depending on $\mathcal{D}$.

Finally, we need to check the third condition of feasibility. Fix some intervals $I \in \mathcal{I}$ and $J \in \mathcal{J}$ with $\mathcal{Q} = (m^{\llcorner}, m^{\lrcorner}, m^{\ulcorner}, m^{\urcorner})$. Since we have a coloring of $\mathcal{I} \times [n]$ and $[n] \times \mathcal{J}$, it suffices to check whether the corresponding grid set is consistent over the elements of π_R precisely as described in the condition. Simulating the nondeterministic recognizer on fixed input can be done in at most $O(n^{c+1})$ time, thus making the total time spent checking the third condition at most $O(g^2 n^{c+1})$. If all three verifications succeed we output the created coloring.

It follows that the total time spent computing Feasible($\mathcal{I}$, $\mathcal{J}$, $\mathcal{Q}$, s), omitting the recursive calls, is $O(f(g)^3h(g)g^2n^{4g+8cg^2+c+c_2})$ where c_2 is a constant independent of g that captures the time spent per subproblem on enumerating all the possible subproblem pairs and testing the first condition. Therefore, the total time required to solve all the problems is at most $O(f'(g)n^{12cg^2+8g+c+c_2})$ where f' is some computable function.

Whenever Feasible outputs a coloring of an initial problem, we verify all the conditions of feasibility and thus we obtain a coloring that witnesses $\pi \in \mathcal{C} \odot \mathcal{D}$. For the converse, suppose that $\pi \in \mathcal{C} \odot \mathcal{D}$ and we aim to show that we obtain a positive answer on the membership problem. Fix a red-blue coloring witnessing $\pi \in \mathcal{C} \odot \mathcal{D}$. Due to Lemma 5, there is a grid set (E, l) with $E = [n+1] \times [n+1]$ that is consistent with the accepting computation of A over π_R. We say that a problem ($\mathcal{I}$, $\mathcal{J}$, $\mathcal{Q}$, s) is *globally feasible* if the problem is feasible, there is an extension of some feasible coloring to the fixed coloring of the whole π and $\mathcal{Q}$ assigns precisely the memory states prescribed by (E, l). As we mentioned before, if π can be properly colored then there has to be some initial state that is globally feasible. We aim to show that for a globally feasible problem ($\mathcal{I}$, $\mathcal{J}$, $\mathcal{Q}$, s), Feasible successfully outputs a feasible coloring which would therefore imply the correctness of the algorithm.

We prove this by induction on the size of $\mathcal{I}$ and $\mathcal{J}$. For any globally feasible problem with $\mathcal{I}$ and $\mathcal{J}$ such that both $\mathcal{I} \times \mathcal{J}$ contain a single point, Feasible clearly outputs some coloring. For a larger globally feasible problem ($\mathcal{I}$, $\mathcal{J}$, $\mathcal{Q}$, s), we first describe how to split the problem into two specific subproblems that are also globally feasible. The splitting of the interval families is uniquely determined by the fixed coloring of the whole permutation together with the first condition of feasibility. Note that the first condition also ensures that we do not reach leaves of the grid tree corresponding to π_B before the interval families get trivial. Second condition determines the states of the tree automaton and we define $\mathcal{Q}$ to be consistent with the grid set (E, l). It is easy to see that these subproblems are also globally feasible and thus the subsequent calls of Feasible return two partial colorings.

Finally, it remains to argue that it does not matter which feasible colorings we obtain from the recursion. Suppose that the subsequent calls of Feasible returned feasible colorings different from our fixed coloring. However, here we can use the global feasibility together with Lemma 6 to see that we can replace the subpermutations box by box and the consistency is preserved. Therefore, the coloring obtained by joining the feasible colorings of the subproblems satisfies all the conditions and is returned by Feasible. ◀

5 Hard cases of merge-recognition

Let us mention, without going into details, that we can also provide examples of merges $\mathcal{C} \odot \mathcal{D}$ whose recognition problem is NP-hard, even when the classes $\mathcal{C}$ and $\mathcal{D}$ are themselves determined by a single forbidden pattern. Specifically, we can prove the following result.

▶ **Theorem 8.** *For any simple permutation α of order at least 4, the recognition problem for the class $Av(\alpha) \odot Av(\alpha)$ is NP-complete.*

6 Concluding remarks and open problems

The complexity of many cases of ($\mathcal{C} \odot \mathcal{D}$)-recognition remains open. One natural question is to consider the merge of GT-recognizable classes that have bounded grid-width but do not belong to NLOL. Classes of this type include many important examples, such as the class Av(2413, 3142) of separable permutations, or the class Av(213) and its symmetries.

▶ **Open problem 1.** *What is the complexity of* $(\mathcal{C} \odot \mathcal{D})$*-recognition when* $\mathcal{C}$ *and* $\mathcal{D}$ *are any two (possibly identical) classes from the set* $\{Av(2413, 3142), Av(213), Av(231), Av(132), Av(312)\}$*?*

It is also natural to consider 'unbalanced' merges, when one of the two classes is very simple, e.g., the class Av(21) of increasing permutations. Our results imply that $(\mathcal{C} \odot \mathrm{Av}(21))$-recognition is tractable when $\mathcal{C}$ is in NLOL or when $\mathcal{C}$ is a GT class of bounded grid-width, but we know nothing about the remaining cases.

▶ **Open problem 2.** *For which classes* $\mathcal{C}$ *is the* $(\mathcal{C} \odot Av(21))$*-recognition polynomial?*

References

1 D. Achlioptas, J. I. Brown, D. G. Corneil, and M. S. O. Molloy. The existence of uniquely $-G$ colourable graphs. *Discrete Math.*, 179(1-3):1–11, 1998. `doi:10.1016/S0012-365X(97)00022-8`.

2 S. Ahal and Y. Rabinovich. On complexity of the subpattern problem. *SIAM J. Discrete Math.*, 22(2):629–649, 2008. `doi:10.1137/S0895480104444776`.

3 M. Albert and V. Jelínek. Unsplittable classes of separable permutations. *Electron. J. Combin.*, 23(2):Paper 2.49, 20, 2016.

4 M. Albert, J. Pantone, and V. Vatter. On the growth of merges and staircases of permutation classes. arXiv:1608.06969, 2016.

5 M. H. Albert. On the length of the longest subsequence avoiding an arbitrary pattern in a random permutation. *Random Structures Algorithms*, 31(2):227–238, 2007. `doi:10.1002/rsa.20140`.

6 V. E. Alekseev, A. Farrugia, and V. V. Lozin. New results on generalized graph coloring. *Discrete Math. Theor. Comput. Sci.*, 6(2):215–221, 2004.

7 D. Bevan, R. Brignall, A. Elvey Price, and J. Pantone. Staircases, dominoes, and the growth rate of 1324-avoiders. *Electronic Notes in Discrete Mathematics*, 61:123–129, 2017. The European Conference on Combinatorics, Graph Theory and Applications (EUROCOMB'17). `doi:10.1016/j.endm.2017.06.029`.

8 M. Bóna. A new upper bound for 1324-avoiding permutations. *Combin. Probab. Comput.*, 23(5):717–724, 2014. `doi:10.1017/S0963548314000091`.

9 M. Bóna. A new record for 1324-avoiding permutations. *Eur. J. Math.*, 1(1):198–206, 2015. `doi:10.1007/s40879-014-0020-6`.

10 P. Borowiecki. Computational aspects of greedy partitioning of graphs. *J. Comb. Optim.*, 35(2):641–665, 2018. `doi:10.1007/s10878-017-0185-2`.

11 P. Bose, J. F. Buss, and A. Lubiw. Pattern matching for permutations. *Inform. Process. Lett.*, 65(5):277–283, 1998. `doi:10.1016/S0020-0190(97)00209-3`.

12 J. I. Brown. The complexity of generalized graph colorings. *Discrete Appl. Math.*, 69(3):257–270, 1996. `doi:10.1016/0166-218X(96)00096-0`.

13 A. Claesson, V. Jelínek, and E. Steingrímsson. Upper bounds for the Stanley–Wilf limit of 1324 and other layered patterns. *J. Comb. Theory A*, 119:1680–1691, 2012.

14 T. Ekim, P. Heggernes, and D. Meister. Polar permutation graphs are polynomial-time recognisable. *European J. Combin.*, 34(3):576–592, 2013. `doi:10.1016/j.ejc.2011.12.007`.

15 A. Farrugia. Vertex-partitioning into fixed additive induced-hereditary properties is NP-hard. *Electron. J. Combin.*, 11(1):Research Paper 46, 9, 2004.

16 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*, pages 82–101. ACM, New York, 2014. `doi:10.1137/1.9781611973402.7`.

17 V. Jelínek and M. Opler. Splittability and 1-amalgamability of permutation classes. *Discrete Math. Theor. Comput. Sci.*, 19(2):Paper No. 4, 14, 2017.

18 V. Jelínek and P. Valtr. Splittings and Ramsey properties of permutation classes. *Adv. Appl. Math.*, 63:41–67, 2015. doi:10.1016/j.aam.2014.10.003.

19 A. E. Kézdy, H. S. Snevily, and C. Wang. Partitioning permutations into increasing and decreasing subsequences. *J. Combin. Theory Ser. A*, 73(2):353–359, 1996.

20 V. Rutenburg. Complexity of generalized graph coloring. In *Mathematical foundations of computer science, 1986 (Bratislava, 1986)*, volume 233 of *Lecture Notes in Comput. Sci.*, pages 573–581. Springer, Berlin, 1986. doi:10.1007/BFb0016284.

21 V. Vatter. An Erdős-Hajnal analogue for permutation classes. *Discrete Math. Theor. Comput. Sci.*, 18(2):Paper No. 4, 5, 2016.