# An Improved Lower Bound on the Number of Pseudoline Arrangements[*]

## Fernando Cortés Kühnast ✉ [ORCID]
Institute of Mathematics, Technische Universität Berlin, Germany

## Justin Dallant ✉ [ORCID]
Algorithms Research Group, Université libre de Bruxelles, Belgium

## Stefan Felsner ✉ [ORCID]
Institute of Mathematics, Technische Universität Berlin, Germany

## Manfred Scheucher ✉ [ORCID]
Institute of Mathematics, Technische Universität Berlin, Germany

──── **Abstract** ────

Arrangements of pseudolines are classic objects in discrete and computational geometry. They have been studied with increasing intensity since their introduction almost 100 years ago. The study of the number $B_n$ of non-isomorphic simple arrangements of $n$ pseudolines goes back to Goodman and Pollack, Knuth, and others. It is known that $B_n$ is in the order of $2^{\Theta(n^2)}$ and finding asymptotic bounds on $b_n = \frac{\log_2(B_n)}{n^2}$ remains a challenging task. In 2011, Felsner and Valtr showed that $0.1887 \leq b_n \leq 0.6571$ for sufficiently large $n$. The upper bound remains untouched but in 2020 Dumitrescu and Mandal improved the lower bound constant to 0.2083. Their approach utilizes the known values of $B_n$ for up to $n = 12$.

We tackle the lower bound by utilizing dynamic programming and the Lindström–Gessel–Viennot lemma. Our new bound is $b_n \geq 0.2721$ for sufficiently large $n$. The result is based on a delicate interplay of theoretical ideas and computer assistance.

## 1 Introduction

Levi [23] introduced arrangements of pseudolines as a natural generalization of line arrangements in 1926. Ringel studied them in the 1950's and Grünbaum [16] popularized them in his 1972 monograph *Arrangements and Spreads*. In the 1980's Goodman and Pollack initiated a thorough study and related them to many other objects of interest in discrete geometry. For a more detailed account to the history and the relevant references we refer the interested reader to the handbook article [11].

───────────

[*] This article is a merge of the following two independent SoCG'24 submissions:
   1) Justin Dallant – Improved Lower Bound on the Number of Pseudoline Arrangements [7]
   2) Fernando Cortés Kühnast, Stefan Felsner, and Manfred Scheucher – An Improved Lower Bound on the Number of Pseudoline Arrangements [22]. First steps were made in the Bachelor's thesis of the first author [6].
   While in 1) and 2) the estimates $b_n \geq 0.2604$ and $b_n \geq 0.2526$ were shown, we here establish $b_n \geq 0.2721$ by combining our two approaches.
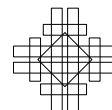
An *arrangement of pseudolines* in the Euclidean plane $\mathbb{R}^2$ is a finite family of simple curves, called pseudolines, such that each curve approaches infinity in both directions and every pair intersects in exactly one point where the two curves cross. More generally, we call a collection of pseudolines *partial arrangement* if every pair intersects in at most one crossing-point. Pseudolines which do not intersect are said to be *parallel*. Note that, while for partial arrangements of *proper* lines the relation 'parallel' is transitive, this is no longer true in partial pseudoline arrangements.

In this article, the focus will be on *simple* arrangements, that is, no three or more pseudolines intersect in a common point (called *multicrossing*). Moreover, we consider all arrangements to be *marked*, that is, they have a unique marked unbounded cell, which is called *north-cell*. Two arrangements are *isomorphic* if one can be mapped to the other by an orientation preserving homeomorphism of the plane that also preserves the north-cell.

While it is known that the number $B_n$ of non-isomorphic arrangements of $n$ pseudolines grows as $2^{\Theta(n^2)}$, it remains a challenging problem to bound the multiplicative factor of the leading term of $\log_2 B_n = \Theta(n^2)$. Determining precise values for small values of $n$ is a challenging task as well, see Table 1. Our focus will be on finding better estimates on the lower bound constant $c^- := \liminf_{n\to\infty} \frac{\log_2 B_n}{n^2}$. One can analogously define the upper bound constant $c^+ := \limsup_{n\to\infty} \frac{\log_2 B_n}{n^2}$. It seems to be open whether $c^+$ and $c^-$ coincide, i.e., whether the limit exists.

■ **Table 1** The known values of $B_n$. Sequence A6245 in the OEIS [28].

| $n$ | $B_n$ | |
| --- | --- | --- |
| 3 | 2 | |
| 4 | 8 | |
| 5 | 62 | |
| 6 | 908 | |
| 7 | 24698 | |
| 8 | 1232944 | |
| 9 | 112018190 | [19, Section 9] |
| 10 | 18410581880 | [10] |
| 11 | 5449192389984 | [29] |
| 12 | 2894710651370536 | [27] |
| 13 | 2752596959306389652 | [18] |
| 14 | 4675651520558571537540 | Yuma Tanaka 2013 |
| 15 | 14163808995580022218786390 | |
| 16 | 76413073725772593230461936736 | Günter Rote 2021 [17] |

A lot of work has been done on finding good estimates for $c^-$ and $c^+$. In the 1980's Goodman and Pollak [15] investigated pseudopoint configurations, which are dual to pseudoline arrangements, and established the lower bound $c^- \geq \frac{1}{8}$. An alternative and slightly easier construction for $c^- \geq \frac{1}{12}$ was given by Matoušek in his *Lectures on Discrete Geometry* [25, Chapter 6]. As pointed out in [8] one also obtains $c^- \geq \frac{1}{8}$ from Matoušek's construction via recursion. Concerning the upper bound, Edelsbrunner, O'Rourke and Seidel [9] showed $c^+ < \infty$ via an algorithm that constructs an arrangement of $n$ pseudolines in $O(n^2)$ time. It is worth noting that an upper bound can also be obtained via planar graphs: The dual graph of arrangement of $n$ pseudolines is planar quadrangulation on $m = \binom{n}{2} + n + 1$ vertices. Since the number of $m$-vertex planar graphs is at most $2^{O(m)}$ [3, 4], it follows that $B_n \leq 2^{O(n^2)}$.

In the 1990's Knuth [19, Section 9] improved the bounds to $c^- \geq \frac{1}{6}$ and $c^+ < 0.7925$, and conjectured that $c^+ \leq 0.5$. The upper bound was lowered to $c^+ < 0.6974$ by Felsner [10]. In 2011, Felsner and Valtr [13] further narrowed the gap by showing $c^- > 0.1887$ and $c^+ < 0.6571$, and, in 2020 Dumitrescu and Mandal [8] proved the currently best lower bound $c^- > 0.2083$.

In this article, we make a substantial step on the lower bound by proving $c^- > 0.2721$.

▶ **Theorem 1.** *The number $B_n$ of non-isomorphic simple arrangements of $n$ pseudolines satisfies the inequality $B_n \geq 2^{cn^2 - O(n \log n)}$ with $c > 0.2721$.*

## 2    Outline

Our approach is in the spirit of several previous bounds. We consider a specific partial arrangement $\mathcal{L}$ of $n$ lines consisting of $k$ *bundles* $\mathcal{L}_1, \ldots, \mathcal{L}_k$ of equally-spaced parallel lines. We then define a class of local perturbations to $\mathcal{L}$ and consider the number of arrangements that can be obtained by these perturbations. This number is a lower bound on $B_n$, and it can be improved by recursively applying the same construction to each of the parallel classes $\mathcal{L}_i$.

The main difference between the approaches lies in the number of bundles $k$ and the choice of locality. Matoušek and also Felsner and Valtr used three bundles but the locality was increased from considering just a triple intersection with its two simple resolutions to the full intersection pattern of three bundles. Dumitrescu and Mandal [8] increased the number $k$ of bundles to 12 but restricted the locality to the possible resolutions of the multicrossings.

Our approach combines higher values of $k$ (we discuss cases with $k$ equal to 4, 6, and 12 in this paper) with an increased locality for the perturbations, i.e., we allow reroutings of sets of pseudolines in designated regions, which we call *patches*. Figure 1 gives an illustration.

When rerouting a partial arrangement within a patch $P$, the order of the crossings along the pseudolines may change. In Section 5.1 we show that the boundary information of $P$ determines which pairs of pseudolines cross within $P$, but the order of crossings along the pseudolines is not determined in general. Outside of $P$, the arrangement remains unaffected, which allows us to count the number of reroutings for each patch independently. The total number of perturbations is obtained as the product of the numbers computed for the individual patches. Details on how we computed the number of possibilities within a patch are given in Section 5 .

To eventually use computer assistance, we choose patches of high regularity and reasonably small complexity. In fact, since the initial arrangement is extremely regular, it is sufficient to determine the rerouting possibilities only for a small number of patch-types. Only a negligible fraction of patches along the boundaries are different. As we only want to find an asymptotic lower bound on $B_n$, the small number of irregular patches along the boundary of the regions will be ignored for the counting.

To eventually prove Theorem 1, we perform the following two steps:

- In the first step (Section 3) we specify the parameters of the construction: For $k = 4$ (Section 3.1), $k = 6$ (Section 3.2), and $k = 12$ (Section 3.3) we construct the initial arrangement with $k$ bundles of $\left\lfloor \frac{n}{k} \right\rfloor$ parallel lines and cover the multicrossing points by patches. By resolving the multicrossing points within the patches, and taking the product over all patches we obtain a lower bound on the number $F_k(n)$ of partial arrangements with $k$ bundles of $\left\lfloor \frac{n}{k} \right\rfloor$ parallel pseudolines.
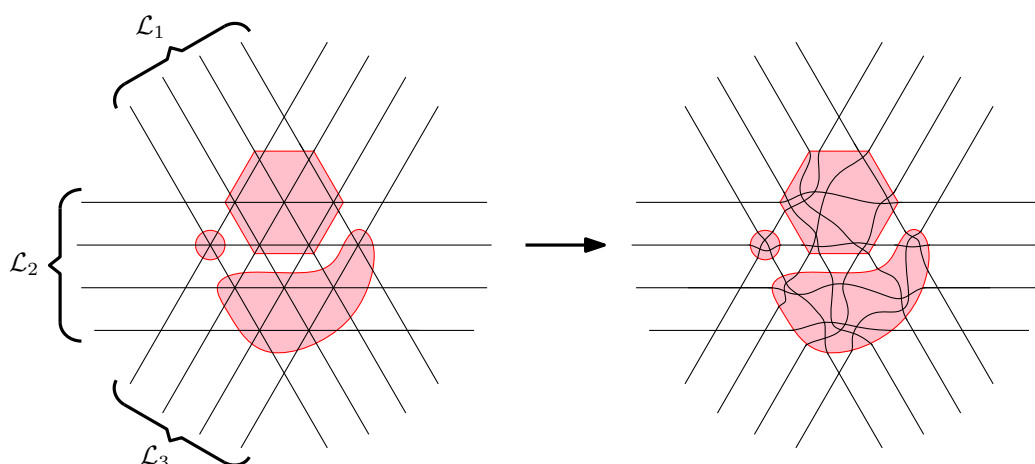
**Figure 1** Left: An arrangement of $k = 3$ bundles of parallel lines and a collection of interior-disjoint patches (highlighted red) such that each multicrossing point is covered by a patch. Right: A partial pseudoline arrangement with the same parallel bundles obtained by rerouting within the patches.

- In the second step (Section 4), we recursively account for crossings in bundles of pseudolines which had been parallel before. The product of the so-computed possibilities yields the improved lower bound on the number $B_n$ of simple arrangements on $n$ pseudolines.

## 3 Step 1: bundles of parallel lines, patches, and perturbations

For the start we fix an integer $k$ and construct an arrangement $\mathcal{L}$ of $k$ bundles of $\left\lfloor \frac{n}{k} \right\rfloor$ parallel lines as in [8]. If $n$ is not an odd multiple of $k$, the remaining lines are discarded, or not used in the counting. We then cover all multicrossing points by a family of disjoint patches, and reroute the line segments within the patches so that all multicrossing points are resolved and the arrangement is simple. Since we use computers it is convenient to construct patches with a high regularity. Moreover, for simplicity in the computational part (cf. Section 5) we ensure that no crossings of $\mathcal{L}$ are located on the boundary of a patch and no patch has a touching with a line.

- **Extremum 1:** If we use one tiny patch for each of the multicrossings, the counting will give the same results as in [8], where each multicrossing was rerouted locally in all possible ways for various configurations with $k$ up to 12.

- **Extremum 2:** If we choose one gigantic patch containing all crossings of $\mathcal{L}$, then all *partial* arrangements of $n$ pseudolines with the same parallel bundles as in $\mathcal{L}$ will be counted, that is $F_k(n)$. For the case of $k = 3$ bundles, Felsner and Valtr [13] determined that $F_3(n) = 2^{cn^2 + o(n^2)}$ where $c = \frac{\log_2(3)}{2} - \frac{2}{3} \approx 0.1258$.

### 3.1 Construction with 4 bundles

First we construct a partial arrangement $\mathcal{L}$ of $n = 4m$ lines, consisting of 4 bundles $\mathcal{L}_1, \ldots, \mathcal{L}_4$ of $m$ parallel lines. See Figure 2 for an illustration.

The construction comes with crossings of order 2, 3, and 4. We restrict our attention to regions with multicrossings since regions with 2-crossings do not allow reroutings. As illustrated on the right-hand side figure, there are two types of regions with multicrossings:
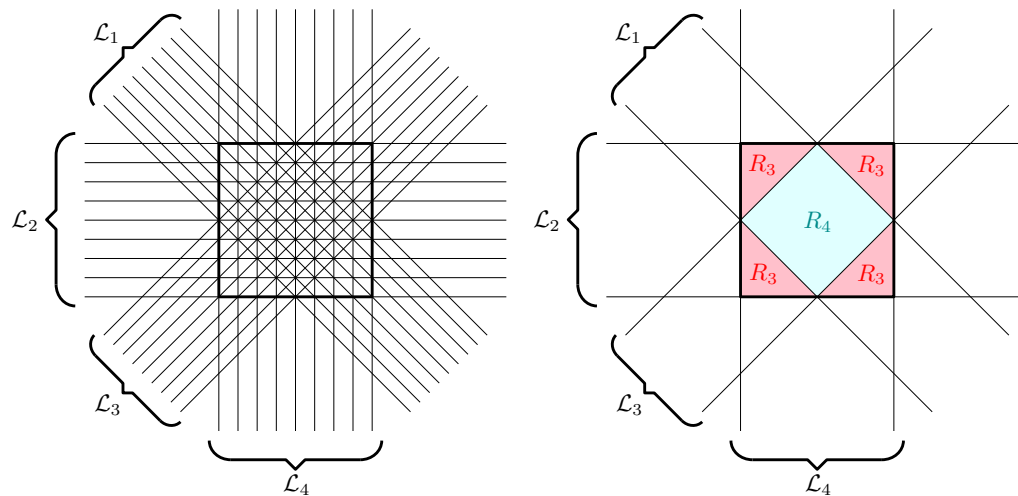
**Figure 2** Construction with 4 bundles as in [8].

- $R_3$ contains multicrossings of degree 3;
- $R_4$ contains multicrossings of degree 4.

We use different patches in the regions $R_3$ and $R_4$ but in both cases rectangular patches can be used to tile the full region apart from a negligible area at its boundary. See Figure 3 for the patch $P_4$ we used to cover $R_4$. Note that each copy of $P_4$ intersects with $\mathcal{L}$ in an equivalent way. This will be true implicitly of all patches we consider from now on. Since $R_3$ has a special pattern of intersection, we were able to compute the number of reroutings using a more efficient method (see Section 5.2) than our more general dynamic programming approach (see Section 5.1). This enabled us to use a square of side length 1000 as our patch $P_3$ for $R_3$, which is much larger geometrically compared to $P_4$.
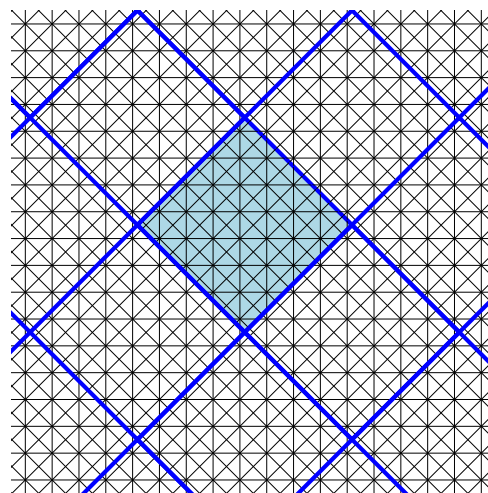


**Figure 3** An illustration of the tiling for region $R_4$ in the 4-slope construction. The patch $P_4$ is a square of side length $8\frac{\sqrt{2}}{2}$ and area 32.

Next we determine the numbers $\mu_i$ of patches of type $P_i$. Since the number of crossings in our construction is asymptotically quadratic in $n$ and each patch contains only a constant number of crossings, the number of patches of type $P_i$ is also quadratic. It is important to

note that the patches along the boundary of $R_i$ behave differently. However, since there are only linearly many of these irregular patches, they only affect the lower order error term. Hence we can omit them in the calculations.

We use a simple area calculation to obtain the $\mu_i$'s. First we fix area$(R_3 \cup R_4) = (m-1)^2$ which implies area$(R_3) =$ area$(R_4) = \frac{(m-1)^2}{2}$. Next we determine the areas of the patches. Since at this point all of our patches are rectangular, this amounts to determining their side lengths. The distance between adjacent lines in $\mathcal{L}_2$ or $\mathcal{L}_4$ is 1 and between those in $\mathcal{L}_1$ or $\mathcal{L}_3$ it is $\frac{\sqrt{2}}{2}$. Thus the areas of the patches are

- area$(P_3) = 1000^2$
- area$(P_4) = (8 \cdot \frac{\sqrt{2}}{2})^2 = 32$.

Now we can easily write down the $\mu_i's$:

- $\mu_3(P_3, n) = \frac{\text{area}(R_3)}{\text{area}(P_3)} - O(n) = \frac{m^2/2}{1000^2} - O(n) = \frac{n^2}{32000000} - O(n)$
- $\mu_4(P_4, n) = \frac{\text{area}(R_4)}{\text{area}(P_4)} - O(n) = \frac{m^2/2}{32} - O(n) = \frac{n^2}{1024} - O(n)$.

To compute the number $F(P_4)$ of all possible reroutings within the patches of type $P_4$, we ran our dynamic program; see Section 5.1 for details. To compute the number $F(P_3)$ of all possible reroutings within the patches of type $P_3$, we made use of the Lindström-Gessel-Viennot lemma; see Section 5.2. We obtained:

- $F(P_3) > 2^{1397192}$
- $F(P_4) = 1023348062661596215589593116398126\,1674$.

Combining the possibilities from the two types of patches according to the estimate $F_k(n) \geq \prod_{i=3}^{k} F(P_i)^{\mu_i(P_i,n)}$ yields

▶ **Proposition 2.** $F_4(n) \geq 2^{cn^2-O(n)}$ with $c > 0.1637$.

By writing $c_i := \lim_{n\to\infty} \frac{\mu_i(n)}{n^2} \cdot \log_2 F(P_i)$, we can see the contributions of the patches $P_3$ and $P_4$ to the leading constant $c = c_3 + c_4$ from Proposition 2; see Table 2.

🟨 **Table 2** Summary of our computational results for each of the patches in the $k = 4$ slope construction. The first entry (marked with a star) is computed using the LGV lemma; see Table 5 for details. The first column shows the name of the region; the second column shows the number of reroutings ($\log_2(F)$); the third column shows the number of patches ($\lim \frac{\mu}{n^2}$); the forth column shows the contribution of the region to the leading constant, i.e., the product of the second and third column; the last column shows the computing time in CPU seconds.

| region | $\log_2(\text{\# of reroutings})$ | \# of patches | contribution | computing time |
|---|---|---|---|---|
| $R_3$* | 1397192.00 | 1/32000000 | 0.04366 | – |
| $R_4$ | 122.94 | 1/1024 | 0.12006 | 20199.30s |
| $\sum$ | – | – | 0.16373 | 20199.30s |

## 3.2 Construction with 6 bundles

In this section we consider a partial arrangement $\mathcal{L}$ of $n$ lines consisting of 6 bundles $\mathcal{L}_1, \ldots, \mathcal{L}_6$ of $m$ parallel lines. See Figure 4 for an illustration.

The construction comes with four types of regions with multicrossings:

- $R_i$ for $i \in \{3, 4, 5\}$ only contains multicrossings of order $i$ and
- $R_6$ contains multicrossings of order 3 and 6.

Note that in contrast to the $k = 4$ bundle construction from Section 3.1, multicrossings of order 3 now occur in $R_3$ and $R_6$.
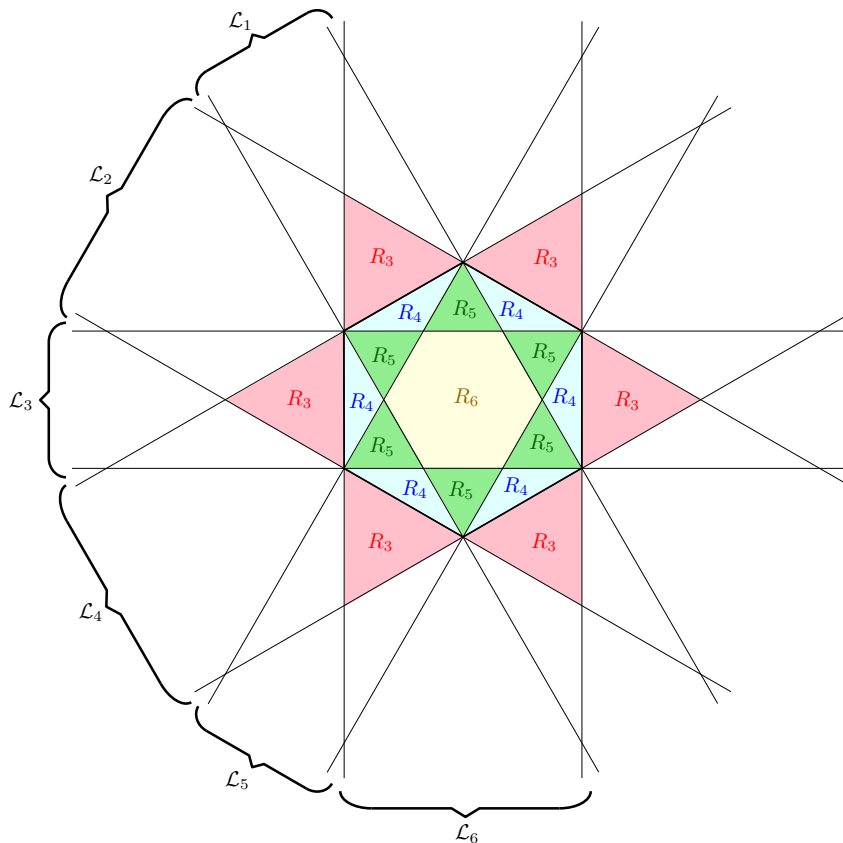
**Figure 4** Construction with 6 bundles as in [8].

For each of the four regions $R_i$ we will use a different type of patch $P_i$ that is based on a regular tiling of the plane to ensure regularity. Figure 5 illustrates the tiling for $R_6$, $R_5$ and $R_4$. For $R_3$ we use a rhombic tiling such that each patch $P_3$ contains exactly $1000 \cdot 1000$ crossings of order 3; see Section 5.2. Pause to note that $P_3$ and $P_4$ are affine transformations of the patches used in the 4 slopes setting in Section 3.1. Therefore they allow the same number of reroutings $F(P_3)$ and $F(P_4)$.

We have to determine the number $\mu_i$ of patches of type $i$. Towards this end we perform a similar area calculation as we did for $k = 4$ bundles. This time both our patches and regions are a little more complex and numerous. To avoid human error, we automated the calculation of both area types in different ways. For the patches, we decided to simply count the number of highest degree crossing within the patch, because that can be done automatically without rounding. There is a one-to-one correspondence between such crossings and the fundamental parallelogram $\sigma$ of the lattice of crossings between lines from the bundles $\mathcal{L}_2, \mathcal{L}_4$ and $\mathcal{L}_6$. Since every patch of the same type contains the same number of such crossings and the patches are part of a potentially infinite tiling, we obtain the area of $P_i$ by simply multiplying the two numbers: $\text{area}(P_i) = \#\{\text{crossings of highest degree}\} \cdot \text{area}(\sigma)$. If we take the area of $\sigma$ to be 1 then the count corresponds exactly to the desired area. We obtain

$$\text{area}(P_3) = 1000^2, \qquad \text{area}(P_4) = 32, \qquad \text{area}(P_5) = 12, \qquad \text{area}(P_6) = 7.$$
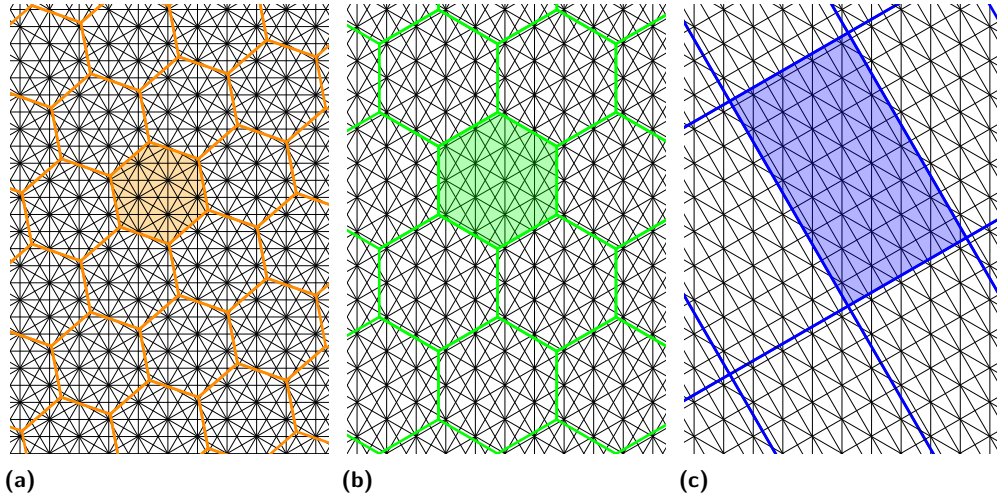
**Figure 5** Three of the four types of patches for our construction on $k = 6$ bundles: (a) For $R_6$ we use a hexagonal tiling such that each patch $P_6$ contains exactly 7 crossings of order 6 and 14 crossings of order 3. (b) For $R_5$ we use a hexagonal tiling such that each patch $P_5$ contains exactly 12 crossings of order 5. (c) For $R_4$ we use a rectangular tiling such that each patch $P_4$ contains exactly $8 \cdot 4 = 32$ crossings of order 4.

Since each region is made up of convex polygons whose supporting lines are known, their areas are easy to compute using computer algebra software. In our supplemental data [1], we provide a Sage script which carries out this task, and can be used for different constructions as well. The areas are

$$\text{area}(R_3) = \tfrac{3(m-1)^2}{4}, \quad \text{area}(R_4) = \tfrac{(m-1)^2}{4}, \quad \text{area}(R_5) = \tfrac{(m-1)^2}{4}, \quad \text{area}(R_6) = \tfrac{(m-1)^2}{4}.$$

Using $n = 6m$ we can calculate the $\mu_i$'s as we did previously:

- $\mu_3(P_3, n) = \frac{\text{area}(R_3)}{\text{area}(P_3)} - O(n) = \frac{3n^2}{144 \cdot 1000^2} - O(n)$
- $\mu_4(P_4, n) = \frac{\text{area}(R_4)}{\text{area}(P_4)} - O(n) = \frac{n^2}{144 \cdot 32} - O(n)$
- $\mu_3(P_5, n) = \frac{\text{area}(R_5)}{\text{area}(P_5)} - O(n) = \frac{n^2}{144 \cdot 12} - O(n)$
- $\mu_3(P_6, n) = \frac{\text{area}(R_6)}{\text{area}(P_6)} - O(n) = \frac{n^2}{144 \cdot 7} - O(n)$.

To compute the numbers $F(P_i)$ of all possible perturbations within the patch type $P_i$ for $i = 4, 5, 6$, we used our dynamic programming approach; see Section 5.1 for more details. For region $F(P_3)$ we were able to use the same number already computed for Section 3.1. The computation was based on the LGV lemma; see Section 5.2. Altogether we computed

- $F(P_3) > 2^{1397192}$
- $F(P_4) = 10233480626615962155895931163981261674$
- $F(P_5) = 322070778554975465081327 40267$
- $F(P_6) = 5489259325039519956333073658900$.

From $F_k(n) \geq \prod_{i=3}^{k} F(P_i)^{\mu_i(n)}$, we can now derive:

▶ **Proposition 3.** $F_6(n) \geq 2^{cn^2 - O(n)}$ *with* $c > 0.2118$.

By writing $c_i := \lim_{n \to \infty} \frac{\mu_i(n)}{n^2} \cdot \log_2 F(P_i)$, we can see the contributions of the patches $P_3$, $P_4$, $P_5$ and $P_6$ to the leading constant $c = c_3 + c_4 + c_5 + c_6$ from Proposition 3; see Table 3.
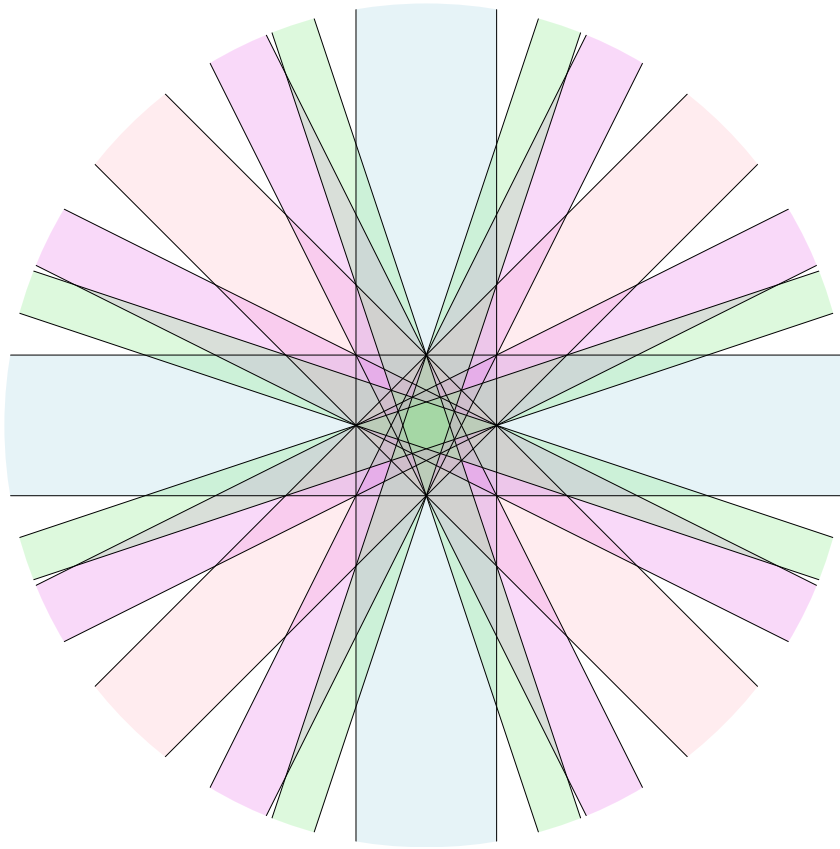
■ **Figure 6** The construction with 12 bundles as in [8].



(a)

(b)
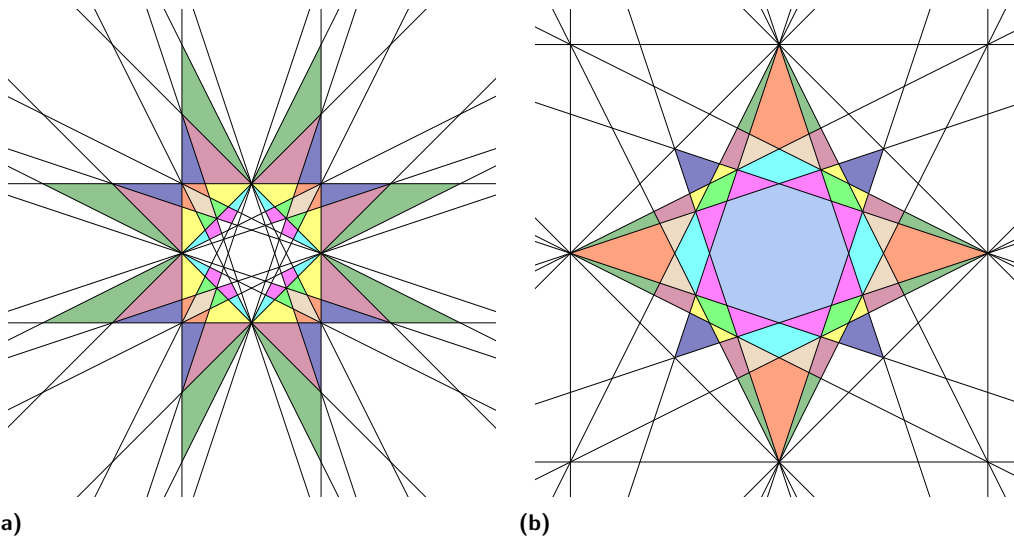
■ **Figure 7** Regions with multi-crossings in the rectangular 12-slope construction.
(a) shows the nine outer regions $R_{3a}, R_{3b}, R_{4a}, R_{4b}, R_{5a}, R_{5b}, R_{6a}, R_{6b}, R_{7a}$.
(b) shows the ten inner regions $R_{7b}, R_{8a}, R_{8b}, R_{8c}, R_{9a}, R_{9b}, R_{10a}, R_{10b}, R_{11}, R_{12}$.

■ **Table 3** Summary of our computational results for each of the patches in the $k = 6$ slope construction. See Table 2 for description of columns.
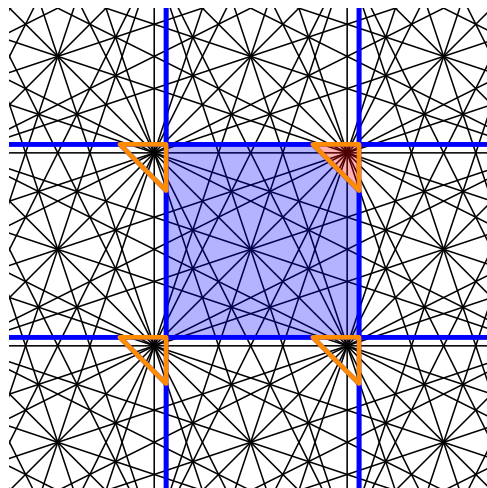
| region | $\log_2(\text{\# of reroutings})$ | # of patches | contribution | computing time |
|---|---|---|---|---|
| $R_3*$ | 1397192.00 | 1/48000000 | 0.02911 | – |
| $R_4$ | 122.94 | 1/4608 | 0.02668 | 18787.90s |
| $R_5$ | 94.70 | 1/1728 | 0.05480 | 4219.49s |
| $R_6$ | 102.11 | 1/1008 | 0.10130 | 868737.00s |
| $\sum$ | – | – | 0.21190 | 891744.39s |



■ **Figure 8** To partition region $R_{12}$ in the $k = 12$ bundle construction, we use a square-shaped patch $P_{12}$ which is further partitioned into two sub-patches.

## 3.3 Rectangular construction with 12 bundles

Finally we consider a partial arrangement $\mathcal{L}$ of $n$ lines consisting of 12 bundles $\mathcal{L}_1, \ldots, \mathcal{L}_{12}$ of $m$ parallel lines. See Figure 6 for an illustration.

Up to symmetries of $\mathcal{L}$, the construction produces 19 regions that contain intersections of at least three bundles, see Figure 7. Since structurally different regions may meet the same numbers of bundles, we differentiate them by adding lowercase Latin letters to their label arbitrarily. Similar as above, we cover each region (except for a small area at its boundary) using a regular tiling and count the numbers of reroutings within each tile using a computer.

For region $R_{12}$ even a $1 \times 1$ square is too complex for our dynamic program to handle in reasonable time/memory. Therefore, we used two sub-patches as illustrated in Figure 8. Multiplying the obtained numbers of reroutings in the two sub-patches yields a lower bound for the $1 \times 1$ square. Note that the smaller patch contains only one crossing of degree 12. Since all pairs of pseudolines cross, the number of reroutings is exactly the number $B_{12}$ of arrangements of 12 pseudolines.

For the other regions we used specific patches which are available in our supplemental data [1]. To obtain the numbers $\mu_{ix}$ of copies of patch $P_{ix}$, we simply divide the area of the whole region $R_{ix}$ by the area of the patch:

$$\mu_{ix} = \frac{\text{area}(R_{ix})}{\text{area}(P_{ix})} - O(n),$$

see Table 4. To compute the areas of the patches we count the number of highest degree crossings they contain again. In this construction, those correspond to $1 \times 1$ squares, so the resulting count will be equal to the patch's area. The areas of the regions can be computed automatically from the specification of the bundles using a Sage script that we provide with our supplemental data [1].

Now we compute the numbers $F(P_{ix})$ of possible reroutings within each patch, see Table 4. To cope with the large patches in the two regions $R_{3a}$ and $R_{3b}$, we again used the Lindström-Gessel-Viennot lemma, see Section 5.2. For all the other patches we used a dynamic programming approach, see Section 5.1. Using the estimate $F_k(n) \geq \prod F(P_{ix})^{\mu_{ix}(n)}$ we derive:

▶ **Proposition 4.** $F_{12}(n) \geq 2^{cn^2 - O(n)}$ with $c > 0.2494$.

By writing $c_{ix} := \lim_{n \to \infty} \frac{\mu_{ix}(n)}{n^2} \cdot \log_2 F(P_{ix})$, we can see the contributions of the patches to the leading constant $c = \sum c_{ix}$ from Proposition 4. All the data discussed above are summarised in Table 4.

■ **Table 4** Summary of our computational results for each of the patches in the $k = 12$ slope construction. See Table 2 for description of columns.

| region | $\log_2(\text{\# of reroutings})$ | # of patches | contribution | computing time |
|---|---|---|---|---|
| $R_{3a}$* | 1397192.00 | 1/144000000 | 0.00970 | – |
| $R_{3b}$* | 1397192.00 | 1/432000000 | 0.00323 | – |
| $R_{4a}$ | 122.94 | 1/6912 | 0.01779 | 18345.70s |
| $R_{4b}$ | 122.94 | 1/69120 | 0.00178 | 18345.70s |
| $R_{5a}$ | 94.70 | 1/6480 | 0.01461 | 4463.95s |
| $R_{5b}$ | 94.70 | 1/17280 | 0.00548 | 4463.95s |
| $R_{6a}$ | 111.79 | 1/17280 | 0.00647 | 98184.90s |
| $R_{6b}$ | 113.74 | 1/17280 | 0.00658 | 38082.70s |
| $R_{7a}$ | 93.74 | 1/7560 | 0.01240 | 1204.09s |
| $R_{7b}$ | 91.12 | 1/20160 | 0.00452 | 8392.94s |
| $R_{8a}$ | 148.77 | 1/24192 | 0.00615 | 148199.00s |
| $R_{8b}$ | 109.24 | 1/15120 | 0.00722 | 57856.50s |
| $R_{8c}$ | 105.70 | 1/6048 | 0.01748 | 142085.00s |
| $R_{9a}$ | 107.55 | 1/30240 | 0.00356 | 8577.60s |
| $R_{9b}$ | 102.98 | 1/10080 | 0.01022 | 316.13s |
| $R_{10a}$ | 144.70 | 1/17280 | 0.00837 | 68991.30s |
| $R_{10b}$ | 70.78 | 1/4320 | 0.01639 | 1430.85s |
| $R_{11}$ | 99.03 | 1/4320 | 0.02292 | 2059.67s |
| $R_{12}$ | 128.89 | 1/1728 | 0.07459 | 47.15s |
| $\sum$ | – | – | 0.24946 | 621047.12s |

## 4 Step 2: resolving parallel bundles

With the second and final step, we obtain a simple arrangement of pairwise intersecting pseudolines from a partial arrangement of $k$ bundles of $m = \lfloor \frac{n}{k} \rfloor$ parallel pseudolines. To do so, we use a recursive scheme as in [13, 8] to make each pair of parallel pseudolines cross: For each $i = 1, \ldots, k$, we consider a disk $D_i$ such that

**(1)** $D_i$ intersects all parallel pseudolines of the bundle $\mathcal{L}_i$ and no other pseudolines, and

**(2)** no two disks overlap.

Within each disk $D_i$ we can place any of the $B_m$ arrangements of $m$ pseudolines. This makes all the pseudolines of a bundle cross. Figure 9 gives an illustration for the case $k = 3$.
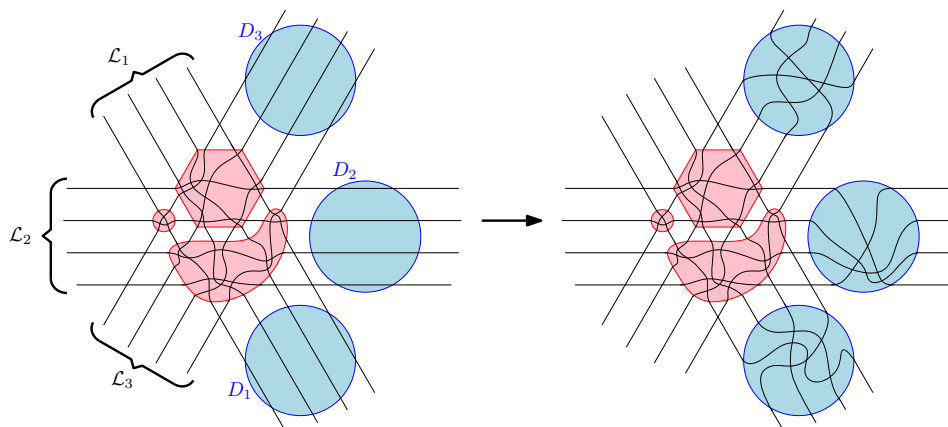


■ **Figure 9** Left: A partial arrangement of 3 bundles of parallel pseudolines and a collection of interior-disjoint disks (highlighted blue) such that each bundle is covered by one disk.
Right: A proper pseudoline arrangement obtained by rerouting within the disks.

Since all $D_i's$ are independent and there are $B_m$ possibilities to reroute within each $D_i$, we obtain the estimate

$$B_n \geq \underbrace{F_k(n)}_{\text{Step 1}} \cdot \underbrace{(B_m)^k}_{\text{Step 2}},$$

where $m = \left\lfloor \frac{n}{k} \right\rfloor$. With the following lemma we can derive $c^- \geq \frac{k}{k-1}c$ where $c$ is the constant obtained in Section 3. The construction with $k = 4$ bundles yields $c^- > 0.2183$, which is already an improvement to the previous best bound by Dumitrescu and Mandal [8]. The construction with $k = 6$ bundles allows a bigger step: It gives the lower bound $c^- > 0.2542$. The biggest improvement comes from the construction with $k = 12$ bundles, which yields $c^- > 0.2721$ and therefore completes the proof of Theorem 1.

▶ **Lemma 5.** *If $F_k(n) \geq 2^{cn^2 - O(n)}$ for some $c > 0$ then $B_n \geq 2^{\frac{k}{k-1}cn^2 - O(n \log n)}$.*

The proof of Lemma 5 is defered to the full version [21].

## 5 Counting all possible reroutings in a patch

In the following we present two approaches how to compute the number $F(P)$ of all possible perturbations within a patch $P$. In Section 5.1 we discuss a general approach based on dynamic programming which can deal with any patch. In Section 5.2 we discuss the special case of 3 slopes, where the reroutings can be derived more efficiently with the Lindström-Gessel-Viennot lemma.

### 5.1 Counting reroutings via dynamic programming

Consider a patch $P$ with boundary curve $C$. Since $C$ has no touching with any line, the intersection of $P$ with any line $L \in \mathcal{L}$ yields proper line segments. And because $C$ passes no crossings all the end points of segments are distinct. We can label the end points on $C$ according to the index of the resulting line segments and since every index occurs twice, we

call the cyclic sequence of end points along $C$ a *bipermutation*. Note that in the case of a non-convex boundary curve $C$, the intersection of a line with $P$ consist of several segments, they have to be distingushed by distinct labels. Figure 10 gives an illustration.
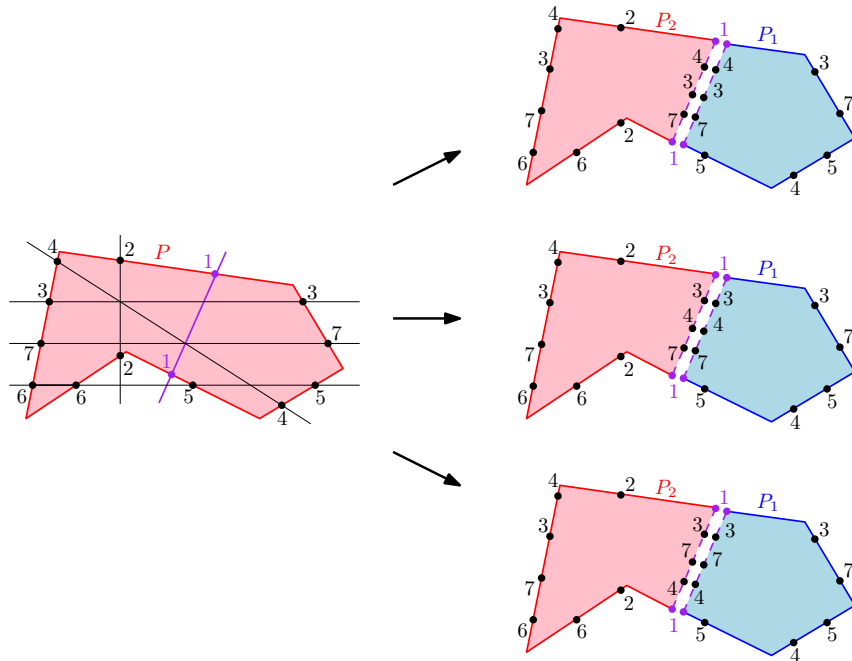


**Figure 10** An illustration of how to recurse on a patch $P$. When cutting along segment 1, highlighted purple, there are intersections with the segments 3, 4, and 7. As the segments 3 and 7 do no cross within $P$, there are only three possibilities for placing the three crossings along the segment 1, namely 4–3–7 (right top), 3–4–7 (right center) and 3–7–4 (right bottom).

The number of possible reroutings within a patch $P$ is computed recursively: Choose a segment $z$, it splits $P$ into two parts. For every segment $a \neq z$ we can determine whether $a$ crosses $z$ by looking at the occurrences of $a$ and $z$ in the bipermutation. If up to cyclic shifts the pattern is $azaz$ they cross in $P$, if the pattern is $aazz$ they are parallel in $P$.

Now for every legal order $\pi$ of the crossing segments on $z$ we combine the two arcs of the bipermutation defined by the two occurrences of $z$ and combine them with $\pi$ to obtain the bipermutations of the two smaller patches $P_1$ and $P_2$. Figure 10 gives an example.

The legal orderings of crossings on $z$ can be found as follows: Consider a pair of segments $a, b$ that both cross $z$. Either $a$ and $b$ are parallel in $P$ (i.e., we see $abzbaz$ in the bipermutation) or they form a crossing in $P$ which we denote by $a \times b$ (i.e., we see $abzabz$). In Figure 10 the segments 3 and 7 are parallel, and segments 3 and 4 cross, and the crossing point $3 \times 4$ can lie on either side of the segment 1. In the case where $a$ and $b$ are parallel, the order of the two crossings $a \times z$ and $b \times z$ along $z$ is uniquely determined. In the case where $a$ and $b$ cross, the crossing point $a \times b$ can lie on either side of $z$, and the choice of the part further determines the order of the two crossings $a \times z$ and $b \times z$ along $z$. Altogether, the bipermutation of $P$ determines a partial order $\prec$ of all crossings on $z$ such that the legal orderings correspond to linear extensions of this partial ordering.

To count all possible reroutings within $P$, we iterate over all linear extensions $\prec^\star$ of $\prec$ and continue recursively. Note that linear extensions can be enumerated efficiently via backtracking: pick a maximal element and recurse on the remaining elements. Each extension $\prec^\star$
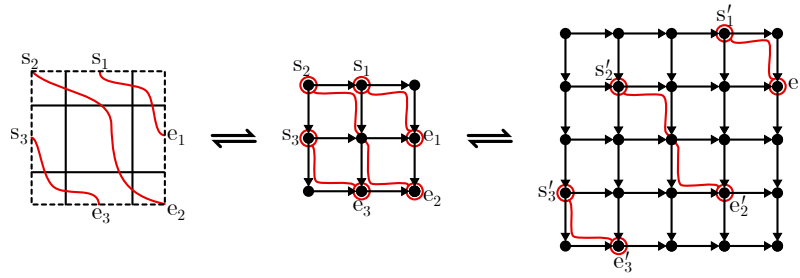
■ **Figure 11** Illustration of the correspondence between arrangements of the chord diagram and disjoint paths in the directed acyclic graph.

uniquely determines the order of crossings along $z$ and therefore the two sides $P_1(z, \prec^\star)$ and $P_1(z, \prec^\star)$ can be explicitly given via their bipermutations. By cumulatively summing up the obtained numbers, we get

$$F(P) = \sum_{\prec^\star \text{ linear extension of } \prec} F(P_1(z, \prec^\star)) \cdot F(P_2(z, \prec^\star)).$$

We provide a computer-assisted framework [1] that allows to fully automatically compute $F(P)$ for a given patch $P$. The input is given as IPE-file[1]. The program reads the collection of lines and the polygonal boundary of $P$, computes the bipermutation, and then performs a dynamic program to determine the number of reroutings within $P$. More specifically, we compute for each bipermutation the lexicographic minimal among all relabelings of the elements, and reuse previously computed values whenever possible.

## 5.2    Counting reroutings via the Lindström-Gessel-Viennot lemma

We illustrate our approach on a small patch defined by a square of side length $l = 2$ (see Figure 11). We can count the number of reroutings in this patch by starting with only the black grid, and counting the number of combinatorially distinct ways to insert three curves starting at $s_1$, $s_2$ and $s_3$ respectively, and ending at $e_1$, $e_2$ and $e_3$ respectively, such that these three curves do not intersect, and each curve crosses every black segment at most once. We can in turn translate this into counting the number of ways to have three paths in the dual grid graph (with edges oriented left to right and top to bottom), starting and ending at prescribed vertices, such that these paths do not cross (middle illustration in Figure 11). Note that here by "not crossing" we do not mean vertex- or edge-disjoint, but that if a path starts above/right of another, it can at no point go below/left of it. Given three such paths, ordered from top right to bottom left, we can shift the second path one unit down and left along the grid and the second path two units down and left along the grid. This results in three vertex-disjoint paths (right illustration in Figure 11). The reverse is also true: given three vertex-disjoint paths on the directed acyclic graph starting at $s_1', s_2', s_3'$ and ending at $e_1', e_2', e_3'$, reversing the shifts produces three non-crossing paths starting at $s_1, s_2, s_3$ and ending at $e_1, e_2, e_3$.

Thus, our question reduces to counting the number of ways to have three such vertex-disjoint paths, whose set of starting vertices is $s_1', s_2', s_3'$ and set of ending vertices is $e_1', e_2', e_3'$. Note that the vertex-disjointness ensures that in such a collection of paths, the path starting at $s_i'$ will end at $e_i'$ for all $1 \leq i \leq 3$. The Lindström–Gessel–Viennot lemma gives us an efficient method to do so:

---

[1] IPE [5] is a drawing editor for creating vector graphics in XML or PDF format. Besides the supplemental input files, also all figures in this article were created with it.
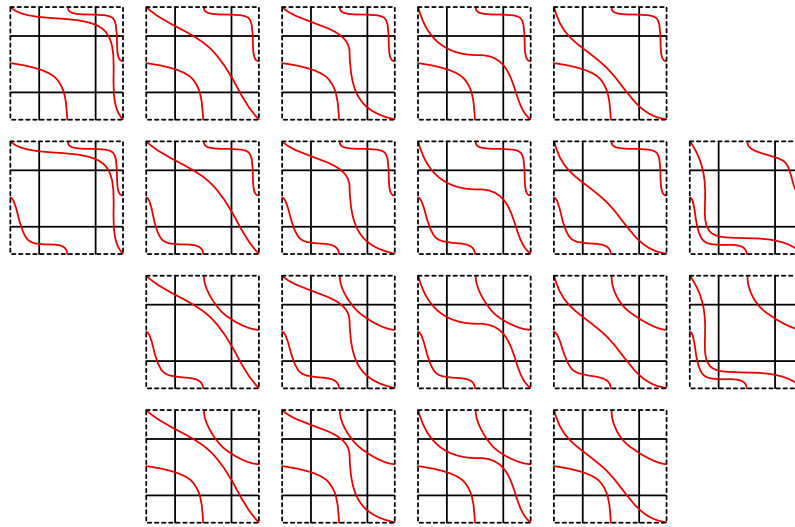
**Figure 12** The 20 reroutings of the patch.

▶ **Lemma 6** (Lindström [24], Gessel & Viennot [14]). *Let $G$ be a finite directed acyclic graph. Consider starting vertices $S = \{s_1, \ldots, s_k\}$ and destination vertices $E = \{e_1, \ldots, e_k\}$. For any two vertices $u$ and $v$, let $p(u, v)$ be the number of paths from $u$ to $v$. Assume that for any tuple of $k$ vertex-disjoint paths starting in $S$ and ending in $E$, the path starting at $s_i$ necessarily ends at $e_i$, for all $1 \le i \le k$. Then the number of distinct such tuples is the determinant of the matrix*

$$M = \begin{pmatrix} p(s_1, e_1) & p(s_1, e_2) & \ldots & p(s_1, e_k) \\ p(s_2, e_1) & p(s_2, e_2) & \ldots & p(s_2, e_k) \\ \vdots & \vdots & \vdots & \vdots \\ p(s_k, e_1) & p(s_k, e_2) & \ldots & p(s_k, e_k) \end{pmatrix}.$$

In our specific case, the entries of this matrix are the number of paths going only down or right between two specified vertices in a grid, which can be easily expressed as a binomial coefficient. We have

$$M = \begin{pmatrix} \binom{2}{1} & \binom{3}{0} & 0 \\ \binom{3}{3} & \binom{4}{2} & \binom{3}{0} \\ 0 & \binom{3}{3} & \binom{2}{1} \end{pmatrix} = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 6 & 1 \\ 0 & 1 & 2 \end{pmatrix}.$$

The determinant of this matrix is $\det(M) = 20$, corresponding to the 20 reroutings of the patch illustrated in Figure 12. In general, for a square patch of size $l \times l$, the corresponding matrix is $M = (m_{ij})_{1 \le i, j \le 2l-1}$, with

$$m_{ij} = \binom{2l - |l - i| - |l - j|}{\frac{2l - |l-i| - |l-j| + 3|i-j|}{2}}.$$

For our actual bounds, we use a square patch of side length $l = 1000$. The number of reroutings of this patch is too big to write explicitly here. In base 10, it is 420 597 digits long. However, it is enough for us to know that $\log_2(\# \text{ of reroutings}) > 1397192$. Table 5 shows the numbers and computation times for other side lengths. The computations suggest that the ratio $\frac{\log_2(\# \text{ of reroutings})}{l^2}$ is converging as $l$ increases. Hence, we expect that increasing $l$ only gives negligible improvements.

■ **Table 5** Summary on the LGV computations for $l \times l$ square patches for different sizes $l$.

| $l$ | $\log_2(\text{\# reroutings})$ | ratio | computing time |
|---|---|---|---|
| 10 | 130.523 | 1.3052 | 0.02s |
| 20 | 539.561 | 1.3489 | 0.09s |
| 50 | 3444.189 | 1.3777 | 0.62s |
| 100 | 13877.972 | 1.3878 | 3.44s |
| 200 | 55719.146 | 1.3930 | 37.96s |
| 500 | 349033.080 | 1.3961 | 2473.19s |
| 1000 | 1397192.106 | 1.3972 | 66467.25s |

## 6   Discussion

All results presented in Tables 2–5 were computed on cluster nodes of TU Berlin with up to 1TB of RAM. The programs can also be run with fewer resources but the dynamic program comes with a memory–time trade-off. Our programs and further details are available in our supplemental data [1]. We also provide simpler patches for which the program only needs few CPU seconds and low RAM. The simpler patches have also been verified by an independent implementation [7]. They, however, give a slightly worse bound.

We performed quite some experiments to optimize the set of parameters. For the 4-slope construction in Section 3.1, we so far manged to obtain $c^- > 0.2183$ and for the 6-slope construction in Section 3.2, $c^- > 0.2542$. For the rectangular 12-slope construction in Section 3.3, we obtained $c^- > 0.2721$, which is the currently best lower bound. It is interesting that already the 4-slope construction gives a significant improvement to the previous best bound by Dumitrescu and Mandal [8]. In the future we plan to investigate also the construction with 16 slopes, from which we expect an improvement to 0.28.

While the results from [8] suggest that larger values of $k$ give better bounds, the computations get more and more complex. In fact, as the number $k$ increases, the number of different shape types and the complexity of the patches increases. Since our program can only deal with patches containing about 30 to 40 segments in reasonable time, depending on the structure of crossings within it, there is a trade-off between the number of crossings and the number of bundles involved in a patch.

As long as one fixes $k$, the counting approach is implicitly limited by $F_k(n)$, which is much smaller than $B_n$. Since $F_3(n) = 2^{cn^2+o(n^2)}$ with $c = \frac{\log_2(3)}{2} - \frac{2}{3} \approx 0.1258$ is known [13], it would be interesting to determine $\lim_{n\to\infty} \frac{\log_2 F_i(n)}{n^2}$ for $i = 4, \dots, 12$. We wonder how far from the truth the constant in Propositions 2 and 3 is. However, since our tiling approach asserts a particular global structure on the arrangement, it is impossible to approximate $F_k$ – even under the assumption that we can process arbitrarily large (but finite) patches.

There is a lot of freedom for choosing the patches for each region, and we indeed experimented with various shapes. The ones presented along Section 3 led to the best results. As mentioned along Section 5.1, when recursively computing $F(B)$, the choice of $z$ can have a significant impact on the computing time. Experiments showed that the best practical performance is obtained by choosing a cutting-segment $z$ such that

- the complexity of the larger part $P_2$ is as small as possible.

In practice this strategy tends to lead to

- *balanced* cuts, that is, $P_1$ and $P_2$ are of similar sizes;
- *short* cuts, that is, the number of intersections on $z$ is relatively small; and
- relatively small numbers of is legal orders.

In general each of those criteria may lead to a different cutting strategy. However, a properly balanced cut may come with larger parts and hence might be less efficient. Also the shortest cut will often be very unbalanced, resulting in one half that is only slightly smaller than the whole path and thus more recursion steps might be required to deal with the patch.

A good splitting strategy should also take the number of legal orders into account because a large number of legal orders will also negatively affect the computation time. The main advantage of our strategy is that it minimizes the complexity of both $P_1$ and $P_2$. This allows the algorithm to reach the bottom of the recursive search tree as fast as possible. In practice it appeared to be more important to reach a low level in the search than to minimize the number of legal orders because the number of cache-hits (pre-computed values) increases fast as the level decreases. To obtain the best possible run-times we performed benchmarks and made statistics for the computing time used in different layers in the recursive search tree. However, in general, it is hard to tell which cutting strategy is the best because we do not have any a priori estimates for the computation time or the number of reroutings for a patch.

Last but not least, since local rerouting is a well-known and frequently used technique in combinatorial geometry, our technique might also be adapted to various other combinatorial structures to derive improved lower bounds. We see for example great potential to improve the lower bounds for arrangements of pseudocircles [12], simple drawings of the complete graph [26, 20], or higher dimensional pseudohyperplane arrangements [2, Corollary 7.4.3], which are all natural generalizations of pseudoline arrangements.

###### ⎯⎯ **References** ⎯⎯

**1** Supplemental data. URL: `https://github.com/fcorteskuehnast/counting-arrangements`.

**2** Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Günter M. Ziegler. *Oriented Matroids*, volume 46 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2 edition, 1999. `doi:10.1017/CBO9780511586507`.

**3** Nicolas Bonichon, Cyril Gavoille, and Nicolas Hanusse. An information-theoretic upper bound of planar graphs using triangulation. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS 2003)*, pages 499–510. Springer, 2003. `doi:10.1007/3-540-36494-3_44`.

**4** Nicolas Bonichon, Cyril Gavoille, and Nicolas Hanusse. *An Information-Theoretic Upper Bound on Planar Graphs Using Well-Orderly Maps*, pages 17–46. Birkhäuser, 2011. `doi: 10.1007/978-0-8176-4904-3_2`.

**5** Otfried Cheong. The Ipe extensible drawing editor. URL: `http://ipe.otfried.org/`.

**6** Fernando Cortés Kühnast. On the number of arrangements of pseudolines. Bachelor's thesis, Technische Universität Berlin, Germany, 2023. URL: `https://fcorteskuehnast.github.io/files/bachelor_thesis.pdf`.

**7** Justin Dallant. Improved Lower Bound on the Number of Pseudoline Arrangements. `arXiv:2402.13923`, 2024.

**8** Adrian Dumitrescu and Ritankar Mandal. New lower bounds for the number of pseudoline arrangements. *Journal of Computational Geometry*, 11:60–92, 2020. `doi:10.20382/jocg.v11i1a3`.

**9** Herbert Edelsbrunner, Joseph O'Rourke, and Raimund Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM Journal on Computing*, 15(2):341–363, 1986. `doi:10.1137/0215024`.

**10** Stefan Felsner. On the Number of Arrangements of Pseudolines. *Discrete & Computational Geometry*, 18(3):257–267, 1997. `doi:10.1007/PL00009318`.

**11** Stefan Felsner and Jacob E. Goodman. Pseudoline Arrangements. In C.D. Toth, J. O'Rourke, and J.E. Goodman, editors, *Handbook of Discrete and Computational Geometry (3rd ed.)*. CRC Press, 2018. `doi:10.1201/9781315119601`.

**12**    Stefan Felsner and Manfred Scheucher. Arrangements of Pseudocircles: On Circularizability. *Discrete & Computational Geometry, Ricky Pollack Memorial Issue*, 64:776–813, 2020. `doi:10.1007/s00454-019-00077-y`.

**13**    Stefan Felsner and Pavel Valtr. Coding and Counting Arrangements of Pseudolines. *Discrete & Computational Geometry*, 46(3), 2011. `doi:10.1007/s00454-011-9366-4`.

**14**    Ira Gessel and Gérard Viennot. Binomial determinants, paths, and hook length formulae. *Advances in Mathematics*, 58(3):300–321, 1985. `doi:10.1016/0001-8708(85)90121-5`.

**15**    Jacob E. Goodman and Richard Pollack. Multidimensional Sorting. *SIAM Journal on Computing*, 12(3):484–507, 1983. `doi:10.1137/0212032`.

**16**    Branko Grünbaum. *Arrangements and Spreads*, volume 10 of *CBMS Regional Conference Series in Mathematics*. AMS, 1972 (reprinted 1980). `doi:10/knkd`.

**17**    Günter Rote. NumPSLA – An experimental research tool for pseudoline arrangements and order types, 2021. URL: `https://github.com/guenterrote/NumPSLA/blob/main/NumPSLA-paper-2024-03-06.pdf`.

**18**    Jun Kawahara, Toshiki Saitoh, Ryo Yoshinaka, and Shin-ichi Minato. Counting Primitive Sorting Networks by $\pi$DDs, 2011. URL: `https://api.semanticscholar.org/CorpusID:9525480`.

**19**    Donald E. Knuth. *Axioms and Hulls*, volume 606 of *LNCS*. Springer, 1992. `doi:10/bwfnz9`.

**20**    Jan Kynčl. Enumeration of simple complete topological graphs. *European Journal of Combinatorics*, 30(7):1676–1685, 2009. `doi:10.1016/j.ejc.2009.03.005`.

**21**    Fernando Cortés Kühnast, Justin Dallant, Stefan Felsner, and Manfred Scheucher. An Improved Lower Bound on the Number of Pseudoline Arrangements. `arXiv:2402.13107`, 2024.

**22**    Fernando Cortés Kühnast, Stefan Felsner, and Manfred Scheucher. An Improved Lower Bound on the Number of Pseudoline Arrangements. `arXiv:2402.13107v1`, 2024.

**23**    Friedrich Levi. Die Teilung der projektiven Ebene durch Gerade oder Pseudogerade. *Berichte über die Verhandlungen der Sächsischen Akademie der Wissenschaften zu Leipzig, Mathematisch-Physische Klasse*, 78:256–267, 1926.

**24**    Bernt Lindström. On the vector representations of induced matroids. *Bulletin of the London Mathematical Society*, 5(1):85–90, 1973. `doi:10.1112/blms/5.1.85`.

**25**    Jiří Matoušek. *Lectures on Discrete Geometry*. Springer, 2002. `doi:10.1007/978-1-4613-0039-7`.

**26**    János Pach and Géza Tóth. How many ways can one draw a graph? *Combinatorica*, 26(5):559–576, 2006. `doi:10.1007/s00493-006-0032-z`.

**27**    Matthew J. Samuel. Word posets, complexity, and Coxeter groups. `arXiv:1101.4655`, 2011.

**28**    Neil J. A. Sloane. The On-Line Encyclopedia of Integer Sequences. `http://oeis.org`.

**29**    Katsuhisa Yamanaka, Shin-ichi Nakano, Yasuko Matsui, Ryuhei Uehara, and Kento Nakada. Efficient enumeration of all ladder lotteries and its application. *Theoretical Computer Science*, 411(16):1714–1722, 2010. `doi:10.1016/j.tcs.2010.01.002`.