Crossing and Independent Families Among Polygons

Anna Brötzner
Malmö University, Sweden

Robert Ganian
Algorithms and Complexity Group, TU Wien, Austria

Thekla Hamm
TU Eindhoven, The Netherlands

Fabian Klute
Universitat Politècnica de Catalunya, Barcelona, Spain

Irene Parada
Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract

Given a set A of points in the plane, a family of line segments forming a matching in A is called crossing (or independent) if each pair of segments in the family intersects (or is non-intersecting, respectively). In past works, these notions have been generalized to polygons by identifying the points in A with the vertices of a given set of polygons and forbidding the line segments from intersecting or overlapping with polygon walls. In this work, we study the computational complexity of computing maximum crossing and independent families in this more general setting.

As our first two results, we show that both problems are NP-hard already when the polygons are triangles. Motivated by this, we turn to parameterized algorithms. For our main algorithmic results, we consider the number of polygons on the input as the natural parameter and under this parameterization obtain a fixed-parameter algorithm for computing a largest crossing family among these polygons, and a separate XP-algorithm for computing a largest independent family that lies in one of the faces of the polygonal domain.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases crossing families, crossing-free matchings, segment intersection graphs, computational geometry, parameterized algorithms

Digital Object Identifier 10.4230/LIPIcs.WADS.2025.11

Funding *Anna Brötzner*: supported by grant 2021-03810 (Illuminate: provably good algorithms for guarding problems) from the Swedish Research Council (Vetenskapsrådet).
Robert Ganian: Projects No. 10.55776/Y1329 and 10.55776/COE12 of the Austrian Science Fund (FWF), Project No. 10.47379/ICT22029 of the Vienna Science Foundation (WWTF).
Fabian Klute: Partially supported by grant PID2023-150725NB-I00 funded by MICIU/AEI/10.13039/501100011033.
Irene Parada: Serra Húnter Fellow. Partially supported by grant PID2023-150725NB-I00 funded by MICIU/AEI/10.13039/501100011033.

1 Introduction

Given n points in the plane, a *crossing family* is a set of straight-line segments with endpoints among these n points such that each pair of segments in the set cross in their interior. The natural counterpart to crossing families are sets of segments which are pairwise non-intersecting (not even at the endpoints); these are studied under the term *crossing-free families*, *plane matchings*, or *independent families* [19].

In 1994, Aronov et al. [4] showed that for any point set in general position there is always a crossing family of size $\Omega(\sqrt{n})$ and since then the problem of quantifying the size of the largest possible crossing family has become a prominent and challenging task in discrete geometry. While it is generally conjectured that the size is in fact $\Theta(n)$ [6], only recently Pach, Rubin, and Tardos [20] achieved a breakthrough result establishing an $n^{1-o(n)}$ bound. Similarly, researchers have investigated crossing-free matchings that can be drawn on a point set. While all point sets admit matchings that cover all but at most one point, a large body of research has been devoted to related problems such as counting how many such matchings exist in a point set [24, 25], considering the lengths of segments in a crossing-free matching [2, 11], or finding large matchings in bicolored point sets [5, 13, 15]. Taking an algorithmic viewpoint, it is natural to ask about the complexity of computing a largest crossing or independent family. While the latter problem (IndependentFamily) is known to be solvable in $\mathcal{O}(n \log n)$-time [13], the complexity of the former (CrossingFamily) remains wide open.

In this work, we consider a more general setting in which the points are the vertices of a set of simple polygons and the segments are segments between two polygon vertices. In particular, we only allow segments which do not intersect the polygon boundary edges in their relative interior. In this perspective, these boundaries can be seen as obstacles in the plane and the segments model visibility, whereas the original problem forms a degenerate case where each polygon is a single point. However, compared to the setting of point sets, the complexity landscape of the problems changes drastically. In fact, already for one polygon the problem of computing a maximum independent family becomes intractable [21], while finding a maximum crossing family in one polygon can be solved in polynomial time [12].

A natural way of tackling these two problems is through the intersection graph of the visibility segments. In particular, one can form a graph by considering each segment as a vertex and connecting two vertices whenever their respective segments intersect. Computing largest crossing and independent families then precisely corresponds to finding a maximum clique or independent set (respectively) in this graph.

The study of these two problems on general segment intersection graphs has a long and storied history. For independent sets in this graph class, Kratochvíl and Nešetřil [14] showed NP-hardness in 1990 and Marx showed that the problem is even W[1]-hard when parameterized by the solution size [16]. For clique, the computational complexity on segment intersection graphs was a long-standing open problem; after partial progress in 1992 [17], this was finally settled via the NP-hardness reduction of Cabello, Cardinal, and Langermann [7]. Unfortunately, none of these lower bounds can be directly carried over to the considered generalization of CrossingFamily and IndependentFamily.

Apart from the considerations above, the problem of finding a large independent family in our setting is closely related to the task of augmenting a given geometric structure by a plane matching. Algorithms and lower bounds for this task have also received considerable attention in the literature [1, 21, 22, 23].

Contributions. Our first result concerns the classical complexity of finding maximum crossing and independent families. For both problems, we demonstrate that they are NP-hard even when all polygons are as simple as possible, namely triangles. Our reductions are from finding a maximum clique and independent set, respectively, in segment intersection graphs [7, 14]. The main idea of the reduction is to "emulate" each segment of the given segment intersection graph using the endpoints of special triangles. And while the principal logic of the reduction can be explained in a streamlined fashion, the underlying gadgets

require a careful construction and the precise placement of every triangle. Especially, it is necessary to find a placement of only triangles such that the vertices responsible for the "emulation" of the segment do not interact with gadgets representing other segments.

To overcome the complexity-theoretic intractability of these problems we turn to the tools of parameterized algorithms. We consider the parameterized problems of finding a largest crossing or independent family among a set of k simple polygons, choosing k as our parameter. For parameterized problems whose unparameterized versions are NP-hard the most desirable runtime behavior of an algorithm is of the form $\mathcal{O}(f(k) \cdot n^c)$ for some computable function f, i.e., a fixed-parameter algorithm. Allowing for more running time, we also consider so-called XP-algorithms where the running time may be in $\mathcal{O}(n^{f(k)})$.

As our first algorithmic result, we show that under this parameterization the problem of finding a largest crossing family indeed admits a fixed-parameter algorithm. For this, we consider the segments in a solution ordered by their slopes, together with the polygons on which the segment endpoints lie. Crucially, we argue that this sequence can in fact be viewed as a sequence of what we call *bundles* which are sets of segments that are consecutive in the sequence and are between a pair of two (not necessarily distinct) polygons. We show that the length of this sequence of bundles is bounded in the number of polygons, which allows us to efficiently branch on the order of pairs of polygons associated with the sequence of bundles. We use the resulting sequence as basis of a dynamic programming approach in which we look at one bundle at a time and compute and tabulate a best option for each last segment (delimiter) of the bundle. We show that it is possible to do that only looking at such tabulated information of the previous bundle and the branched information. The main challenge here is that choices made for a bundle that are compatible with preceding bundles may not be compatible with later bundles or segments in them; carefully filtering our choices without missing a potential solution, we can guarantee that this does not happen.

Turning to our second algorithm, we first realize that for independent families, the above mentioned result by Pilz et al. [21] indicates that the problem is already NP-hard for one polygon, ruling out the existence of even an XP-algorithm. Crucial to their reduction is the fact that segments may appear on the outside and the inside of the polygon. Consequently, we consider the case when segments may only appear on one side of each polygon. We give an XP-algorithm solving this case, again with the number k of polygons on the input as the parameter. The idea is to partition a hypothetical solution into $\mathcal{O}(k^2)$ independent parts, each of which is bounded by two segments of the solution. Branching over which segments act as these boundaries allows us to solve the instance by leveraging polynomial-time algorithms for finding maximum independent sets in circle graphs.

2 Preliminaries

We assume familiarity with the foundations of *parameterized complexity* [8] and standard terminology used in the area of computational geometry, including the notions of *(simple) curves* and *(straight-)line segments*, *polygons* and their *vertices* [10].

All of our objects are placed in the plane. We say that a simple closed curve τ *encloses* an object X if X is contained in the face bounded by τ. Two points a, b are *visible* to (or *see*) each other if the a-b line segment neither touches nor crosses any other object in the plane; we call such segments *visibility segments*. Extending this, a set of points is *visible* from a point a if each point in the set is visible from a. We assume all polygons in this work to be simple (i.e., with no self-intersections) and pairwise non-intersecting. Moreover, we assume w.l.o.g. that the vertices of each polygon are in general position, meaning that no

three vertices lie on a straight line. We explicitly note that polygons are allowed to nest each other, i.e., one may be drawn in the interior of another. We use the notation P° to denote the interior of a polygon.

We say that a *crossing family* among a set of polygons is a set of pairwise crossing visibility segments between vertices of the polygons. Similarly, an *independent family* (also called *non-crossing matching*) among a set of polygons is a set of pairwise non-intersecting visibility segments between vertices of the polygons; we explicitly remark that such segments cannot intersect even at the endpoints. We can now define our main problems of interest:

▶ **Problem 1** (MaxCrossingFamily). *Given a set $\mathcal{P}$ of polygons, find a largest crossing family C among $\mathcal{P}$.*

▶ **Problem 2** (MaxIndependentFamily). *Given a set $\mathcal{P}$ of polygons, find a largest independent family F among $\mathcal{P}$.*

Unfortunately, solving MaxIndependentFamily is known to be NP-hard even if $|\mathcal{P}| = 1$ [21], precisely because of the fact that visibility segments may occur both inside and outside of polygons. Hence, it will be useful to also consider a variant of the problem where polygons represent bounded-sized regions of the plane that are inaccessible, i.e., F does not contain any segment that intersects the interior of a polygon. Formally:

▶ **Problem 3** (MaxIndependentFamilyWithHoles). *Given a set $\mathcal{P}$ of polygons, find a largest independent family F among $\mathcal{P}$ that lies completely in the unbounded face.*

We remark that – in contrast to MaxIndependentFamily– the above problem can be solved in polynomial time for $|\mathcal{P}| = 1$, as it can be reduced to the well-studied problem of finding a maximum independent set in a circle graph [12].

3 Finding maximum crossing and independent families is hard

In this section, we prove that both MaxIndependentFamily and MaxCrossingFamily are NP-hard, even if the polygons on the input are triangles. We present reductions from the corresponding problems of finding a maximum independent set and maximum clique, respectively, in segment intersection graphs. Both are well known to be NP-hard [7, 14]. Here we give an overview of the reductions.

In either reduction, we start from a given set of segments $\mathcal{S}$ and construct a set of triangles $\mathcal{T}$ such that a maximum set of pairwise crossing or non-crossing segments in $\mathcal{S}$ corresponds to a maximum crossing family and independent family, respectively, among $\mathcal{T}$.

We first sketch the construction for MaxIndependentFamily. The idea of the reduction is to replace each $s = pq \in \mathcal{S}$ by a so-called *segment gadget*. Each segment gadget consists of ten triangles, divided into two groups of five triangles each, where each group forms an *endpoint gadget*. As the name suggests, the two endpoint gadgets which we denote by $\mathcal{T}_p$ and $\mathcal{T}_q$ are placed at the endpoints p and q of s, respectively. As every endpoint gadget contains exactly 15 vertices, there is at least one vertex in any independent family among the triangles of the endpoint gadget which can never be matched to a vertex also in the same endpoint gadget. The goal is to place the triangles in such a way that a solution will match a pair of vertices in endpoint gadgets if and only if the underlying segment s lies in a corresponding set of non-crossing segments in $\mathcal{S}$.

An endpoint gadget $\mathcal{T}_p$ contains three types of triangles: three *outer* triangles, one *inner triangle*, and one *shooting* triangle. The outer triangles are three congruent triangles placed around the inner triangle in such a way that each of them has one side on the boundary of the

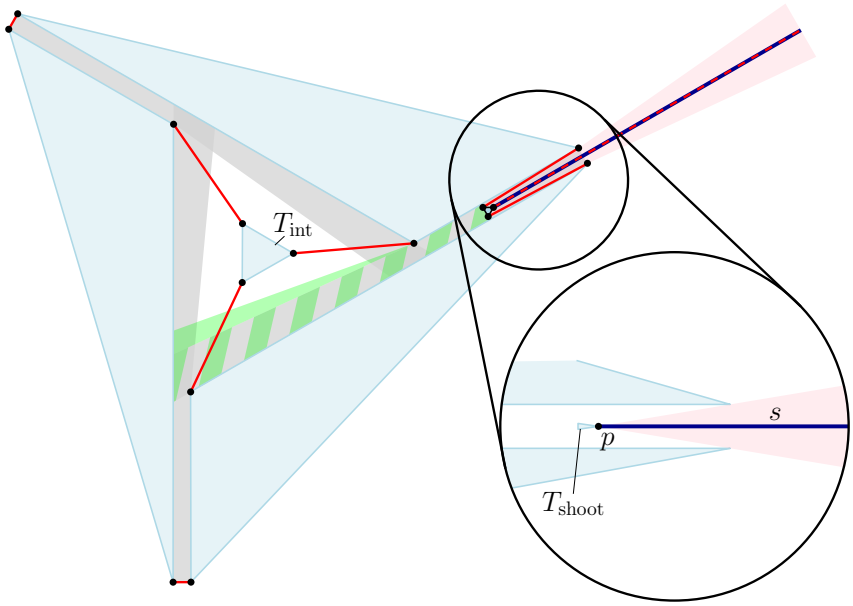

Figure 1 Endpoint gadget with three outer triangles, inner triangle T_{int}, and shooting triangle T_{shoot}. The original segment s is indicated in dark blue. The areas marked in gray are visible from the exterior of the gadget; the area marked in green is the interior area visible to T_{shoot}; the area marked in pink is the area outside the gadget visible from T_{shoot}. The red segments are segments of a maximum independent family within the gadget.

gadget's convex hull, and such that every pair of outer triangles forms a narrow rectangular corridor. One of these corridors is parallel to the supporting line of s and contains the endpoint p in its interior. At the position of p, we place the shooting triangle T_{shoot}. The construction can be seen in Figure 1. An analogous endpoint gadget $\mathcal{T}_q$ is placed at q. By carefully scaling the corridor, we achieve that the vertices of the shooting triangles only see some vertices in their own endpoint gadget, and vertices in the corridor around q in $\mathcal{T}_q$.

Finally, the three outer triangles delimit a triangular region in the interior of the convex hull of $\mathcal{T}_p$ that is visible neither via the corridors nor from the vertices of T_{shoot}. Consequently, the vertices of the inner triangle only see the three vertices of outer triangles that are interior to the convex hull of the endpoint gadget. This forces the corresponding three segments between these vertices to occur in every maximum independent family among $\mathcal{T}$.

Given a set of k pairwise non-crossing segments in $\mathcal{S}$ it is straight-forward to construct an independent family of size $14 \cdot |\mathcal{S}| + k$ among the triangles in $\mathcal{T}$. Vice versa, for any independent family F of size $14 \cdot |\mathcal{S}| + k$ among $\mathcal{T}$ we find that for k segment gadgets a segment in F has to have both its endpoints among the vertices of the two shooting triangles in the segment gadget. Since two segments connecting shooting triangles in different segment gadgets cross if and only if the corresponding segments in $\mathcal{S}$ cross we obtain a set of pairwise non-crossing segments of size k in $\mathcal{S}$ from F.

▶ **Theorem 1.** MaxIndependentFamily *is* NP*-hard even if all polygons are triangles.*

For MaxCrossingFamily we need to modify the above construction, as vertices of the outer triangles may be matched to each other and thereby create a large crossing family independent from the shooting triangles. Therefore, we adapt the endpoint gadgets to ensure that many pairwise crossing visibility segments have to be added for each segment gadget. To this end, we replace in each endpoint gadget the shooting triangle by a set of $7 \cdot |\mathcal{S}|$ shooting

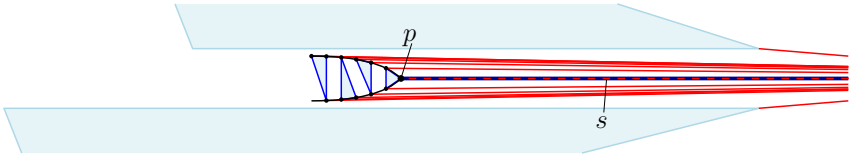

Figure 2 Modification of endpoint gadget. The original segment s is indicated in dark blue. The red segments are segments of a maximum crossing family within the segment gadget.

triangles such that the segments between shooting triangles of the two endpoint gadgets of each segment gadget together form a crossing family of size $21 \cdot |\mathcal{S}|$, see Figure 2. Additionally, four segments between vertices of the outer triangles can be added to this crossing family. Then, a maximum crossing family of segments within a segment gadget has size $21 \cdot |\mathcal{S}| + 4$.

In the whole set of triangles $\mathcal{T}$, a maximum crossing family consists of segments between two vertices in the same segment gadget, and segments between two vertices in different segment gadgets. We can upper-bound the number of the latter ones by $6 \cdot |\mathcal{S}|$, since every endpoint gadget contributes at most six vertices as endpoints of these segments. Consequently, even if all segments between all vertices of outer triangles in $\mathcal{T}$ could be added, this would not offset the contribution of just one segment gadget. Arguing as for independent families, we obtain that $\mathcal{T}$ contains a maximum crossing family of size at least $(21 \cdot |\mathcal{S}| + 4)k$ if and only if there is a set of k pairwise crossing segments in $\mathcal{S}$.

▶ **Theorem 2.** MaxCrossingFamily *is* NP*-hard even if all polygons are triangles.*

4 Finding crossing families among k polygons

Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be a set of k non-intersecting polygons, possibly nesting. In this section we present a fixed-parameter algorithm to solve MaxCrossingFamily when parameterized by k. Thus, the number k of polygons is bounded, but the numbers $n_1, n_2, \ldots, n_k$ of vertices per polygon may be unbounded. The input size is $n = \sum_{i=1}^{k} n_i$.

The $k > 0$ polygons in $\mathcal{P}$ define $k+1$ faces. Our algorithm will compute a largest crossing family among $\mathcal{P}$ in each face independently. Since segments in different faces cannot cross, the solution to MaxCrossingFamily is the largest computed crossing family. For the remainder of this section, we therefore ask for a largest crossing family in a given face f.

For purely technical reasons, we assume w.l.o.g. that there is no vertical segment connecting two polygon vertices (else we perform a slight rotation of the whole instance). This merely ensures that each segment has a well-defined *slope*, which we will later use to obtain an ordering of the segments. For polygons $P, Q \in \mathcal{P}$ (not necessarily distinct) we define their *class* (P, Q) to be the set of segments in which the right endpoint is a vertex of P and the left endpoint is a vertex of Q. In a crossing family C, we consider the segments uniquely sorted by decreasing slope; note that segments with the same slope cannot cross. This naturally gives rise to the sequence of classes of these segments which we compress to a sequence Π in which consecutive occurrences of a class are merged. Note also that a class can appear more than once in Π; each such appearance corresponds to a *bundle* of segments in C, formally a maximal set of segments of the same class that is consecutive in the ordering of the crossing family by slopes. We use their corresponding order in Π to label the bundles $B_1, \ldots B_\ell$. Despite the repetitions, we can provide the following bound on ℓ:

▶ **Lemma 3.** *The length ℓ of Π is at most $4k - 3$.*

Proof. Let Π_r (Π_l) be the sequence of polygons that arises when picking only the first (resp. second) elements of the classes in Π. Since no two polygons intersect each other and no polygon intersects a segment in the crossing family, neither of these sequences contains a subsequence of the form $PQPQ$ with polygons $P \neq Q$. Indeed, two segments that cross and have one of their endpoints in P, together with the boundary of P, define a bounded region. Polygon Q can either lie inside it or outside it, but not both.

This lack of $PQPQ$ subsequences implies that Π_r and Π_l are Davenport-Schinzel sequences of order 2 on the k polygons, and their length is therefore at most $2k-1$ [9]. In the combined sequence Π, two consecutive classes differ in at least one of the polygons. As there are at most $2k-2$ changes in each of the two sequences of polygons, there are at most $2 \cdot (2k-2) = 4k-4$ changes in Π. Therefore, Π has length at most $4k-3$. ◀

We now describe our algorithm, which on a high level works by dynamically computing the largest crossing family of a consecutive sequence of bundles with specific first and last segments under the condition that the crossing family is consistent with the selected Π.

Step 0: Precomputation. We precompute which pairs of vertices of polygons see each other (without crossing any polygon boundary). These are the $\mathcal{O}(n^2)$ *valid* segments and they can trivially be computed in $\mathcal{O}(n^3)$ time.

Step 1: Branching. In the next step of the algorithm, we guess the sequence Π for a solution of MaxCrossingFamily, that is, we run the algorithm for each possible sequence Π. This fixes the order of the bundles and also determines the subset of polygons that contribute to the solution. We call them the *contributing polygons*, $\mathcal{P}'$. By Lemma 3 and the fact that there are k different polygons, we have at most $k^{2(4k-3)}$ possibilities for Π.

For a crossing family, the *extreme delimiters* d_0 (*start delimiter*) and d_ℓ (*end delimiter*) are the segments with maximum and minimum slopes, respectively. We branch on d_0 in a solution. For a class K_i, let $|K_i|$ denote the product of the number of vertices of the two polygons in class K_i. Since d_0 must be in the first class K_1 of Π, the number of choices is bounded by $|K_1|$. In the worst case, this is $\mathcal{O}(n^2)$.

Thus, we run the algorithm in the next step at most $\mathcal{O}(n^2 k^{2(4k-3)})$ times and select the largest crossing family computed among them.

Step 2: Optimizing delimiters. The key information for the solution are the extreme segments for each bundle. A crossing family C *respects* a sequence of classes Π if the compressed sequence of classes of the segments of C sorted by decreasing slope is Π. For a crossing family C that respects Π and for a bundle B_i, the *delimiter* d_i of B_i in C is the segment of C in B_i that has the smallest slope. For notational convenience, we preface the sequence of classes Π by class K_0, which is equal to K_1. We refer to the resulting extended sequence as Π^+. We say that the bundle and delimiter of K_0 is the start delimiter d_0.

Given two segments a and b we say that a point p lies (strictly) *between* them if p lies in the (open) region bounded by the supporting lines of a and b that does not contain any vertical ray, the *horizontal double wedge* of a and b. (We call the (open) interior of the complement of a horizontal double wedge the *vertical double wedge*.) We say that a segment c is between a and b if both endpoints of c are between a and b, one in each side of the horizontal double wedge. Note that then all the segments of C between two consecutive delimiters d_i and d_{i+1} together with d_{i+1} (and d_0 if $d_0 = d_i$) form the bundle B_{i+1}. We also say that a polygon lies between a and b if all its vertices lie strictly between a and b, on the same side of the horizontal double wedge.

We now outline the procedure of our dynamic programming algorithm. Given a set $\mathcal{P}$ of polygons, a sequence of classes Π, and a segment d_0, we aim to find a largest crossing family C among $\mathcal{P}$ that respects Π and begins with delimiter d_0. For this, we dynamically find an optimal sequence of bundle delimiters. Our algorithm processes the classes in the order given by Π. When processing class $i \in [1, \ell]$, we assume that we have solved the subproblems for every valid delimiter d_{i-1} in class K_{i-1}, and that these solutions are stored in a table with one row per delimiter d_{i-1}. In that row we store the maximum *possible* size $|C_{i-1}(d_0, d_{i-1})|$ of a crossing family between (i.e., with extreme delimiters) d_0 and d_{i-1}. With *possible* we here mean that at each step we already take into account that we will need to complete the crossing family to one that respects Π. This table also includes in each row an iteratively built crossing family with extreme delimiters d_0, d_{i-1}, for convenience of solution reconstruction.

At the beginning of the algorithm, the table only contains one row for delimiter d_0, with size 1 and sequence d_0. In step i this table gets updated to one that contains the maximum possible size of a crossing family between d_0 and d_i, $|C_i(d_0, d_i)|$, for each valid delimiter d_i in class K_i. We calculate the values $|C_i(d_0, t)|$ for each t in class K_i from the previously tabulated values $|C_{i-1}(d_0, s)|$ with s in class K_{i-1}. At step i we compute, for each possible pair s, t of delimiters of classes K_{i-1} and K_i, respectively, the best possible crossing family extension $C_i(s, t)$ consisting of segments of class K_i between delimiters s and t, not including s. We denote its size as $|C_i(s, t)|$. In this computation, we take into account that the final sequence needs to respect Π. Also, not all (s, t) pairs are possible: they need to cross and be sorted by slope, but also t should cross the sequence of previous delimiters of s. Crucially, by checking certain necessary conditions, we can make this check independent of the sequence of previous delimiters of s. This restricts the sizes at each step and disallows certain delimiters (which we mark as having size $-\infty$). With the above notation, for each t in class K_i,

$$|C_i(d_0, t)| = \max_{s \text{ delimiter of } K_{i-1}} \{|C_{i-1}(d_0, s)| + |C_i(s, t)|\}.$$

In this way, our algorithm assigns to each delimiter t in class K_i with $i \geq 1$ a delimiter s in the previous class K_{i-1}. To obtain the crossing family in the row corresponding to t, we take the one corresponding to s and append $C_i(s, t)$.

In the final step of the algorithm, we select a delimiter d_ℓ that maximizes $|C_\ell(d_0, d_\ell)|$; the corresponding crossing family is stored in the last table in the row for d_ℓ.

The next lemma specifies how we compute each value $|C_i(s, t)|$, for s and t in classes K_{i-1} and K_i, respectively, and the underlying crossing family $C_i(s, t)$. Moreover, it shows that such a crossing family is optimal if s and t are the delimiters of the corresponding classes. In particular, all other segments in a crossing family respecting Π will cross them.

▶ **Lemma 4.** *Assume we are given a family of polygons $\mathcal{P} = \{P_1, \ldots, P_k\}$, the start delimiter d_0, an extended sequence of classes $\Pi^+ = (K_0 = K_1, \ldots, K_\ell)$, a superset $\mathcal{Q}$ of the contributing polygons of Π^+, and the two delimiters d_i and d_{i+1} of (consecutive bundles corresponding to) classes K_i and K_{i+1}, respectively, satisfying*

(⋆) *$\{d_0, d_i, d_{i+1}\}$ form a crossing family, are sorted by decreasing slope, and there is no polygon in $\mathcal{Q}$ between d_i and d_{i+1}.*

Then we can compute, in time polynomial in the total number n of polygon vertices, a largest crossing family C of $\mathcal{P}$ such that

(i) *its extreme delimiters are d_i and d_{i+1},*

(ii) *all its segments except maybe d_i are in class K_{i+1}, and*

(iii) *no polygon in $\mathcal{Q}$ lies in a triangle bounded by three segments of C.*

Moreover, for all crossing families among $\mathcal{P}$ that: respect the extended sequence Π^+, no polygon in $\mathcal{Q}$ lies between two consecutive delimiters or in a bounded cell, and have as delimiters d_i and d_{i+1} for the bundles of classes K_i and K_{i+1}, respectively, it is always possible to replace the segments between and including d_i and d_{i+1} by C without decreasing the size of the crossing family, still respecting Π^+ and that no polygon in $\mathcal{Q}$ lies in a bounded cell.

In our algorithm we use Lemma 4 with $\mathcal{Q}$ equal to the contributing polygons of Π^+ (same as of Π). The motivation behind Condition iii and the condition involving $\mathcal{Q}$ in the second part of the statement is that no segments in a solution crossing family can define a region enclosing a contributing polygon. If such a region existed, there would be three segments defining a triangular region enclosing the polygon, which cannot be the case since any segment incident to an endpoint inside the region can only cross two of the segments.

▶ **Observation 5.** *Given a crossing family C whose set of contributing polygons is $\mathcal{P}$, no (three) segments of C define a (triangular) region enclosing a polygon in $\mathcal{P}$.*

Proof of Lemma 4. We first check whether Condition $\star$ holds in $\mathcal{O}(n)$ time. If Condition $\star$ does not hold and $\mathcal{Q}$ is the set of contributing polygons of Π, d_i and d_{i+1} cannot be delimiters for classes K_i and K_{i+1}.

Let $\mathcal{S}$ be the set of segments containing each segment $s \neq d_i, d_{i+1}$ such that $\{d_i, d_{i+1}, s\}$ are a crossing family that fulfills Conditions i–iii. Whether a segment lies in $\mathcal{S}$ can be tested in $\mathcal{O}(n)$ time per segment s and there are at most $\mathcal{O}(n^2)$ such segments. The segments in $\mathcal{S}$ form a circle graph, since their endpoints lie either on the boundaries of two disjoint polygons, or on the boundary of one polygon. Consequently, a maximum crossing family C^- consisting of segments in $\mathcal{S}$ is a clique in this graph and can be computed in time $\mathcal{O}(|\mathcal{S}|^2)$ [3]. Let C be the crossing family $C^- \cup \{d_i, d_{i+1}\}$, which we computed in $\mathcal{O}(n^4)$ time.

We now show that the algorithm above computes the crossing family C as per the lemma statement. First, note that by definition of $\mathcal{S}$, Conditions i and ii hold. In the definition of $\mathcal{S}$, we only checked that Condition iii holds for triangular regions bounded by segments d_i and d_{i+1}. We show that this implies that it also holds for any other triple of segments in C.

Assume for contradiction that a polygon $Q \in \mathcal{Q}$ lies in a triangular region bounded by three segments in C. By Condition $\star$, Q must lie either above the supporting lines of d_i and d_{i+1}, or below both. Being in such a bounded region implies that it lies below or above (respectively) the supporting line of a segment $x \in \mathcal{S}$. In both cases, Q lies in the triangle bounded by x, d_i, and d_{i+1}, which we checked is not the case when we included x in $\mathcal{S}$.

For the second part of the lemma, consider a crossing family X among $\mathcal{P}$ that respects Π^+, no polygon in $\mathcal{Q}$ lies between consecutive delimiters or in a bounded cell, and has delimiters d_i and d_{i+1} for the bundles of classes K_i and K_{i+1}, respectively. Let C' be the set of segments of X between d_i and d_{i+1}. For each segment $y \in C'$ it must hold, by definition, that $\{d_i, d_{i+1}, y\}$ is a crossing family satisfying Conditions i–iii. Thus, $C' \subseteq \mathcal{S}$ and replacing C' by C^- does not decrease the size of the crossing family.

We next show that all segments in $X \setminus C'$ cross all the segments in C^-; refer to Figure 3 (left). This is trivially true for d_i and d_{i+1}. Consider a different such segment z in X. By definition, z cannot lie between d_i and d_{i+1}. Moreover, none of its endpoints can lie between d_i and d_{i+1} since z must cross both d_i and d_{i+1}. Thus, by construction of $\mathcal{S}$, and concretely Condition iii, no endpoint of z can lie in a triangle bounded by d_i, d_{i+1}, and a segment in C^-. This implies that one endpoint of z lies below the supporting line of each segment in $\mathcal{S}$ and the other above. Since z crosses d_{i+1} and no segment in X can cross the boundary of a polygon, no endpoint of a segment in C^- can lie in a triangle bounded by z, d_i, and d_{i+1}.

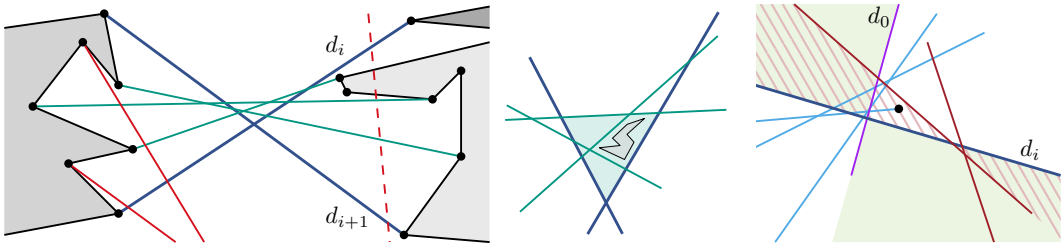

Figure 3 Illustration of the correctness of the FPT-algorithm for MaxCrossingFamily. Left: Segments in red cannot be in a crossing family with the blue consecutive delimiters. The dashed segment is invalid. Middle: a violation of Condition iii. Right: Red segments do not cross all blue delimiters. It can be detected considering the previous dark blue delimiter and the relevant polygons.

As the segments in C^-, by definition, lie between d_i and d_{i+1}, this means that every such segment has one endpoint on either side of z. We can therefore conclude that z crosses all segments in C^-.

By construction and Conditions i and ii, after replacing C' by C^- Π^+ is still respected. It remains to prove that no polygon $Q \in \mathcal{Q}$ lies in a bounded cell. By Condition $\star$, Q cannot lie in the horizontal double wedge of d_i, d_{i+1}. Assume w.l.o.g. that it lies in the top region of the vertical double wedge of d_i, d_{i+1}. If Q lies below the supporting line of a segment in C^-, then that segment together with d_i and d_{i+1} would contradict Condition iii; see Figure 3 (middle). But if it lies above the supporting lines of all segments in C^- then Q can only lie in a bounded cell if it already was in $X \setminus C'$, contradicting the assumption of X. ◀

In our algorithm, when computing $C_i(s,t)$ using Lemma 4, if Condition $\star$ does not hold we do not allow such a pair of delimiters (we can return an empty set and size $-\infty$). The next lemma establishes the correctness of our procedure.

▶ **Lemma 6.** *For a given set $\mathcal{P}$ of polygons, a sequence of classes Π^+, and a segment d_0, our algorithm computes a valid crossing family C among $\mathcal{P}$ that respects Π^+ and has as start delimiter d_0, or reports that no such family exists.*

Proof. Assume first that a solution exists, and let $d_0, d_1, \ldots d_\ell$ be its sequence of delimiters sorted by decreasing slope. Any two consecutive delimiters (together with d_0) must fulfill Condition $\star$ in Lemma 4, and thus our algorithm considered them and a largest crossing family between them, since all conditions are necessary. In particular, by Observation 5 Condition iii must hold for the set of contributing polygons. It might be that our algorithm picks a different sequence of delimiters or crossing family between them than the one in the solution assumed, but it will output an at-least-as-large crossing family.

In the other direction, assume that our algorithm produced a crossing family. We want to show that it is indeed a valid crossing family that respects Π^+ and has as start delimiter d_0. We construct the extended sequence of classes Π^+ from Π and d_0. The computed crossing family corresponds to a sequence of delimiters $d_0, d_1, \ldots d_\ell$. If those would indeed form a crossing family sorted by decreasing slope that respects Π^+ and has as start delimiter d_0, then by the second part of Lemma 4, the whole crossing family computed would be valid. The sequence of delimiters is sorted by decreasing slope and has as start delimiter d_0 by Condition $\star$ at each step. It also respects Π^+ by Condition ii.

It remains to prove that all the delimiters pairwise cross. Assume for contradiction that they do not, and let d_{i+1} be the first delimiter in the sequence that does not cross all the previous delimiters. Thus, $d_0, d_1, \ldots, d_i$ form a crossing family. By Condition $\star$, d_0, d_i, and

d_{i+1} are sorted by decreasing slope and there is no contributing polygon between d_i and d_{i+1}. In particular, the endpoints of d_{i+1} lie in the top and bottom region of the vertical double wedge of d_i and d_0, see Figure 3 (right). In contrast, delimiters $d_1, \ldots, d_{i-1}$ have their endpoints in the left and right region of the horizontal double wedge of d_i and d_0. Since we only consider valid segments as delimiters and there is no contributing polygon between d_i and d_{i+1}, there is no endpoint of the delimiters $d_1, \ldots, d_{i-1}$ enclosed by $\{d_0, d_i, d_{i+1}\}$. Thus, $d_0, d_1, \ldots, d_{i+1}$ form a crossing family, contradicting the assumption. ◀

▶ **Lemma 7.** *For a given set $\mathcal{P}$ of polygons, a sequence of classes Π, and a segment d_0, our algorithm computes a largest crossing family C among $\mathcal{P}$ that respects Π and has as start delimiter d_0.*

Proof. Let $\Pi_i^+ = (K_0 = K_1, \ldots, K_\ell)$ be the extended sequence of classes of Π and let $\mathcal{Q}$ be the set of contributing polygons of Π. We prove by induction on i that our algorithm computes (in step i) a largest crossing family C among $\mathcal{P}$ that respects the extended sequence $\Pi_i^+ = (K_0 = K_1, \ldots, K_i)$, has as start delimiter d_0, as end delimiter a given segment d_i (of class K_i) and the following property: ($\diamond$) there is no polygon in $\mathcal{Q}$ that lies between the extreme delimiters of C or in a triangular region bounded by three segments of C. Note that condition ($\diamond$) is necessary to obtain a crossing family fulfilling the conditions of the lemma.

The induction base trivially holds for $i = 0$ and by Lemma 4, case $i = 1$ also holds. Assume the induction hypothesis for step i, and assume for contradiction that in step $i+1$ there is a larger crossing family C' that respects Π_{i+1}^+, has as start delimiter d_0, as end delimiter d_{i+1}, and fulfills ($\diamond$). Let d_i' be the previous delimiter in C'; it might not coincide with the one chosen by our algorithm, but, for a similar argument as in Lemma 6, it is in the table at the start of step $i+1$. By the induction hypothesis, until d_i' the crossing family computed and tabulated by our algorithm cannot be smaller than the one of C' until d_i'. Moreover, by Lemma 4, between delimiters d_i' and d_{i+1} our algorithm computes a largest crossing family that can replace the one of C' while maintaining its properties (respects Π_{i+1}^+, has as start delimiter d_0, as end delimiter d_{i+1}, and fulfills ($\diamond$)). Thus, the crossing family that our algorithm computes for d_{i+1} in step $i+1$ is at least as large as C'; contradiction.

This implies that in the last step our algorithm computes a largest crossing family for each possible end delimiter. Lemma 6 ensures that we will succeed, and assuming we chose the end delimiter yielding a largest crossing family, the statement follows. ◀

We are now ready to establish our first algorithmic result:

▶ **Theorem 8.** MaxCrossingFamily *is fixed-parameter tractable with respect to the number of polygons k.*

Proof. Given a set $\mathcal{P}$ of polygons, consider a crossing family C' solving MaxCrossingFamily. In our branching step, we consider all possible sequences of classes and start delimiters. In particular, the sequence of classes Π and the start delimiter d_0 of C'. By Lemmas 6 and 7, Step 2 in our algorithm computes a largest valid crossing family C with $|C| \geq |C'|$ respecting Π and with start delimiter d_0, thus solving MaxCrossingFamily.

The algorithm runs in time $f(k) \cdot n^{\mathcal{O}(1)}$ where n is the number of vertices of $\mathcal{P}$. The precomputation step takes $\mathcal{O}(n^3)$ time. We run the polynomial (in n and k) algorithm in Step 2 $\mathcal{O}(n^2 k^{2(4k-3)})$ times. Without trying to optimize its running time, it comprises at most $4k-3$ steps corresponding to the classes, and in each step we consider at most $\mathcal{O}(n^4)$ pairs of delimiters (assuming that we choose the 4 endpoints of the delimiters from all n points). For each such pair, the procedure in Lemma 4 takes $\mathcal{O}(n^4)$ time. ◀

5 Finding independent families among k holes

As our final contribution, we provide an XP-algorithm for MaxIndependentFamilyWithHoles when parameterized by the number of holes – a result which contrasts the known NP-hardness of MaxIndependentFamily with a single polygon [21].

Our proof is based on decomposing sets of independent segments into what we refer to as *bunches*. On a high level, bunches are inclusion-maximal sets of segments that are enclosed by one segment and one connected hole boundary part, or by two segments and two connected hole boundary parts that do not enclose segments of the independent family with endpoints on other hole boundary parts. We will argue that each hypothetical solution can be decomposed into a number of bunches quadratic in the number of polygons, each of which is enclosed by at most two *delimiting* segments of the hypothetical solution and connected parts of at most two polygon boundaries. In XP-time we can branch on a choice of such delimiting segments and compute the other edges of each bunch as an independent set of a circle graph. We formalize below.

▶ **Definition 9.** *Given a family of polygons $\mathcal{P} = \{P_1, \ldots, P_k\}$ and a set $\mathcal{S}$ of pairwise independent segments between vertices of $\mathcal{P}$ that do not intersect P° for any $P \in \mathcal{P}$, a* bunch *B is a maximal subset of $\mathcal{S}$ that is enclosed by at most two elements b_1, b_2 of B and connected parts Q, Q' of the boundaries of at most two arbitrary polygons $P, P' \in \mathcal{P}$ such that the endpoints of each $b \in B$ lie in $Q \cup Q'$. b_1, b_2 are called the* delimiters *of B, and $b_1 \cup b_2 \cup Q \cup Q'$ is called the* boundary *of B.*

The next lemma, whose proof uses similar ideas as that of Lemma 3, allows us to branch on boundaries of bunches that partition a hypothetical target maximum independent family.

▶ **Lemma 10.** *Given a family of polygons $\mathcal{P} = \{P_1, \ldots, P_k\}$ and a set of pairwise independent segments $\mathcal{S}$ between vertices of $\mathcal{P}$ that do not intersect P° for any $P \in \mathcal{P}$, $\mathcal{S} = \dot{\bigcup}_{i \in [\ell]} B_i$ where each B_i is a bunch and $\ell \leq 4k^2$.*

Knowing the boundaries of bunches allows us to compute their other contained segments, or an independent family of at least as many segments in polynomial time. Note that here it is crucial that we are considering holes.

▶ **Theorem 11** ([18])**.** *Given a polygon, a maximal family of independent segments in its interior can be computed in polynomial time.*

The proofs for Lemma 10 and Theorem 11 can be found in the appendix. Combining the above, we obtain our final algorithmic result.

▶ **Theorem 12.** MaxIndependentFamilyWithHoles *is in* XP *parameterized by the number of holes.*

Proof. We can assume a partition into bunches of a maximum independent family of segments between the holes according to Lemma 10, and branch in $n^{2^{\mathcal{O}(k^2)}}$ (n denotes the overall number of polygon vertices) ways on the boundaries of the corresponding bunches: we branch on a set of $\mathcal{O}(k^2)$ pairwise independent delimiting segments, on $\mathcal{O}(k^k)$ pairings of them, and $\mathcal{O}(1)$ hole boundary parts to complete the boundary. Using Theorem 11, we can compute in polynomial time a maximum independent family of segments within each boundary. We return the union of all families computed in this way together with the branched delimiting segments obtained in this way with maximum size.

The described procedure obviously runs in XP-time parameterized by k. Further, the returned family of segments is indeed independent: Segments which are in the solution of the same invocation of Theorem 11 are independent by the correctness of Theorem 11. Segments which are in a solution of an invocation of Theorem 11 are independent from all branched on delimiting segments because they are prohibited from crossing the boundary for which they are in the solution and all delimiting segments lie on or beyond that boundary. Further, the boundary for which a segment was in the solution of an invocation of Theorem 11 separates this segment from all segments in solutions of other invocations of Theorem 11. Independence of branched-on delimiting segments is ensured at branching.

A maximum independent family is returned because in some branch, the delimiting segments will coincide with those that exist in a bunch partition of a maximum independent family according to Lemma 10, and any other segments in a bunch can be replaced by an arbitrary maximum independent family of segments in its boundary. ◀

6 Discussion and open questions

Our reductions establish the intractability of the considered problems even if the polygons are triangles. Nevertheless, the complexity of computing a maximum crossing family among a set of points remains as a prominent open question.

On the algorithmic front, we leave open whether MaxIndependentFamilyWithHoles is fixed-parameter tractable w.r.t. the number of holes. While it seems conceivable that the boundaries of bunches employed in the proof of Theorem 12 could instead be processed via dynamic programming, it is entirely unclear what processing order one should use. Another open question is whether MaxIndependentFamily is fixed-parameter tractable w.r.t. the number of polygons (as opposed to paraNP-hard) if all polygons are convex.

References

1 Hugo A. Akitaya, Matias Korman, Oliver Korten, Mikhail Rudoy, Diane L. Souvaine, and Csaba D. Tóth. Circumscribing polygons and polygonizations for disjoint line segments. *Discrete & Computational Geometry*, 68(1):218–254, 2022. doi:10.1007/S00454-021-00355-8.

2 Noga Alon, Sridhar Rajagopalan, and Subhash Suri. Long non-crossing configurations in the plane. *Fundamenta Informaticae*, 22(4):385–394, 1995. doi:10.3233/FI-1995-2245.

3 Alberto Apostolico, Mikhail J. Atallah, and Susanne E. Hambrusch. New clique and independent set algorithms for circle graphs. *Discrete Applied Mathematics*, 36(1):1–24, 1992. doi:10.1016/0166-218x(92)90200-t.

4 Boris Aronov, Paul Erdös, Wayne Goddard, Daniel J. Kleitman, Michael Klugerman, János Pach, and Leonard J. Schulman. Crossing families. *Combinatorica*, 14(2):127–134, 1994. doi:10.1007/bf01215345.

5 Mikhail J. Atallah. A matching problem in the plane. *Journal of Computer and System Sciences*, 31(1):63–70, 1985. doi:10.1016/0022-0000(85)90065-0.

6 Peter Brass, William OJ Moser, and János Pach. *Research problems in discrete geometry*, volume 18. Springer, 2005. doi:10.1007/0-387-29929-7.

7 Sergio Cabello, Jean Cardinal, and Stefan Langerman. The clique problem in ray intersection graphs. *Discrete & Computational Geometry*, 50(3):771–783, 2013. doi:10.1007/s00454-013-9538-5.

8 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

9 Harold Davenport and Andrzej Schinzel. A combinatorial problem connected with differential equations. *American Journal of Mathematics*, 87(3):684–694, 1965. doi:10.2307/2373068.

10 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications, 3rd Edition*. Springer, 2008. doi:10.1007/978-3-540-77974-2.

11 Adrian Dumitrescu and Csaba D. Tóth. Long non-crossing configurations in the plane. *Discrete & Computational Geometry*, 44(4):727–752, 2010. doi:10.1007/S00454-010-9277-9.

12 Fanica Gavril. Algorithms for a maximum clique and a maximum independent set of a circle graph. *Networks*, 3(3):261–273, 1973. doi:10.1002/net.3230030305.

13 John Hershberger and Subhash Suri. Applications of a semi-dynamic convex hull algorithm. *BIT Numerical Mathematics*, 32(2):249–267, 1992. doi:10.1007/BF01994880.

14 Jan Kratochvíl and Jaroslav Nešetřil. Independent set and clique problems in intersection-defined classes of graphs. *Commentationes Mathematicae Universitatis Carolinae*, 31(1):85–93, 1990. URL: http://dml.cz/dmlcz/106821.

15 Chi-Yuan Lo, Jirí Matousek, and William L. Steiger. Algorithms for ham-sandwich cuts. *Discrete & Computational Geometry*, 11:433–452, 1994. doi:10.1007/BF02574017.

16 Dániel Marx. Parameterized complexity of independence and domination on geometric graphs. In *Proceedings of the Second International Workshop on Parameterized and Exact Computation (IWPEC'06)*, volume 4169 of *LNCS*, pages 154–165. Springer, 2006. doi:10.1007/11847250_14.

17 Matthias Middendorf and Frank Pfeiffer. The max clique problem in classes of string-graphs. *Discrete Mathematics*, 108(1-3):365–372, 1992. doi:10.1016/0012-365X(92)90688-C.

18 Nicholas Nash and David Gregg. An output sensitive algorithm for computing a maximum independent set of a circle graph. *Information Processing Letters*, 110(16):630–634, 2010. doi:10.1016/j.ipl.2010.05.016.

19 Monroe Newborn and William O. J. Moser. Optimal crossing-free hamiltonian circuit drawings of k_n. *Journal of Combinatorial Theory, Series B*, 29(1):13–26, 1980. doi:10.1016/0095-8956(80)90041-6.

20 János Pach, Natan Rubin, and Gábor Tardos. Planar point sets determine many pairwise crossing segments. *Advances in Mathematics*, 386:107779, 2021. doi:10.1016/j.aim.2021.107779.

21 Alexander Pilz, Jonathan Rollin, Lena Schlipf, and André Schulz. Augmenting geometric graphs with matchings. In *Proceedings of the 28th International Symposium on Graph Drawing and Network Visualization (GD'20)*, volume 12590 of *Lecture Notes in Computer Science*, pages 490–504. Springer, 2020. doi:10.1007/978-3-030-68766-3_38.

22 David Rappaport. Computing simple circuits from a set of line segments is np-complete. *SIAM Journal on Computing*, 18(6):1128–1139, 1989. doi:10.1137/0218075.

23 David Rappaport, Hiroshi Imai, and Godfried T. Toussaint. On computing simple circuits on a set of line segments. In *Proceedings of the 2nd Annual Symposium on Computational Geometry (SCG'86)*, pages 52–60. ACM Press, 1986. doi:10.1145/10515.10521.

24 Micha Sharir, Adam Sheffer, and Emo Welzl. Counting plane graphs: Perfect matchings, spanning cycles, and kasteleyn's technique. *Journal of Combinatorial Theory, Series A*, 120(4):777–794, 2013. doi:10.1016/j.jcta.2013.01.002.

25 Micha Sharir and Emo Welzl. On the number of crossing-free matchings, cycles, and partitions. *SIAM Journal on Computing*, 36(3):695–720, 2006. doi:10.1137/050636036.

A Omitted proofs from Section 5

▶ **Lemma 10.** *Given a family of polygons $\mathcal{P} = \{P_1, \ldots, P_k\}$ and a set of pairwise independent segments $\mathcal{S}$ between vertices of $\mathcal{P}$ that do not intersect P° for any $P \in \mathcal{P}$, $\mathcal{S} = \dot{\bigcup}_{i\in[\ell]} B_i$ where each B_i is a bunch and $\ell \leq 4k^2$.*

Proof. We show the statement by induction on k.

For the base case, $k = 1$ and $\mathcal{S} = B$ with the delimiting segment of B being given by an arbitrary segment in $\mathcal{S}$ whose endpoints are consecutive in a traversal of the boundary $P \cap \mathcal{S}$ along the boundary of P (if no such segment exists then $\mathcal{S} = \emptyset$ and hence $\ell = 0$ is valid). The boundary of B is given by the above delimiting segment and the part of the boundary of P between its endpoints that contains other endpoints of segments in $\mathcal{S}$ (if there is no other segment in $\mathcal{S}$ we can choose between the two parts of the boundary of P between the endpoints of the delimiting segment arbitrarily).

For the inductive step, assume the statement to hold for independent families on holes $P_1, \ldots, P_{k-1}$. Consider all elements of $\mathcal{S}$ with an endpoint on P_k and an endpoint on P_i for any $i \in [k-1]$ and label them with $i \in [k-1]$ based on the containment of their other endpoint in P_i.

Along a traversal of the boundary of P_k, any distinct labels i, i' do not interleave (i.e., occur in the order i, i', i, i') as this would induce a crossing between two closed curves, one formed by segments with the same label i and a part of the boundary between their endpoints on P_k and P_i and the other formed by segments with the same label i' and a part of the boundary between their endpoints on P_k and $P_{i'}$, contradicting that $\mathcal{S}$ is independent or that P_i, $P_{i'}$, and P_k are pairwise non-intersecting. Now we remove all but the first and last segment or each subsequence of each maximal subsequence of the traversal which has one unique label. We call the resulting sequence σ.

A non-interleaving sequence over $[k-1]$ with at most two consecutive repetitions of each element has length at most $4k-2$ because without repetitions it is a Davenport-Schinzel sequence of order 2, and its length is therefore at most $2k-1$ [9]. Thus, the length of σ is at most $4k-2$.

By construction, a desired partition into bunches can be achieved by combining a desired partition of the subset of $\mathcal{S}$ that does not contain segments with endpoints on P_k and bunches whose delimiting segments are in σ; here, delimiting segments are associated to the same bunch if they are consecutive with the same label. The former partition exists by inductive hypothesis, and by inductive hypothesis we get an overall size bound of $4(k-1)^2+4k-2 \leq 4k^2$ as desired. ◀

▶ **Theorem 11** ([18]). *Given a polygon, a maximal family of independent segments in its interior can be computed in polynomial time.*

Proof. The set of segments in the interior of a polygon can be viewed as the vertex set of a circle graph in a natural way. Finding a maximum independent family among these segments then directly corresponds to finding a maximum independent set in the circle graph. The latter is known to be possible in polynomial time [18]. ◀