A QPTAS for the Base of the Number of Crossing-Free Structures on a Planar Point Set

Marek Karpinski[1] ⋆, Andrzej Lingas[2] ⋆⋆, and Dzmitry Sledneu[3]

[1] Department of Computer Science, University of Bonn
marek@cs.uni-bonn.de
[2] Department of Computer Science, Lund University
andrzej.lingas@cs.lth.se
[3] Centre for Mathematical Sciences, Lund University
dzmitry@maths.lth.se

Abstract. The number of triangulations of a planar n point set S is known to be c^n, where the base c lies between 8.65 and 30. Similarly, the number of spanning trees on S is known to be d^n, where the base d lies between 12.52 and 141.07. The fastest known algorithm for counting triangulations of S runs in $O^*(2^n)$ time while that for counting spanning trees on S enumerates them in $\Omega(12.52^n)$ time. The fastest known arbitrarily close approximation algorithms for the base of the number of triangulations of S and the base of the number of spanning trees of S, respectively, run in time subexponential in n. We present the first quasi-polynomial approximation schemes for the base of the number of triangulations of S and the base of the number of spanning trees on S, respectively.

1 Introduction

By a crossing-free structure in the Euclidean plane, we mean a *planar straight-line graph* (PSLG), i.e., a plane graph whose edges $\{v, u\}$ are represented by properly non-intersecting straight-line segments with endpoints v, u, respectively. Triangulations and spanning trees on finite planar point sets are the two most basic examples of crossing-free structures in the plane, i.e., PSLGs. The problems of counting the number of such structures for a given planar n-point set belong to the most intriguing in Computational Geometry [2–5, 7, 11, 12].

Counting triangulations. A *triangulation* of a set S of n points in the Euclidean plane is a PSLG on S with a maximum number of edges. Let $F_t(S)$ stand for the set of all triangulations of S.

The problem of computing the number of triangulations of S, i.e., $|F_t(S)|$, is easy when S is convex. Simply, by a straightforward recurrence, $|F_t(S)| = C_{n-2}$, where C_k is the k-th Catalan number, in this special case. However, in the general case, the problem of computing the number of triangulations of S is

⋆ Research partially supported by DFG grants and the Hausdorff Center grant.
⋆⋆ Research supported in part by VR grant 621-2011-6179.

neither known to be $\#P$-hard nor known to admit a polynomial-time counting algorithm.

It is known that $|F_t(S)|$ lies between $\Omega(8.65^n)$ [4] and $O(30^n)$ [11]. See also Table 1 in Appendix. Since the so called flip graph whose nodes are triangulations of S is connected [15], all triangulations of S can be listed in exponential time by a standard traversal of this graph. Only recently, Alvarez and Seidel have presented an elegant algorithm for the number of triangulations of S running in $O^*(2^n)$ time [3] which is substantially below the aforementioned lower bound on $|F(S)|$ (the O^* notation suppresses polynomial in n factors).

Also recently, Alvarez, Bringmann, Ray, and Seidel [2] have presented an approximation algorithm for the number of triangulations of S based on a recursive application of the planar simple cycle separator [9]. Their algorithm runs in subexponential $2^{O(\sqrt{n}\log n)}$ time and over-counts the number of triangulations by at most a subexponential $2^{O(n^{\frac{3}{4}}\sqrt{\log n})}$ factor. It also yields a subexponential-time approximation scheme for the base of the number of triangulations of S, i.e., for $|F_t(S)|^{\frac{1}{n}}$. The authors of [2] observe also that just the inequalities $\Omega(8.65^n) \leq |F_t(S)| \leq O(30^n)$ yield the large exponential approximation factor $O(\sqrt{30/8.65}^n)$ for $|F_t(S)|$ trivially computable in polynomial time.

Counting spanning trees. A *spanning tree* U on a set S of n points in the Euclidean plane is a connected PSLG on S that is cycle-free, equivalently, that has $n-1$ edges. Let $F_s(S)$ stand for the set of all spanning trees on S.

It is known that $|F_s(S)|$ lies between $\Omega(12.52^n)$ [8] and $O(141.07^n)$ [7]. The fastest known algorithms for computing $|F_s(S)|$ enumerate $F_s(S)$. It is still an open problem if the enumeration method can be beaten for spanning tree.

The aforementioned approximation algorithm for $|F_t(S)|$ due to Alvarez, Bringmann, Ray, and Seidel can be adapted to compute $|F_s(S)|$ in the same asymptotic subexponential $2^{O(\sqrt{n}\log n)}$ time within the same asymptotic subexponential $2^{O(n^{\frac{3}{4}}\sqrt{\log n})}$ approximation factor [2]. The adaption also yields a subexponential-time approximation scheme for the base of the number of spanning trees on S, i.e., for $|F_s(S)|^{\frac{1}{n}}$.

Our contributions. We take a similar approximation approach to the problems of counting triangulations of S and counting spanning trees on S as Alvarez, Bringmann, Ray, and Seidel in [2]. However, importantly, instead of using recursively the planar simple cycle separator [9], we shall apply recursively the so called balanced α-cheap l-cuts of maximum independent sets of triangles within a dynamic programming framework developed by Adamaszek and Wiese in [1]. By using the aforementioned techniques, the authors of [1] designed the first quasi-polynomial time approximation scheme (QPTAS, see Appendix for the definition) for the maximum weight independent set of polygons belonging to the input set of polygons with poly-logarithmically many edges.

Observe that a triangulation of S can be viewed as a maximum independent set of triangles drawn from the set of all triangles with vertices in S that are free from other points in S (triangles, or in general polygons, are identified with their open interiors). Also, a spanning tree on S can be easily complemented to a full

triangulation on S. These simple observations enable us to use the aforementioned balanced α-cheap l-cuts recursively in order to bound an approximation factor of our approximation algorithm. The parameter α specifies the maximum fraction of an independent set of triangles that can be destroyed by the l-cut, which is a polygon with at most l vertices in a specially constructed set of points of polynomial size.

Similarly as the approximation algorithm from [2], our algorithm may over-count the true number of triangulations or spanning trees because the same triangulation or spanning tree, respectively, can be partitioned recursively in many different ways. In contrast with the approximation algorithm in [2], our algorithm may also under-count the number of triangulations of S or spanning trees on S, since our partitions generally destroy a fraction of edges in a triangulation or a spanning tree on S.

Our approximation algorithm for the number of triangulations of (or, the number of spanning trees on, respectively) a set S of n points with integer coordinates in the plane runs in $n^{(\log(n)/\epsilon)^{O(1)}}$ time. For $\epsilon > 0$, it returns a number at most $2^{\epsilon n}$ times smaller and at most $2^{\epsilon n}$ times larger than the number of triangulations of S (or, the number of spanning trees on S, respectively). Note that even for $\epsilon = (\log n)^{-O(1)}$, the running time is still quasi-polynomial.

As a corollary, we obtain quasi-polynomial approximation schemes for the base of the number of triangulations of S, i.e., for $|F_t(S)|^{\frac{1}{n}}$, and the base of the number of spanning trees on S, i.e., for $|F_s(S)|^{\frac{1}{n}}$, respectively. This implies that the problems of approximating $|F_t(S)|^{\frac{1}{n}}$ and $|F_s(S)|^{\frac{1}{n}}$ cannot be APX-hard (under standard complexity theoretical assumptions).

Organization of the paper. In Preliminaries, we introduce basic concepts of the dynamic programming framework from [1]. In the following section, we present five properties of an abstract family of (crossing-free) structures on which the analysis of our approximation algorithm relies. Section 4 presents our approximation counting algorithm for the number of such structures on S and its time-complexity analysis. A comparison of our algorithm with prior algorithms is moved to Appendix. In Sections 5, upper bounds on the under-counting and the over-counting of the algorithm are derived, respectively. In Section 6, we obtain our main results by showing that planar triangulations and spanning trees satisfy these five properties. A short discussion on possible improvements and extensions of our results is moved to Appendix.

2 Preliminaries

The Maximum Weight Independent Set of Polygons Problem (MWISP) is defined as follows [1]. We are given a set Q of n polygons in the Euclidean plane. Each polygon has at most k vertices, each of the vertices has integer coordinates. Next, each polygon P in Q is considered as an open set, i.e., it is identified with the set of points forming its interior. Also, each polygon $P \in Q$ has weight $w(P) > 0$ associated with it. The task is to find a maximum weight independent

set of polygons in Q, i.e., a maximum weight set $Q' \subseteq Q$ such that for all pairs $P_i,\ P_j$ of polygons in Q', if $P_i \neq P_j$ then it holds $P_i \cap P_j = \emptyset$.

The *bounding box* of Q is the smallest axis aligned rectangle containing all polygons in Q.

Note that in particular if Q consists of all triangles with vertices in a finite planar point set S such that no other point in S lies inside them or on their perimeter, each having weight 1, then the set of all maximum independent sets of polygons in Q is just the set of all triangulations of S. Recall that the latter set is denoted by $F_t(S)$ while the set of all spanning trees on S is denoted by $F_s(S)$.

Adamaszek and Wiese have shown that if $k = poly(\log n)$ then MWISP admits a QPTAS [1].

Fact 1 ([1]). *Let k be a positive integer. There exists a $(1+\epsilon)$-approximation algorithm with a running time of $(nk)^{(\frac{k}{\epsilon}\log n)^{O(1)}}$ for the Maximum Weight Independent Set of Polygons Problem provided that each polygon has at most k vertices.*

Recently, Har-Peled generalized Fact 1 to include arbitrary polygons [6].

We need the following tool from [1].

Definition 1. *Let $l \in \mathbb{N}$ and $\alpha \in \mathbb{R}$ where $0 < \alpha < 1$. Let T be a set of pairwise non-touching triangles. A polygon Γ is a balanced α-cheap l-cut of T if*

- *Γ has at most l edges,*
- *the total weight of all triangles in T that intersect Γ does not exceed an α fraction of the total weight of triangles in T,*
- *the total weight of the triangles in T contained in Γ does not exceed two thirds of the total weight of triangles in T,*
- *the total weight of the triangles in T outside Γ does not exceed two thirds of the total weight of triangles in T.*

For a set of triangles T in the plane, the set of *DP-points* consists of *basic DP-points* and *additional DP-points*. The set of basic DP-points contains the four vertices of the bounding box of T and each intersection of a vertical line passing through a corner of a triangle in T with any edge of a triangle in T or a horizontal edge of the bounding box. The set of additional DP-points consists of all intersections of pairs of straight-line segments whose endpoints are basic DP-points. The authors of [1] observe that the total number of DP-points is $O(n^4)$.

Fact 2 (Lemma 3.6 in [1]). *Let $\delta > 0$ and let T be a set of pairwise non-touching triangles in the plane such that the weight of no triangle in T exceeds one third of the weight of T. Then there exists a balanced $O(\delta)$-cheap $(\frac{1}{\delta})^{O(1)}$-cut with vertices at basic DP-points.*

3 An abstract crossing-free structure

Triangulations and spanning trees are special cases of planar straight-line graphs (PSLGs). We shall consider an abstract family F_a of finite PSLGs having five properties (satisfied by triangulations and spanning trees as shown in Section 6).

We shall use the following conventions in order to specify these properties and design an approximation algorithm for counting the number of PSLGs in F_a whose vertex set is an n-point planar point set S. We shall denote the latter set by $F_a(S)$.

We shall call a member in F_a a (crossing-free) *structure*, and a member in $F_a(S)$ a structure on S. Next, we shall call any subgraph of a structure a *substructure*.

Let P be a polygon with holes. The restriction of a structure G to P is the substructure consisting of all edges and vertices of G within P. (E.g., if G is a triangulation then the restriction is a partial triangulation, and if G is a spanning tree then the restriction is a forest, in general).

We say that a substructure is within P if all its vertices and all its edges are within P. Next, we shall call a substructure $H = (V_H, E_H)$ within P *maximal* if there is no other substructure $H' = (V_{H'}, E_{H'})$ within P, where $V_H = V_{H'}$, and $E_H \subsetneqq E_{H'}$. (E.g., if H is a partial triangulation within P then it cannot be extended to any larger partial triangulation by adding more edges, similarly, if H is a forest within P then it cannot be extended to any larger forest within P by adding more edges.)

We shall assume that the family F_a has the following properties.

1. One can decide if a PSLG with at most n vertices is a structure, i.e., belongs to F_a, in at most $2^{O(n \log n)}$ time.
2. If a structure has n vertices then it has $\Omega(n)$ edges. Two structures with the same set of vertices have the same number of edges.
3. Any substructure is in particular a substructure of a structure on the vertex set of the substructure.
4. Any extension of the restriction of a structure G to a simple polygon P with holes to a maximal substructure on the vertices of G within P uses at most $O(l)$ additional edges, where l is the number of edges of G with endpoints in P crossed by the boundaries of P.
5. Suppose that polygons P_1, P_2 with holes form a partition of a polygon P with holes. The union of a substructure within P_1 with a substructure within P_2 is a substructure.

By the definitions, F_a has also the following properties.

Lemma 1. *(Property 6) A maximal substructure H within the bounding box of the structure that H is a subgraph is a structure.*

Lemma 2. *(Property 7). Suppose that for $j = 1, \ldots, l$, R_j is a maximal substructure within the polygon P_j with holes, and the polygons P_1 through P_l are pairwise non-overlapping and their union forms a polygon P with holes. Let*

$R'_1, \ldots, R'_l$ be another sequence of maximal substructures within $P, \ldots, P_l$, respectively, where R_j and R'_j have the same vertex set for $j = 1, \ldots, l$. If $R_i \neq R'_i$ for some $i \in \{1, \ldots, l\}$, each edge extension of $\bigcup_{j=1}^{l} R_j$ to a maximal substructure within P is different from any edge extension of $\bigcup_{j=1}^{l} R'_j$ to a maximal substructure within P.

Proof. The proof is by contradiction. The joint edge extension of both sequences would contain $R_j \cup R_{j'}$ within P_j which would contradict the maximality of both R_j and $R_{j'}$ within P_j. □

4 Dynamic programming

Our dynamic programming approximation algorithm for $|F_a(S)|$ is termed Algorithm 1 and it is depicted in Fig. 1. For the comparison of our algorithm with those due to Adamaszek and Wiese [1] and to Alvarez, Bringmann, Ray, and Seidel [2], see Appendix.

Input: A set S of n points with integer coordinates in the Euclidean plane and natural number parameters k and Δ.
Output: An approximate number of structures on the vertex set S, i.e., an approximate $|F_a(S)|$.
1: $T \leftarrow$ the set of all triangles with vertices in S that do not contain any other point in S;
2: $P \leftarrow$ a list of polygons (possibly with holes) with at most k vertices in total at DP points induced by T, topologically sorted with respect to geometric containment;
3: **for** each polygon $Q \in P$ containing at most Δ points in S **do**
4: $as(Q) \leftarrow$ exact number of maximal substructures on the vertex set $S \cap Q$ within Q;
5: **end for**
6: **for** each polygon set $Q \in P$ containing more than Δ points in S **do**
7: $as(Q) \leftarrow 0$;
8: **for** each partition of Q into polygons $Q_1, \ldots, Q_l \in P$, where $l \leq k$, no Q_j contains more than two thirds of points in $S \cap Q$, and $as(Q_1)$ through $as(Q_l)$ are defined **do**
9: $as(Q) \leftarrow as(Q) + \prod_{j=1}^{l} as(Q_j)$;
10: **end for**
11: **end for**
12: Output $as(B)$, where B is the bounding box of T.

Fig. 1. Algorithm 1 for approximately counting structures on a finite planar point set.

Time complexity. The cardinality of T does not exceed n^3. Then, by the analogy with the dynamic programming algorithm of Adamaszek and Wiese for nearly maximum independent set of triangles [1], we call a polygon in the list P in Algorithm 1 a *DP cell* and observe that the number of DP cells is

$(3n^3)^{O(k)} = n^{O(k)}$ (see Proposition 2.1 in [1]). Consequently, the number of possible partitions of a DP cell into at most k DP cells is $O(\binom{n^{O(k)}}{k})$, i.e., $n^{O(k^2)}$.

It follows that if we neglect the cost of computing the exact number of maximal substructures contained within a DP cell including at most Δ input points, then Algorithm 1 runs in $n^{O(k^2)}$ time.

We can compute the exact number of maximal substructures contained within a DP cell with at most Δ input points in $2^{O(\Delta \log \Delta)}$ time as follows. By enumerating all PSLGs on the subset of S contained in the DP cell, and using Property 1 and the fact that the number of PSLGs on at most Δ vertices is $2^{O(\Delta \log \Delta)}$, we can list all structures on this subset in $2^{O(\Delta \log \Delta)} \times 2^{O(\Delta \log \Delta)} = 2^{O(\Delta \log \Delta)}$ time. Hence, by Property 3, we can exactly count all maximal substructures (on this subset) within the cell by pruning the aforementioned structures and checking maximality also in $2^{O(\Delta \log \Delta)}$ time. We conclude with the following lemma.

Lemma 3. *Algorithm 1 runs in $n^{O(k^2)}2^{O(\Delta \log \Delta)}$ time.*

5 Approximation factor

Under-counting. The potential under-counting stems from the fact that when a DP cell is partitioned into at most k smaller DP cells then the possible combinations of structure edges crossing the boundaries of the cells are not counted. Furthermore, in the leaf DP cells, i.e., those including at most Δ points from S, we count only maximal substructures while the restriction of a structure on S to a DP cell does not have to be a maximal substructure within the cell. See Q_5 in Fig. 2.

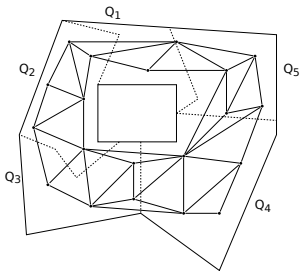

Fig. 2. An example of a maximal partial triangulation within a DP cell and a partition of the DP cell into smaller DP cells $Q_1, \ldots, Q_5$ crossing some triangles in the triangulation.

Intuitively, the general idea of the proof of our upper bound on under-counting is as follows. For each structure $W \in F_a(S)$, there is a substructure counted by Algorithm 1 that can be obtained by removal $O(\epsilon n)$ edges from W and augmenting the resulting substructure with $O(\epsilon n)$ other edges. The final substructure is a union of maximal substructures contained in leaf DP cells.

Lemma 4. *Let S be a set of n points in the plane and let $\epsilon > 0$. For each $W \in F_a(S)$, there is a substructure $W^* \subseteq W$ on S containing at least a $1-O(\epsilon)$ fraction of the edges of W and a substructure $M(W^*)$ on S which is an extension of W^* by $O(\epsilon n)$ edges such that the estimation returned by Algorithm 1 with k set to $\log^{O(1)}(n)/\epsilon^{O(1)}$ is not less than $|\bigcup_{W \in F(S)}\{M(W^*)\}|$.*

Proof. Let $W \in F_a(S)$ and let $T(W)$ be any triangulation of S that is an extension of W. By adapting the idea of the proof of the approximation ratio of the QPTAS in [1], consider the following tree U of DP cells obtained by recursive applications of balanced α-cheap l-cuts.

At the root of U, there is the bounding box. By Fact 2, there is a balanced α-cheap l-cut, where $l = \alpha^{-O(1)}$, that splits the box into at most k children DP cells such that only α fraction of the triangular faces of $T(W)$ is crossed by the cut. The construction of U proceeds recursively in children DP cells and stops in DP cells that contain at most Δ points in S.

Note that the height of U is not greater than $\log_{3/2} n$.

For a node u of U, let W_u be the substructure that is the restriction of W to the vertices and edges of W contained in the DP cell Q_u associated with u. Analogously, let $T(W)_u$ be the partial triangulation of the points in $S \cap Q_u$ that is the restriction of $T(W)$ to (the vertices and edges of $T(W)$ contained in) Q_u. Clearly, W_u is a subgraph of $T(W)_u$. Next, let W^*_u be the substructure that is the union of W_t over the the leaves t of the subtree of U rooted at u. Note that W^*_u is a subgraph of W_u. Analogously, let $T(W)^*_u$ denote the restriction of $T(W)_u$ to the union of $T(W)_t$ over the the leaves t of the subtree of U rooted at u. Clearly, W^*_u is a subgraph of $T(W)^*_u$.

By induction on the height $h(u)$ of u in U, we obtain that the partial triangulation $T(W)^*_u \subseteq T(W)_u$ contains a $(1-\alpha)^{h(u)}$ fraction of triangular faces of $T(W)_u$. Set α to $\frac{O(\epsilon)}{\log(n/\epsilon)}$. It follows in particular that for the root r of U, $T(W)^*_r \subseteq T(W)$ contains at least a $(1-\alpha)^{\log_{3/2} n/\epsilon} \geq 1 - O(\epsilon)$ fraction of triangular faces in $T(W)$. Set $T(W)^*$ to $T(W)^*_r$ and W^* to W^*_r. By Property 2 ensuring that W has $\Omega(n)$ edges and the fact that each triangular face has three edges, we conclude that analogously W^* contains a $1-O(\epsilon)$ fraction of the edges of W. Thus, the number of edges in W missing in W^* is $O(\epsilon n)$.

For a leaf t of U, let $M(W_t)$ be an (edge) extension of W_t to a maximal substructure within the leaf cell Q_t. By Property 4, the number of edges extending W_t to $M(W_t)$ is bounded by a constant times the number of edges in W crossing the boundary of Q_t and having an endpoint within Q_t.

For a node u of U, let $M(W^*_u)$ be a substructure within Q_u that is the union of $M(W_t)$ over the leaves t of the subtree of U rooted at u. We have also $M(W^*) = M(W^*_r)$ by $W^* = W^*_r$. It follows that the number of edges extending W^* to $M(W^*)$ is bounded by a constant times the number of edges of W missing in W^*, i.e., $O(\epsilon n)$.

We shall show by induction on $h(u)$ that Algorithm 1 counts at least the number of $M(W^*_u)$ while computing an estimation for Q_u.

If $h(u) = 0$, i.e., u is a leaf in U then $W_u^* = W_u$ and consequently in particular $M(W_u^*) = M(W_u)$ is counted by Algorithm 1.

Suppose in turn that u is an internal node in U with l children $u_1, \ldots, u_l$. When the estimation for Q_u is computed by Algorithm 1, the sum of products of estimations yielded by different partitions of Q_u into at most k DP cells is computed. In particular, the partition into $Q_{u_1}, \ldots, Q_{u_l}$ is considered. By the induction hypothesis, the estimation for Q_{u_j} includes $M(W_{u_j}^*)$ for $j = 1, \ldots, l$. Hence, the product of these estimations counts also $M(W_u^*) = \bigcup_{j=1}^{l} M(W_{u_j}^*)$.

By $M(W^*) = M(W_r^*)$, to obtain the lemma it remains to show that the bound $\log^{O(1)}(n/\epsilon)/\epsilon^{O(1)}$ on k is sufficiently large. Following the proof of Lemma 2.1 in [1], observe that each DP cell Q_u at each level of U is an intersection of at most $O(\log(n/\epsilon))$ polygons, each with at most l edges and vertices at basic DP points. Hence, by $\alpha = \frac{O(\epsilon)}{\log(n/\epsilon)}$ and $l = \alpha^{-O(1)}$, the resulting polygons have at most $O(l^2 \log^2(n/\epsilon)) = \log^{O(1)}(n/\epsilon)/\epsilon^{O(1)}$ edges and vertices at basic and additional DP points. □

Theorem 1. *The under-counting factor of Algorithm 1 with k set to $\log^{O(1)}(n/\epsilon)/\epsilon^{O(1)}$ is at most $2^{O(\epsilon n \log n)}$.*

Proof. Consider any structure $W \in F_a(S)$. By Lemma 4, the number of edges of W that are missing in the substructure $W^* \subseteq W$ is $O(\epsilon n)$. Since all structures in $F_a(S)$ have the same number of edges by Property 2, the number of edges completing W^* to any structure is $O(\epsilon n)$. It follows that the number of ways of completing W^* to a structure in $F_a(S)$ is not greater than the number of subsets of at most $O(\epsilon n)$ edges of the complete Euclidean graph on S, which is $2^{O(\epsilon n \log n)}$.

By Lemma 4, the estimation returned by Algorithm 1 with k set to $\log^{O(1)}(n/\epsilon)/\epsilon^{O(1)}$ is not less than $|\bigcup_{W \in F(S)} \{M(W^*)\}|$.

Now it remains to show that the maximum number of substructures $(W')^*$, $W' \in F_a(S)$, for which $M((W')^*) = M(W^*)$ is at most $2^{O(\epsilon n \log n)}$. By Lemma 4, the number of edges extending $(W')^*$ to $M((W')^*)$ is at most $O(\epsilon n)$. Consequently, the maximum number of such substructures $(W')^*$ is upper bounded by the number of subsets of at most $O(\epsilon n)$ edges of $M(W^*)$ (whose removal may form a substructure $(W')^*$ satisfying $M((W')^*) = M(W^*)$). The latter number is $2^{O(\epsilon n \log n)}$.

We conclude that for $W \in F(S)$, the number of other structures $W' \in F(S)$ for which $M((W')^*) = M(W^*)$ is at most $2^{O(\epsilon n \log n)} 2^{O(\epsilon n \log n)} = 2^{O(\epsilon n \log n)}$. Now, the theorem follows from Lemma 4. □

Over-counting. The reason for over-counting in the estimation returned by our algorithm is as follows. The same structure or more generally substructure within a DP cell may be cut in the number of ways proportional to the number of considered partitions of the DP cell into at most k smaller DP cells. This reason is similar to that for over-counting of the approximation triangulation counting algorithm of Alvarez, Bringmann, Ray, and Seidel [2] based on the planar simple

cycle separator theorem. Therefore, our initial recurrences and calculations are similar to those derived in the analysis of the over-counting from [2].

Lemma 5. *Let Q be an arbitrary DP cell processed by Algorithm 1 which contains more than Δ input points. Recall the calculation of the estimation for Q by summing the products of estimations for smaller DP cells $Q_1, \ldots, Q_l$ over $n^{O(k^2)}$ partitions of Q into $Q_1, \ldots, Q_l$, $l \leq k$. Substitute the true value of the number of maximal substructures (on input points) within each such smaller cell Q_i for the estimated one in the calculation. Let r be the resulting value. The number of maximal substructures (on input points) within Q is at least $r/n^{O(k^2)}$.*

Proof. Note that r is the sum of the number of different combinations of maximal structures within smaller DP cells $Q_1, \ldots, Q_l$ over $n^{O(k^2)}$ partitions of Q into smaller cells $Q_1, \ldots, Q_l$, $l \leq k$. Importantly, each such combination can be completed to some maximal substructure within Q (Property 5) but no two different combinations coming from the same partition $Q_1, \ldots, Q_l$ can be extended to the same maximal substructure within Q by Property 7 (Lemma 2).

Let M be the set of maximal substructures W within Q for which there is a partition into smaller DP cells $Q_1, \ldots, Q_l$, $l \leq k$, such that for $i = 1, \ldots, l$, W constrained to Q_i is a maximal substructure within Q_i. Note that for each $W \in M$, the number of the combinations that can be completed to W cannot exceed that of the considered partitions, i.e., $n^{O(k^2)}$, as each of the combinations has to come from a distinct partition $Q_1, \ldots, Q_l$.

Thus, there is a binary relationship between maximal substructures within Q that belong to M and the aforementioned combinations. It is defined on all the maximal substructures in M and on all the combinations, and a maximal substructures in M is in relation with at most $n^{O(k^2)}$ combinations. This yields the lemma. □

By Lemma 5, we can express the over-counting factor $L(Q, \Delta)$ of Algorithm 1 for a DP cell Q by the following recurrence:

$$L(Q, \Delta) = \sum_{(Q_1, \ldots, Q_l)} \prod_{j=1}^{l} L(Q_j, \Delta) \leq n^{O(k^2)} \prod_{j=1}^{l^*} L(Q_j^*, \Delta)$$

where the summation is over all partitions of Q into DP cells $Q_1, \ldots, Q_l$, such that $l \leq k$, and $Q_1^*, \ldots, Q_{l^*}^*$ is a partition that maximizes the term $\prod_{j=1}^{l} L(Q_j, \Delta)$. When Q contains at most Δ input points, Algorithm 1 computes the exact number of maximal substructures on these points within Q. Thus, we have $L(Q, \Delta) = 1$ in this case.

Following [2], it will be more convenient to transform our recurrence by taking logarithm of both sides. For any DP cell P, let $L'(P, \Delta) = \log L(P, \Delta)$. We obtain now:

$$L'(Q, \Delta) \leq O(k^2 \log n) + \sum_{j=1}^{l^*} L'(Q_j^*, \Delta)$$

Lemma 6. *Let B be a bounding box for a set S of n points in the plane. The following equality holds*

$$L'(B, \Delta) = O(k^2 n \log^2 n/\Delta)$$

Proof. Let U be the recurrence tree and let D be the set of direct ancestors of leaves in U. For each node $d \in D$, the corresponding DP cell includes at least $\Delta+1$ points in S. It follows that $|D| \leq n/\Delta$. Also, any node in D has depth $O(\log n)$ in U. Consequently, the contribution of the subproblems corresponding to nodes in D and their ancestors to the estimation for $L'(B, \Delta)$ can be upper bounded by $O(k^2 \log n \times (n/\Delta) \log n)$. Finally, recall that the subproblems corresponding to leaves of U do not contribute to the estimation. □

Lemma 6 and Property 6 (Lemma 1) immediately yield the following corollary.

Theorem 2. *Let B be a bounding box for a set of n points in the plane. Set the parameter k in Algorithm 1 as in Theorem 1. If for $\epsilon > 0$ the parameter Δ in Algorithm 1 is set to $\frac{c}{\epsilon} k^2 \log^2 n$ for sufficiently large constant c then the over-counting factor is at most $2^{\epsilon n}$.*

6 Main results

Lemma 7. *Triangulations and spanning trees on finite planar point sets satisfy the five properties of F_a.*

Proof. Properties 1, 2, 3 and 5 are clearly satisfied by triangulations and spanning trees.

To show that Property 4 holds for triangulations, consider an extension of the restriction of a triangulation G to a simple polygon P with holes to a maximal partial triangulation on the vertices of G within P. All the edges within P added by the extension have to be incident to vertices of triangular faces of G with at least one edge crossed by the boundary of P. Observe, that such a triangular face has to have at least one vertex within P that is an endpoint of an edge of G crossed by the boundaries of P. Let l be the number of edges of G with an endpoint within P crossed by the boundaries of P. It follows that the number of aforementioned triangles is at most $2l$ and consequently the number of the endpoints of the edges within P added by the extension does not exceed $3 \times 2l = O(l)$. Hence, the total number of the added edges is also $O(l)$.

To show in turn that Property 4 holds for spanning trees, consider the forest which is the restriction of a spanning tree G to a simple polygon P with holes. Let t be the number of connected components of the forest. It follows that the number l of edges of the spanning tree G with at least one endpoint within P crossed by the boundaries of P is at least $t - 1$. On the other hand, any edge extension of the forest to a maximal forest within P may add at most $t - 1 \leq l$ edges to the forest. □

By combining Lemmata 7 and 3 with Theorems 1, 2 with ϵ set to $\epsilon/\log n$, we obtain our main result.

Theorem 3. *There exists an approximation algorithm for the number of triangulations of (or, the number of spanning trees on) a set S of n points with integer coordinates in the plane with a running time of at most $n^{(\log(n)/\epsilon)^{O(1)}}$ that returns a number at most $2^{\epsilon n}$ times smaller and at most $2^{\epsilon n}$ times larger than the number of triangulations of (or, spanning trees on, respectively) S.*

Corollary 1. *There exists a $(1+\epsilon)$-approximation algorithm with a running time of at most $n^{(\log(n)/\epsilon)^{O(1)}}$ for the base of the number of triangulations of (or, spanning trees on) a set of n points with integer coordinates in the plane.*

Proof. Let c^n be the number of triangulations of (or, the number of spanning trees on) the input n point set, and let Λ be the number returned by the algorithm from Theorem 3. We have $\max\{\frac{c^n}{\Lambda}, \frac{\Lambda}{c^n}\} \leq 2^{\epsilon n}$ by Theorem 3. By taking the n-th root on both sides, we obtain $\max\{\frac{c}{\Lambda^{\frac{1}{n}}}, \frac{\Lambda^{\frac{1}{n}}}{c}\} \leq 2^{\epsilon}$. Now it is sufficient to observe that $2^{\epsilon} < 1+\epsilon$ for $\epsilon < \frac{1}{2}$. □

References

1. Adamaszek, A., Wiese, A.: A QPTAS for maximum weight independent set of polygons with polylogarithmically many vertices. SODA 2014.
2. Alvarez, V., Bringmann, K., Ray, S., Seidel, R.: Counting triangulations approximately. CCCG 2013.
3. Alvarez, V., Seidel, R.: A simple aggregative algorithm for counting triangulations of planar point sets and related problems. SoCG'13.
4. Dumitrescu, A., Schulz, A., Sheffer, A., Tóth, C. D.: Bounds on the maximum multiplicity of some common geometric graphs. SIAM J. Discrete Math. 27(2), 802–826. (2013)
5. Flajolet, P., Noy, M.: Analytic combinatorics of non-crossing configurations. Discrete Mathematics 204(1-3), 203–229. (1999)
6. Har-Peled, S.: Quasi-polynomial time approximation scheme for sparse subsets of polygons. SoCG'14.
7. Hoffmann, M., Sharir, M., Sheffer, A., Tóth, C. D., Welzl, E.: Counting plane graphs: Flippability and its applications. WADS 2011.
8. Huemer, C., de Mier, A.: Lower bounds on the maximum number of non-crossing acyclic graphs. `http://arxiv.org/abs/1310.5882`. (2013)
9. Miller, G. L.: Finding small simple cycle separators for 2-connected planar graphs. J. Comput. Syst. Sci. 32(3), 265–279. (1986)
10. Olaverri, A. G., Noy, M., Tejel, J.: Lower bounds on the number of crossing-free subgraphs of K_n. Comput. Geom. 16(4), 211–221. (2000)
11. Sharir, M., Sheffer, A.: Counting triangulations of planar point sets. Electr. J. Comb. 18(1). (2011)
12. Sharir, M., Sheffer, A., Welzl, E.: On degrees in random triangulations of point sets. J. Comb. Theory, Ser. A 118(7), 1979–1999. (2011)
13. Sharir, M., Sheffer, A., Welzl, E.: Counting plane graphs: Perfect matchings, spanning cycles, and kasteleyn's technique. J. Comb. Theory, Ser. A 120(4), 777–794. (2013)
14. Sharir, M., Welzl, E.: On the number of crossing-free matchings, cycles, and partitions. SIAM J. Comput. 36(3), 695–720. (2006)
15. Sibson, R.: Locally equiangular triangulations. Comput. J. 21(3), 243–245. (1978)

Appendix

Bounds on the number of different types of plane graphs

The following table is based on [4].

Table 1. Bounds on the number of different types of plane graphs

Graph type	Lower bound	Reference	Upper bound	Reference
Triangulations	$\Omega(8.65^n)$	[4]	30^n	[11]
Spanning cycles	$\Omega(4.64^n)$	[10]	$O(54.55^n)$	[13]
Perfect matchings	$\Omega(3^n)$	[10]	$O(10.05^n)$	[14]
Spanning trees	$\Omega(12.52^n)$	[8]	$O(141.07^n)$	[11], [7]

QPTAS definition

An algorithm is called *quasi-polynomial-time* if its worst case running time is $n^{(\log n)^c}$ for some fixed c.

A *quasi-polynomial-time approximation scheme* (QPTAS) for an optimization or counting problem P is a family of algorithms $\{A_\epsilon\}$ satisfying the following condition. For every $\epsilon > 0$, there is a natural number N such that for each instance I of P with size $n \geq N$, if $Opt(I)$ is the measure of an optimal solution to I when P is an optimization problem or just the exact (positive) number in case of counting problem then the measure $A_\epsilon(I)$ of the approximation solution or just the approximate solution returned by A_ϵ, respectively, satisfies $\max\{\frac{A_\epsilon(I)}{Opt(I)}, \frac{Opt(I)}{A_\epsilon(I)}\} \leq 1 + \epsilon$ and A_ϵ runs in quasi-polynomial-time for the fixed ϵ.

A comparison of Algorithm 1 with prior algorithms

The QPTAS of Adamaszek and Wiese for maximum weight independent set of polygons [1] is based on dynamic programming. For each polygon (possibly with holes) with at most k vertices at the DP points induced by the input polygons, termed a DP cell, an approximate maximum weight independent subset of the input polygons contained in the DP cell is computed. The computation is done by considering all possible partitions of the DP cell into at most k smaller DP cells. For each such partition, the union of the approximate solutions for the component DP cells is computed. Then, a maximum weight union is picked as the approximate solution for the DP cell.

Our algorithm, termed Algorithm 1, is in part similar to that of Adamaszek and Wiese [1]. For each DP cell, an approximate number of maximal substructures within the cell is computed instead of an approximate maximum number of non-touching input triangles within the cell. Further modification of the dynamic programming of Adamaszek and Wiese are as follows.

1. Solely those partitions of a DP cell into at most k component DP cells are considered where no component cell contains more than two thirds of the input points in the partitioned cell. (Alternatively, one could generalize the concept of a DP cell to a set of polygons with holes and consider only partitions into two DP cells obeying this restriction.)
2. While a partition of a DP cell into at most k cells is processed, instead of the union of the solutions to the subproblems for these cells, the product of the numerical solutions for the component DP cells is computed.
3. Instead of taking the maximum of the solutions induced by the partition of a DP cell into at most k DP cells, the sum of the numerical solutions induced by these partitions is computed.
4. When the number of points contained in a DP cell does not exceed the threshold number Δ then the exact number of maximal substructures within the cell is computed.

Algorithm 1 also in part resembles the approximation counting algorithm for the number of triangulations of a planar point set due to Alvarez, Bringmann, Ray, and Seidel [2]. The main difference is in the used implicit recursive partition tool. Algorithm 1 uses balanced α-cheap l-cuts within the dynamic programming framework from [1] instead of the simple cycle planar separator theorem [2, 9]. Thus, Algorithm 1 recursively partitions a DP cell defining a subproblem into at most k smaller DP cells while the algorithm in [2] recursively splits a subproblem by a simple cycle that yields a balanced partition. The new partition tool gives a better running time since the number of possible partitions is much smaller so the dynamic programming/recursion has lower complexity and the threshold for the base case can be much lower. Since the algorithm in [2] in particular lists all simple cycles on $O(\sqrt{n})$ vertices, it runs in at least $2^{O(\sqrt{n}\log n)}$ time independently of the precision of the approximation.

Improvements and extensions

One can a bit refine the dynamic programming (Algorithm 1) and consider solely partitions of a DP cell obtained by intersection with polygons with holes on most k DP-points. The number of such partitions is only $n^{O(k)}$, so the whole dynamic programming would take $n^{O(k)}2^{O(\Delta \log \Delta)}$ time. This however does not change the form of the main results.

Adamaszek and Wiese presented also an extension of their theorem on α-cheap cut of an independent set of triangles (Fact 2) to include independent polygons with at most K edges (Lemma 3.1 [1]). This makes possible to generalize our QPTAS for counting triangulations to include the approximation of the number of maximum weight partitions into K-gons.

The other popular crossing-free structures like perfect matchings and cycle covers (see also Table 1 in Appendix) do not satisfy all the five properties of F_a. It is an intriguing open problem if they admit similar quasi-polynomial time approximation algorithms.