Peeling the Cactus: Subexponential-Time Algorithms for Counting Triangulations*

Dániel Marx† Tillmann Miltzow‡

Abstract

Given a set of n points S in the plane, a triangulation T of S is a maximal set of non-crossing segments with endpoints in S. We present an algorithm that computes the number of triangulations on a given set of n points in time $n^{(11+o(1))\sqrt{n}}$, significantly improving the previous best running time of $O(2^n n^2)$ by Alvarez and Seidel [SoCG 2013]. Our main tool is identifying separators of size $O(\sqrt{n})$ of a triangulation in a canonical way. The definition of the separators are based on the decomposition of the triangulation into nested layers ("cactus graphs").

1 Introduction

Given a set of n points in the plane, a triangulation T of S is defined to be a maximal set of non-crossing line segments with both endpoints in S. This set of segments together with the set S defines a plane graph. It is easy to see that every bounded face of a triangulation T is indeed a triangle. Triangulations are one of the most studied concepts in discrete and computational geometry, studied both from combinatorial and algorithmic perspectives. It is well known that the number of possible triangulations of n points in convex position is exactly the $(n-2)$-th Catalan number, but counting the number of triangulations of arbitrary point sets seems to be a much harder problem. There is a long line of research devoted to finding better and better exponential-time algorithms for counting triangulations. The sequence of improvements culminated in the $O(2^n n^2)$ time algorithm of Alvarez and Seidel [2], winning the best paper award at SoCG 2013. Our main result significantly improves the running time of counting triangulations by making it subexponential:

Theorem 1 (General Plane Algorithm) *Given a set S of n points in the plane, there exists an algorithm that computes the number of all triangulations of S in $n^{(11+o(1))\sqrt{n}}$ time.*

*Supported by the ERC grant "PARAMTIGHT: Parameterized complexity and the search for tight complexity results", no. 280152.

†Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), dmarx@cs.bme.hu

‡Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), t.miltzow@gmail.com

It is very often the case that restricting an algorithmic problem to planar graphs allows us to solve it with much better worst-case running time than what is possible for the unrestricted problem. One can observe a certain "square root phenomenon": in many cases, the best known running time for a planar problem contains a square root in the exponent. For example, the 3-Coloring problem on an n-vertex graph can be solved in subexponential time $2^{O(\sqrt{n})}$ on planar graphs (e.g., by observing that a planar graph on n vertices has treewidth $O(\sqrt{n})$), but only $2^{O(n)}$ time algorithms are known for general graphs. Moreover, it is known that if we assume the Exponential-Time Hypothesis (ETH), which states that there is no $2^{o(n)}$ time algorithm for n-variable 3SAT, then there is no $2^{o(\sqrt{n})}$ time algorithm for 3-Coloring on planar graphs and no $2^{o(n)}$ time algorithm on general graphs [7]. The situation is similar for the planar restrictions of many other NP-hard problems, thus it seems that the appearance of the square root of the running time is an essential feature of planar problems. A similar phenomenon occurs in the framework of parameterized problems, where running times of the form $2^{O(\sqrt{k})} \cdot n^{O(1)}$ or $n^{O(\sqrt{k})}$ appear for many planar problems and are known to be essentially best possible (assuming ETH).

A triangulation of n points can be considered as a planar graph on n vertices, hence it is a natural question whether the square root phenomenon holds for the problem of counting triangulations. Indeed, for the related problem of finding a minimum weight triangulation, subexponential algorithms with running time $n^{O(\sqrt{n})}$ are known [4, 5]. These algorithms are based on the use of small balanced separators. Given a plane triangulation on n points in the plane, it is well known that there exists a balanced $O(\sqrt{n})$-sized separator that divides the triangulation into at least two independent graphs [6]. The basic idea is to guess a correct $O(\sqrt{n})$-sized separator of a minimum weight triangulation and recurse on all occurring subproblems. As there are only $n^{O(\sqrt{n})}$ potential graphs on $O(\sqrt{n})$ vertices, one can show that the whole algorithm takes $n^{O(\sqrt{n})}$ time [4, 5].

Unfortunately, this approach has serious problems when we try to apply it to counting triangulations. The fundamental issue with this approach is that a triangulation of course may have more than one $O(\sqrt{n})$-

sized balanced separators and hence we may overcount the number of triangulations, as a triangulation would be taken into account in more than one of the guesses. To get around this problem, an obvious simple idea would be to say that we always try to guess a "canonical" separator, for example, the lexicographically first separator. However, it is a complete mystery how to guarantee in subsequent recursion steps that the separator we have chosen is indeed the lexicographic first for all the triangulations we want to count.

Perhaps the most important technical idea of the paper is finding a suitable way of making the separators canonical. For this purpose, we define a decomposition of a triangulation into nested layers of cactus graphs. (A plane graph is a cactus graph if all vertices *and edges* are incident to the outer face.) The first layer is defined by the set of vertices and edges incident to the outer face. Inductively, the i-th layer is defined by the vertices and edges incident to the outer faces after the first $i-1$ layers are removed. Note that this definition may look similar to onion layers, but actually is very different. The outerplanar index of a graph is defined by the number of non-empty layers.

Given a triangulation T, we define small *canonical* separators by distinguishing two cases. If T has more than $\sqrt{n}$ cactus layers, then one of the first $\sqrt{n}$ layers has size at most $\sqrt{n}$ and we can define the one with smallest index to be the canonical separator. Using such a separator, we *peel off* some cacti to reduce the problem size. In the case when we have only a few cactus layers, we can define short canonical separator paths from the interior to the outer face of the triangulation. We formalize both ideas into a dynamic programming algorithm. The main difficulty is to define the subproblems appropriately. We use the so-called ring subproblems for the layer separators and ring sector subproblems for the path separators.

As a byproduct of this algorithmic scheme, we can efficiently count triangulations with a small number of layers. This is similar to previous work on finding a minimum weight triangulation [3] and counting triangulations [1] for point sets with a small number of onion layers.

Theorem 2 (Thin Plane Algorithm) *Given a set S of n points in the plane, there exists an algorithm that computes the number of all triangulations of S with outerplanar index k in $n^{O(k)}$ time.*

2 Ring Subproblems

Our algorithm is based on dynamic programming: we define a large number of subproblems that are *more general* than the problem we are trying to solve. We generalize the problem by considering *rings:* we need to triangulate a point set in a region between two polygons. Additionally, we may have layer constraints prescribing that a certain number of vertices should appear on certain layers.

In this section, we give a vague definition of the ring subproblems used by the algorithm and sketch how an algorithm that can solve those problems implies Theorem 1 for counting triangulations. In Section 3 we will sketch how to solve these subproblems for "thin rings". Section 4 sketches how to solve ring subproblems in full generality using the algorithm from Section 3 as an important subroutine.

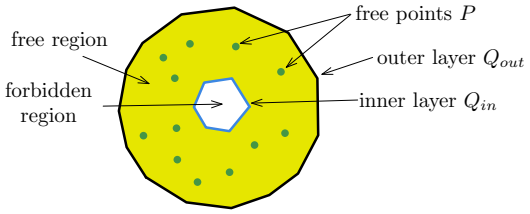

Figure 1: A simple ring subproblem.

See Figure 1 for an illustration of the following definition. A *ring subproblem* consists of: an outer layer Q_{out}, which consists of one or more simple polygons (a ring subproblem is *simple* in case that there is only one polygon.); an inner layer Q_{in}, which is a cactus graph and potentially empty; an inner and outer layer index, which serve to determine the width and conveniently combine solutions of ring subproblems; and potentially layer-constraints that indicate the size of each layer. A *valid triangulation* of a ring subproblem is a triangulation of the area and points between the inner and outer layer, which satisfies the width conditions and in case that layer-constraints are present: each layer should have an appropriate size.

Theorem 3 *Given a layer-unconstrained ring subproblem $\mathcal{S}$ with n free points, there exists an algorithm, denoted by* GeoRing, *that computes the number of all triangulations of $\mathcal{S}$ in $n^{(11+o(1))\sqrt{n}}$ time.*

Proof. [Sketch Theorem 1] The way, we use Theorem 3 is to define for each k an layer-unconstrained ring subproblem $\mathcal{O}_k$ such that each k-outerplane triangulation of S corresponds to a valid k-outerplane triangulation of $\mathcal{O}_k$ and vice versa. Then the algorithm to count all triangulations of S is to count all triangulations of $\mathcal{O}_k$, for each $k = 1, \ldots, n$. It takes $n^{O(\sqrt{n})}$ time for each k. We define $\mathcal{O}_k$ as follows. The outer layer Q_{out} is the boundary of the convex hull of S. The inner layer Q_{in} is empty. The free points P are S without the points on the boundary of the convex hull of S. The inner and outer layer index are in-index$(\mathcal{O}_k) = k + 1$ and out-index$(\mathcal{O}_k) = 1$. □

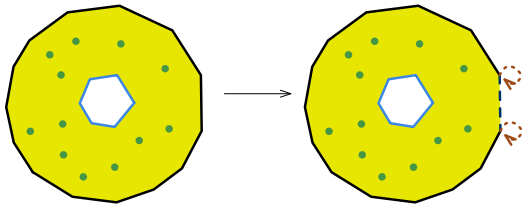

Figure 2: left: a simple ring subproblem right: the transformed ring sector subproblem.

3 Thin Rings

This section is devoted to the proof of the following theorem, which gives an algorithm for solving ring subproblems with a certain width w. This algorithm will be invoked by the main algorithm for values $w \leq \sqrt{n}$. We will sketch the main ideas of the algorithm and its runtime analysis.

Theorem 4 *Given a simple (layer-constrained or layer-unconstrained) ring subproblem $\mathcal{S}$ with width w, there exists an algorithm* RingSec *that computes the number of all valid triangulations of $\mathcal{S}$ in time $n^{(5+o(1))w}$.*

Theorem 4 implies easily Theorem 2 in a similar fashion as Theorem 3 implies Theorem 1.

We use a different kind of separator for this algorithm. This requires a yet more specialized definition of subproblems for our dynamic programming scheme: *ring sector subproblems*. We sketch the proof of Theorem 5. It easily implies Theorem 4, using a simple transformation from ring subproblems to ring sector subproblems, see Figure 2.

Theorem 5 *Given a ring sector subproblem $\mathcal{S}$ with width w on a set of n points, there exists an algorithm, denoted* RingSec *that computes the number of all valid triangulations of $\mathcal{S}$ in $n^{(5+o(1))w}$ time.*

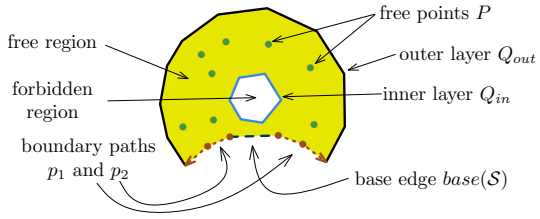

Figure 3: A ring sector subproblem.

Ring Sector subproblems are defined very similar to ring subproblems, see Figure 3. The essential difference is that some part of the outer layer is removed and replaced by two boundary paths and a base edge. These components are introduced for the recursion step as illustrated in Figure 4. A *valid triangulation* of a ring sector subproblem needs to satisfy some additional boundary constraints to ensure the boundary paths are indeed canonical separators of all counted triangulations.

Given a valid triangulation T of a ring sector problem $\mathcal{S}$ it is not hard to show that every vertex on layer i has a neighbor w.r.t. T in layer $i-1$. Furthermore, there is a unique triangle Δ incident to the base edge. From the vertex v of Δ that is not incident to the base edge, exists a path to the outer layer of $\mathcal{S}$ by always choosing an adjacent vertex closer to the outer layer. There is *exactly one* such path p, if we further require that the vertex with lowest order label is taken. (The order label is some distingued number from {1,. . . ,n} that was fixed in advance for each vertex.) Such paths are called *canonical outgoing paths*. We recurse on a ring sector subproblem by guessing all potential such triangles Δ and all potential canonical paths p as described above. For each such path, we can define two subproblems $\mathcal{S}_{\text{right}}$ and $\mathcal{S}_{\text{left}}$, see Figure 4. We can

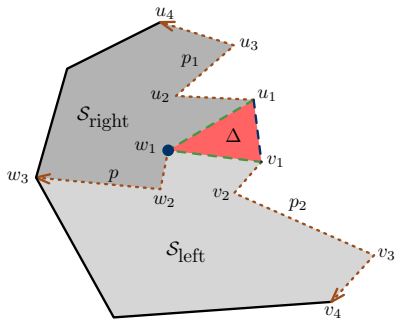

Figure 4: Splitting a ring sector subproblem.

restrict our triangulation T to these subproblems and receive two new triangulations T_{left} and T_{right}, and conversely, given two triangulations T_{left} and T_{right}, we can combine it to a triangulation T. Thus if we recursively count the number of valid triangulations of subproblems $\mathcal{S}_{\text{right}}$ and $\mathcal{S}_{\text{left}}$, then we get *exactly* the number of valid triangulations of $\mathcal{S}$ where Δ is the triangle incident to the base edge and p is the canonical outgoing path starting at vertex v or Δ. Summing up for every possible triangle Δ and path p, we get exactly the number of valid triangulations of $\mathcal{S}$.

If there are layer constraints in $\mathcal{S}$, then we have to do some more work. Let d, d_{left}, and d_{right} be the vectors that indicate the size of the layers for T, T_{left}, and T_{right} respectively. Except for the vertices shared by T_{left} and T_{right}, it holds that d equals $d_{\text{left}} + d_{\text{right}}$. Now, let us go back to our subproblems $\mathcal{S}_{\text{left}}$ and $\mathcal{S}_{\text{right}}$. Let c be the layer-constraint vector of $\mathcal{S}$. Then, we define all pairs of compatible layer-constraints $(c_{\text{left}}, c_{\text{right}})$ such that two valid triangulations for $\mathcal{S}_{\text{right}}(c_{\text{right}})$ and $\mathcal{S}_{\text{left}}(c_{\text{left}})$ respectively give a triangulation for $\mathcal{S}$ with the correct number of vertices on each layer. Here, the technical difficulty is

to take into consideration the vertices shared by both subproblems. Further we need to ensure that vertices in the i-th layer of $\mathcal{S}_{\text{right}}$ will also be in the i-th layer of $\mathcal{S}$. We recurse on all subproblems occurring in this way.

Proof. [Sketch Theorem 5] The correctness of the algorithm follows from the correctness of the recursion. The bound on the running time follows from bounding the time required to solve a subproblem times the number of subproblems. We save our intermediate results in a search tree in order to prevent to handle any subproblem more than once. The bound on the number of subproblems follows from the fact that all of their components are defined by at most two path separators, and from the fact that a separator has at most length w there are at most $n^{O(w)}$ of them. The number of layer constraints is bounded by the assumption that at most $\sqrt{n}$ layers are non-zero. The time for the recursive steps for one subproblem can be asymptotically bounded by the number of recursions, which in turn depends only on the number of potential canonical paths and ways to split the layer constraints in a compatible way. □

4 General Ring Subproblems

We briefly sketch the main algorithm in this section and estimate its running time.

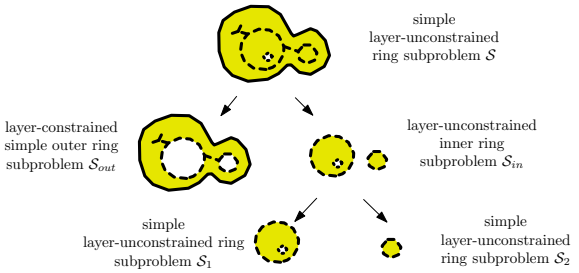

Figure 5: Overview of the algorithm.

The way we solve general ring subproblems is to distinguish two cases. In the case that the ring subproblem is thin, that is, has only few layers ($\leq \sqrt{n}$), we will use the algorithm of Theorem 4 as explained in Section 3. In case that we have many layers ($> \sqrt{n}$), we know that one of the outermost $\sqrt{n}$ layers must be of size $\leq \sqrt{n}$ by the pigeon hole principle. We use this layer as a separator that splits the problem into a thin outer part and an inner part.

To be more explicit, let $\mathcal{S}$ be a ring problem and T be valid triangulation of $\mathcal{S}$. By the outer and inner layer index of $\mathcal{S}$, we already know exactly the number of layers that T has. Consider the case that T has more than $\sqrt{n}$ layers. Then among the $\sqrt{n}$ layers closest to the outer layer, one must have size less than or equal to $\sqrt{n}$. Note that the layer L that is actually closest to the outer layer of $\mathcal{S}$ is *uniquely* determined. We try to guess this layer L, which requires guessing at most $\sqrt{n}$ points. The guess defines an inner ring subproblem $\mathcal{S}_{\text{in}}$ and an outer ring subproblem $\mathcal{S}_{\text{out}}$, as depicted in Figure 5. In case the cactus layer L is disconnected or has disconnected bounded faces, the inner ring subproblems consist of several components. In this case, we split it into smaller subproblems, before we proceed with the recursion.

We can restrict T to $\mathcal{S}_{\text{out}}$ to attain a triangulation T_{out}. It is clear that all layers different from L in T_{out} have size larger than $\sqrt{n}$. Therefore, we want to count only those triangulations of $\mathcal{S}_{\text{out}}$ that have all layers (except L) of size larger than $\sqrt{n}$. We use layer constraints for this purpose: we solve $\mathcal{S}_{\text{out}}$ with every possible layer constraint where every layer is required to have size greater than $\sqrt{n}$.

The running time can be estimated by bounding the total number of ring subproblems times the time spent per ring subproblem. Each ring subproblem is defined by an inner and outer layer and a layer constraint. In the course of the algorithm only inner and outer layers of size less than or equal to $\sqrt{n}$ are guessed, and there are at most $n^{O(\sqrt{n})}$ of them. As we will never constrain more than $\sqrt{n}$ layers to be non-zero, the total number of layer constraints is bounded by $n^{O(\sqrt{n})}$. For the case of rings with width smaller than $\sqrt{n}$, the runtime is given by Theorem 4. In the other case, the bound stems from the total number of layers of size $\sqrt{n}$, which is $n^{O(\sqrt{n})}$.

References

[1] V. Alvarez, K. Bringmann, R. Curticapean, and S. Ray. Counting triangulations and other crossing-free structures via onion layers. *DℰC*, 53(4):675–690, 2015.

[2] V. Alvarez and R. Seidel. A simple aggregative algorithm for counting triangulations of planar point sets and related problems. In *SoCG'13*, pages 1–8, 2013.

[3] E. Anagnostou and D. Corneil. Polynomial-time instances of the minimum weight triangulation problem. *Computational Geometry*, 3(5):247–259, 1993.

[4] C. Knauer and A. Spillner. A fixed-parameter algorithm for the minimum weight triangulation problem based on small graph separators. In *WG'06*, pages 49–57, 2006.

[5] A. Lingas. Subexponential-time algorithms for minimum weight triangulations and related problems. In *CCCG'98*, 1998.

[6] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

[7] D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the Exponential Time Hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.