Counting Triangulations Approximately

Victor Alvarez* Karl Bringmann† Saurabh Ray‡ Raimund Seidel§

Abstract

We consider the problem of counting straight-edge triangulations of a given set P of n points in the plane. Until very recently it was not known whether the *exact* number of triangulations of P can be computed asymptotically faster than by enumerating all triangulations. We now know that the number of triangulations of P can be computed in $O^*(2^n)$ time [2], which is less than the lower bound of $\Omega(2.43^n)$ on the number of triangulations of *any* point set [11]. In this paper we address the question of whether one can approximately count triangulations in sub-exponential time. We present an algorithm with sub-exponential running time and sub-exponential approximation ratio, that is, if we denote by Λ the output of our algorithm, and by c^n the exact number of triangulations of P, for some positive constant c, we prove that $c^n \leq \Lambda \leq c^n \cdot 2^{o(n)}$. This is the first algorithm that in sub-exponential time computes a $(1 + o(1))$-approximation of the base of the number of triangulations, more precisely, $c \leq \Lambda^{\frac{1}{n}} \leq (1 + o(1))c$. Our algorithm can be adapted to approximately count other crossing-free structures on P, keeping the quality of approximation and running time intact. Our algorithm may be useful in guessing, through experiments, the right constants c_1 and c_2 such that the number of triangulations of any set of n points is between c_1^n and c_2^n. Currently there is a large gap between c_1 and c_2. We know that $c_1 \geq 2.43$ and $c_2 \leq 30$.

1 Introduction

Let P be a set of n points on the plane. A crossing-free structure on P is a straight-line plane graph with vertex set P. Examples of crossing-free structures include triangulations and spanning cycles, also known as polygonizations, among others. Let $\mathcal{C}$ denote a certain class of crossing-free structures, and let $\mathcal{F}_{\mathcal{C}}(P)$ denote the set of all crossing-free structures on P belonging to class $\mathcal{C}$. Recently, there has been a significant amount of work regarding the question: "Can the cardinality of $\mathcal{F}_{\mathcal{C}}(P)$ be computed faster than by enumerating $\mathcal{F}_{\mathcal{C}}(P)$?".

In this paper we focus on the particular class $\mathcal{C}$ of all straight-edge triangulations. To the best of our knowledge the first result regarding this question was by O. Aichholzer in '99 [1], where he introduced a geometric structure called "the path of a triangulation", or T-path for short. Using T-paths he showed a divide-and-conquer algorithm that experimentally indicated that triangulations could be counted in time sub-linear in the number of triangulations, that is, the algorithm was apparently faster than enumeration. However, a formal proof of this - or even a good analysis of its running time - seems hard to obtain, since it is not clear how to bound the number of triangulations containing a single T-path. Later, in '05, S. Ray and R. Seidel [8] presented a new algorithm for counting triangulations that, in practice, appeared to be substantially faster than Aichholzer's algorithm. This algorithm is also very hard to analyze, and thus no good analysis of its running time has been presented so far. It took until '12 for new counting algorithms to come up that could be analyzed properly. The first new algorithm is also based on T-paths but uses the sweep-line paradigm [4]. This algorithm was proven to count triangulations in time $O^*(9^n)$. The second new algorithm, based on the onion layers of P and the divide-and-conquer paradigm, was proven to count triangulations in time $O^*(3.1414^n)$ [3]. From the experimental point of view, the second algorithm turned out to be significantly faster, for certain configurations of points, than the one shown in [8]. These experiments can be found in the full version of [3] available on the authors' websites. The third and, so far, last new algorithm for counting triangulations runs in time $O^*(2^n)$ [2]. This last algorithm finally shows that enumeration algorithms for triangulations can indeed *always* be beaten, as all point sets with n points have at least $\Omega(2.43^n)$ triangulations [11]. From an experimental point of view it was also shown to be significantly faster than all previous algorithms on a variety of inputs [2].

1.1 Our Contribution

The $O^*(2^n)$ algorithm of [2] for counting triangulations *exactly* seems hard to beat at this point. However, it would be very interesting to see whether the number

*Fachrichtung Informatik, Universität des Saarlandes, Germany, alvarez@cs.uni-saarland.de.

†Max Planck Institute for Informatics, Karl Bringmann is a recipient of the *Google Europe Fellowship in Randomized Algorithms*, and this research is supported in part by this Google Fellowship. kbringma@mpi-inf.mpg.de.

‡Ben Gurion University of the Negev, Israel saurabh@math.bgu.ac.il.

§Fachrichtung Informatik, Universität des Saarlandes, Germany, rseidel@cs.uni-saarland.de.

of triangulations of P can be *approximated*. The result presented in this paper is, to the best of our knowledge, the first result in this new line of research.

Note that since for all sets of n points the number of triangulations is $\Omega(2.43^n)$ [11] and $O(30^n)$ [10], the quantity $\Theta(\sqrt{30 \times 2.43}^{\,n})$ approximates the exact number of triangulations within a factor of $O(\sqrt{30/2.43}^{\,n})$. Thus, one can trade the exponential time of an exact algorithm for a polynomial time algorithm with exponential approximation ratio. In this paper we bridge the gap between these two solutions by presenting an algorithm with sub-exponential running time and sub-exponential approximation ratio.

Let $\mathcal{F}_T(P)$ denote the set of all triangulations of P. The main result of this paper, whose proof is shown in § 4, is the following:

Theorem 1 *Let P be a set of n points on the plane. Then a number Λ can be computed in time $2^{o(n)}$ such that $|\mathcal{F}_T(P)| \leq \Lambda \leq |\mathcal{F}_T(P)|^{1+o(1)} = 2^{o(n)}|\mathcal{F}_T(P)|$.*

While the approximation factor of Λ is rather big, it is within the same order of magnitude as the real value of $|\mathcal{F}_T(P)|$, that is, we compute a $(1+o(1))$-approximation of the base of the exponentially large value $|\mathcal{F}_T(P)|$. More precisely, for $|\mathcal{F}_T(P)| = c^n$ we have $c \leq \Lambda^{\frac{1}{n}} \leq c^{1+o(1)} \leq (1 + o(1))c$. Also, this approximation can be computed in sub-exponential time, which, at least theoretically, is asymptotically faster than the worst-case running times of the algorithms presented in [4, 3, 2]. This is certainly very appealing.

2 Preliminaries

Our algorithm uses simple cycle separators as the main ingredient, originally presented in [7] by G. L. Miller, and improved in [5] by H. N. Djidjev and S. M. Venkatesan. The following theorem accounts for both results:

Theorem 2 (Separator Theorem) *Let T be a triangulation of a set of n points in the plane such that the unbounded face is a triangle. Then there exists a simple cycle C of size at most $\sqrt{4n}$, that separates the set A of vertices of T in its interior from the set B of vertices of T in its exterior, such that the number of elements of each one of A and B is always at most $\frac{2n}{3}$.*

Observe that the Separator Theorem does not imply that *every* triangulation of a set of points contains a *unique* simple cycle separator. One can easily come up with examples in which a triangulation contains more than one simple cycle separator. The important part here is that *every* triangulation contains *at least* one simple cycle separator.

3 The Algorithm

The idea for an approximate counting algorithm is suggested by the Separator Theorem: We enumerate all possible simple cycle separators C of size at most $\sqrt{4n}$ that we can find in the given set P. Then we recurse in each of the parts A and B of the Separator Theorem delimited by C[I], and add or multiply the numbers thus obtained appropriately, that is, we multiply the number we obtain for the sub-problem $A \cup C$ by the number we obtain for sub-problem $B \cup C$, and we add these products over all cycle separators C. With this algorithm we clearly over-count the triangulations of P, and it remains to show that we do not over-count by too much. We will later see that in order to keep over-counting small, we have to solve small recursive sub-problems exactly. Note that problems of size smaller than a threshold Δ can be solved exactly in time $2^{O(\Delta)}$ [2].

However, there are some technicalities that we have to overcome first. For starters, the Separator Theorem holds only if the unbounded face of T is also a triangle. Thus, if we add a dummy vertex v_∞ outside $\text{Conv}(P)$, along with the adjacencies between v_∞ and the vertices of $\text{Conv}(P)$, to make the unbounded face a triangle, we can apply the Separator Theorem. Once a simple cycle with the dividing properties of a separator is found, by the deletion of v_∞ we are left with a separator that is either the original cycle that we found, if v_∞ does not appear as a vertex of the separator, or a path otherwise. Thus, when guessing a separator we have to consider that it might be a path instead of a cycle. This brings us to the second technical issue. As we go deeper in the recursion we might create "holes" in P whose boundaries are the separators that we have considered thus far. That is, the recursive problems are polygonal regions, possibly with holes, containing points of P. Therefore, when guessing a separator, cycle or path, we have to arbitrarily triangulate the holes first. This does not modify the size of the sets we guess for a separator in a sub-problem.

We can now prove the first lemma:

Lemma 3 *Let $\mathcal{F}_T(P)$ be the set of triangulations of a set P of n points. Then* all *separators, simple cycles or paths, among* all *the elements of $\mathcal{F}_T(P)$ can be computed in time $2^{o(n)}$.*

Proof. We know by the Separator Theorem and the discussion beneath that *every* element of $\mathcal{F}_T(P)$, a triangulation, contains at least one separator C, simple cycle or path. Moreover, the size of C is at most $\sqrt{4n}$. Thus, searching by brute-force will do the job. We can enumerate all the subsets of P of size at most $\sqrt{4n}$ along with their permutations. A permutation is what tells us

[I] Thus separator C also forms part of the two sub-problems.

how to connect the points of the guessed subset, after also guessing whether we have a path or cycle. We can then verify if the constructed simple cycle, or path, fulfils the dividing properties of a separator, as specified in the Separator Theorem.

It is not hard to check that the total number of guessed subsets and their permutations is $2^{O\left(\sqrt{n}\log(n)\right)}$. Verifying whether a cycle, or a path, is indeed a separator can be done in polynomial time. So the total time spent remains being $2^{O\left(\sqrt{n}\log(n)\right)}$. □

We can now proceed with the proof of Theorem 1.

4 Proof of Theorem 1

We will first prove that the approximation ratio is sub-exponential and then prove that the algorithm has sub-exponential running time.

4.1 Quality of approximation

By the proof of Lemma 3 we also obtain that the number of simple cycle separators cannot be larger than $2^{O\left(\sqrt{n}\log(n)\right)}$. Since at *every* stage of the recursion of the counting algorithm *no* triangulation of P can contain more than the total number of simple cycle separators found at that stage, we can express the *over-counting factor* of the algorithm by the following recurrence:

$$\begin{aligned} S(P,\Delta) &\leq \sum_{C} S(A\cup C,\Delta)\cdot S(B\cup C,\Delta) \\ &\leq 2^{O\left(\sqrt{n}\log(n)\right)}\cdot S(A\cup C^*,\Delta)\cdot S(B\cup C^*,\Delta), \end{aligned}$$

where the summation is over all separators C available at the level of recursion, $A\cup C$, $B\cup C$ are the sub-problems as explained before, C^* is the cycle that maximizes $S(A\cup C,\Delta)\cdot S(B\cup C,\Delta)$ over all C, and Δ is a stopping size. Specifically, whenever the current recursive sub-problem contains $\leq \Delta$ points we stop the recursion and compute the number of triangulations of the sub-problem exactly. Hence, we have $S(P',\Delta)=1$ whenever $|P'|\leq\Delta$. We can now write

$$\begin{aligned} S'(P,\Delta) := \log(S(P,\Delta)) \leq O\left(\sqrt{n}\log(n)\right) &+ \\ S'(A\cup C^*,\Delta) &+ \\ S'(B\cup C^*,\Delta). & \end{aligned}$$

Our goal now is to prove the following lemma:

Lemma 4 *Let P be a set of n points on the plane and assume $\Delta = n^{\Omega(1)}$, $n>\Delta$, and Δ is at least a sufficiently large constant. Then we have*

$$S'(P,\Delta) = O\left(\left(\frac{n}{\sqrt{\Delta/3}}-\sqrt{n}\right)\log\Delta\right).$$

Proof. We use induction over the size of P. Let $P'\subseteq P$ of size $m\leq n$. We have,

$$\begin{aligned} S'(P',\Delta) &\leq O\left(\sqrt{m}\log(m)\right) + S'(A\cup C^*,\Delta) + \\ &\quad S'(B\cup C^*,\Delta) \\ &\leq O\left(\sqrt{m}\log(m)\right) + \\ &\quad c\left(\frac{m_1}{\sqrt{\frac{\Delta}{3}}}-\sqrt{m_1}+\frac{m_2}{\sqrt{\frac{\Delta}{3}}}-\sqrt{m_2}\right)\log\Delta, \quad (1) \end{aligned}$$

where m_1, m_2 are the sizes of the sub-problems $A\cup C^*$ and $B\cup C^*$ of P', respectively, and c is some sufficiently large positive constant. By the Separator Theorem, we can express $m_1\leq \alpha m+\sqrt{4m}$ and $m_2\leq \beta m+\sqrt{4m}$, such that: (1) α,β are constants that depend on the instance, so $\alpha=\alpha\left(A\cup C^*\right)$ and $\beta=\beta\left(B\cup C^*\right)$, (2) $0<\beta\leq\alpha\leq\frac{2}{3}$, and (3) $\alpha+\beta=1$.

Now let us for the moment focus on the term $\frac{m_1}{\sqrt{\Delta/3}}-\sqrt{m_1}+\frac{m_2}{\sqrt{\Delta/3}}-\sqrt{m_2}$ of equation (1) above:

$$\begin{aligned} &\frac{m_1}{\sqrt{\frac{\Delta}{3}}}-\sqrt{m_1}+\frac{m_2}{\sqrt{\frac{\Delta}{3}}}-\sqrt{m_2} \\ &\quad= \frac{m_1+m_2}{\sqrt{\frac{\Delta}{3}}}-\sqrt{m_1}-\sqrt{m_2} \\ &\quad\leq \frac{\alpha m+\sqrt{4m}+\beta m+\sqrt{4m}}{\sqrt{\frac{\Delta}{3}}}-\sqrt{m_1}-\sqrt{m_2} \\ &\quad\leq \frac{m+4\sqrt{m}}{\sqrt{\frac{\Delta}{3}}}-\sqrt{\alpha m}-\sqrt{\beta m} \\ &\quad= \frac{m+4\sqrt{m}}{\sqrt{\frac{\Delta}{3}}}-\sqrt{m}\left(\sqrt{\alpha}+\sqrt{\beta}\right) \\ &\quad\leq \frac{m+4\sqrt{m}}{\sqrt{\frac{\Delta}{3}}}-\sqrt{m}\,(1+\varepsilon) \end{aligned}$$

The last inequality is obtained by minimizing $\sqrt{\alpha}+\sqrt{\beta}$. Since we mentioned before that $0\leq\beta\leq\alpha\leq\frac{2}{3}$ and $\alpha+\beta=1$, the minimum of $\sqrt{\alpha}+\sqrt{\beta}$ is attained at $(\alpha,\beta)=\left(\frac{2}{3},\frac{1}{3}\right)$, and is strictly larger than one, so we can choose $\varepsilon>0$. Now, since Δ is sufficiently large, we have $\frac{4\sqrt{m}}{\sqrt{\Delta/3}}\ll\varepsilon\sqrt{m}$, so $\frac{4\sqrt{m}}{\sqrt{\Delta/3}}-\varepsilon\sqrt{m}=-\varepsilon'\sqrt{m}$, for some positive constant ε'. Thus we can continue as follows:

$$\begin{aligned} \frac{m_1}{\sqrt{\frac{\Delta}{3}}}-\sqrt{m_1}+\frac{m_2}{\sqrt{\frac{\Delta}{3}}}-\sqrt{m_2} &\leq \frac{m+4\sqrt{m}}{\sqrt{\frac{\Delta}{3}}}-\sqrt{m}\,(1+\varepsilon) \\ &\leq \frac{m}{\sqrt{\frac{\Delta}{3}}}-\sqrt{m}-\varepsilon'\sqrt{m}. \quad (2) \end{aligned}$$

Combining equations (1) and (2) we obtain

$$S'(P',\Delta) \leq O\left(\sqrt{m}\log(m)\right) + c\left(\frac{m}{\sqrt{\frac{\Delta}{3}}} - (1+\varepsilon')\sqrt{m}\right)\log\Delta$$
$$\leq c\left(\frac{m}{\sqrt{\frac{\Delta}{3}}} - \sqrt{m}\right)\log\Delta + O\left(\sqrt{m}\log(m)\right) - c\cdot\varepsilon'\sqrt{m}\log\Delta$$

If we choose Δ to be sufficiently large, say $\Delta \geq n^\delta$, for some constant $\delta > 0$, then we have $\Delta \geq n^\delta \geq m^\delta$, and the negative term $-c\cdot\varepsilon'\sqrt{m}\log\Delta$ is larger, for appropriately large c, than the $O\left(\sqrt{m}\log(m)\right)$ term. Hence, we can conclude that $S'(P',\Delta) \leq O\left(\left(\frac{m}{\sqrt{\Delta/3}} - \sqrt{m}\right)\log\Delta\right)$, which proves the induction step.

It still remains to prove that the inductive claim holds for the boundary condition, so let Q be a recursive sub-problem of size $\leq \Delta$. As Q stems from an application of the Separator Theorem, it is easy to see that $|Q| \geq \frac{\Delta}{3}$. Thus, we have $S'(Q,\Delta) = 0 \leq c\left(\frac{|Q|}{\sqrt{\Delta/3}} - \sqrt{|Q|}\right)\log\Delta$. Lemma 4 follows. □

Now, let Λ be the number computed by our algorithm. Recall that $|\mathcal{F}_T(P)|$ is the exact number of triangulations of P. By setting $\Delta = \sqrt{n}\log(n)$ we obtain an over-counting factor of the algorithm of:

$$S(P,\Delta) = 2^{S'(P,\Delta)} = 2^{O\left(\frac{n\log\Delta}{\sqrt{\Delta}}\right)} = 2^{O\left(n^{\frac{3}{4}}\sqrt{\log(n)}\right)}$$

Hence $|\mathcal{F}_T(P)| \leq \Lambda \leq |\mathcal{F}_T(P)|\cdot 2^{O\left(n^{\frac{3}{4}}\sqrt{\log(n)}\right)} = |\mathcal{F}_T(P)|^{1+o(1)}$. This completes the qualitative part of Theorem 1. It remains to discuss the running time of our algorithm.

4.2 Running time

The running time of the algorithm can be expressed with the following recurrence:

$$T(n) < 2^{O\left(\sqrt{n}\log(n)\right)}T\left(\frac{2n}{3} + \sqrt{4n}\right)$$

Taking again $T'(n) = \log(T(n))$ yields $T'(n) := T'\left(\frac{2n}{3} + \sqrt{4n}\right) + O\left(\sqrt{n}\log(n)\right)$, which can then be solved using the well-known Akra-Bazzi Theorem for recurrences, see [6]. This yields $T'(n) = O\left(\sqrt{n}\log(n)\right)$. There is, however, one detail missing, the stopping condition Δ. In order to use the Akra-Bazzi Theorem we need a boundary condition of $T(n) = 1$ for $1 \leq n \leq n_0$ (for some constant n_0), but in the algorithm we stop the recursion whenever a sub-problem Q is of size $\leq \Delta$ (which is dependent on the size n of the original point set). At that point we compute the exact number of triangulations of Q, which gives $T(|Q|) = 2^{O(|Q|)} = 2^{O(\Delta)}$. Hence the exponent in the running time of the algorithm is upper-bounded by the solution of $T'(n)$, as given by the Akra-Bazzi Theorem, plus a factor of $O(\Delta)$, *i.e.*, $T(n) = 2^{O\left(\sqrt{n}\log(n)+\Delta\right)}$. If as before we assume that $\Delta = \sqrt{n}\log(n)$ then we end up with $T(n) = 2^{O\left(\sqrt{n}\log(n)\right)} = 2^{o(n)}$, which concludes the proof of Theorem 1.

As a final remark observe that we could have used other values for Δ, rather than $\sqrt{n}\log(n)$, without violating any argument in the proofs. This yields a trade-off with running time $2^{O(\Delta)}$ and approximation ratio $2^{O(\frac{n\log\Delta}{\sqrt{\Delta}})}$ for any $\sqrt{n}\log(n) \leq \Delta \leq n$. Although the quality of the approximation improves with larger Δ, the running time increases. Since we see no way of not having over-counting with this algorithm, we regard $\Delta = \sqrt{n}\log(n)$ as the most reasonable setting. Note that it remains an open problem to find an algorithm with sub-exponential approximation ratio and running time $2^{o(\sqrt{n}\log(n))}$, e.g, polynomial.

5 Extensions and Conclusions

We note that with the techniques of [3] one can generalize our algorithm for approximately counting triangulations to approximately counting other crossing-free structures. See [9] for other related results. In the following we sketch this for spanning cycles. We embed a spanning cycle in its contrained Delauney triangulation, annotating the edges by whether they belong to the spanning cycle. To approximate the number of spanning cycles of a given set of points, we enumerate all possible cycle separators *together with all incident triangles* (similar to *sn*-paths versus triangular paths in [3]). Moreover, we enumerate all annotations of this structure by which edges belong to the spanning cycle and the topology of the seen parts of the spanning cycle. This gives an algorithm with the same asymptotic running time and approximation ratio as for triangulations. The full details of this, along with the details of how to approximately count crossing-free matchings and spanning trees, will appear in the full version of this paper.

In summary, in this paper we presented an approximation algorithm for the number of triangulations of a given point set. This algorithm has sub-exponential running time and sub-exponential approximation ratio. Although its approximation ratio is rather large, the algorithm computes a $(1+o(1))$-approximation of the base c of the number of triangulations c^n, and it does so in sub-exponential time. No algorithm with this property was known before.

References

[1] O. Aichholzer, "The path of a triangulation," in *Symposium on Computational Geometry*, pp. 14–23, 1999.

[2] V. Alvarez and R. Seidel, "A Simple Aggregative Algorithm for Counting Triangulations of Planar Point Sets and Related Problems," in *29th ACM Symposium on Computational Geometry* To appear, ACM, 2013.

[3] V. Alvarez, K. Bringmann, R. Curticapean, and S. Ray, "Counting crossing-free structures," in *Symposium on Computational Geometry* (T. K. Dey and S. Whitesides, eds.), pp. 61–68, ACM, 2012.

[4] V. Alvarez, K. Bringmann, and S. Ray, "A simple sweep line algorithm for counting triangulations and pseudo-triangulations", *Submitted*, 2012.

[5] H. Djidjev and S. M. Venkatesan, "Reduced constants for simple cycle graph separation," *Acta Inf.*, vol. 34, no. 3, pp. 231–243, 1997.

[6] T. Leighton, "Notes on better master theorems for divide-and-conquer recurrences," tech. rep., Massachusetts Institute of Technology, 1996.

[7] G. L. Miller, "Finding small simple cycle separators for 2-connected planar graphs," *JCSS*, vol. 32, pp. 265–279, June 1986.

[8] S. Ray and R. Seidel, "A simple and less slow method for counting triangulations and for related problems," in *EuroCG*, 2004.

[9] A. Razen and E. Welzl, "Counting Plane Graphs with Exponential Speed-Up" in *Rainbow of Computer Science*, pp. 36-46, 2011.

[10] M. Sharir and A. Sheffer, "Counting triangulations of planar point sets," *Electr. J. Comb.*, vol. 18, no. 1, 2011.

[11] M. Sharir, A. Sheffer, and E. Welzl, "On degrees in random triangulations of point sets," *J. Comb. Theory, Ser. A*, vol. 118, no. 7, pp. 1979–1999, 2011.