Computational Geometry 48 (2015) 386–397

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

Counting triangulations and other crossing-free structures approximately

Victor Alvarez [a], Karl Bringmann [b,1], Saurabh Ray [c], Raimund Seidel [d]

[a] Information Systems Group, Universität des Saarlandes, Germany
[b] Max Planck Institute for Informatics, Germany
[c] Ben Gurion University of the Negev, Israel
[d] Fachrichtung Informatik, Universität des Saarlandes, Germany

ARTICLE INFO

Article history:
Available online 18 December 2014

Keywords:
Algorithmic geometry
Crossing-free structures
Counting algorithms
Triangulations
Approximation algorithms

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 [9], which is less than the lower bound of $\Omega(2.43^n)$ on the number of triangulations of *any* point set [30]. 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, denoting 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. In this paper we show how to do this for matchings and spanning trees.

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, trees, matchings, and spanning cycles, also known as the polygonizations of P, among others. Let X denote certain type of crossing-free structures and let $\mathcal{F}_X(P)$ denote the set of all crossing-free structures on P of type X. One of the most intriguing problems in Computational Geometry is the following: given P and a particular type of crossing-free structures X, how fast can we compute the cardinality of $\mathcal{F}_X(P)$?

Among all kinds of crossing-free structures on P, triangulations are perhaps the most studied ones, so let us first focus on them. How fast can we compute the number of triangulations of P? For starters, although triangulations are the most studied, the problem of computing the number of *all* triangulations of P is, in general, neither known to be #P-hard, nor do we know of a polynomial time algorithm to even approximate that number. When the point set is in convex position,

E-mail addresses: alvarez@cs.uni-saarland.de (V. Alvarez), kbringma@mpi-inf.mpg.de (K. Bringmann), saurabh@math.bgu.ac.il (S. Ray), rseidel@cs.uni-saarland.de (R. Seidel).

[1] Karl Bringmann is a recipient of the *Google Europe Fellowship in Randomized Algorithms*, and this research is supported in part by this Google Fellowship.

http://dx.doi.org/10.1016/j.comgeo.2014.12.006
0925-7721/

an easy recurrence relation can be derived showing that the number of triangulations spanned by n points in convex position is C_{n-2}, where C_k is the k-th Catalan number. F. Hurtado and M. Noy [21] were able to find further formulas to exactly compute the number of triangulations of "almost convex sets". Unfortunately, the approach of finding exact formulas does not seem to take us much further.

For *any* given set of points P it is possible to enumerate all its triangulations in time proportional to its number of triangulations. This is because the *flip graph*[2] of triangulations is connected, see [23,33]. Thus any graph traversal algorithm like DFS or BFS can be used to enumerate the vertices of the flip graph of triangulations of P. One limitation of such traversal algorithms is, however, that the amount of memory used is proportional to the number of vertices in the graph — which is known to *always* be exponential, because the number of triangulations of P lies between $\Omega(2.43^n)$ [30] and $O(30^n)$ [29]. Using a general technique due to D. Avis and K. Fukuda, called Reverse Search [10], it is possible to enumerate triangulations while keeping the memory usage polynomial in n. This technique has been further improved in [11,22]. However, it is important to observe that, since the number of triangulations is exponential in n, counting triangulations by enumeration takes exponential time, so a natural question is whether one can do significantly better.

At this point let us mention that it is not known whether the aforementioned bounds of $\Omega(2.43^n)$ and $O(30^n)$ for the number of triangulations are realizable, i.e., whether there are indeed set of points having that many triangulations. In terms of realizability, the best we know at this point is that there exists a set of n points having $O^*(3.47^n)$[3] [3] triangulations, but there is also a set of points where this number grows to at least $\Omega(8.65^n)$ [15]. This leads to the interesting observation that the number of triangulations is not minimized when the set of points is in convex position; we already mentioned that the number of triangulations of set of n points in convex position is C_{n-2}, which asymptotically grows as $\Theta^*(4^n)$.

In [1] O. Aichholzer introduced the notion of the "path of a triangulation", T-path from now on, that allows a divide-and-conquer approach to speed up counting. From empirical observations, this approach seems to count triangulations in time sub-linear in the number of triangulations, that is, apparently faster than enumeration. Unfortunately, no proof is known that this algorithm is *always* faster than enumeration, and no good analysis of its running time has been obtained yet. Subsequently, a simple algorithm based on dynamic programming was presented in [26]. This algorithm empirically appears to be substantially faster than Aichholzer's algorithm. From the limited empirical data, the running time seems to be proportional to the square root of the number of triangulations. However, as with Aichholzer's algorithm, the worst case running time of this algorithms seems difficult to analyze and no bound on the running time has been found.

Very recently two algorithms whose running times could actually be analyzed were presented. The first algorithm, presented in [8], combines Aichholzer's idea of T-paths with a sweep-line algorithm and runs in time proportional to the largest number of T-paths found during execution (within a $poly(n)$ factor). In [8] the number of T-paths is shown to be at most $O(9^n)$. It is important to observe that, for very particular and well-studied configurations of points, the number of T-paths can be shown to be significantly smaller than the number of triangulations. Thus, at least for those configurations, the algorithm of [8] counts triangulations faster than by using enumeration techniques. Unfortunately, this algorithm turned out to be very slow in practice, which is most probably due to the fact that the number of T-paths can still be very large — in [14] a configuration having at least $\Omega(4^n)$ T-paths was shown.

The second algorithm, presented in [6], uses a divide-and-conquer approach based on the onion layers of the given set of points. This algorithm was shown to have a running time of at most $O^*(3.1414^n)$ and is likely to have a running time sub-linear in the number of triangulations since it is widely believed that the number of triangulations spanned by *any* set of n points is at least $\Omega^*(3.47^n)$,[4] see [3,4,28]. 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 [26]. These experiments can be found in [7]. Finally, in 2013, an algorithm with running time $O^*(2^n)$ was presented [9]. This last algorithm 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. From an experimental point of view it was also shown to be significantly faster than all previous algorithms on a variety of inputs [9].

With respect to crossing-free structures other than triangulations the situation is very similar. In [22] a general framework for enumerating crossing-free structures (including spanning trees and perfect matchings) was presented. However, as for triangulations, the number of enumerated objects is again in general exponential. Table 1 gives the latest asymptotic bounds on the number of triangulations, perfect matchings, and spanning trees, all crossing-free, found on a set of n points P in the plane. The symbols $\leq$, $\geq$ should be understood as upper and lower bound, respectively.

We remark that it has been proven that the number of perfect matchings, and the number of spanning trees, among others, is minimized when P is in convex position, see [2] and references therein. That is, the selected (emphasized) lower bounds in Table 1 are tight, i.e., there exists a set of n points P having at most that many perfect matchings [17], and spanning trees [16]. The interested reader is referred to [2] for a list of other classes of crossing-free structures on P whose cardinality is minimized when P is in convex position, and to [15,32] for up-to-date lists of asymptotic bounds for other crossing-free structures.

[2] The flip graph is a graph whose vertex set is the set of triangulations of P and where there is an edge between vertices of the flip graph if both corresponding triangulations can be transformed into each other by flipping exactly one of their edges.

[3] In the notation $\Omega^*(\cdot)$, $O^*(\cdot)$, and $\Theta^*(\cdot)$ we neglect polynomial terms and we just present the dominating exponential term.

[4] The exact term is $\sqrt{12}^{n-\Theta(\log n)}$ [3].