Generating Random Star-Shaped Polygons (Extended Abstract)

Christian Sohler
Heinz Nixdorf Institute and Dept. of Mathematics and Computer Science,
University of Paderborn, D-33095 Paderborn, Germany
e-mail: csohler@uni-paderborn.de

Abstract

In this paper we deal with two problems on star-shaped polygons. First, we present a Las-Vegas algorithm that uniformly at random creates a star-shaped polygon whose vertices are given by a point set S of n points in the plane that does not admit degenerate star-shaped polygons. The expected running time of the algorithm is $O(n^2 \log n)$ and it uses $O(n)$ memory. We call a star-shaped polygon degenerate if its kernel has 0 area.

Secondly, we show how to count all star-shaped polygons whose vertices are a subset of S in $O(n^5 \log n)$ time and $O(n)$ space. The algorithm can also be used for random uniform generation. We also present lower and upper bounds on the number of star-shaped polygons.

1 Introduction

The random generation of geometric objects recently has received some attention by researchers [1][2][4]. One of the most challenging among these problems is the generation of simple polygons uniformly at random. Since it is not known whether there exists a polynomial time algorithm to solve this problem, researchers either try to use heuristics that do not have a uniform distribution or restrict on certain classes of polygons such as monotone or star-shaped polygons.

Recently, Aichholzer [5] discovered the first algorithm to count triangulations and simple polygons that has a running time sublinear in the number of triangulations (but still exponential in the number of points).

The main application for the random generation of geometric objects is to simplify the evaluation of algorithms. It is often not possible to get data from real world applications when testing algorithms and a random test set is a good alternative.

In this paper we discuss the special case of star-shaped polygons. A polygon P is *star-shaped*, if there exists a point p such that all line segments pv are inside P for each vertex v of P. The *kernel* of a star-shaped polygon P is the union of all points that satisfy the above condition. It is *degenerate* if the kernel has zero area (is a single point or line segment).

Given a point set S of n points in the plane in general position we discuss lower and upper bounds on the number of degenerate and non-degenerate star-shaped polygons whose vertex set is S.

Then we present a Las-Vegas style algorithm (an algorithm that with constant probability a computes a solution while with probability $1 - a$ it answers "did not find a solution; try again") to compute a random star-shaped polygon uniformly distributed among the set of all non-degenerate star-shaped polygons over the point set S. The algorithm needs $O(n^2 \log n)$ expected running time and $O(n)$ space improving over the $O(n^4)$ time and space algorithm of Auer and Held [1] (which can be generalized to deal with degenerate cases). If we generalize our algorithm to degenerate cases, the distribution of the polygons will no longer be uniform. Although the restriction to non-degenerate polygons reduces the worst case number of polygons from an exponential size to a polynomial, there are good reasons to consider this special case. First of all, for random point sets (e.g., uniformly distributed in the unit cube) the probability of degenerate cases is 0. Secondly, when designing prototypes of algorithms, we often do not want to consider degenerate cases since this increases implementation time.

Finally, we show that there is a polynomial time algorithm to count all (including degenerate ones) different star-shaped polygons whose vertex set is a subset of S. The running time of this algorithm is $O(n^5 \log n)$ and it uses $O(n)$ space. It can be used for uniform generation, as well.

2 Lower and Upper Bounds on the Number of Star-Shaped polygons

2.1 Non-degenerate Case

Auer and Held [1] showed that the $O(n^4)$ complexity of the induced line arrangement obtained by drawing lines through each pair of points of S is an asymptotic upper bound on the number of non-degenerate star-shaped polygons.

Certain point sets achieve this bound: place $n-2$ points on the parabola $y = x^2$, half of them to the left and the other half to the right of the y-axis. Finally, add the two remaining points at $(-\infty, -\infty)$ and $(\infty, -\infty)$. Then the resulting point set has $\Omega(n^4)$ different non-degenerate star-shaped polygons.

This can be verified using the following construction that defines all kernels of non-degenerate star-shaped polygons of a point set S: for any pair of points take the two half-lines defined by the line through these points without the segment connecting them. Intersect with $CH(S)$ to get two segments. Put these segments and all bounding segments of the convex hull into a set T. The subdivision induced by T then defines all kernels (the figure below describes a kernel subdivision for a set of five points). Using this construction we can immediately apply the line segment intersection algorithm of Bentley and Ottmann[3]. Then we have an $O((n^2+k)\log n)$ time and $O(n^2)$ space (using the method from [6]) algorithm to count all star-shaped polygons (including degenerate ones), where k is the number of non-degenerate star-shaped polygons. The algorithm of Agarwal [7] can also be used to obtain an $O(n^{8/3} polylog\ n)$ time algorithm to count all non-degenerate star-shaped polygons.

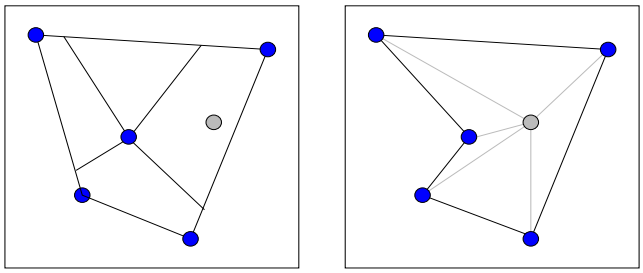

figure 1: a kernel subdivision(all points within a cell define exactly one star-shaped polygon); drawing lines from a point within $CH(S)$ defines a star-shaped polygon; when the grey point moves across the boundary of the polygon's kernel, the polygon changes.

2.2 Degenerate Case

The following construction creates a point set with $\Omega(2^{n/2})$ different star-shaped polygons: Take a random point p in the plane (this point is not put into S) and shoot $\frac{n}{2}$ half - lines away from p. Choose two points on each half-line and put them into S. If necessary, put one or two points into S such that p is in $CH(S)$. Then S has $\Omega(2^{n/2})$ valid star-shaped polygons, since any two points on a half-line can appear in two different orders on the boundary of the polygon (p is the kernel for these polygons). On the other hand, the induced line arrangement of S has at most $O(n^4)$ vertices. Each vertex has at most degree $n/2$. Thus there is an upper bound of $O(2^{n/2}n^4)$ on the number of degenerate star-shaped polygons.

Note that it is possible to check in $O(n \log n)$ time whether a given vertex of the kernel subdivision is degenerate (this can be done by sorting the point set clockwise around the vertex).

3 Generating Star-Shaped Polygons

First we describe how the algorithm works in general. Details and analysis are given later in this section. The idea is to select a random face of the kernel subdivision without computing the whole subdivision. Let S be a point set that does not admit any degenerate star-shaped polygons. At the top level the algorithm can be described as follows:

Step 1: Compute $CH(S)$, if S is in convex position, return $CH(S)$.

Step 2: Randomly select two segments from T.

Step 3: if the two segments do not intersect in a valid point p, repeat from step 2.

Step 4: construct the faces incident to p.

Step 5: for each such face normalize the probability that it is selected using the number of valid incident points.

Step 6: use the normalized probabilities to select a face (further explained below); if a face is selected construct the corresponding polygon, if not restart.

Now the parts of the algorithm are explained in more detail.

3.1 Selecting Segments

To randomly select a cell from the kernel subdivision we first try to find a vertex of this subdivision. Valid vertices are intersections between segments of T except for the points in S. Note that any face is incident to at least one valid vertex since S is in general position. To find a valid vertex we randomly select two segments from T (to be more precise, we select halflines and compute their intersection with $CH(S)$ in $O(\log n)$ time) .

If these two segments do not intersect, we try again. We will now give an upper bound on the number of tries needed to find such a segment. Obviously, each segment except for the boundary segments of $CH(S)$ has at least one valid intersection point (the intersection with the boundary edges of $CH(S)$). Let k be the number of points of S in the interior of $CH(S)$ and $n = |S|$. Then there are $k(n-1) + (n-k) = k(n-2) + n$ segments with at least $k(n-1)$ valid intersections. Overall, the probability that an intersection point is found, is at least $\frac{k(n-1)}{(k(n-2)+n)^2} = \Omega(\frac{1}{kn}) = \Omega(\frac{1}{n^2})$. Thus the expected number of tries until an intersection point is selected is $O(n^2)$.

3.2 Face Construction

Since S does not admit any degenerate star-shaped polygon, there are exactly 4 faces incident to a given valid vertex v. We can construct these faces using $O(n^2 \log n)$ time and $O(n)$ space. For each point p in the interior of $CH(S)$ we compute all $n-1$ half-lines starting at p. Obviously, there is a sector bounded by two half-lines h_1, h_2 that contains v. The two half-lines h_1 and h_2 together with v define two half-planes (we extend the half-lines to lines). We then compute the intersection of the current convex region with this these two half-planes (or equivalently, with the sector bounded by h_1 and h_2). Finally, we add the two segments defining v and we constructed the faces incident to v. Time and space bounds are immediate.

3.3 Uniform Distribution

Once the faces incident to the selected intersection point are constructed, we count the number of valid points on the boundary of each face. We then select one of the faces according to the following simple rule: The probability that a face f is selected as the kernel of a polygon is $\frac{1}{4 \cdot v}$ where v is the number of valid points on the boundary of f. Note that the probabilities do not usually sum up to 1. There is a chance that none of the faces is selected. Then the algorithm restarts.

The overall probability that a certain face f is selected is $\frac{v}{q} \cdot \frac{1}{4 \cdot v} = \frac{1}{4 \cdot q}$ where q is the number of overall valid points. Clearly, two segments have at most one intersection point and vice versa every intersection point has exactly two defining segments. Therefore, the valid points are uniformly distributed.

The overall probability that any face is selected is $\frac{F}{4 \cdot q}$ where F is the overall number of faces. Since $F = \Theta(q)$ this probability is constant and thus the expected number of tries to select a face is also constant.

3.4 Summary

Our algorithm selects in $O(n^2 \log n)$ expected time a valid intersection point. Then in $O(n^2 \log n)$ time the incident faces are constructed and with constant probability one of them is selected. Otherwise, the algorithm restarts. When a cell has been selected the corresponding polygon can be constructed in $O(n \log n)$ time. Thus, the overall expected running time of the algorithm is $O(n^2 \log n)$.

4 Counting Star-Shaped Polygons Whose Vertices are a Subsets of S

In this section we describe how to count all star-shaped polygons whose vertex set is a subset of S. First, take a look at the planar subdivision K which is the intersection between $CH(S)$ and the induced line arrangement of S. Observe that K is a refinement of all possible kernels of subsets of S. In this case, we must also consider segments between points in S, because each segment is a boundary segment of $CH(S')$ for some subset $S' \subset S$ and thus it is a boundary segment for some kernel.

Let L be the set of lines induced by S. The defining segment of a line in L is the segment between the two points of S defining this line. To avoid counting polygons multiple times, we implicitly assign to each intersection point p of two lines l_1, l_2 from L a number of polygons. To be more precise, a polygon is assigned to such a point p, if p is the rightmost vertex of the polygon's kernel. Then for each such point we have to count the number of assigned polygons and we are done.

The lines l_1 and l_2 induce 4 sectors on the plane. We call the sector left of the vertical line through their intersection point the kernel sector. Observe that any kernel is convex. Then it follows that only those polygons are charged to p whose kernel is a subset of the kernel sector and also has p as one of its vertices. The algorithm works like the following: For each pair of lines $l_1, l_2 \in L$ with defining segments s_1 and s_2 we compute the intersection point p of l_1 and l_2. Then we distinguish the following 4 cases:

Case 1: s_1 and s_2 intersect; their intersection point is in the interior of both segments.

Case 2: s_1 and s_2 intersect in a common endpoint. This point is a point of S.

Case 3: s_1 and s_2 do not intersect, but l_1 intersects s_2 or l_2 intersects s_1.

Case 4: only l_1 and l_2 intersect.

First, observe that s_1 and s_2 are always boundary segments of a polygon whose rightmost kernel vertex is the intersection of l_1 and l_2. Thus in case 1 no valid polygon exists, since s_1 and s_2 intersect. In the second case

(see also the figure below) all sets including the 3 endpoints of the segments and a subset of vertices in the kernel sector form a unique star shaped polygon with rightmost kernel vertex p. Thus the number of these polygons is 2^k where k is the number of vertices in the kernel sector.

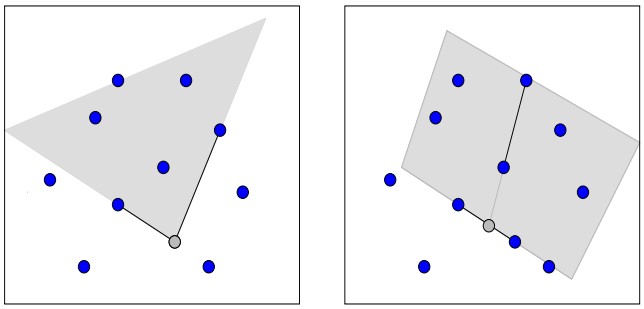

figure 2: Case 2 and 3

Case 3 can be handled similarly. W.l.o.g. let s_1 intersect l_2 and let h be the halfplane that contains s_2 and is bounded by l_1. Then any subset of points of $S \cap h$ together with the 4 defining points of l_1 and l_2 defines a star-shaped polygon. Let k be the number of these points. Then there are 2^k such polygons.

The last case is slightly more difficult. Here any subset $S' \subset S$ defines a polygon charged to p, if p is in $CH(S')$ and if the 4 defining points of l_1 and l_2 are in S'. Now assign the numbers 1 to 4 to the sectors induced by l_1 and l_2 in clockwise order starting with the kernel sector. From now on we regard only subsets S' of S that include the 4 defining points of l_1 and l_2. Then it is obvious, that p is in $CH(S')$ if there exists a vertex in S' that is located in the third sector. If there is no such vertex, p is inside $CH(S')$ if there is an edge between a sector 2 and a sector 4 vertex of S' that does not cross sector 1.

The number of choices for the first subcase is easy to count: let k_i be the number of points in the i-th sector, then we have $(2^{k_3}-1)\cdot 2^{n-4-k_3}$ different valid polygons. To compute the number of valid polygons that do not have a vertex in sector 3, we proceed as follows:

First sort the points in sector 2 and 4 in clockwise order around v. Then for each point q in sector 4 compute the number $k_{2,invalid}$ of points in sector 2 that are on the wrong side of the line through p and q, that is the number of points such that a segment between q and any of these points crosses the kernel sector. Let $k_{4,invalid}$ be the number of points smaller or equal to q in clockwise order around v. Then the number of valid polygons that have q as the smallest boundary vertex in sector 4 is given by $2^{k_1+k_{2,invalid}+k_4-k_{4,invalid}} \cdot (2^{k_2-k_{2,invalid}}-1) = 2^{k_1+k_2+k_4-k_{4,invalid}} - 2^{k_1+k_{2,invalid}+k_4-k_{4,invalid}}$. Obviously, we can evaluate the positive and negative terms separately and then compute their difference. Since we only have to add powers of two and compute one time the difference of two large numbers, the whole computation time for a single point p requires only $O(n \log n)$ time dominated by sorting.

Now that we can deal with all cases, we simply compute all intersection points one by one and deal with each intersection point separately using the above case distinction. Each such case requires at most $O(n \log n)$ time. Thus the running time of the algorithm is $O(n^5 \log n)$ since there are $O(n^2)$ different segments and its space requirement is $O(n)$. We can adapt the algorithm to deal with degenerate cases without further time or space requirements. In this abstract we omit the details.

5 Acknowledgment

The author would like to thank Tamás Lukovszki for several useful discussions.

References

[1] T. Auer, M. Held, Heuristics for the Generation of Random Polygons, Proc. 8th Canadian Conference on Computational Geometry (CCCG'96), pages 38-44, 1996

[2] C. Zhu, G. Sundaram, J. Snoeyink, and J.S.B. Mitchell, Generating Random Polygons with Given Vertices, Comput. Geom. Theory and Appl., 6(5):277-290, 1996

[3] J.L. Bentley and T.A. Ottmann, Algorithm for Reporting and Counting Geometric Intersections, IEEE Trans. Comput., C-28:643-647, 1979

[4] J. O'Rourke and M. Virmani, Generating Random Polygons, Technical Report 011, CS Dept. Smith College, Northhampton, MA 01063, July 1991

[5] O. Aichholzer, The Path of a Triangulation, Proc. 15th Annu. ACM Symposium Comput. Geom., June 1999

[6] J.Pach, M.Sharir, On vertical visibility in arrangements of segments and the queue size in the Bentley-Ottmann line sweeping algorithm, SIAM J. Comput., 20:460-470, 1991

[7] P.K. Agarwal, Partitioning arrangements of lines: II. Applications, Discrete Comput. Geom., 5:533-573, 1990