

## A generator of random convex polygons in a disc

Olivier Devillers, Philippe Duchon, Rémy Thomasse

► **To cite this version:**

Olivier Devillers, Philippe Duchon, Rémy Thomasse. A generator of random convex polygons in a disc. AofA 2014- 25th International Conference on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms, Jun 2014, Paris, France. <hal-01015603>

**HAL Id: hal-01015603**

**<https://hal.inria.fr/hal-01015603>**

Submitted on 26 Jun 2014

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# A generator of random convex polygons in a disc<sup>†</sup>

Olivier Devillers<sup>1</sup>, Philippe Duchon<sup>2</sup>, Rémy Thomasse<sup>1</sup>

<sup>1</sup>INRIA Sophia Antipolis - Méditerranée

<sup>2</sup>Univ. Bordeaux, LaBRI, UMR 5800, F-33400 Talence, France  
CNRS, LaBRI, UMR 5800, F-33400 Talence, France

---

**Abstract** Using the theory of random polytopes, we propose an algorithm that generates a random polygon in  $\mathbb{R}^2$  as the convex hull of  $n$  points in a disc, without explicitly generating  $n$  points. As the size complexity is only  $O(n^{\frac{1}{3}})$  and the time complexity only  $O(n^{\frac{1}{3}} \log^2 n)$ , this algorithm allows to generate random polygons with very large  $n$ .

**Keywords:** Random Convex polygon, Generator, Convex hull, Random Analysis

---

## 1 Introduction

Let  $\mathcal{D}$  be a disc in  $\mathbb{R}^2$  with radius  $R$  centered at  $\sigma$ , and  $(x_1, \dots, x_n)$  a sample of  $n$  points uniformly and independently distributed in  $\mathcal{D}$ . Let's define the polygon  $P_n$  as the convex hull of  $(x_1, \dots, x_n)$ , and  $f_0(P_n)$  its number of vertices. This kind of polygon has been well studied, and using Bárány (1992); Reitzner (2010) one can easily see that

$$\mathbb{E}f_0(P_n) = c n^{\frac{1}{3}} + o(n^{\frac{1}{3}}) \quad (1)$$

where  $c = 2^{\frac{4}{3}} 3^{-\frac{1}{3}} \Gamma(\frac{5}{3}) \pi^{\frac{5}{3}} \approx 3.383228964$  and  $\Gamma$  denotes the usual Gamma function.

To generate such a polygon, one can explicitly generate  $n$  points uniformly in  $\mathcal{D}$  and compute the convex hull. For a very large number of points, it could be interesting to generate fewer points to get the same polygon, for example to evaluate the constant that are not explicitly known for the asymptotic distribution of the perimeter, or other parameters such as the higher moments of the extremal points.

In this note, we propose an algorithm that generates far fewer points at random in order to get  $P_n$ , so that the time and the memory needed is reduced for  $n$  large.

## 2 Algorithm

We start with random polygon  $P_i$  in  $\mathcal{D}$  where  $i$  is very small, and we increase the number of points until we get  $P_n$ .

---

<sup>†</sup>Supported by ANR blanc PRESAGE (ANR-11-BS02-003) and Région PACA.

The idea is that given the convex hull of small number of points, the number of points generated in  $\mathcal{D}$  that are deeply inside (and so won't change the convex hull) is a random variable that we can easily simulate, so that we need to generate only the small number of points that are close to the convex hull.

The outline of the algorithm is the following :

- Generate a small number of points in  $\mathcal{D}$  and compute its convex hull;
- Compute the radius of the largest disc centered at  $\mathfrak{o}$  inscribed in the convex hull;
- Choose a number of points to simulate at this step;
- Simulate the number of points that fall in this inscribed disc at this step;
- Generate the rest of the random points in the annulus defined by these two discs and update the convex hull.
- continue this process until the sum of the simulated and generated points is equal to  $n$ .

To simplify the notations, we assume  $\mathcal{D}$  to have radius 1.

#### Notations

- $n$  is the total number of points simulated and  $m_i$  is the total number of points from step 1 to step  $i$ ;
- $k_i$  is the number of generated points at step  $i$ , and  $k = \sum_i k_i$  is the total number of generated points;
- $h_i$  is size of the convex hull at step  $i$ , and  $h$  is the size of the final convex hull.
- $p_i$  is the probability to fall in the annulus at step  $i$ .

**Initialization** First, we have to generate a random polytope  $P_i$  with a small number of points in  $\mathcal{D}$  such that  $\mathfrak{o}$  is inside the random polytope. This is not too much to ask, as (see Alon and Spencer (2008))

$$\mathbb{P}(\mathfrak{o} \notin P_i) = 2^{-(i-1)} \quad (2)$$

We initialize the random polygon by generating 100 points in the disc. As the probability that  $\mathfrak{o} \notin P_{100}$  is lower than  $1.6 \times 10^{-28}$ , it's very unlikely that this is not enough. Otherwise, we add another sample of 100 points, until  $\mathfrak{o} \in P_{100,j}$  for some  $j$ .

**Simulation of Points** At the beginning of each step  $i$  of the loop, we are given a polygon  $P_{m_i}$ , which is the convex hull of  $m_i$  points. Let  $s_i$  the number of new points simulated at step  $i$ . We choose  $s_i = m_i$  if  $m_i < n \log^{-2} n$ , and  $s_i = n \log^{-2} n$  otherwise. Let  $\mathcal{D}_i$  be the largest inscribed disc in  $P_{m_i}$  centered in  $\mathfrak{o}$ , and  $r_i$  its radius.

Using a simulation of a binomial variable of parameter  $s_i$  and  $1 - r_i^2$ , we can evaluate the number of points that falls in  $\mathcal{D} \setminus \mathcal{D}_i$ . As the points in  $\mathcal{D}_i$  won't change the convex hull, we don't need to generate them. Then, we generate the rest of the points uniformly in the annulus  $\mathcal{D} \setminus \mathcal{D}_i$ , and we update the convex hull using a Graham scan.

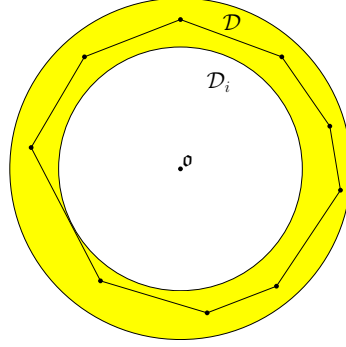


Fig. 1: At step  $i$ , we simulate the number of points that falls in  $\mathcal{D}_i$  and we generate points uniformly in the yellow annulus

**Generation of random points** To generate random points in an annulus with radii  $r_i$  and 1, one need to generate the polar angles uniformly in  $[-\pi, \pi)$  and the squared radii uniformly in  $[r_i^2, 1)$ . As we want to perform a Graham scan in linear time, the points have to be sorted by their polar angles. This can be done in expected linear time and size, using a bucket sort, as the angles are uniformly chosen, see Cormen et al. (2001).

#### Full algorithm

```

Data: integer  $n$ 
Result: Convex hull of  $n$  uniformly chosen points in the disc  $\mathcal{D}$ 
 $Simulated\_Points \leftarrow 0$ ;
do
  | Generate  $\min(100, n - Simulated\_Points)$  points in the disc of radius 1;
  |  $Simulated\_Points \leftarrow Simulated\_Points + \min(100, n - Simulated\_Points)$ ;
while  $o$  is not in the convex hull and  $Simulated\_Points < n$ ;
while  $Simulated\_Points < n$  do
  | Compute  $inscribed\_radius$ ;
  |  $p \leftarrow inscribed\_radius^2$ ;
  | if  $Simulated\_Points < n \log^{-2} n$  then
  | |  $k \leftarrow Simulated\_Points$ ;
  | else
  | |  $k \leftarrow \min(\lfloor n \log^{-2} n \rfloor, n - Simulated\_points)$ ;
  | end
  |  $X \leftarrow$  Simulation of Binomial variable with parameters  $k, 1 - p$ ;
  | Generate  $X$  points uniformly and sorted in the annulus of radii  $inscribed\_radius, 1$ ;
  |  $Simulated\_Points \leftarrow Simulated\_Points + k$ ;
  | Merge the list of the convex hull and the new points;
  | Update the convex hull with a Graham scan on the list;
end
return Convex hull

```

**Algorithm 1:** Algorithm of the Generator of Random Polygon in a disc

### 3 Complexity

Clearly the size complexity is  $\max_i(h_i + k_i)$  and the time complexity is  $\sum_i(h_i + k_i)$  since the Graham scan and the points generation are linear in the number of points Graham (1972).

For the initialization, as the probability that  $\mathfrak{o} \notin P_n$  decreases exponentially, it is very unlikely that more than one loop is necessary. Let's call  $\mathfrak{p}$  the minimal number of points such that  $\mathfrak{o} \in P_{\mathfrak{p}}$ .

Using formula (2), the expectation of  $\mathfrak{p}$  is very small :

$$\begin{aligned}
 \mathbb{E}(\mathfrak{p}) &= \sum_{j=1}^{\infty} j \mathbb{P}(\mathfrak{p} = j) = \sum_{j=1}^{\infty} \mathbb{P}(\mathfrak{p} \geq j) \\
 &= \sum_{i=1}^3 \mathbb{P}(\mathfrak{p} \geq j) + \sum_{j=4}^{\infty} \mathbb{P}(\mathfrak{o} \notin P_{j-1}) \\
 &= 3 + \sum_{j=3}^{\infty} \mathbb{P}(\mathfrak{o} \notin P_j) \\
 &= 3 + \sum_{j=3}^{\infty} j 2^{-(j-1)} = 3 + 2 = 5.
 \end{aligned} \tag{3}$$

Thus, the expected size and time complexity of the initialization is  $O(1)$ .

For  $i$  big enough, we have Bárány (1989):

$$\mathbb{E} d_H(P_{m_i}, \mathcal{D}) = \Theta \left( \frac{\log m_i}{m_i} \right)^{\frac{2}{3}} \tag{4}$$

where  $d_H$ , the Hausdorff distance, is the maximal distance between a point in  $P_i$  and the boundary of  $\mathcal{D}$ . Recall that  $\mathcal{D}_i$  is the annulus with radii  $r_i = 1 - d_H(P_{m_i}, \mathcal{D})$  and 1 and let  $p_i$  be the probability that a random point in  $\mathcal{D}$  falls in  $\mathcal{D}_i$ . Using (4), there exist a constant  $c_0 > 0$  such that, for  $i$  large  $\mathbb{E}(p_i) =$

$$1 - r_i^2 < 2(1 - r_i) < c_0 \left( \frac{\log m_i}{m_i} \right)^{\frac{2}{3}}.$$

Let's call  $i_{\tau}$  the last step  $i$  where  $m_i < \frac{n}{\log^2 n}$ .

At each step  $i \leq i_{\tau}$ ,  $k_i$  is a binomial variable with parameter  $p_i$  and  $m_i < \frac{n}{\log^2 n}$ . Thus, for  $i$  large enough,

$$\mathbb{E}(k_i) = \mathbb{E}(\mathbb{E}(k_i | p_i)) = m_i \mathbb{E} p_i = O(m_i^{\frac{1}{3}} \log^{\frac{2}{3}}(m_i)) = O \left( \left( \frac{n}{\log^2 n} \right)^{\frac{1}{3}} \log^{\frac{2}{3}} n \right) = O(n^{\frac{1}{3}}). \tag{5}$$

As we choose  $m_{i+1} = 2m_i$ ,  $i_{\tau}$  is bounded by  $\log_2(n)$ .

For  $i > i_{\tau}$ ,  $k_i$  is a binomial variable with parameter  $p_i$  and  $\frac{n}{\log^2 n}$ , so using (5) and the fact that  $m_i > \frac{n}{\log^2 n}$ , we get  $\mathbb{E} k_i = O(n^{\frac{1}{3}})$ . As we simulate at each step  $i > i_{\tau}$  at least  $\frac{n}{\log^2(n)}$ , the number of step after  $i_{\tau}$  is bounded by  $\log^2(n)$ .

Now,

$$\begin{aligned}
\mathbb{E}k &= \sum_{i=1}^{i_\tau} \mathbb{E}k_i + \sum_{i>i_\tau} \mathbb{E}k_i \\
&\leq O\left(\log^{\frac{2}{3}} n \left(\sum_{i=1}^{\log_2 n} m_i^{\frac{1}{3}}\right)\right) + \log^2 n O(n^{\frac{1}{3}}) \\
&\leq O\left(\log^{\frac{2}{3}} n \left(\sum_{i=1}^{\log_2 n} (2^i)^{\frac{1}{3}}\right)\right) + O(n^{\frac{1}{3}} \log^2 n) = O(n^{\frac{1}{3}} \log^2 n)
\end{aligned} \tag{6}$$

At each step  $i$ , the expected size of the convex hull is  $O(m_i^{\frac{1}{3}}) = O(n^{\frac{1}{3}})$ , and  $\mathbb{E}k_i = O(n^{\frac{1}{3}})$ . Thus, using Chernoff bound, the expected size of the largest list, that is the expected size complexity, is  $O(n^{\frac{1}{3}})$ . As our points are sorted according to their polar angle, computing the convex hull with a Graham scan is done in linear time ( $O(n^{\frac{1}{3}} \log^2 n)$ ), the generation of the  $k$  points and the computation of the largest annulus as well ( $O(n^{\frac{1}{3}})$ ). Thus, the expected time complexity is  $O(n^{\frac{1}{3}} \log^2 n)$ .

## 4 Experiments

This algorithm has been implemented in C++, and is submitted for integration in the CGAL library. As expected, the distribution of  $\mathbb{E}f_0(P_n)$  is asymptotically the same as the theoretical one, see equation (1), with the same constant :

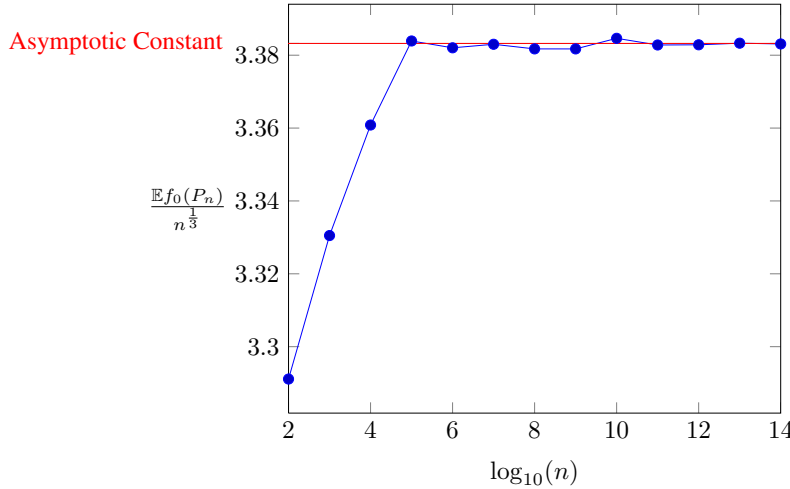


Fig. 2: Average size of  $P_n$  divided by  $n^{\frac{1}{3}}$

This estimation has been done one 1000 experiments for each value of  $n$ .

**Complexities** To evaluate the expected size complexity, we estimate the average size of the largest list of points used in the loop, see Figure 3. The time complexity can be evaluated by estimating the total number of points generated, see Figure 4.

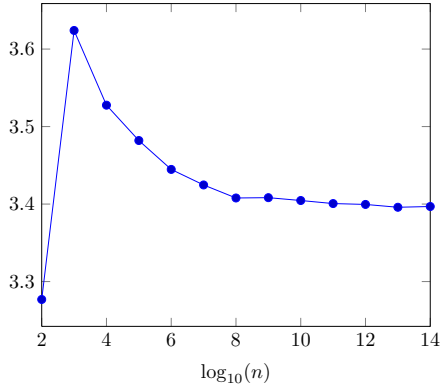


Fig. 3: Average maximal size of the list divided by  $n^{1/3}$

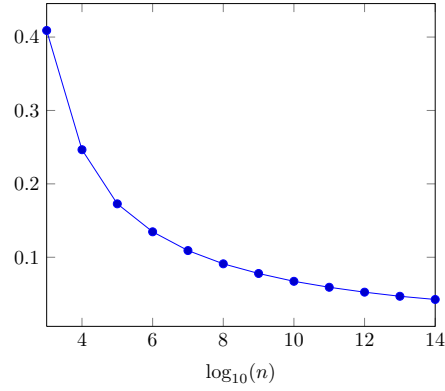


Fig. 4: Average number of generated points divided by  $n^{1/3} \log^2 n$

As a first application, we can estimate the distribution of the smallest and the largest edge of a random polytope, see Figure 5 and Figure 6. The average have been done on 100 experiments for each value of  $n$ . The expected minimal edge seems to be  $O(n^{-1/3})$ , and the maximal  $O\left(\frac{\log^2 n}{n}\right)^{1/3}$ .

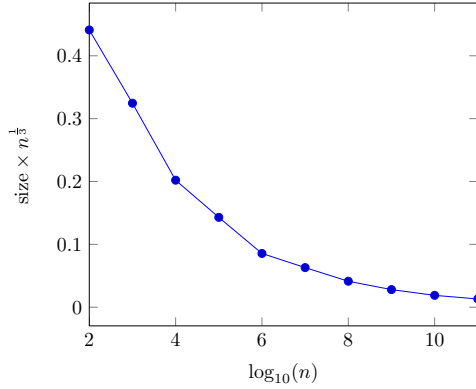


Fig. 5: Average length of the minimal edge

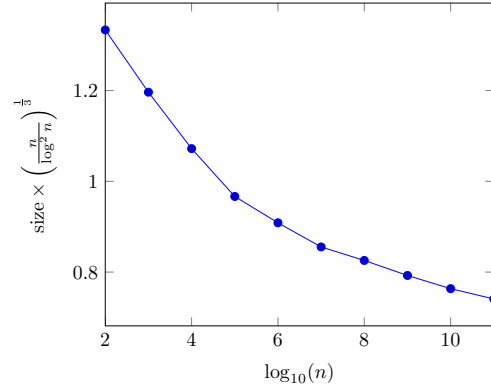


Fig. 6: Average length of the maximal edge

## 5 Conclusion

We propose an algorithm that generates random polygons given by the convex hull of random points, without generating all the points. There is no theoretical obstacle to generalize to higher dimension. The theoretical results used for the evaluation of the complexity are known in arbitrary dimension Bárány (1989); Reitzner (2010), so the analysis can be done as well.

We can reduce the expected time complexity by a logarithmic factor if we allow to increase the expected size complexity by a logarithmic factor. Instead of bounding the simulated points at step  $i$  by  $\frac{n}{\log^2 n}$  when  $m_i$  becomes bigger than  $\frac{n}{\log^2 n}$ , we can choose to always simulate  $m_i$  points. In this case,

$$\mathbb{E}k_i = O(m_i^{\frac{1}{3}} \log^{\frac{2}{3}} m_i) = O(n^{\frac{1}{3}} \log^{\frac{2}{3}} n) \quad (7)$$

so the expected size complexity becomes  $O(n^{\frac{1}{3}} \log^{\frac{2}{3}} n)$ . On the other hand, the expected time complexity is reduced to  $O(n^{\frac{1}{3}} \log^{\frac{2}{3}} n)$ , as

$$\begin{aligned} \mathbb{E}k &= \sum_i \mathbb{E}k_i \\ &= O\left(\log^{\frac{2}{3}} n \left(\sum_{i=1}^{\log_2 n} m_i^{\frac{1}{3}}\right)\right) = O\left(n^{\frac{1}{3}} \log^{\frac{2}{3}} n\right). \end{aligned}$$

## Acknowledgments

This work was initiated during the 2013 Presage Workshop on computational geometry and probability in Valberg. The authors wish to thank all the participants for creating a pleasant and stimulating atmosphere and Pierre Calka for discussions about potential use of the generator.

## References

- N. Alon and J. H. Spencer. *The probabilistic method*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons Inc., Hoboken, NJ, third edition, 2008. ISBN 978-0-470-17020-5. doi:10.1002/9780470277331. URL <http://dx.doi.org/10.1002/9780470277331>. With an appendix on the life and work of Paul Erdős.
- I. Bárány. Intrinsic volumes and  $f$ -vectors of random polytopes. *Mathematische Annalen*, 285(4):671–699, 1989.
- I. Bárány. Random polytopes in smooth convex bodies. *Mathematika*, 39:81–92, 6 1992. ISSN 2041-7942. doi:10.1112/S0025579300006872. URL [http://journals.cambridge.org/article\\_S0025579300006872](http://journals.cambridge.org/article_S0025579300006872).
- T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001. ISBN 0070131511.
- R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information processing letters*, 1(4):132–133, 1972.
- M. Reitzner. Random polytopes. In *New perspectives in stochastic geometry*, pages 45–76. Oxford Univ. Press, Oxford, 2010.