Generating Empty Convex Polygon Randomly from a Subset of Given Point Set

Manas Kumar Mohanty, Sanjib Sadhu, Niraj Kumar and Kamaljit Pati

Abstract In computational geometry, problem of generating random geometric objects are very interesting and extensively studied problems. In this paper we propose a new algorithm to generate an empty convex polygon from a subset of given point set. Let $S = \{p_1, p_2, \ldots, p_n\}$ be the given point set lying in $\mathbb{R}^2$. The proposed algorithm generates a random empty convex polygon consisting of k vertices in S. In preprocessing phase we compute Convex Hull Layers $CL(S)$ in $O(n \log n)$ time. By using the visibility relationship in Convex Layers, the proposed algorithm generates a random empty convex polygon in $O(\log n + k)$ time which is improved over the existing solution for this problem.

Keywords Polygon · Convex polygon · Convex hull layers · Visibility

1 Introduction

In computational geometry, generating random geometric objects like simple polygon, monotone polygon are extensively studied problems. The theoretical applications of geometric objects includes testing of various computational geometry algorithms and verification of the algorithm being tested. There exist a lot of heuristics [1] to generate a random polygon from n points. Much efforts has also

M.K. Mohanty (✉) · S. Sadhu · N. Kumar · K. Pati
National Institute of Technology, Durgapur, India
e-mail: munkun41@gmail.com

S. Sadhu
e-mail: sanjibsadhu411@gmail.com

N. Kumar
e-mail: nirajcse08@gmail.com

K. Pati
e-mail: kamaljit.igit@hotmail.com

J.K. Mandal et al. (eds.), *Information Systems Design and Intelligent Applications*,
Advances in Intelligent Systems and Computing 339,
DOI 10.1007/978-81-322-2250-7_30

been applied to generate a specific class of polygons like monotone polygons [2], star shaped polygons [1].

In this paper, our focus is on convex polygons with vertices in S rather than simple polygons with vertices of S. Generation of empty convex k-gons and counting of empty k-gons in given point set are the problems of this category.

In [3] problem of large convex holes in random point set has been discussed. Counting version of this problem is also quite popular, some of the recent studies includes [4, 5]. A random convex polygon has been generated by Zhu et al. [2]. After $O(n^3)$ preprocessing on a set S of n points, a convex polygon can be generated in $O(n^2)$ time whose vertices are in S [2]. In this paper, we compute a k-gon efficiently than that of existing algorithms. Our algorithm involves two phases. The preprocessing phase involves computing convex layers, which requires $O(n \log n)$ time. From convex layers we compute visibility information. Visibility information will be used to compute a k-gon in $O(\log n + k)$ time. Hence, proposed algorithm computes empty k-gon in given point set in $O(\log n + k)$ time after a preprocessing of $O(n \log n)$.

Convex layers finds a lot of applications in a variety of fields. Chazelle [6] provides some applications of convex layers. In [7] convex layers has been used to solve TSP problem. In [8] Sadhu et al. applied convex layers to design a heuristic to generate simple polygon from S.

The outline of the paper is as follows. Section 2 provides some definitions and other concepts required to solve problem. Section 3 describes the algorithm designed to solve the problem along with its complexity analysis. Finally in Sect. 4, we conclude with possible future works.

2 Preliminaries

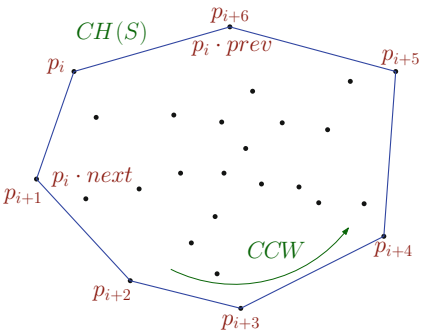
We use the term points and vertices interchangeably throughout the paper. For a pair of point p and q, the line segment $\ell(p, q)$ is defined as directed line segment from point p to point q. Henceforth we assume that set $S = \{p_1, p_2, \ldots, p_n\}$ is given point set containing n points in $\mathbb{R}^2$.

Definition 1 A polygon P is said to be simple [9] if it consists of straight, non-intersecting line segments, called edges that are joined pair-wise to form a closed path.

An edge connecting two points p and q is denoted by $e(p, q)$. The x-coordinate and y-coordinate of a point p are denoted by $x(p)$ and $y(p)$, respectively. Here and throughout the paper, unless qualified otherwise, by polygon we mean simple polygon. By empty polygon we mean no point $p \in S$ lies inside the polygon.

Definition 2 A subset S of the plane is called **convex** if and only if for any pair of points $(p, q) \in S$ the line segment $\ell(p, q)$ is completely contained in S [9].

Fig. 1 Convex hull $CH(S)$

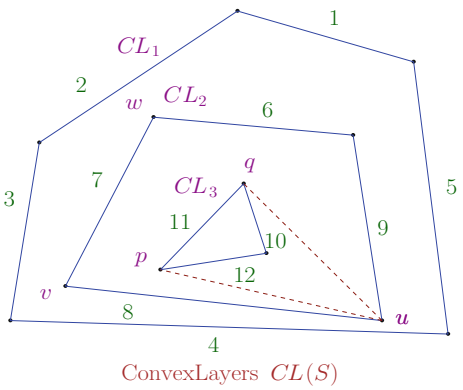


In other terms, a convex hull of given point set is smallest convex set that contains all points of S. Convex hull of given points is represented by set of vertices, in order, that defines hull edges. The convex hull of point set S, denoted by $CH(S)$, is shown in Fig. 1. The $CH(S)$ represents a chain of vertices $\{p_i, p_{i+1}, \ldots, p_{j-1}, p_j, p_i\}$ such that $1 \leq i \leq j \leq n$. From now on we assume that vertices of convex hull are given in counterclockwise order, as shown in Fig. 1. The lower bound to find $CH(S)$ of a given point set with n points, is $\Omega(n \log n)$ [10].

Definition 3 A **convex chain** of a polygon is a sequence of consecutive edges where the internal angle between two edges is less than 180°.

Consecutive vertices of convex hull forms a convex chain. Convex layers is the extension to convex hull concept. For the given point set S, its convex layers set $CH(S)$ is set of convex layer CL_i such that each layer $CL_i \in CL(S)$ is computed by removing points of all previous layers, i.e. CL_j, where $j < i$ from point set S. Hence, convex layers set $CH(S)$ can be obtained by recursively computing convex hull of point set and removing points lying on computed convex hull, until point set becomes empty. The convex layers of a point set are shown in Fig. 2. In [11] convex layer mentioned to define *depth* of a point as

Fig. 2 The edge $e(p, q)$ is not visible to point u, however it is visible to points v and w



Definition 4 The depth of a point p in a set S is the number of convex hulls (convex layers) that have to be stripped from S before p is removed. The depth of S is the depth of its deepest point.

For a given point set $S = \{p_1, p_2, \ldots, p_n\}$ we represent its convex hull by $CH(S)$, whereas set of convex layers will be represented by $CL(S)$. The convex layers set, or simply convex layers, $CL(S)$ is set of convex layers $\{CL_1, CL_2, \ldots, CL_m\}$, where m is the number of convex layers in given point set. In $CL(S)$, we follow the inward ordering, hence the outermost layer will be referred as CL_1 and innermost layer as CL_m, as shown in Fig. 2. Any layer $CL_i \in CL(S)$ is defined by sequence of vertices. From now on the vertices of convex hull are assumed to be in counterclockwise sequence. Hence, a convex layer CL_i is represented by counterclockwise circular sequence of vertices.

A point $p \in S$ is said to be visible [12] from a point $q \in S$ if the line segment $\ell(p, q)$ does not intersect any other line segment or does not pass through a third point $r \in S$.

Definition 5 (*Edge Visibility*). An edge $e(p, q) \in CL_i$ is said to be visible from a point $u \in CL_{i-1}$, if p, q and any point r (say, midpoint of $e(p, q)$) lying on $e(p, q)$ is visible from u.

Consider Fig. 2, the Convex Layer $CL(S)$ of the point set S consists of three Convex Layers CL_1, CL_2 and CL_3. Figure 2 shows that the edge $e(p, q) \in CL_3$ is "not visible" to the point $u \in CL_2$, however edge $e(p, q)$ is visible to points v and w lying on CL_2.

3 Algorithm

The objective is to generate a random empty convex polygon from a subset of points of S. Preprocessing phase requires computation of convex layers followed by visibility computation.

The convex layers will be computed using the optimal algorithm proposed in [6]. It is an optimal time algorithm, which computes convex layers in $O(n \log n)$ time. The lower bound to compute convex hull of a given point set with n points is $\Omega(n \log n)$ [10]. Hence, the convex layer algorithm with $\Omega(n \log n)$ time is an optimal algorithm. The pseudocode in Algorithm 1 describe the preprocessing part before computing a random convex empty polygon from the subset of point set.

Observation 1 *With respect to an edge $e(p, q) \in CL_i$, there must exist at least one vertex $v \in CL_{i+1}$ visible from the edge $e(p, q)$.*

Lemma 1 *Line number 6 of the Algorithm 1 takes $O(\log n_{i-1})$ time, where n_{i-1} is the number of edges in CL_{i-1}.*

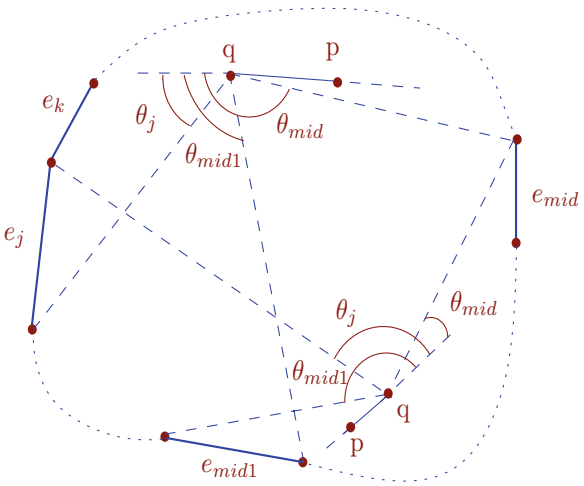


Fig. 3 The edge $e(p, q)$ lies inside the convex polygon whose edges e_j, e_{mid}, e_{mid1} and e_k are shown. Its other edges are represented by *dotted curves*

Proof Let the edges of the layer CL_{i-1} be $\{e_j, e_{j+1}, \ldots, e_k\}$, where $k = (j + n_{i-1} - 1)$. Consider Fig. 3. These sequence of edges forms an initial convex chain. Now, the line $\ell(p, q)$ passing through the edge $e(p, q) \in CL_i$ must intersect the CL_{i-1} at two points which are to be found. We compute an index $mid = (j + k)/2$. We check in $O(1)$ time whether the line $\ell(p, q)$ passes through e_j and(or) e_{mid}. If yes, we found intersection(s). Otherwise, we test in which side of the line $\ell(p, q)$, the edges e_j and e_{mid} lies. □

There are two possible cases:

Case 1 The edges e_j and e_{mid} lie on the same side of $\ell(p, q)$. Here, $\ell(p, q)$ intersect either the convex chain $\{e_{j+1}, e_{j+2}, \ldots, e_{mid-1}\}$ or the convex chain $\{e_{mid+1}, e_{mid+2}, \ldots, e_k\}$ depending on the angles θ_j, θ_{mid} and θ_{mid1} (Refer to Fig. 3), where $mid1$ is the index of the edge e_{mid1} and is given by $mid1 = j + (mid - j)/2$. Find out the angle θ_j formed by the edge $e(p, q)$ and line segment $\ell(p, r)$ where r is any one of the end points of the edge e_j. Similarly find out the angles θ_{mid} and θ_{mid1} formed by the edge $e(p, q)$ with the line segments $\ell(p, s)$ and $\ell(p, t)$ respectively, where s and t are any one of the end points of the edge e_{mid} and e_{mid1} respectively. These three angles can be computed in $O(1)$ time. Now, if the value of θ_{mid1} lies in between θ_j and θ_{mid}, then we have to consider only the convex chain $\{e_{mid+1}, e_{mid+2}, \ldots, e_k\}$, because the line $\ell(p, q)$ will pass through this convex chain only. Hence we reject other convex chain. However, if θ_{mid1} is either greater than or less than both θ_j and θ_{mid}, we will consider the convex chain $\{e_{j+1}, e_{j+2}, \ldots, e_{mid-1}\}$ rejecting the other convex chain.

Case 2 The edges e_j and e_{mid} lie on the different sides of $\ell(p, q)$. In this case the line $\ell(p, q)$ will intersect both the convex chains $\{e_{j+1}, e_{j+2}, \ldots, e_{mid-1}\}$ and $\{e_{mid+1}, e_{mid+2}, \ldots, e_k\}$. Hence both the convex chains are to be considered. We search for intersecting point by recursive procedure by assuming each such convex

chain as the initial convex chain. The intersection points are determined by binary search like procedure and hence, this operation will take $O(\log n)$ time.

Let m be the number of convex layers in convex layers set $CL(S)$. In step 1 of the Algorithm 1, convex layers set $CL(S)$ computed using optimal algorithm to compute convex layer proposed in [6].

Algorithm 1: Preprocessing

Input: Point Set $S = \{p_1, p_2, \ldots, p_i, \ldots, p_n\}$
Output: Convex Layers Set $CL(S) = \{CL_1, CL_2, \ldots, CL_m\}$ and $ArrayTable[\ n\ ][\ n\ ], ArrayB[n]$

1 Compute the Convex Layers $CL(S)$ of S
2 Initialize each entry of the array $B[\ n\ ]$ to zero
3 **for** $i = 2$ **to** m **do**
4 **for** *each edge* $e(p, q) \in CL_i$ **do**
5 Find the number (say t) of vertices $v \in CL_{i-1}$, from which the edge $e(p, q)$ is completely visible
6 $B[\ t\ ] \leftarrow B[\ t\ ] + 1$
7 Store the edge id $e.id$ of the edge $e(p, q)$ in $Table[[\ t\ ][B[\ t\ ]]$

In Algorithm 1, the tth row of the *Table*, i.e. *Table*[t] store the edge-id of those *edges* $\in$ *CL*(*S*) which are visible from exactly t number of vertices. The number of edge ids stored in each tth row of *Table*[][] are stored in *B*[t].

Observation 2 *The Algorithm 1 takes* $O(n \log n)$ *time.*

Proof Step 1 of the Algorithm 1 requires computation of Convex Layers of *S*. To compute convex layers set *CL*(*S*) we use optimal algorithm proposed in [6], which computes *CL*(*S*) in $O(n\ \ell ogn)$ time. Then it computes the set of visible vertices for all the edges lying on the Convex Layer, except outermost layer CL_1. Hence by lemma 2, the time complexity of the Algorithm 1 is $O(n \log n)$. □

Once the preprocessing part is over, we can generate a random convex (empty) polygon using the Algorithm 2.

Theorem 3 *The Algorithm 2 computes a random convex polygon (consisting of k vertices) in* $O(\log n + k)$ *time after the preprocessing part is over.*

Proof Line number 4 of the Algorithm 2 takes $O(\log n)$ time as proved in lemma 2. The if-else statement of the Algorithm 2 takes $O(k)$ time to process the visible vertices. Hence, the overall time complexity of Algorithm 2 is $O(\log n + k)$. □

The Fig. 4 shows how an empty Convex Pentagon is randomly generated by the Algorithm 2 after the preprocessing part is over. As shown in Fig. 4, one can generate an empty convex quadrilateral using the edge $e(p, q)$ and any two vertices

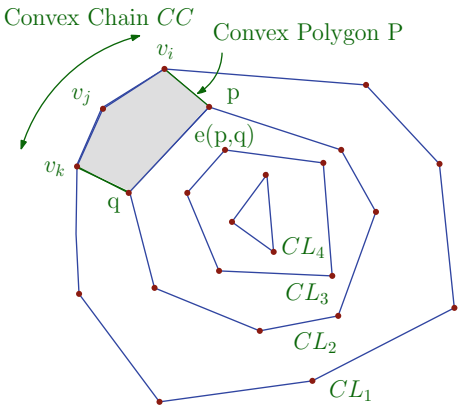


Fig. 4 The set of vertices visible from the edge $e(p, q) \in CL_3$ is $V_e = \{v_i, v_j, v_k\}$ and a convex pentagon (i.e. $k = 5$) P is generated after selecting the random edge e

from the set V_e. Our algorithm constructs an empty convex polygon using two adjacent Convex Layers CL_i and CL_{i+1}.

Algorithm 2: Convex Polygon Generation

Input: $CL(S), Table[\ n\ ][\ n\ ], B[\ n\ ], k$
Output: Convex Polygon P consisting of k vertices

1 $u \leftarrow (k-2)$ ▷ Since the convex polygon P will be generated after selecting an edge e which will be part of the P , we will consider $(k-2)$ points from the $CL(S)$ (except the two end points of the edge e)
2 Randomly select an edge e from the t_{th} row (where $t \geq u$ and $B[\ t\ ] > 0$) of the Table[ ][ ]
3 $V_e \leftarrow$ Set of vertices visible from $e \in CL_i$, where $V_e \in CL_{i+1}$ and $e \in CL_i$
4 **if** *(*$|t| == |u|$*)* **then**
5 Connect the vertices V_e in the order in which they appear in the layer CL_i to form a convex chain CC
6 **else**
7 Select randomly $|u|$ number of vertices from the V_e and connect them in the order in which they appear in the layer CL_i to form a convex chain CC;
8 Connect the two end vertices of CC with the two end points of e to get a convex Polygon P
9 **return** *Polygon* P

4 Conclusion and Future Works

Proposed algorithm computes empty convex polygon in given points set. In pre-processing part, convex layers set $CL(S)$ and the visibility relations among its edges are computed. After $O(n \log n)$ preprocessing task, a random empty convex

polygon consisting of k vertices is generated in $O(\log n + k)$ time which is much faster than the existing algorithm [2].

Proposed algorithm computes empty convex polygon by considering two adjacent convex layers. However, in some cases it may be possible that the empty convex polygon consisting of k vertices cannot be generated using only two adjacent Convex Layers. Hence, as a future work, we can use more than two neighboring Convex layers of $CL(S)$ to generate a random empty convex polygon. However, it would be a challenging task to compute empty convex polygon by considering all layers in $O(\log n + k)$ time.

References

1. Auer, T., Held, M.: RPG: heuristics for the generation of random polygons. In: Proceedings of 8th Canadian Conference Computational Geometry, pp. 38–44 (1996)
2. Zhu, C., Sundaram, G., Snoeyink, J., Mitchel, J.S.B.: Generating random polygons with given vertices. Comput. Geom. Theory Appl. 6:277–290 (1996)
3. Balogh, J., Gonzlez-Aguilar, H., Salazar, G.: Large convex holes in random point sets. Comput. Geom. **46**(6), 725–733 (2013)
4. Mitchell, J.S., Rote, G., Sundaram, G., Woeginger, G.: Counting convex polygons in planar point sets. Inf. Process. Lett. **56**(1), 45–49 (1995)
5. Rote, G., Woeginger, G.: Counting convex k-gons in planar point sets. Inf. Process. Lett. **41** (4), 191–194 (1992)
6. Chazelle, B.: On the convex layers of a planar set. Trans. Inf. Theory IEEE **31**(4), 509–517 (1985)
7. Liew, S.: Applying convex layers, nearest neighbor and triangle inequality to TSP problem. arXiv preprint arXiv:1204.2350 (2012)
8. Sadhu, S., Kumar, N., Kumar, B.: Random polygon generation through convex layers. Proc. Technol. **10**, 356–364 (2013)
9. De Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O., Overmars, M.H.: Computational geometry: algorithms and applications. New York, New York (2000)
10. Yao, A.C.C.: A lower bound to finding convex hulls. J. ACM **28**(4), 780–787 (1981)
11. Preparatata, F.P., Shamos, M.I.: Computational Geometry: An Introduction. Springer, Berlin (1985)
12. Ghosh, S.K.: Visibility Algorithm in the Plane. Cambridge University press, Cambridge (2007)