単純多角形の生成に関する発見的手法

寺本 幸生　元木 光雄　上原 隆平　浅野 哲夫
北陸先端科学技術大学院大学 情報科学研究科
〒 923-1292 石川県能美市旭台 1-1, {s-teramo,mmotoki,uehara,t-asano}@jaist.ac.jp

Abstract. 平面上の n 点からなる集合 S が与えられた時, S の点を頂点として持つ単純 n 角形をランダムに生成する問題について考える. これまで, カウンティング問題でさえ難しいと思われているため, 発見的な手法の開発が行なわれてきた. 本論文では、三角形分割を用いる発見的手法を提案する. これは, 可能なすべての単純多角形を非負の確率で生成する $O(n\log n+f)$ 時間アルゴリズムである. ただし、f は辺のフリップ操作の回数とする. この手法はこれまで知られている $O(n^4\log n)$ 時間アルゴリズムよりも効率的であると考えられる.

Key words: 計算幾何学, 幾何学的数え上げ, 単純多角形, 三角形分割, 発見的手法.

Heuristics for Generating a Simple Polygonalization

Sachio Teramoto　Mitsuo Motoki　Ryuhei Uehara　Tetsuo Asano
School of Information Science, Japan Advanced Institute of Science and Technology (JAIST),
1-1, Asahidai, Nomi, Ishikawa, 923-1292 Japan, {s-teramo,mmotoki,uehara,t-asano}@jaist.ac.jp

Abstract. Given a set S of n points in the plane, randomly generate a simple polygon with n sides using the points of S as its vertices, or compute a simple polygonalization of S. Since the counting problem seems to be quite difficult, heuristic approaches have been adopted during the past decade. We propose a triangulation-base heuristic algorithm which generates each possible simple polygon with a non-zero probability in $O(n\log n+f)$ time, where f is the number of edge-flippings.This time complexity has an advantage over a known $O(n^4\log n)$ algorithm.

Key words: Computational geometry, geometric enumeration, simple polygon, polygonalization, triangulation, heuristics.

1 Introduction

Geometric Enumeration deals with problem of listing all geometric objects that satisfy a specified property. Typical objects to be enumerated are triangulations of a set of planar points. Triangulation is one of the most important geometric structures in Computational Geometry [6, 32], so that a number of literature have proposed good enumerating algorithms [4, 5, 1].

In addition to theoretical interest, generation of random geometric objects has applications that include testing and verification of time-complexity of computational geometric algorithms for practical rather than worst-case behavior. In that sense, generating a simple polygon seems to be practical and more important than a triangulation, since there are many applications which treat a simple polygon as an instance in geometric optimization, such as Art Gallery Problems [30, 36], Polygon Partitioning [21], Geometric Shortest Paths [18], and so on [33, 17]. This paper considers how to generate random simple polygons efficiently.

Problem statement Given a set S of n points in general position in the plane, randomly generate a simple polygon whose polygon vertices are precisely of S, or compute a random *simple polygonalization* of S.

More precisely, we would like polygonalize a set S of points so that each member of all polygonilizations of S is generated uniformly at random. However, examining the set of all simple polygonalizations of S is quite difficult even in the counting problem which asks how many simple polygonalizations are there in S, see [27]. A brief history of the asymptotic bounds on this number are summarized by Demaine [12]. The currently known approximate upper and lower bounds are $O(86.81^n)$ due to Sharir and Welzl [34] and $\Omega(4.642^n)$ due to García, Noy and Tejel [16], respectively. Due to the high upper bound heuristic approaches have been adopted to generate a simple polygonalization during the past decade.

Related works Simple polygonalizations are also called simple polygonizations, or crossing-free Hamiltonian cycles, or planar traveling salesman tours. There are a few related optimization problems such as the Traveling salesman problem. In fact, the problems for computing a simple polygonalization with the minimum total polygon edge length [39] or with the minimum or maximum area [14] are $\mathcal{NP}$-complete.

As we mentioned, it is an outstanding open problem whether the number of simple polygonalizations of S can be computed in polynomial time. There are two different approaches in the literature: one is to investigate a subclass of simple polygons such as x-monotone polygons [26, 40], and start-shaped polygons [2, 3, 37, 38]; the other is to design an efficient heuristics [2, 3, 10, 31, 40]. Auer and Held [2], and Zhu, Sundaram, Snoeyink and Mitchel [40] independently proposed a practically useful heuristic, so called 2-opt Moves. It can generate every possible simple polygonalization with a *positive probability* (this implies there is no possible simple polygons which cannot be generated by the heuristic). Any other heuristics cannot generate all possible simple polygonalizations, or are impractical by the experimental results of Auer [3].

2-opt Moves is experimentally good, but, it requires $\Theta(n^3)$ times "untangling 2-opt" moves before convergence to a simple polygon in the worst case, as is proved by Van Leeuwen and Schoone [23]. Hence, the time complexity of 2-opt Moves is $O(n^4 \log n)$ with sweep-line technique (see e.g., [6]) in the worst case. In particular, an implementation of 2-opt Moves has been included in CGAL [8]. Unfortunately it takes too much time to implement 2-opt Moves for a larger instance, and any algorithms proposed so far seem to be impractical. We propose a simple heuristic algorithm which runs fast enough even for a large instance.

Our contributions We propose a triangulation-base heuristic algorithm which generates each possible simple polygon with a positive probability in $O(n \log n + f)$ time, where f is the number of edge-flipping operations which may have effect on the randomness. This may improve the time complexity $O(n^4 \log n)$ of 2-opt Moves.

Our algorithm consists of three phases: first, it generates a triangulation T of given point set at random; next computes a random maximal *polygon tree* on the dual $\mathcal{D}(T)$ of T. A polygon tree $\mathcal{T}$ on the dual $\mathcal{D}(T)$ of a triangulation T is a tree such that $\cup_{v \in \mathcal{T}} g^{-1}(v)$ is a simple polygon, where g is bijection from a face in T to a vertex in $\mathcal{D}(T)$; and finally constructs a simple polygon by traversing on the maximal polygon tree with depth-first search.

In section 3, we describe our heuristic algorithm in detail. Our heuristic algorithm may applicable to the generalized simple polygonalization problems. We discuss in Section 4. Finally, we show future works in Section 5.

2 Preliminaries

A *polygon* is a region of the plane bounded by a finite collection of line segments forming a closed curve. A polygon P is said to be *simple* if points of the plane belonging to two polygon edges of P are limited to the polygon vertices of P. Hence, there are no self-intersections or no holes (see Fig. 1), and then a simple polygon is topologically homeomorphic image of a disk. Fig. 1 (a) and (b) depict a simple polygon with 64 vertices, and a nonsimple polygon with 5 self-intersections and 3 holes, respectively. In this paper, polygons mean simple polygons unless it is stated. We sometimes treat a polygon P with k vertices, or k-gon, as a circular list of k polygon vertices $(v_1, v_2, \ldots, v_k, v_{k+1} = v_1)$ in clockwise-order and denote the number of polygon vertices or polygon edges by $|P|$.

Throughout this paper we let S stand for a finite set of n points in the plane. We assume that S is in general position, i.e., no three points are collinear and no four points are cocircular.

A *triangulation* of planar points S, denoted by $T(S)$, is a simplicial decomposition of its convex hull $\mathcal{CH}(S)$ whose vertices are precisely the points in S. In other words, a triangulation is a maximal crossing-free geometric graph on S (in a geometric graph the edges are realized by straight line segments). To distinguish the terminologies of 'vertex' and 'edge' between polygon and graph, we explicitly specify the modifier "polygon" for vertex and edge of polygon.

In a triangulation $T(S)$, an edge e of $T(S)$ is *flippable* if it is adjacent to two triangles whose union is a convex quadrilateral C. By *flipping* e we mean an operation of removing e for $T(S)$ and replacing it by the other diagonal of C. In this way we obtain a new triangulation $T'(S)$, and we say that $T'(S)$ has been obtained from $T(S)$ by means of a *flip*. Lawson [22] showed that any two triangulations of a planar point set can be transformed into each other by flipping edges. Fortune [15] showed that at most $\binom{n}{2}$ flips are sufficient to compute Delaunay triangulation. This implies there exists a sequence of $O(n^2)$ flips which transforms a triangulation to any other. More precisely,

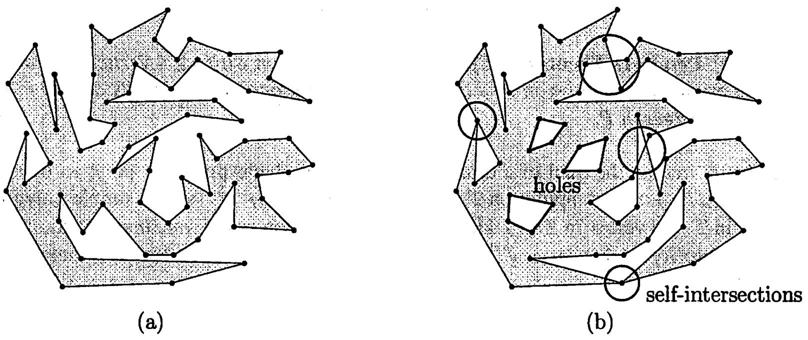

Fig. 1. Simple and nonsimple polygons

Hurtado, Noy and Urrutia [19] showed that if a set of n points has k *convex layers*[1], then one triangulation can be transformed into the other triangulation using $O(kn)$ flips.

We denote a set of vertices, edges and faces of $T(S)$ by V, E and F, respectively. A triangulation $T(S)$ is always associated with a *dual graph*. Let $\mathcal{D}(T) = (V_{\mathcal{D}}, E_{\mathcal{D}})$ be the dual graph of $T(S)$. We can define a bijection $g : F \hookrightarrow V_{\mathcal{D}}$, and then $(v,w) \in E_{\mathcal{D}}$ for any distinct $v, w \in V_{\mathcal{D}}$ if and only if the triangles, or the reverse images, $g^{-1}(v)$ and $g^{-1}(w)$ share a common edge in E. Fig.2 (a) and (b) depict a triangulation of planar points and its dual graph, respectively.

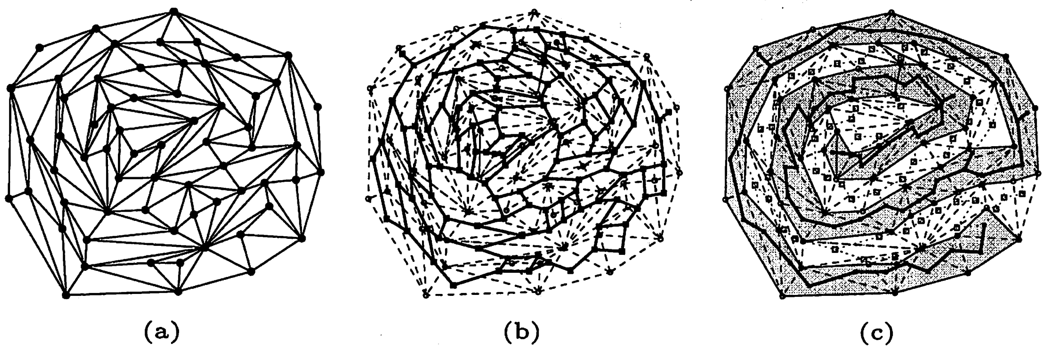

Fig. 2. Triangulation $T(S)$ of planar points, its dual graph $\mathcal{D}(T)$, and a polygon tree on $\mathcal{D}(T)$.

We neglect the unbounded face in a triangulation and its dual. So, we will refer each face in a triangulation as triangle, and the degree of each vertex in the dual is at most 3. Since a triangulation $T(S)$ is a planar graph, the size of a triangulation and its dual can be derived from the Euler's formula[2]. Hence we have that $|V| = n$, $|E| = 3n-3-k$, $|F| = |V_{\mathcal{D}}| = 2n-2-k$, and $|E_{\mathcal{D}}| = 3n-3-2k$, where $k = |\mathcal{CH}(S)|$.

We also consider a *polygon triangulation* of a simple polygon P, denoted by $T(P)$, that is, a simplicial decomposition of P whose vertices are precisely the polygon vertices of P, or the subdivision of P into non-overlapping triangles using diagonals only. The following theorems are important to design our heuristic algorithm.

Theorem 1 (Triangulation Theorem, see e.g., [30]). *Every simple polygon admits a polygon triangulation. A simple polygon of n vertices may be partitioned into $n-2$ triangles by additional $n-3$ internal diagonals.* □

[1] convex layers in a set of planar points are obtained from removing the convex hull (the first layer) and repeating the operation with the remaining point set until no point is left

[2] For any planar graph $G = (V, E)$, we always have $|V| - |E| + |F| = 1$.

Theorem 2 **(see e.g., [30]).** *The dual graph of a triangulation of a simple polygon forms a tree.* □

Theorem 3 **(Meisters' Two Ears Theorem [25]).** *Every simple polygon with $n \geq 4$ vertices has at least two non-overlapping ears. An* ear *of simple polygon P is a triangle such that one of its edges is a diagonal of P and the remaining two edges are edges of P.* □

We define a *polygon tree* on the dual of a triangulation $T(S)$ of planar points. A polygon tree $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$ on the dual $\mathcal{D}(T) = (V_{\mathcal{D}}, E_{\mathcal{D}})$ of a triangulation T is a tree such that $\cup_{v \in \mathcal{T}} g^{-1}(v)$ is a simple polygon, where g is bijection from a face in T to a vertex in $\mathcal{D}(T)$. Hence, we say a polygon tree $\mathcal{T}$ *maximal* if and only if appending any edge $e \in E_{\mathcal{D}} \setminus E_{\mathcal{T}}$ into $E_{\mathcal{T}}$ makes $\cup_{v \in \mathcal{T}} g^{-1}(v)$ nonsimple polygon. Fig.2 (c) depicts a maximal polygon tree and the dual polygon.

3 Heuristic Algorithm

In this section, we describe a triangulation-base heuristic algorithm for computing a simple polygonalization of S. We assume that a triangulation $T(S)$ is stored in a canonical data structure for maintaining *Planar Straight Line Graphs* such as *Halfedge* data structure [9] or doubly connected edge list [6]. Halfedge supports efficient local modifications with constant time such as edge-flipping operation. Note that we do not explicitly maintain $\mathcal{D}(T)$, since Halfedge provides efficient functions for the bijection between $T(S)$ with $\mathcal{D}(T)$. Algorithm 1 shows an outline of our heuristic algorithm for computing a simple polygonalization.

Algorithm 1: Heuristic for computing a simple polygonalization

```
Input  : A set $S$ of points in general position in the plane
Output: A random simple polygonalization
1 Let initialize $T(S)$ with a randomly generated triangulation of $S$;
2 Construct the dual graph $\mathcal{D}(T)$ of $T(S)$;
3 Compute a random maximal polygon tree $\mathcal{T}$ on $\mathcal{D}(T)$;
4 Construct a simple polygon by traversing the tree $\mathcal{T}$ with depth-first search;
```

3.1 On generating a random triangulation

In the first of Algorithm 1, we generate a random triangulation $T(S)$. Recently, considering a random triangulation has been received various attentions. For instance, Sharir and Welzl [35] show several results on the numbers of planar triangulations by using some properties of random triangulations. However, it does not seem that many have been known about generating a random triangulation, although Epstein and Sack [13] propose efficient $O(n^3)$ and $O(n^4)$ time algorithms for counting triangulations and generating a random one of a given simple polygon with n vertices, respectively. Aichholzer [1, Section 4.3] suggests an idea for generating a random triangulation: first, enumerate all possible triangulations, and number them in generating order; next, generate a random number $r \in \{i\}_{i=1}^{t(S)}$, where $t(S)$ is the number of triangulations of S; finally, report the r-th triangulation. It seems to be expensive to compute all triangulations for larger instances.

Our idea of generating a random triangulation is to perform random edge-flipping operations. However, it is a folklore open problem to determine the mixing rate of the Markov process that starts at some triangulation and keeps flipping a random flippable edge; see [24, 28] where this is treated for points in convex position. In fact, the mixing rate of triangulation of planar n points in convex position is bounded by $O(n^4 \log n)$.

This leads that our $O(n \log n + f)$ heuristic algorithm has the same time complexity as that of 2-opt Moves even in the special case for which all planar points are in a convex position. However, we can generate any triangulation of S with a positive probability by performing random $O(n^2)$ edge-flipping operations in the worst case. Since the expected number of convex layers for n uniformly and independently distributed points is $\Theta(n^{2/3})$, due to Dalal [11], random $O(n^{5/3})$ edge-flipping operations may be required in the average case. Therefore, when we stand for the sense that we want to generate a polygon as an instance, or do not require fairness of generated polygons, we can consider our heuristic algorithm has advantage to 2-opt Moves.

3.2 Computing a random polygon tree

We describe a procedure for computing a random polygon tree $\mathcal{T}$ on the dual $\mathcal{D}(T) = (V_{\mathcal{D}}, E_{\mathcal{D}})$ of triangulation T of S. Algorithm 2 is a pseudo-code for computing a random polygon tree $\mathcal{T}$. Throughout the Algorithm 2, we maintain two sets X and Y: X is a set of vertices in $V_{\mathcal{D}}$ visited so far; Y is a set of current candidate edges in $E_{\mathcal{D}}$ which connect between vertices $v^* \in X$ and $w^* \in V_{\mathcal{D}} \setminus X$. In the line 6 – 14, the procedure augments the current polygon tree by adding an appropriate edge one at a time. An edge $(v^*, w^*) \in E_{\mathcal{D}}$ is admitted as of polygon tree $\mathcal{T}$, where $v^* \in X$ and $w^* \notin X$, if a triangle $g^{-1}(w^*)$ has a vertex marked unvisited. Otherwise, the union of triangles $\bigcup_{v^* \in X \cup \{w^*\}} g^{-1}(v^*)$ induces a nonsimple polygon.

Algorithm 2: Computing a random polygon tree $\mathcal{T}$

```
   Input  : A triangulation $T(S) = (V, E)$ and its dual graph $\mathcal{D}(T) = (V_{\mathcal{D}}, E_{\mathcal{D}})$.
   Output: A random polygon tree $\mathcal{T}$.
1  forall $v \in V$ do mark $v$ as the label unvisited ;
2  forall $e^* \in E_{\mathcal{D}}$ do mark $e^*$ as the label dual_edge ;
3  Randomly choose a vertex $v^* \in V_{\mathcal{D}}$ and mark to all vertices of triangle $g^{-1}(v^*)$ as visited ;
4  $X \leftarrow \{v^*\}$;
5  $Y \leftarrow \{(v^*, w^*) \in E_{\mathcal{D}} \mid v^* \in X,\ w^* \in V_{\mathcal{D}} \setminus X\}$;
6  while $Y \neq \emptyset$ do
7      Randomly choose an edge $e^* = (v^*, w^*)$ from $Y$, and $Y \leftarrow V \setminus \{e^*\}$;
       (*Assumption: Assumption: $v^* \in X,\ w^* \notin X$.   *)
8      if $g^{-1}(w^*)$ has a vertex $v$ marked unvisited then
9          Mark visited to $v$;
10         Mark polygon_edge to $e^*$;
11         $Y \leftarrow Y \cup \{$edges in $E_{\mathcal{D}}$ incident to $w^*$ but not $e^*\}$;
12     end
13     $X \leftarrow X \cup \{w^*\}$;
14 end
15 if $|X| \neq |V_{\mathcal{D}}|$ then return false;
16 else
17     $E_{\mathcal{T}} \leftarrow \{e^* \in E_{\mathcal{D}} \mid e^*$ is marked polygon_edge$\}$;
18     $V_{\mathcal{T}} \leftarrow \{v^* \in V_{\mathcal{D}} \mid v^*$ is a vertex of $e^* \in E_{\mathcal{T}}\}$;
19     return $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$;
20 end
```

We can show a few properties of polygon tree generated by Algorithm 2.

Lemma 1. *A subgraph $\mathcal{T}$ of $\mathcal{D}(T)$ which is generated by Algorithm 2 is a maximal polygon tree.*

Proof. It is obviously that $\mathcal{T}$ is connected, since the procedure grows the current polygon tree by adding an edge incident to a vertex in X with the other in $V_{\mathcal{D}} \setminus X$ one by one.

It can be completed by showing there is no cycle in $\mathcal{T}$. When we assume that $\mathcal{T}$ contains a cycle, intermediately we have a contradiction. To construct a cycle, we have to violate at least two times for the condition of line 8 in Algorithm 2.

When the while-loop, i.e. at the line 14, ends, the maximality can be satisfied since the procedure has tried each adjacent triangles to check whether or not it is admitted. □

Unfortunately, there exist undesirable situations for which this procedure goes into a deadlock: the simple polygonalization induced by the reported maximal polygon tree of Algorithm 2 cannot cover all given points. Fig.3 shows the undesirable situation. Therefore, Algorithm 1 is a Monte-Carlo algorithm [29] which does not guarantee always to compute a feasible simple polygonalization of S.

However, our heuristics can generate each possible simple polygonalization when it does not return false.

Lemma 2. *Algorithm 1 generates each possible simple polygonalization of given set S of planar points.*

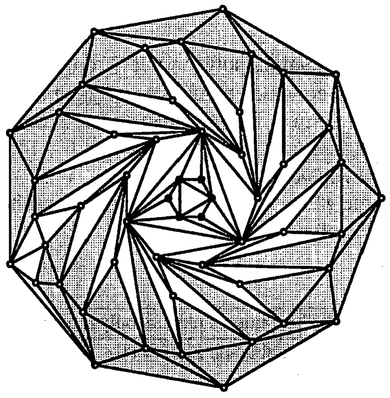

Fig. 3. An example in which Algorithm 2 goes into a deadlock; there exist points in S which cannot be covered by resulting simple polygonalization. White points are the vertices of the resulting simple polygon, Black points are unvisited by Algorithm 2.

Proof. Let P be an arbitrary simple polygonalization of S. We consider a triangulation $T(S)$ which contains a triangulation $T(P)$ of P as a subgraph.

From Theorem 3, $T(P)$ has at least two ears for $n > 3$. Let v_i, v_{i+1} and v_{i+2} be one of the ears in $T(P)$. Cutting the ear from $T(P)$, we obtain a $(n-1)$-gon $(v_1, v_2, \ldots, v_i, v_{i+2}, \ldots, v_n)$. We repeat this procedure until P converges to a triangle (v_a, v_b, v_c) while maintaining the order performing ear-cutting operations. We can obtain a simple polygonalization P, when Algorithm 2 is executed as follows: starting at the vertex corresponding to triangle (v_a, v_b, v_c); and growing the current polygon tree in the reverse order of ear-cuttings. Note that Algorithm 2 computes a polygon tree $\mathcal{T}$ without going into a deadlock. Hence this completes the proof. □

Now, we estimate the time and space complexities of Algorithm 1. We can see that the space complexity is $O(n)$ for given set of n planar points from observations in Section 2. For the time complexity, the most expensive part is to compute a triangulation of S, or to perform random flipping-edges, in Algorithm 1, We assume that the number f of edge-flipping operations are given in advance. Since there are a number of algorithms for constructing a triangulation $T(S)$ in the optimal time $\Theta(n \log n)$ in the worst case (see e.g., [32, 6]), the time complexity is $O(n \log n + f)$.

Theorem 4. *Algorithm 1 can compute a simple polygonalization of given set of n planar points in $O(n \log n + f)$ time when it does not returns* false.

4 Simple Polygonalizations Problem

We consider the following extension, we call Simple Polygonalizations Problem. Given a set S of n points in the plane, generate a set of disjoint simple polygons $\{P_1, P_2, \ldots, P_k\}$ such that the union of all vertices of each P_i is exactly S. The random simple polygonalizations may be an instance of Robot Motion Planning Problems (see e.g., [6, Chapter 13, 15]).

Algorithm 3 shows our idea for this generalized problem. First, we compute a random triangulation $T(S)$ of S. This step is the same as in Algorithm 2. Second, we consider the dual $\mathcal{D}(T) = (V_{\mathcal{D}}, E_{\mathcal{D}})$ and the *map graph* $\mathcal{M}(T) = (V_{\mathcal{D}}, E_{\mathcal{M}})$ of $T(S)$. Map graph is a generalized dual of planar graph G in which two regions of G are adjacent when they share any vertex of their boundaries (not an edge, as standard planarity requires), see Chen, Grigni, and Papadimitriou [7]. Third, we compute an independent set $I_{\mathcal{M}(T)} = \{v_1^*, \ldots, v_k^*\}$ on the map graph $\mathcal{M}(T)$. Note that $g^{-1}(v_i^*) \cup g^{-1}(v_j^*) = \emptyset$ for any two different vertices $v_i^*, v_j^* \in I_{\mathcal{M}(T)}$. Finally, we compute a random maximal polygon forest $\mathcal{F}$ which is defined as a set of maximal polygon trees. The maximal polygon forest can be computed by Algorithm 2 with $X = I_{\mathcal{M}(T)}$ and $Y = \{(v^*, w^*) \in E_{\mathcal{D}} \mid v^* \in X, w^* \in V_{\mathcal{D}} \setminus X\}$.

In the simple polygonalizations problem, Algorithm 3 can always generate a number of simple polygons, since there is no deadlocks if we restart growing a new polygon tree from $v^* \notin X$ and $g^{-1}(v^*)$ has an unvisited vertex.

Algorithm 3: Heuristic for computing simple polygonalizations

```
Input  : A set $S$ of points in general position in the plane
Output: A random simple polygonalization
1 Let initialize $T(S)$ with a randomly generated triangulation of $S$;
2 Construct the dual graph $\mathcal{D}(T)$ and the map graph $\mathcal{M}(T)$ of $T(S)$;
3 Compute a random independent set $I_{\mathcal{M}}(T) = \{v_1^*, \ldots, v_k^*\}$ on $\mathcal{M}(T)$;
4 Compute a random maximal polygon forest $\mathcal{F}$ on $\mathcal{M}(T)$;
5 Construct simple polygons by traversing each tree in $\mathcal{F}$ with depth-first searches;
```

This heuristic algorithm solves a relaxed problem proposed as a future work by Auer and Held [2]: Given a set S of n points and a natural number $k \leq \frac{n}{3}$, generate k random polygons on S. If we apply our idea to this problem, we must solve a difficult problem for generating an independent set on a map graph whose cardinarity is just k.

5 Concluding remarks and Future works

We have presented a triangulation-base heuristic algorithm for computing a simple polygonalization of given planar point set in $O(n \log n + f)$ time, where f is the number of edge-flipping operations. Our heuristic algorithm is Monte-Carlo Algorithm, that is, it has undesirable situations. Hence we have to estimate how often our heuristics goes into a deadlock with computer experiments. Furthermore, we have to evaluate our algorithm from various points of view including its running time, the fairness of generating simple polygonalization, and behavior for larger instances by experiments.

One of the most important future works is to guarantee that Algorithm 2 always can compute a simple polygonalization of S. In other words, it may be necessary to backtrack and restore while computing a maximal polygon tree if it is trapped by a deadlock. We have designed a recursive procedure to restart growing polygon tree, but there still exist undesireble situations.

In the simple polygonalizations problem, we are also interested in designing an efficient algorithm for enumerating maximal independent sets in the map graph, although there exists an algorithm for generating all maximal independent sets in a polynomial delay [20].

References

1. O. Aichholzer. The path of a triangulation. In *Proc. 15th Ann. ACM Symp. Computational Geometry*, pp. 14–23, Miami Beach, Florida, USA, 1999.
2. T. Auer and M. Held. Heuristics for the Generation of Random Polygons. *Proc. 8th Canadian Conference on Computational Geometry*, pp. 38–44, 1996.
3. T. Auer. Heuristics for the Generation of Random Polygons. Master's thesis, Computerwissenschaften, U. Salzburg, A-5020 Salzburg, Austria, June 1996.
4. D. Avis and K. Fukuda. Reverse Search for Enumeration. *Discrete Applied Mathematics*, 65:21–46, 1996.
5. S. Bespamyatnikh. An efficient algorithm for enumeration of triangulations. *Computational Geometry Theory and Application*, 23(3):271–279, 2002.
6. M. de Berg, O. Schwarzkopf, M. van Kreveld and M. Overmars. Computational Geometry: Algorithms and Applications 2nd edition, *Springer-Verlag*, 2000.
7. Z.-Z. Chen, M. Grigni and C.H. Papadimitriou. Map Graphs. *Journal of ACM*, 49(2):127–138, 2002.
8. CGAL: Computational Geometry Algorithms Library. `http://www.cgal.org/`.
9. S.-W. Cheng. Planar Straight Line Graphs. Handbook of Data Structures and Applications, Edited by D.P. Mehta and S. Sahni, *Chapman & Hall*, 2005.
10. I.K. Crain. The Monte-Carlo generation of random polygons. *Computers and Geosciences*, 4:131–141, 1978.
11. K. Dalal. Counting the Onion. *Random Structures and Algorithms*, 24(2):155–165, 2004.
12. E.D. Demaine. Simple Polygonizations. `http://theory.lcs.mit.edu/~edemaine/polygonization/`.
13. P. Epstein and J. Sack. Generating triangulations at random. *Proc. 4th Canadian Conference on Computational Geometry*, pp. 305–310, 1992.
14. S.P. Fekete. On Simple Polygonalizations with Optimal Area. *Discrete Comput. Geom.*, 23:73–110, 2000.
15. S. Fortune. Voronoi Diagrams and Delaunay Triangulations. Computing in Euclidean Geometry, Edited by Ding-Zhu Du and Frank Hwang, World Scientific, *Lecture Notes Series on Computing – Vol. 1*, pp. 193–233, 1992.
16. A. García, M. Noy and J. Tejel. Lower bounds on the number of crossing-free subgraphs of K_N. *Computational Geometry*, 16:211–221, 2000.

17. J.E. Goodman and J. O'Rourke. Handbook of Discrete and Computational Geometry 2nd Edition. *Chapman & Hall,* 2004.
18. J. Hershberger and S. Suri. Matrix Searching with the Shortest Path Metric. *SIAM Journal on Computing,* 26:1612–1634, 1997.
19. F. Hurtado, M. Noy and J. Urrutia. Flipping Edges in Triangulations. *Discrete Comput Geom.,* 22:333–346, 1999.
20. D.S. Johnson, M. Yannakakis and C.H. Papadimitriou. On generating all maximal independent sets. *IPL,* 27(3):119–123, 1988.
21. M. van Kreveld and I. Reinbacher. Good NEWS: Partitioning a Simple Polygon by Compass Directions. *International Journal of Computational Geometry & Applications,* 14:233–259, 2004.
22. C.L. Lawson. Transforming triangulations. *Discrete Math.,* 3:365–372, 1972.
23. J. van Leeuwen and A.A. Schoone. Untangling a traveling salesman tour in the plane, In J.R. Muhlbacher, editor, *Proc. 7th Conf. Graph-theoretic Concepts in Comput. Sci.,* pp. 87–98, 1981.
24. L. McShine and P. Tetali. On the mixing time of the triangulation walk and other Catalan structures. DIMACS-AMS volume on Randomization Methods in Algorithm Design (Eds. P.M. Pardalos, S. Rajasekaran, and J. Rolim) *DIMACS Series in Discrete Mathematics and Theoretical Computer Science,* 43:147–160, 1998.
25. G.H. Meisters. Polygons have ears. *American Mathematical Monthyly,* 82:648–651, 1975.
26. H. Meijer and D. Rappaport. Upper and lower bounds for the number of monotone crossing free Hamiltonian cycles from a set of points. *ARS Combinatoria,* 30:203–208, 1990.
27. J.S.B. Mitchell and J. O'Rourke. Computational geometry column 42. *International Journal of Computational Geometry and Applications,* 11(5):573–582, 2001.
28. M. Molloy, B. Reed and W. Steiger. On the mixing rate of the triangulation walk. DIMACS-AMS volume on Randomization Methods in Algorithm Design (Eds. P.M. Pardalos, S. Rajasekaran, and J. Rolim) *DIMACS Series in Discrete Mathematics and Theoretical Computer Science,* 43:179–190, 1998.
29. R. Motwani and P. Raghavan. Randomized Algorithms. *Cambridge University Press,* 1995.
30. J. O'Rourke. The art gallery theorems and algorithms. *Oxford University Press,* 1987.
31. J. O'Rourke and M. Virmani. Generating Random Polygons, Technical Report 011, CS Dept. Smith College, Northhampton, MA 01063, July 1991.
32. F.P. Preparata and M.I. Shamos. Computational Geometry: An Introduction, *Springer-Verlag,* 1985.
33. J. -R. Sack and J. Urrutia. Handbook of Computational Geometry. *Elsevier Science Publishers,* 2000.
34. M. Sharir and E. Welzl. On the number of crossing-free matchings, cycles, and partitions. *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms,* pp. 860–869, 2006.
35. M. Sharir and E. Welzl. Random Triangulations of Planar Point Sets. to appear in *Proc. 22nd Ann. ACM Symp. Computational Geometry,* 2006.
36. T. Shermer. Recent results in art galleries. *Proc. IEEE,* 80:1384–1399, 1992.
37. C. Sohler. Generating Random Star-Shaped Polygons. *Proc. 11th Canadian Conference on Computational Geometry,* pp. 174–177, 1999.
38. R. Ulber. On The Number Of Star-Shaped Polygons And Polyhedra. *Proc. 11th Canadian Conference on Computational Geometry,* pp. 170–173, 1999.
39. V.V. Vazirani. Approximation Algorithms. *Springer-Verlag,* Berlin, 2001.
40. C. Zhu, G. Sundaram, J. Snoeyink, and J.S.B. Mitchel. Generating Random Polygons with Given Vertices. *Computational Geometry Theory and Application,* 6(5):277–290, 1996.