GRP_CH Heuristic for generating Random Simple Polygon

Sanjib Sadhu, Subhashis Hazarika, Kapil Kumar Jain,
Saurav Basu, and Tanmay De

Department of Computer Science and Engineering,
National Institute of Technology, Durgapur
{sanjibsadhu411,hsubhashis,jainkkapil,
saurabh.basu.cs,tanmoydeynitd}gmail.com

Abstract. A heuristic 'GRP_CH' has been proposed to generate a random simple polygon from a given set of 'n' points in 2-Dimensional plane. The "2-Opt Move" heuristic with time complexity $\mathcal{O}(n^4)$ is the best known (referred in [1]) among the existing heuristics to generate a simple polygon. The proposed heuristics, 'GRP_CH' first computes the convex hull of the point set and then generates a random simple polygon from that convex hull. The 'GRP_CH' heuristic takes $\mathcal{O}(n^3)$ time which is less than that of "2-opt Move" heuristic. We have compared our results with "2-Opt Move" and it shows that the number of unique polygons generated in 'GRP_CH' is more than that of "2-Opt Move".

Keywords: Simple polygon, Convex Hull,Visibility of a line segment

1 Introduction

There are several importances of generating a simple polygon randomly from a given set of points, e.g. verifying time complexity for geometric algorithm and generating the test instances of geometric problem. In this paper, our attempt is to generate randomly a simple polygon from a set S=$\{s_1,s_2,\ldots,s_n\}$ of n points which lie on a 2-dimensional plane.

Although there exists a lot of heuristics to generate a random polygon from n points, still it is an open problem to generate it uniformly at random, i.e. generating a polygon with probability $(\frac{1}{j})$ if there exists j simple polygons on set S in total. We have designed a new heuristic which generate a large number of unique random polygons than the existing heuristic and also with less time complexity. Our heuristic is based on the construction of convex hull of point set and the visibility of a line segment from a point.

A polygon P is said to be **simple**[8] if it does not contain any hole and no edges of it intersect with each other, except that neighbouring edges meet only at their common end point known as vertices of the polygon.

A subset S of the plane is called convex[8] if and only if for any pair of points $p, q \in S$ the line segment $l(p, q)$ is completely contained in S. The Convex Hull $CH(S)$ of a point set S is the smallest convex set that contains S.

An edge $e(u, v)$ of a simple polygon is said to be **fully visible** from a point r if and only if both the end points u & v of the edge e are visible [7] to the point r.

In this paper, we have designed a new heuristic known as "GRP_CH" for generating a random simple polygon from a set of n points and this runs with time complexity $\mathcal{O}(n^3)$. Since the "2-Opt Move" heuristic[1] is the best known (referred in [1]) among the existing heuristics with time complexity $\mathcal{O}(n^4)$ to generate a simple polygon, we have compared our result with that of "2-Opt Move" heuristic. Our result shows that the number of unique simple polygon generated through "GRP_CH" heuristic is more than that of "2-Opt Move" heuristic.

This paper is organized as follows: In Section 2, the existing heuristics are described. Section 3 deals with detailed description of our proposed heuristic "GRP_CH". In Setion 4 results are discussesed. We conclude our paper in Section 5.

2 Literature survey

From 1992, the generation of geometric objects became an interesting research topic to the researchers. Epstein[4] studied random generation of triangulation. Zhu[6] designed an algorithm to generate an x-monotone polygon on a given set of vertices uniformly at random. A heuristics for generating simple polygons was investigated by ORourke et. al.[2] in 1991. However, the vertices move while creating a polygon in their algorithm. The 2-Opt Moves heuristic was first proposed to solve the traveling salesman problem by J.van Leeuwen et. al.[5] In 1996, Thomas Auer et. al.[1] presented a study of all heuristics present at that time and reported a variety of comparison among them. There implementations are now part of RANDOMPOLYGONGENERATOR, RPG which is publically available via `http://www.cosy.sbg.ac.at/~held/projects/rpg/rpg.html`. The existing heuristics for random polygon generating simple polygon are discussed below:

- Permute & Reject
  "Permute & Reject" has been designed by Thomas Auer et. al[1]. The algorithm creates a permutation of S and check whether this permutation corresponds to a simple polygon. If the polygon is simple then it is output; otherwise a new polygon is generated. Obviously, the actual running time of this method mainly depends on how many polygons need to be generated in order to encounter a simple polygon. Clearly, "Permute & Reject" produce all possible polygons with a uniform distribution, but this algorithm is not applicable to anything but extremely small set of points.
- Space partitioning
  To generate a random simple polygon Thomas Auer et. al.[1] designed "Space partitioning" algorithm which recursively partitions a set of point S into subsets and those subsets have disjoint convex hulls. The algorithm has been described in detail in [1]. In the worst case, this algorithm takes $\mathcal{O}(n^2)$ time.

Unfortunately, Space Partitioning does not generate every possible polygon on S.

– Steady Growth

"Steady Growth" has also been designed by Thomas Auer et. al. [1]. As initialization, Steady Growth randomly select three points s1, s2, s3 from the set S such that no other point of S lies within convex hull, CH(s1, s2, s3). This convex hull is taken as a start polygon. In each of the following iteration steps a point s is chosen in such a way that by appending s to the polygon, it's convex hull again does not contain any further points of the point set. Then an edge (u,v) of the polygon that is completely visible from s is searched and replaced by the chain (u,s,v). This way the polygon is extended with the point s. By using "Steady Growth", one can compute a simple polygon in at most $\mathcal{O}(n^2)$ time; since all that has to be done during each iteration is to compute all edges which are completely visible. Unfortunately "Steady Growth" does not generate every possible polygon on S.

– 2-Opt Move

Although Zhu et al.[6] gave the idea of "2-Opt Move" first, it was designed by Thomas Auer et. al.[1] in 1992. This algorithm first generates a random permutation of S, which again is regarded as the initial polygon P. Any self intersections of P are removed by applying so called "2-Opt Move". Every "2-Opt move" replaces a pair of intersecting edges (v_i,v_{i+1}), (v_j,v_{j+1}) with the edges (v_{j+1},v_{i+1}) and (v_j,v_i) as shown in fig1. In this application, at each iteration of the algorithm one pair of intersecting edge is chosen at random and the intersection is removed. Leeuwen et al.[5] has shown that

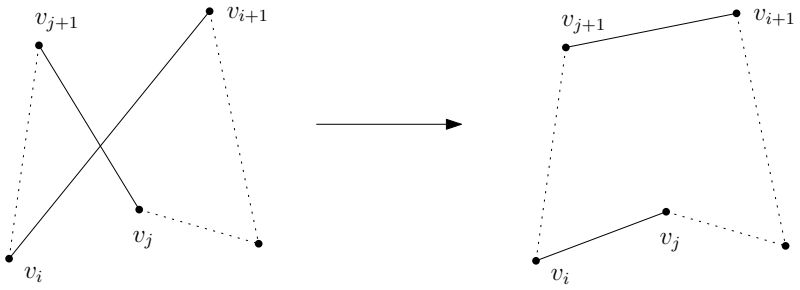


Fig. 1. An example of a 2-opt move.

for obtaining a simple polygon, at most $\mathcal{O}(n^3)$ many "2-Opt move" required to be applied. Thus, an overall time complexity of $\mathcal{O}(n^4)$ can be achieved. The "2-Opt Move" heuristic will produce all possible polygons, but not with a uniform distribution[6]. However, this is reported as the best heuristic[1] among all the existing ones in the sense that it produces varities of different simple polygons from the point set S.

3 Proposed Heuristics

The proposed heuristic **GRP_CH** is a randomized algorithm which generates a random simple polygon from a set of n points lying on a 2-dimensional plane.

3.1 Assumption

The point set S lie on a plane in general position that means no three points or vertices are co-linear. The polygon is represented by clockwise orientation of its vertices.

3.2 GRP_CH Algorithm

The convex hull of the point set S is first constructed by using Graham's scan[8] algorithm. Therefore all the remaining points will lie inside the convex hull. This convex hull is taken as polygon P which will be modified later. Now randomly select a point v_i and find out how many edges of the polygon P are fully visible from that point v_i. Select any of those visible edges randomly, say $e(v_1, v_2)$. Therefore, each point as well as each visible edges has equal chance of being selected. Now connect the two ends (v_1, v_2) of that edge with the corresponding point v_i and delete that selected edge $e(v_1, v_2)$ of P. After this, the modified polygon will remain simple one. Repeat the same procedure for the remaining points.

```
Procedure GRP_CH
Input: A set S of points in 2-D Plane.
Output: A simple polygon P with vertex set V
represented in clockwise order.
 Begin
  Q = CH (S);     // Q is the vertices on the convex hull of the point set S
  P'= ConvexHull_Edges(S); //P' is set of edges of the convex hull
  S = S \ Q ;
  while (S is not empty) do
    v = Random (S);       // randomly select one point  from the set S
    E = FullyVisibleEdges(P', m, v);        //m is no of vertices of P'
      // E is the set of edges fully visible from the point v
    if (E==NULL) then exit; // simple polygon is not possible in this case
   e = Random (E);        // randomly select one edge from the set E
   P' = P' \ e;
   P' = P' U {v1, v} U {v, v2};
   S = S \ v;
  EndWhile
  P= P'
 End
```

```
FullyVisibleEdges (P, m, v)
Input: Polygon P with m vertices, a point v from which visibility is to be checked.
Output: the edge set E, visible from point v.
 Begin
   E=NULL;
   for i=1 to m
         if (the two end points of edge Ei are visible from v)
            // Edge Ei is visible from point v
            E=E U { ei }
         Endif
   Endfor
   Return E
 End
```

The Fig 2 describes the algorithm step by step graphically on a set of 8 points.

3.3 Analysis of GRP_CH algorithm

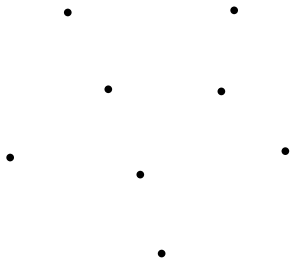
Computation of the convex hull [Graham's Scan algorithm] takes $\mathcal{O}(n\ logn)$ time. To find out whether the two lines intersect or not can be implemented in constant time. So, a particular edge which is visible or not from a point, can be checked in $\mathcal{O}(k)$ time where k is the convex hull edges. Since k$\leq$ n, this time will be $\mathcal{O}(n)$ and hence, the set of edges which are fully visible from a point can be computed in $\mathcal{O}(n^2)$ time. The while loop runs over $n-k$ times, hence overall GRP_CH algorithm takes $\mathcal{O}(n^3)$ time.

3.4 Limitation of GRP_CH algorithm

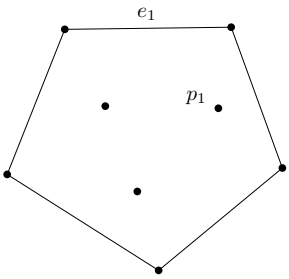
There may arise a particular situation when there is no fully visible edge of polygon P for the randomly selected point p as shown in Fig.3. In such cases, our algorithm cannot generate a simple polygon. We stop and restart our algorithm. It is less likely that again the same points and the same edges will be selecteded in ame order like before.

3.5 Probability of obtaining a simple polygon using GRP_CH Algorithm

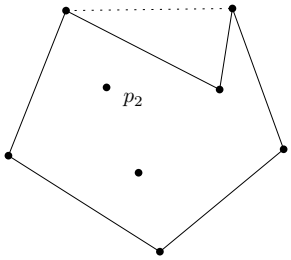
After each iteration of the "RPG_CH" algorithm, a simple polygon P is generated. Now a new point v_i will arrive inside or outside the polygon (however not outside the convex hull of the polygon vertices). Now we have to compute the probability of generation of the new polygon (P') from old polygon P after



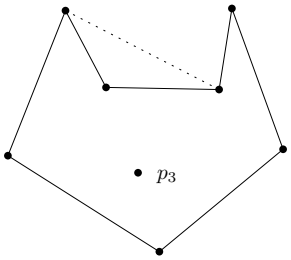
(a) Set of 8 points



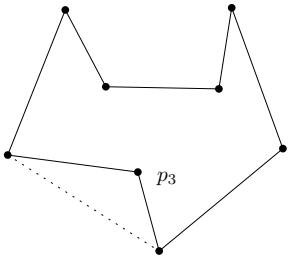
(b) Convex Hull of 8 points. Randomly select point p_1 .



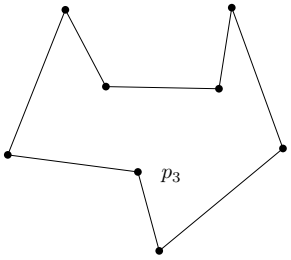
(c) Edge e_1 is selected randomly and removed.



(d)



(e)



(f)

Fig. 2. Successive step of the GRP_CH heuristic.

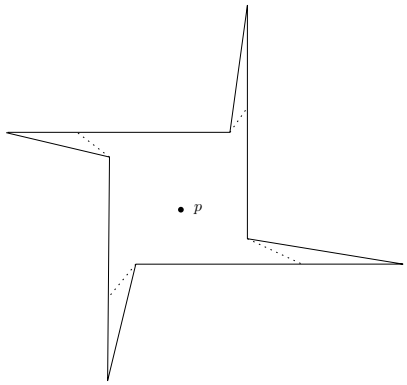


Fig. 3. The point p cannot see any edge of the polygon fully.

selecting the point v_i. If this is successful, a random point v_j will be considered next in same way just described above. Suppose at any instance there is a simple polygon P $\{v_1, v_4, v_2, v_5, v_3, v_6, v_1\}$ like Fig 4. Next a new point v_7 is to be selected. Using GRP_CH Algorithm the new polygon P can be generated including v_7 if and only if at least one edge of polygon P is fully visible from the point v_7. In the Fig 4 the new polygon P can be generated if and only if the new point v_7 does not lie inside Δ IJK where Δ IJK is that region from which no edges of polygon P is fully visible. This type of zone (Δ IJK) will be termed as "**non-visible-zone**" with repect to the polygon constructed so far. Therefore, P can be generated if and only if v_7 lies anywhere inside the convex hull of the given point set, except in "non-visible-zone". The area of the convex hull and the area of "non-visible-zone" for the polygon constructed so far (before adding this i^{th} point) are taken as A and a_i respectively.

In GRP_CH algorithm, after computing the convex hull, the remaining points and the visible edges are selected alternately. When chosing a point v_i randomly, the probability p_i that this point v_i will not lie within the the "non-visible-zone" is $p_i = \left(1 - \frac{a_i}{A}\right)$. After the point selection (v_i), one visible edge of the polygon (constructed so far) is chosen randomly and this probability is $q_i = \frac{1}{m}$, where m (say) is the total number of visible edges from the point v_i. The prbability p_i depends on the previous event of edge selection, because area a_i changes if different edge was selected in previous step. Also q_i depends on the point selected at this step, because if different point was selected, the number of visible edges from the point might differ. The sequence of the chosen points and visible edges will determine whether the simple polygon can be drawn or not. Therefore the probability of success to generate a simple polygon depends on the alternate events of point selection and visible edge selection. Hence each such events are conditional one.

Let $(n-k)$ points lie on convex hull, so k point will be inside the convex hull. If a particular sequence of k points (say $v_1, v_2, v_3, \ldots, v_k$) are chosen one after another, the probability P_r of getting a polygon by following that particular

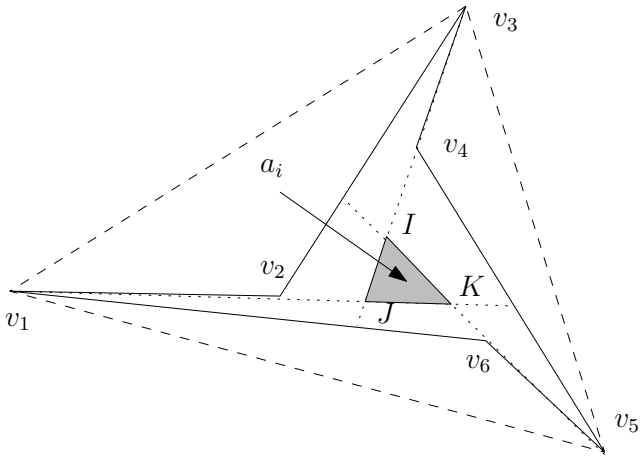


Fig. 4. The shaded area indicates "non-visible-zone" from which no edge of the polygon is fully visible.

sequence of point is

$$P_r = \left(p_1 \sum_{i_2=1}^{e_1} q_{i_2} * \left(p_2 \sum_{i_3=1}^{e_2} q_{i_3} * \left(\ldots * \left(p_{k-1} \sum_{i_{k-1}=1}^{e_{k-2}} q_{i_{k-1}} * (p_k) \right) \right) \right) \right) \quad (1)$$

where e_i is the no. of visible edges from i^{th} point v_i. Here p_i is the probability of getting a simple polygon after adding i^{th} point in the constructed polygon so far. So, by this way we can determine the probability of obtaining a polygon for a particular order of point sequence. Since there are k number of points inside the convex hull so point sequence may arrive in $k!$ ways. Hence, the probability of occurrence of each sequences is $(1/k!)$.Therefore, summing up all the probability P_r of individual sequences and multiplying with $(1/k!)$ the probability of generating a simple polygon from 'n' number of point set is obtained.

4 Result and Discussion

GRP_CH has been implemented in C++ programming language. For random number generation rand () function has been used and this rand () function belong to standard C library. Our code handles input and output in floating point format. Also, all numerical calculations are based on standard floating point arithmetic. We emphasize that we have not experienced any robustness problems in use of our code. All other software, hardware specification given below

Platform:

Operating System (Ubuntu), Ubuntu Release 10.04 (lucid), Kernel Linux 2.6.32-30-generic,GNOME 2.30.2

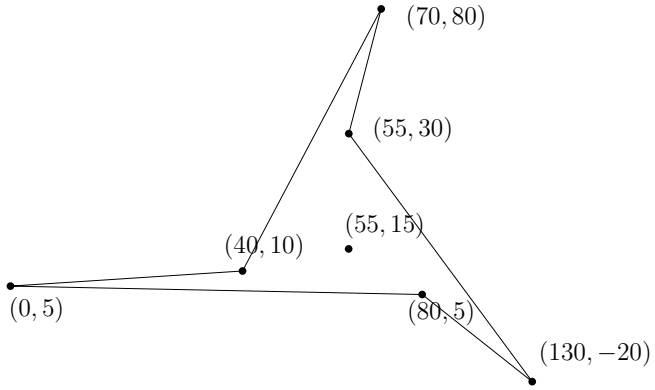


Fig. 5. An instance to show that the GRP_CH heuristic fails, but with this point set the probability of failure is 0.008

Hardware:
Memory : 2.0 GiB, Processor 0 : Intel(R) Core(TM)2 Duo CPU T5550 @ 1.83 GHz, Processor 1 : Intel(R) Core(TM)2 Duo CPU T5550 @ 1.83 GHz
Software:
GNU g++ 4.4.3

Two different series of experiments has been performed:

1. **Probability of Success of GRP_CH algorithm**
   The GRP_CH algorithm takes the random point set as input. The cardinality of point set in each simulation are 5, 10, 15, ..., 195, 200. Each simulation consists of 10,000 runs and the position of the point sets remains unaltered throughout a simulation. However, in next simulation new position of the point sets are taken. For each such simulation, a log file has been generated to count the number of simple polygon generated. It has been observed that the success rate to obtain a simple polygon is 100 percent when cardinalities of point set is not large. However the success rate of generation of random simple polygon drops slightly for large point set. This is due to the limitation of our algorithm (Section 3.4). The Fig 6 shows the probability of success of generating a random simple polygon from the point set S_i.
   The Fig 5 shows a set of seven points(along with their co-ordinates) from which a polygon consisting of six vertices has been generated using "GRP_CH" heuristic; however in next step the simple polygon will not be generated since the next point to be selected lies at position (55,15) which is within the "non-visible-zone" of the generated polygon. For this point set, the GRP_CH heuristic has been executed 10,000 times out of which simple polygons are generated 99472 times. Therefore, the probability of getting a simple polygon is 0.994. Using the mathematical formula, the probability of success for the

same set of points is found to be 0.992. This indicates that the simulation result (probability of success) matches very close with that of theoretical result.

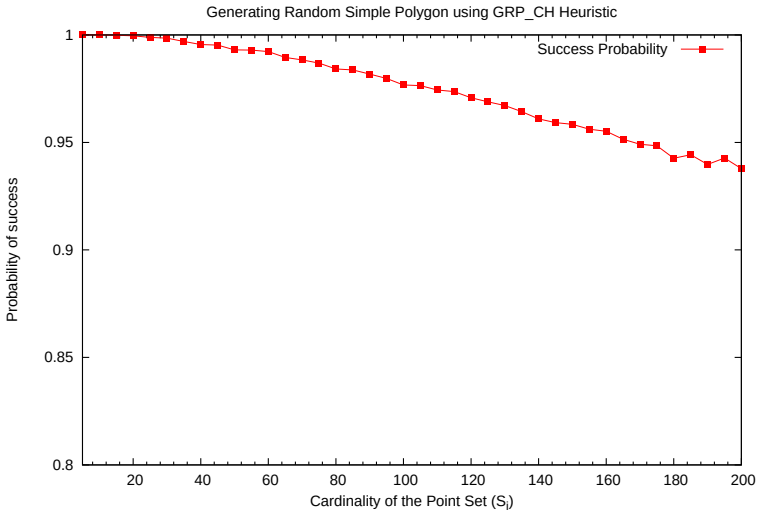


Fig. 6. An instance to show that the GRP_CH heuristic fails, but with this point set the probability of failure is 0.008

2. **Unique polygons generated for given cardinality of the point set** We found out the number of unique polygons generated from the set of the all simple polygons generated on each simulation for the point sets of cardinalities 5, 6, 7,. . ., 12, 13. Here also, for each simulation which consists of 10,000 runs, the position of the point sets remains unaltered. The results of "GRP_CH" algorithm and "2-Opt Move" for the same set of input configuration has been shown in the Table 1. This shows that the number of unique polygons generated by GRP_CH is larger than that of "2-Opt Move". However, "2-Opt Move" has success probability one.

5 Conclusion and future work

We have presented a heuristic for generating random simple polygons. It has been shown that the probability of successful generation of a random simple polygon by "GRP_CH" is very good for point sets of cardinality below 100. However it tends to decrease as the cardinality of point sets increases. The experiment has been carried out also to find out the number of unique polygons generated on 10,000 runs for the given set of point and compared the results with existing

Simulation Number	|S| $\longrightarrow$	5	6	7	8	9	10	11	12	13
1	2-Opt	4	12	17	57	83	135	167	1171	3569
	GRP_CH	4	12	17	57	83	136	179	1792	6125
2	2-Opt	8	5	19	60	251	341	1987	813	2065
	GRP_CH	8	5	19	60	254	359	2696	976	3277
3	2-Opt	4	5	17	57	105	888	484	2611	2022
	GRP_CH	4	5	17	57	105	980	534	4061	3558
4	2-Opt	4	13	19	67	210	440	552	2578	3615
	GRP_CH	4	12	17	57	83	136	179	1792	6125
5	2-Opt	4	13	17	72	34	590	1041	2450	2225
	GRP_CH	4	13	17	72	34	631	1358	3874	3341
6	2-Opt	4	5	19	61	95	357	1227	681	3197
	GRP_CH	4	5	19	61	95	366	1606	939	5911
7	2-Opt	4	12	19	26	30	127	1168	3913	2777
	GRP_CH	4	12	19	26	30	130	1507	6052	4777
8	2-Opt	4	12	6	27	83	312	562	1694	2128
	GRP_CH	4	12	6	27	83	326	611	2424	3353
9	2-Opt	8	13	19	61	216	372	499	2491	4614
	GRP_CH	8	13	19	61	217	405	557	4026	7076
10	2-Opt	8	12	46	163	102	343	2137	760	2042
	GRP_CH	8	12	46	163	102	361	2790	936	3536

Table 1. The number of unique polygons generated by "2-Opt Moves" and "GRP_CH" algorithm

"2-Opt Move" method. We have shown also that the number of unique simple polygons generated by "GRP_CH" algorithm are larger than that of "2-Opt Move".

From a theoretical point of view, it remains an open problem to generate polygons on a given set of points uniformly at random. Our heuristic has improved the present existing heuristics although it has a limitation and the removal of that limitation will be a future work.

References

1. Auer, T., Held, M.: Heuristics for the Generation of Random Polygons: In Proc. 8th Canadian Conference computational Geometry, pp. 38-44 ,Ottawa,Canada, Carleton University Press,(1996).
2. O'Rouke, J.,Virmani, M.: Generating Random Polygons. Technical Report 011,CS Dept., Smith College,Northampton,MA 01063,(1991).
3. Chazelle, B., Incerpi, J.: Triangulation and Shape complexity. ACM Trans.Graph., 3(2):pp. 135-152, (1984).
4. Epstein, P., Sack, J.: Generating triangulation at random : ACM Transaction on Modeling and Computer Simulation, Vol 4, No. 3. pp. 267-278 (1994).
5. Leeuwen, J.V., Schoone, A.A. : Untangling a travelling salesman tour in the plane. In J. R. Muhlbacher, editor. Proc.: 7th Conference Graph-theoretic Concepts in Computer Science.(WG81), pp 87-98 (1982).

6. Zhu, C., Sundaram, G., Snoeyink, J., Mitchel, J.S.B.: Generating random polygons with given vertices: Comput. Geom. Theory and Application, Vol. 6, Issue 5: pp. 277-290, (1996).
7. Ghosh, S.K.: Visibility Algorithm in the plane: Cambridge University press, 2007.
8. Kreveld, M.V., Berg, M.D., Schwartskopf, O., Overmars, M.: Computational Geometry, Algorithm and Application: Spinger publication (1996).