Malaysian Journal of Mathematical Sciences 10(S) March : 131-142 (2016)
Special Issue: The 10th IMT-GT International Conference on Mathematics, Statistics and its Applications 2014 (ICMSA 2014)

**MALAYSIAN JOURNAL OF MATHEMATICAL SCIENCES**

Journal homepage: http://einspem.upm.edu.my/journal

# Heuristic Algorithms for Finding Area Constrained Non-Convex *k*-gons

**Amal Dev Parakkat and Philumon Joseph**

*Department of Computer Science and Engineering,*
*Government Engineering College Idukki,*
*Kerala, India.*

*E-mail: adp.upasana@gmail.com*

## ABSTRACT

The non-convex *k*-gon of a point set *S* of size *n* ($n \geq k$) is a non-convex simple polygon which spans *k* vertices. In this work, we address the problems to compute the optimum area, maximum/minimum area non-convex *k*-gon and then we propose a heuristic algorithm to obtain the area optimized *k*-gons.

Keywords: NP class, non-convex polygons, k-gons, computational mathematics,Heuristic algorithm, area optimized polygons.

## 1. Introduction

A *k*-gon of a point set S of size $n \geq k$ can be defined as a simple non-convex polygon which spans *k* vertices, where a simple polygon (Figure 1) is a polygon in which no non-adjacent segments intersect each other (O'Rourke, 1998). A non-convex polygon (Figure 2) is a polygon which has at least one internal angle greater than $\pi$ (O'Rourke, 1998). The work/research on finding *k*-gons started from the question raised by Esther Klein in 1931. His question was "Whether there exists a constant $n_k$ such that all point set of size *n* in general position contains a convex *k*-gon?" (Erdos, 1975). Starting from Esther Klein's finding of $n_4 = 5$, various works have been conducted to bound the values of $n_k$ (Kalbeisch and Stanton, 1970) (Szekeres and Peters, 2006).

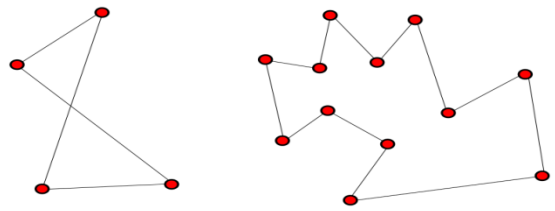

Figure 1: Examples of non-simple (Left) and simple (Right) polygon.

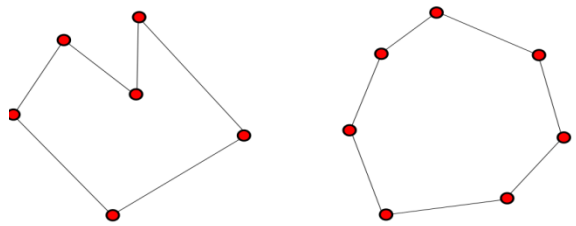

Figure 2: Examples of non-convex (Left) and convex (Right) polygon.

Most of the existing works are concentrated on tightening bounds of $n_k$ (Erdos and Szekeres, 1960; Tothand Valtr, 2005). In this work, we are going to propose a heuristic algorithm for finding approximate maximum and minimum area non-convex *k*-gons. David Eppstein et. al. introduced an $O(n^3)$ algorithm for finding convex *k*-gons (Eppstein et al., 1992). Their algorithm can only be used to find "convex" *k*-gons. But, the problem to compute maximum area non-convex *k*-gon is extremely difficult. In Eppstein et al., 1992, authors pointed out that computing area constrained non-convex *k*-gon is challenging because of its self-intersecting nature. To the best of our knowledge, no one has yet proposed any algorithm for finding area constrained non-convex k-gons.

The paper is organized as follows: Section 2 describes the naive approach for finding maximum area non-convex *k*-gon, in Section 3 we present our heuristic algorithm to compute approximate maximum (minimum) area *k*-gon. Section 4 illustrates the experimental studies along with the results and discussions and section 5 concludes the paper.

## 2. Naïve Algorithm

Naïve algorithm to find maximum area non-convex *k*-gon can be informally depicted as selecting *k* points from a set of *n* points, which gives us a maximum area simple non-convex *k*-gon. The pseudocode for finding maximum area *k*-gon through exhaustive searching is given in Algorithm 1.

**Algorithm 1. Algorithm for finding maximum area non-convex *k*-gon**

1 : Procedure EXHAUSTIVE *k*-gon (Point Set *P*, Integer *k*)
2 : Find all the combinations of *k* vertices form *P*
3 : Construct a set, *C* of all valid non-convex permutations of each of these combinations.
4 : Find the polygon with the largest area from *C*.
5 : End procedure.

The algorithm uses simple permute and reject method on all possible combinations of size *k*, and returns valid *k*-gon with maximum area. A polygon constructed from a combination is said to be valid if and only if it is simple, non-convex and spans *k* vertices.

Suppose we have a point set *P* of size *n* and our aim is to find maximum area *k*-gon, then we choose *k* points in ${}_nC_k$ different ways. In each combination, we don't know which arrangement of points will result in a valid polygon. Hence, we have to check all possible permutations (k!) of each combinations. In total we have to go through all ${}_nC_k*k!$ possibilities to find maximum area *k*-gon. The problem might sound trivial but Table 1 shows the underlying complexity.

Given a polygon *P* and area *A*, we can easily check whether it is a non-convex *k*-gon with area *A*, which classifies the problem under Non-Polynomial class (*NP* class).

This naive algorithm cannot be used for computing maximum area *k*-gon because of its huge complexity. Hence, it is difficult to find the optimum solution for larger values of *n* and *k* (even impossible for $n = 20$ and $k = 10$). A similar approach can be used to find the minimum area k-gon.

Table 1: Number of possible configurations for various values of n and k.

| Index | Value of | | ${}_nC_k$ | ${}_nC_k$*k! |
|---|---|---|---|---|
| | n | k | | |
| 1 | 2 | 1 | 2 | 2 |
| 2 | 5 | 2 | 10 | 20 |
| 3 | 10 | 5 | 252 | 302490 |
| 4 | 20 | 10 | 184756 | 6.70443E+11 |
| 5 | 40 | 20 | 1.38E+11 | 3.35E+29 |

## 3. Heuristic Algorithm

From the previous section we have realized that finding an optimum solution for area optimized *k*-gon is difficult. Hence, we have to formulate an algorithm that runs in polynomial time. In order to do that, we have to make a tradeoff with the area constraint.

In this section, we propose a heuristic algorithm for finding near optimal maximum and minimum area non-convex *k*-gons. The algorithm to compute maximum area k-gon can be divided into three parts, each depending on the value of *k* and the size of convex hull. Suppose, the size of convex hull is *CHS*, then the different cases are:

(i) $CHS < k$-1, (ii) $CHS = k$-1 and (iii) $CHS > k$-1.

***Observation 1:*** Convex hull results in the largest polygon that can be made from a point set.

***Observation 2:*** If vertex $v \notin$ Convex Hull, then adding $v$ to convex hull makes it non-convex.

According to Observation 1, if the size of convex hull is *k* then it is the maximum area "convex" *k*-gon. However, we are dealing with non-convex *k*-gons. Hence, we have to convert the convex polygon to non-convex. We use observation 2 to do this conversion. The entire algorithm is built over these two observations and the only change lies in using these observations for different cases.

**Case 1:** Size of convex hull *CHS* = *k*-1

Given a polygon of maximum area, the issues to be addressed are size and convex nature. Hence we add one more vertex to convex hull for making its size '*k*'. Owing to the property of convex hull, all other points will be inside convex hull. Hence by Observation 2, adding any point to convex hull makes the resulting polygon non-convex. Thus, satisfying both the conditions for making the resulting polygon a valid one. Our objective is to design an algorithm such that the resulting polygon has a near-optimal maximum area. This is done by adding an internal point which carves out a smaller region from a convex hull. In simple words, the triangle with a smaller area is removed from convex hull. The algorithm is described in Algorithm 2.

**Algorithm 2. Non-convex *k*-gon (Case 1)**

1: Procedure Create_*k*-gon (Point Set *P*, Integer *k*)
2: *CH*= Convex_Hull (*P*)
3: If size (*CH*) = = *k*-1, then
4: Remove the smallest area triangle $\Delta T_a T_b T_c$ such that $T_a, T_b \in CH$ and $T_c \notin CH$
5: End if
6: End procedure

**Case 2:** Size of convex hull *CHS* > *k*-1

In this case, we have to remove vertices from the hull to make its size *k*. Inorder to achieve our objective to maximize the area, it would be sensible to remove ears of the convex hull with smaller area. To make the polygon non-convex, we use the Observation 2. Adding an internal point to polygon of size '*k*' makes its size '*k*+1'. Hence, to address the issue, we remove ears until the size of the hull becomes '*k*-1'. Inorder to apply Observation 2, we need some internal points and there is a chance for all internal points to lie within the small ear, which will then be removed as part of the ear removal procedure. Figure 3, shows an example of a situation in which all the internal points lies within the smallest ear $\Delta abe$.

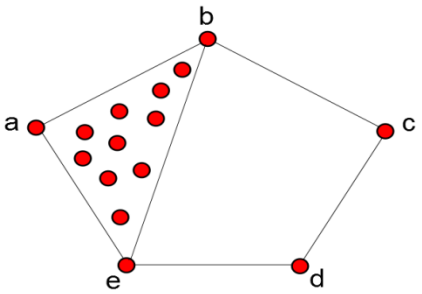


Figure 3: Minimum ear enclosing all points in a convex hull.

**Algorithm 3. Non-convex *k*-gon (Case 2)**

1: Procedure Create_ *k*-gon (Point Set *P*, Integer *k*)
2: $CH$ = Convex_Hull ($P$)
3: $H = CH$
4: If size $(CH) > k-1$ then
5: While size $(H) > k-1$
6: Remove the smallest area triangle $\Delta T_{i-1}T_iT_{i+1}$ from $H$ such that $H$ remains non-empty
7: End while
8: Removes the smallest area triangle $\Delta T_aT_bT_c$ such that $T_a, T_b \in H$ and $T_c$ lies inside $H$
9: End if
10: Return $H$
11: End procedure

Inorder to address this issue, we introduce one more constraint to our algorithm which always ensures the non-emptiness of resulting polygon after an ear removal procedure. But checking non-emptiness after removing each ear increases the overall complexity. In this situation, we adopt the method proposed by David et.al., 1992, in which the emptiness of each triangle can be determined in constant time, with a preprocessing complexity of O($n^2$). Therefore, after running ear removal procedure, internal points are added until the size of hull becomes *k-1*. The pseudocode is given in Algorithm 3.

**Case 3:** Size of convex hull *CHS* < *k*-1

In this case, we have to dig into convex hull for adding remaining (*k*-1-*CHS*) points to hull. But we are unsure about getting an optimal result, so we dig through the point which carves out the smallest area. To ensure the non-intersecting nature of resulting *k*-gon, we check whether an intersection occurs while adding an internal point. An example is illustrated in Figure 4, in which adding vertex *j* to edge *ac* of hull {*abcdefghi*} will result in an intersecting polygon. The pseudocode is given in Algorithm 4.

**Algorithm 4. Non-convex *k*-gon (Case 3)**

1: Procedure Create_ *k*-gon (Point Set *P*, Integer *k*)
2: *CH* = Convex_Hull (*P*)
3: *H* = *CH*
4: If size $(CH) < k-1$ then
5: While size $(H) < k$ - 1 do
6: Remove the smallest area triangle $\Delta T_a T_b T_c$ from *H* such that $T_a, T_b \in H$, $T_c \notin H$ remains simple after adding $T_c$
7: End while
8: End if
9: Return *H*
10: End procedure

The general idea to compute minimum area *k-gon* is as follows: Starting from the minimum area triangle, add external points until the size of polygon becomes *k*. Ensure the non-convexity during the first step by adding point to triangle in such a way that the resulting polygon becomes non-convex, such that all further polygons also becomes non-convex. The minimum area triangle can be found in O($n^3$) time. The pseudocode to obtain approximate minimum area *k-gon* is given in Algorithm 5.

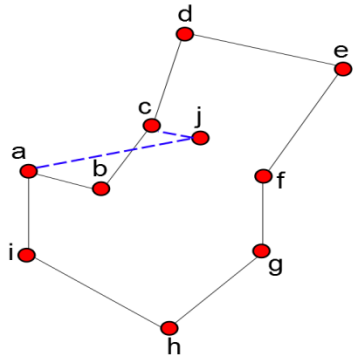


Figure 4: Illustration of an invalid triangle

**Algorithm 5. Heuristic for finding minimum area non-convex *k*-gon**

1: Procedure Approx_Min_ *k*-gon (Point Set *P*, Integer *k*)
2: Find minimum area triangle $T = \Delta abc$
3: Add_First(*T*)
4: H = *CH*
5: While size $T < k$ do
6: Continue_Adding(*T*)
7: End while
8: Return *T*
9: End procedure

In Algorithm 5, we use two functions namely *Add_First(T)* and *Continue_Adding(T)*: Where *Add_First(T)* function adds a point to *T* which makes *T* simple and non-convex. Since our aim is to find approximate minimum area *k*-gon, we add the point in such a way that after adding the new point, polygon *T* will be of minimum area. The idea is to find an approximate minimum area non-convex *4*-gon from the set of points. Function *Continue_Adding(T*) continues adding points to polygon *T* until its size becomes *k*. This is achieved by replacing the edge by two new edges in such a way that the newly added triangle will be of minimum area. The constraint of adding minimum area triangle helps to get approximate minimum area *k*-gon. The resulting polygon *T* will be approximately of minimum area, because we are expanding *T* by adding minimum area triangles.

## 4. Empirical Study

In this section, we discuss the complexity of the proposed algorithm, the results generated by the algorithm and finally the possible future works.

### 4.1 Complexity

All algorithms for finding maximum area non-convex *k*-gon from a point set of size *n* initially takes an O(*n* log*n*) time to compute the convex. Moreover, an additional complexity of any of the following case has to be added:

*Case 1:* The case takes O ($n^2$) time for selecting the smallest area triangle.

*Case 2:* The algorithm for this case takes O(*kn*) for slicing operation and takes O($n^2$) for selecting minimum area triangle. The O($n^2$) preprocessing mentioned in Eppstein et al., 1992, can be used for checking emptiness of triangles.

*Case 3:* The algorithm for this case takes O($n^2$) for selecting a single point to be added to previous hull and the same procedure has to be repeated for all points in worst case. Hence, it takes O($n^3$) time in the worst case.

Minimum area *k-gon* construction algorithm takes O($n^4$) in the worst case.

### 4.2 Results

The program is implemented in *C++* and the results are displayed using *OPENGL*. Figures 5 and 6 show the results of our algorithms on randomly generated input point set. Figure 5 shows the output of APPROX_*k*-GON() algorithm to find maximum area *k*-gon, where the convex hull size is 9. First row shows the output of Case 1 where $k < 10$. Second row shows the output of Case 2 where $k = 10$ and third row shows the output of Case 3 where $k > 10$. Figure 6 shows the output of APPROX_MIN_*k*-GON for different values of *k* on a randomly generated point set.

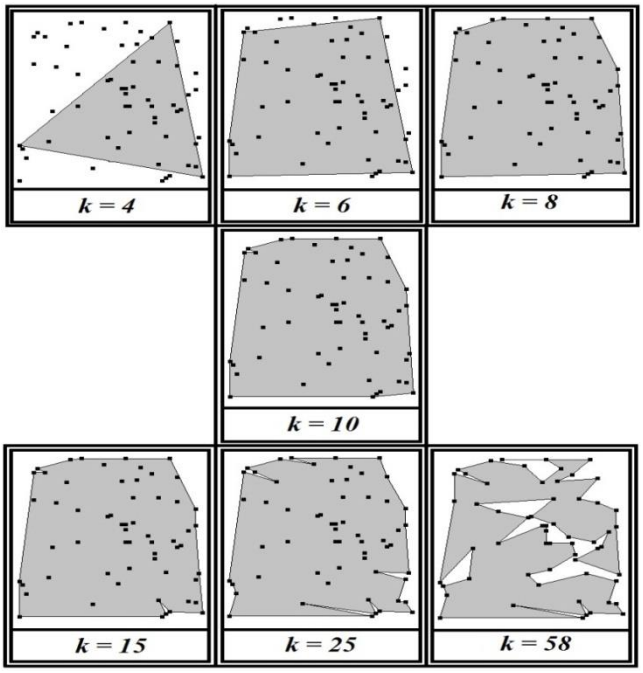


Figure 5: Output of APPROX_k-GON() algorithm for different values of $k$ on a randomly generated point set.

The output of APPROX_$k$-GON() is optimum for Case 2, as by Observation 1 convex hull is the largest polygon that can be constructed from a point set and we remove smallest triangle from convex hull. Hence it definitely results in a largest non-convex hull which can be obtained from the given point set.

## 4.3 Future Works

Possible directions for future works are:

1. To our knowledge, nobody has yet proved whether the problem is NP-Complete or not.
2. The approximation factors for the proposed algorithms are unknown.
3. The proposed algorithm have complexity of $O(n^3)$ and $O(n^4)$ which can be reduced.

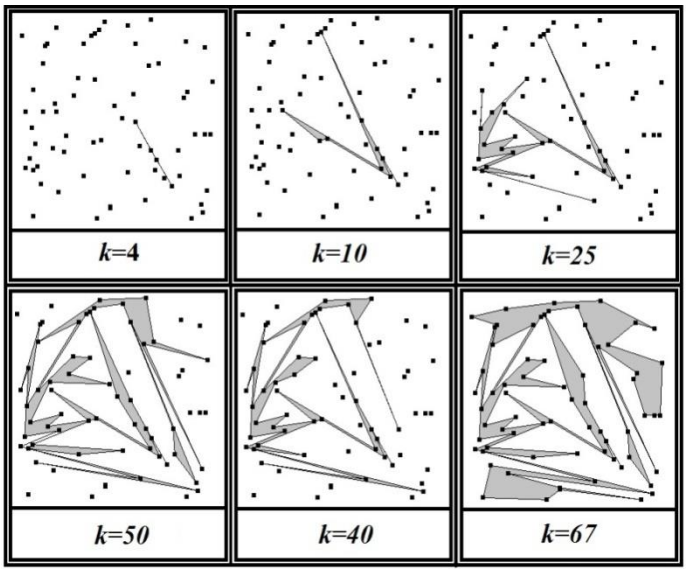


Figure 6: Output of APPROX_MIN_k-GON() algorithm for different values of *k* on a randomly generated point set.

## 5. Conclusion

In this paper, we infer that an optimal algorithm for finding maximum area *k*-gon is infeasible. Inorder to get a near optimal result, we introduced heuristic algorithms which approximate the optimal result. The results obtained using the proposed algorithm are promising. The complexity of the proposed algorithm is comparatively low and the computation is done in at most $O(n^3)$ time for maximum area *k*-gon and $O(n^4)$ for minimum area *k*-gon. Moreover, we infer that a near optimal solution for otherwise a computationally infeasible problem can be given.

## References

Eppstein, D., Overmars, G. R. and Woeginger, G.J. (1992).Finding minimum area k-gons.*Discrete and Computational Geometry*, **7**:45-58.

Erdos, P. (1975). On Some of Elementary and Combinatorial Geometry, *Annali di Mathematica puraedapplicata* (IV), Vol. CIII, pp. 99-108.

Erdos, P. and Szekeres, G.(1960)On some extremum problems in elementary geometry. *Ann. Univ. Sci. Budapest. Eotvos, Sect. Math.*,**3**(4):53-62.

Kalbeisch, J., Kalbeisch, J. and Stanton, R. (1970). A combinatorial problem on convex n-gons. In Proc. Louisiana Conference on Combinatorics, *Graph Theory and Computing*, pages 180-188, Louisiana State University.

O'Rourke, J. (1998). *Computational Geometry in C.*2nd edition.New York, NY, USA: Cambridge University Press.

Szekeres,G. and Peters, L. (2006) Computer solution to the 17- point Erdos-Szekeres problem. *The ANZIAM Journal*, **48**(2):151-164.

Toth,G. and Valtr, P. (2005) The Erdos-Szekeres theorem: upper bounds and related results. *Combinatorial and Computational Geometry*, J.E. Goodman, J. Pach, and E.Welzl (Eds.), 52:557-568.