Sets without empty convex 6-gons

Mark Overmars, Bertha Scholten and Ingrid Vincent

RUU-CS-88-12
March 1988

Rijksuniversiteit Utrecht

Vakgroep informatica

Padualaan 14 3584 CH Utrecht
Corr. adres: Postbus 80.089, 3508 TB Utrecht
Telefoon 030-531454
The Netherlands

Sets without empty convex 6-gons

Mark Overmars, Bertha Scholten and Ingrid Vincent

Technical Report RUU-CS-88-12
March 1988

Department of Computer Science
University of Utrecht
P.O.Box 80.089
3508 TB Utrecht
the Netherlands

Sets without empty convex 6-gons

Mark Overmars, Bertha Scholten and Ingrid Vincent

March 1988

1 Introduction

Given a set V of n points in the plane, no three of which are collinear, we consider subsets $\Phi_k(V)$ of V of cardinality k that are convex, i.e., they form the vertices of some convex k-gon, and there is no point of V in the interior of this polygon (the polygon is empty). Let F_k denote the smallest value such that any set V of cardinality at least F_k contains some subset $\Phi_k(V)$. (Note that, when all sets of size n contain an empty k-gon then any set of size larger than n contains an empty k-gon. Hence, we could also define F_k to be the smallest value such that any set of size F_k contains some Φ_k.) Erdös[3] proposed the study of finding bounds on F_k. It is trivial to prove that $F_1 = 1$, $F_2 = 2$, $F_3 = 3$ and $F_4 = 5$. The following results are known:

Theorem 1.1 *(Harborth[4])* $F_5 = 10$.

Theorem 1.2 *(Horton[5])* $F_7 = \infty$. *(In other words, for each n there exists a set of n points without empty convex 7-gon.)*

Clearly this result also holds for $k > 7$. For $k = 6$ no exact bounds on F_k are known. In [1] Avis and Rappaport give a method to determine whether a given set of points does contain an empty convex 6-gon. Using this method they succeeded in finding a set of 20 points that does not contain an empty convex 6-gon, showing that $F_6 \geq 21$. The following table summerizes the known results.

k	F_k
1	1
2	2
3	3
4	5
5	10
6	≥ 21
≥ 7	∞

In this paper we describe a new algorithm, based on a method by Dobkin, Edelsbrunner and Overmars[2] that is much faster than the technique described in [1]. Using this algorithm, implemented on a simple micro computer, we were able to perform many more tests. Moreover, a user interface was designed that allowed the user to collaborate with the program in testing promizing position or areas for points. Also some backtracking techniques were used. This finally resulted in a set of 26 points that does not contain an empty convex 6-gon. This shows that $F_6 \geq 27$.

The paper is organized as follows. In section 2 we describe the algorithm used. In section 3 we compare our method with the one presented in [1]. Finally, in section 4 we present the set of points found.

2 The algorithm

After some experiments with random sets, it was clear that it is impossible (well, almost) to improve F_6 by testing large collections of random points. In fact, we did not even result in finding a set of 17 points this way. (The same observation was made in [1].) We decided to use an incremental approach instead. Starting with a set of n points without an empty convex 6-gon, try to find a new point that does not introduce an empty convex 6-gon. (That this is a reasonable approach also follows from the fact that any set of $n+1$ points without an empty convex 6-gon can be obtained in this way. This follows from the fact that removing any point of the convex hull of the set does not introduce a new empty convex 6-gon.) So our basic problem is the following:

Problem: Given a set V of n points without an empty convex 6-gon, and a point p, test whether $V \cup \{p\}$ contains an empty convex 6-gon.

To be able to test many different points an efficient algorithm (and, in particular, efficiently implementable) for this problem was required. To this end an algorithm by Dobkin, Edelsbrunner and Overmars[2] for finding empty polygons was adapted to our incremental approach.

2.1 Reducing the problem

It is obvious that when the new set $V' = V \cup \{p\}$ contains an empty convex 6-gon, p must be one of the vertices. Now sort all points of V by angle around p. Let the counter-clockwise sorted set be $p_1, ..., p_n$. When p lies on the convex hull of V' we construct one polygon $P^1 = pp_1p_2...p_n$. When p does not lie on the convex hull of V' we construct two polygons. First the polygon P^1. Secondly, let r be a ray from p that runs between p_n and p_1. Consider the ray r' that runs from p in the opposite direction. It will run between two points p_i and p_{i+1}. Now let $P^2 = pp_{i+1}...p_np_1...p_i$. See figure 1 for the two polygons we get.

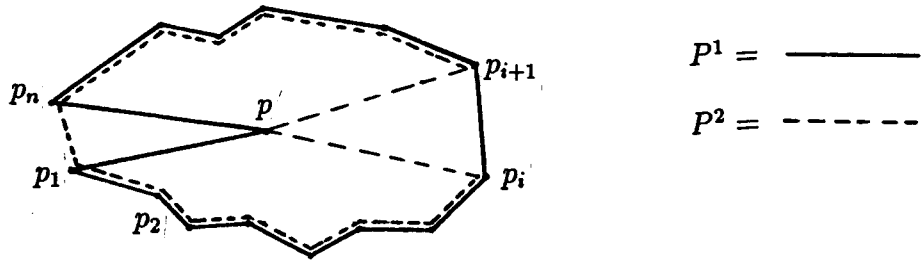


Figure 1: The two polygons P^1 and P^2.

Lemma 2.1 *When V' contains an empty convex 6-gon, it lies either inside P^1 or inside P^2.*

Proof. Because p will be on the 6-gon, it will have the form $pp_{i_1}...p_{i_6}$ where $p_{i_1}, ..., p_{i_6}$ appear in this order in the cyclic sorted list of vertices.

Because p lies in the kernel of both polygons, it is easy to see that the only (parts of) edges that could lie inside the 6-gon can be the outgoing edges of p. Now the 6-gon can never intersect both rays r and r'. Hence, it can never intersect edges in both P^1 and P^2. □

So we are left with (at most) two instances of the following problem:

Problem: Given a polygon P of $n+1$ vertices with one vertex p in the kernel, determine whether an empty convex 6-gon exist including p.

Let VG be the graph consisting of the vertices $p_1, ..., p_n$ as nodes and an edge between p_i and p_j if this edge lies completely inside (or on the boundary of) P. This graph is called the *visibility graph* of P (not including the edges starting at p). Note that any edge of an empty 6-gon, except for the outgoing edges from p is an edge of VG. See figure 2 for an example of the visibility graph.

Lemma 2.2 *$pp_{i_1}...p_{i_6}$ is an empty convex 6-gon if and only if $p_{i_1}...p_{i_6}$ is a convex chain in VG and the angle $p_{i_6}pp_{i_1}$ is convex.*

Proof. Follows trivially. □

This leads to the following algorithm.

1. **Sort the points in V by angle around p.**

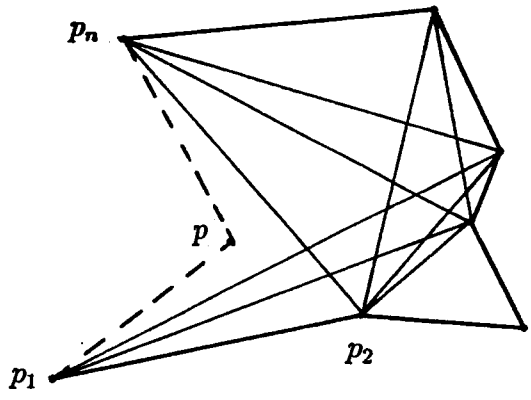


Figure 2: The visibility graph of a starshaped polygon.

2. Form P^1 and P^2.

3. Compute the visibility graph VG of P^1.

4. Check whether VG contains a chain of length 4 (edges) that forms an empty convex 6-gon with p.

5. If neccessary repeat step 3 and 4 with P^2.

2.2 Constructing the visibility graph

The method for computing the visibility graph in a star-shaped polygon is taken from [2]. The only difference is that in our case p not neccessarily lies on the convex hull. Hence, we have to add a test to check whether an edge does not intersect one of the outgoing edges of p.

Let the polygon be given by its points $p, p_1, p_2, ..., p_n$. We construct the visibility graph VG during one counter-clockwise scan around the polygon starting at p_1 and ending at p_n. When we visit p_i we construct all incoming edges of p_i. With each vertex p_i we maintain a queue Q_i that stores the starting points of some of the incoming edges of p_i in counter-clockwise order. It contains those points p_j such that $\overline{ji}$ is an edge of the visibility graph and we have not yet reached another point p_k with $k > i$ such that $\overline{jk}$ is an edge of the visibility graph. Hence, Q_i is a kind of waiting list. It contains those points that could be seen by p_i but could not be seen since, because p_i blocks their view. The required operations that can be performed in constant time are the following:

1. ADD($\overline{ij}$): it creates an edge from i to j. This edge will be stored at both p_i and p_j for use later.

2. TURN($\overline{ij}$,$\overline{jk}$): it returns *left* or *right* depending on whether p_k lies to the left or to the right of $\overline{ij}$. (Note that it cannot lie on the line.)

3. FRONT(Q): it returns the index of the first point in queue Q.

4. DEQUEUE(Q): it removes the first point from queue Q.

5. ENQUEUE(k,Q): it adds the point p_k to queue Q.

The algorithm now looks as follows:

```
PROCEDURE VISIBILITY;
    FOR i := 1 TO N − 1 DO Q_i := ∅;
    FOR i := 1 TO N − 2 DO PROCEED(i,i + 1)

PROCEDURE PROCEED(i,j);
    IF TURN(ip→,pj→)=right THEN
        WHILE Q_i ≠ ∅ AND TURN(FRONT(Q_i)i→,ij→)=left DO
            PROCEED(FRONT(Q_i),j);
            DEQUEUE(Q_i);
        ADD(ij→);
        ENQUEUE(i,Q_j)
```

PROCEED adds an edge from point p_i to p_j. It first checks whether p lies on the correct side of the edge. If so $\overrightarrow{ij}$ can indeed be added. It also checks whether any of the points in the waiting queue of p_i are visible from p_j and, if so, recursively calls PROCEED. Because the points in the queue are sorted counter-clockwise only a first portion of the queue needs to be checked.

It is easy to see that the method runs in time O(size of VG).

2.3 Determining convex chains

The next step is to determine whether a convex chain exists inside the visibility graph of length 4. To this end we again walk counter-clockwise along the vertices of the polygon. For each edge we will determine the "best" convex chain of length 3 and 2 that ends on this edge. With "best" we mean: that starts the latest. This means that the chance that p forms a convex angle with it is the greatest. For an edge e we use $C3_e$ to be the starting point of the best chain of length 3 and $C2_e$ to be the starting point of the best chain of length 2. For consistency we define $C1_e$ to be the starting point of edge e but we do not actually have to store it. When we are at a point p_x we assume that for all incoming edges the $C3$ and $C2$ values are known (if they exist).

Note that both the incoming edges and outgoing edges are sorted by angle (that is the way the visibility graph algorithm produces them). We are now going to look which outgoing edges form a convex angle with what incoming edges. We start with the first outgoing edge e, which can be connected to the smallest number of incoming

edges, and determine all incoming edges with which it forms a convex angle. Of these we determine the best $C3$, $C2$ and $C1$ values. If $C3$ exists and together with e and the point p forms an empty convex 6-gon we are done. Otherwise, we set $C3_e$ to $C2$ and $C2_e$ to $C1$. We then continue with the next outgoing edge. We do not have to recheck all the incoming edges cause we already know the best values. We only have to check new incoming edges that form a convex angle with it. The algorithm looks as follows:

PROCEDURE FINDCHAIN;
  FOR $z := 1$ TO N DO TREAT(p_z)

PROCEDURE TREAT(p_z);
  Let $i_1, ..., i_{imax}$ be the incoming edges of p_z.
  Let $o_1, ..., o_{omax}$ be the outgoing edges of p_z.
  Both ordered counter-clockwise.
  $C3 := -1; C2 := -1; C1 := -1;$
  $l := 1;$
  FOR $j := 1$ TO $omax$ DO
    WHILE $l \leq imax$ AND TURN(i_l,o_j)=*left* DO
      IF $C3_{i_l} > C3$ THEN $C3 := C3_{i_l}$;
      IF $C2_{i_l} > C2$ THEN $C2 := C2_{i_l}$;
      IF $C1_{i_l} > C1$ THEN $C1 := C1_{i_l}$;
      $l := l + 1$
    IF $C3 \neq -1$ AND TURN($\overline{C3p}, \overline{po_j}$)=*right* THEN
      Empty 6-gon found;
    $C3_{o_j} := C2$;
    $C2_{o_j} := C1$;
    $C1_{o_j} := z$;

Correctness can easily be shown. As each incoming and each outgoing edges is checked only once, the amount of time is bounded by O(size of VG).

2.4 Puting it together

Theorem 2.3 *Given a set V of n points without an empty convex 6-gon and a point p, it can be determined in time $O(n^2)$ whether $V \cup \{p\}$ contains an empty convex 6-gon.*

Proof. The sorting takes time $O(n \log n)$. The two other steps take time O(size of VG). Clearly, the size of VG is bounded by $O(n^2)$. □

In practise the routine works a lot faster (see section 3). The reason is that the visibility graph tends to have a size much smaller than n^2. This raises an interesting

question: Assuming that the set does not have an empty convex 6-gon, how large can the visibility graph be?

In the implementation of the method, the construction of the visibility graph and the determining of convex chains are mixed. So for every edge of the visibility graph that is created, immediately the chain information is computed. As a result, the program can stop when it finds a chain of length 4 without constructing the rest of the visibility graph. On the average this will be about half-way so the running time is almost halved.

3 The experiments

The above algorithm was implemented using MODULA-2 on an ATARI-ST micro computer (68000 based). The routine did run very fast. For example, 1000 tests for adding a point to a set of 20 points took an average of 88 seconds. Compare this with the method of [1] that, running on a VAX-750, required 1300 seconds for the same task.

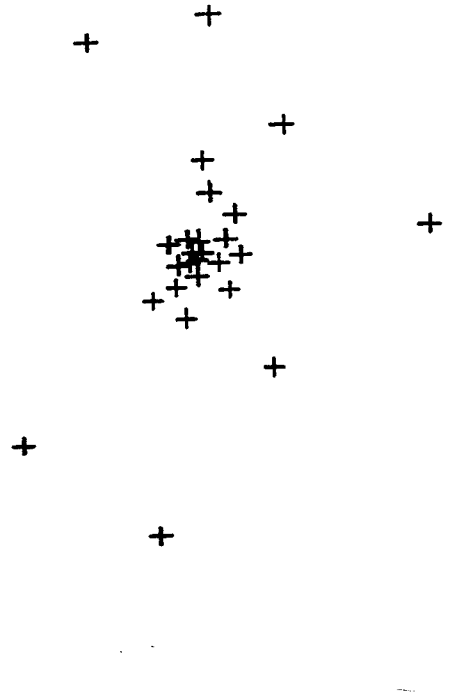
The following table shows the average amount of time required for 1000 tests for (unsuccessfully) adding a point to a set V of size n.

n	time in sec. (1000 tests)
11	57
13	69
16	76
18	80
20	88
22	95
24	100
26	106

From these experiments one can conclude that the running time of the algoritm tends to behave almost linear on the average. Compare this with the worst-case of $\Omega(n^2)$. Of course, it is impossible that the method runs in linear time because of the sorting involved. But the constants in the sorting are very small compared to the other steps of the algorithm.

4 The result

Starting with one point and randomly adding points in a slowly growing region we soon found sets of 21 and 22 points without empty convex 6-gons. To obtain larger sets we used a kind of random backtrack method that randomly removed some points from a large set without creating empty 6-gons and tried to add, preferably more, new points. Also the user was involved who could add points, delete points, indicate



(1181, 727)
(1327, 921)
(1138, 1011)
(1870, 946)
(1339, 1359)
(1365, 482)
(1420, 707)
(2025, 410)
(771, 1066)
(1090, 457)
(731, 1594)
(1054, 2285)
(2284, -117)
(2102, 1640)
(573, 560)
(1534,-1320)
(2413, 793)
(7148, 195)
(1701, -603)
(3216, 3448)
(139, 1853)
(3521,-2068)
(-3176, 5105)
(-1347,-4021)
(1767,-4587)
(207, 7135)

Figure 3: A set of 26 points without empty convex 6-gons.

where to search for new points, etc. This was done using a graphical interface. This gave as some insight in the "form" of "good" sets. After about two weeks of (micro) computer time the method resulted in a set of 26 points without empty convex 6-gons, proving the following theorem:

Theorem 4.1 $F_6 \geq 27$.

Proof. See figure 3 for a set of 26 points that does not contain any empty convex 6-gon. □

During the months after that the program never found a larger set, although it did find many more sets of 26 points. We will continue running the program to try and obtain larger sets of points. Who is interested in doing the same can obtain a copy of the program by writing to the authors. (You will need an ATARI-ST with a monochrome monitor to run it.)

References

[1] Avis, D. and D. Rappaport, Computing the largest empty convex subset of a set of points, *Proc. first ACM Symp. on Computational Geometry,* Baltimore, 1985, 161-167.

[2] Dobkin, D.P., H. Edelsbrunner and M.H. Overmars, Searching for empty convex polygons, *Proc. 4th ACM Symp. on Computational Geometry,* Urbana-Champaign, 1988, 224-228.

[3] Erdös, P., Combinatorial problems in geometry and number theory, Proc. Symp. Pure Math. 34 (1979), 149-162.

[4] Harborth, H., Konvex Funfecke in ebene Punkmengen, *Elem. Math.* 33 (1978) 116-118.

[5] Horton, J.D., Sets with no empty convex 7-gons, *C. Math. Bull.* 26 (1983) 482-484.