An Algorithm for the MaxMin Area Triangulation of a Convex Polygon

J. Mark Keil, Tzvetalin S. Vassilev
Department of Computer Science
University of Saskatchewan, 57 Campus Drive,
Saskatoon, Saskatchewan, S7N 5A9
e-mail: keil@cs.usask.ca, tsv552@mail.usask.ca

Abstract

Given a convex polygon in the plane, we are interested in triangulations of its interior, i.e. maximal sets of non-intersecting diagonals that subdivide the interior of the polygon into triangles. The MaxMin area triangulation is the triangulation of the polygon that maximizes the area of the smallest area triangle in the triangulation. There exists a dynamic programming algorithm that computes the optimal triangulation with respect to a number of optimality criteria in $\Theta(n^3)$ time and $\Theta(n^2)$ space, [4]. We present an algorithm that constructs the MaxMin area triangulation of a convex polygon in $O(n^2 \log n \log \log n)$ time and $O(n^2)$ space. The algorithm is based on the dynamic programming approach and uses a number of problem-specific geometric properties that are established within the paper.

1 Introduction

Triangulations of point sets in the plane have been studied as one of the important structures in computational geometry. There are optimality criteria based on edge length, angles, areas and other elements of the individual triangles in a triangulation, for an overview see [1]. Usually, in connection with these criteria, we consider MinMax and MaxMin problems. The first quantifier defines an optimization that is done over all possible triangulations of the given point set and the second quantifier specifies the optimization that is done within the respective elements (edges, angles, triangles) of a particular triangulation. For example, MinMax angle stands for the triangulation that minimizes the maximum angle in a triangulation over all possible triangulations of the given point set.

If the point set is a convex polygon, there is a dynamic programming algorithm by Klincsek, described in [4], that finds the optimal triangulation with respect to a large number of criteria. The algorithm runs in $\Theta(n^3)$ time and requires $\Theta(n^2)$ space. Many of the optimal triangulations, though, admit better problem-specific algorithms. The Greedy and the Delaunay triangulations are computable in linear time and space for convex polygons. Some other optimal triangulations can be computed within time and space bounds that are better than those of the general algorithm, for example by edge insertion [2].

We study the problems of optimizing the area of the triangles in the triangulation, and particularly the MaxMin area triangulation of a convex polygon. In the following section we outline the approach used to compute the optimal triangulation solving the so-called threshold problem. We discuss the relationship between the optimization and the threshold problems. In section 3 we provide the geometric background for the algorithm, properties and the structure of the triangulation. In section 4 we present the main result of the paper the algorithm for computing the MaxMin area triangulation of a convex polygon, and prove the $O(n^2 \log n \log \log n)$ time and $O(n^2)$ space bounds. Section 5 concludes the paper with discussion on the extensions of this algorithmic approach to other optimal triangulations, directions for future work and open problems.

2 Optimization and threshold problems

We can consider two types of problems with respect to optimal triangulations.

Definition 1. (Optimization problem): *Given a planar set of points S, and an optimality criterion represented by the quality measure μ, the* **optimization problem** *is:*
"Find the triangulation(s) T of S that optimize(s) the value of μ over all possible triangulations of S."

Definition 2. (Threshold problems): *Given a planar set of points S, an optimality criterion represented by the quality measure μ, and a threshold τ, the* **threshold problems** *are:*
"Is there a triangulation T of S such that $\mu(T) \geq \tau$ (or $\mu(T) \leq \tau$) ?" (Yes/No problem)
"Find a triangulation T of S such that $\mu(T) \geq \tau$." (Construction problem).

3 Geometric properties of the MaxMin area triangulation

The following is a well-known result from elementary geometry [3].

Property 1. *Given a triangle $\triangle DEF$ in the plane and a triangle $\triangle PQR$ inscribed in it so that $P \in EF$, $Q \in FD$, $R \in DE$ if we consider the area of the triangles, denoted by A, then:*
$A_{\triangle PQR} \geq \min(A_{\triangle DQR}, A_{\triangle ERP}, A_{\triangle FPQ})$

In other words, if we inscribe a triangle inside another triangle, the inscribed triangle is not the smallest in terms of area. Using this property we will establish an interesting fact about the "worst" triangle in the MaxMin area triangulation of a convex polygon.

Lemma 1. *Given a convex polygon P in the plane and a triangulation T of P, the triangle in T that has smallest area, has at least one edge on the boundary of P.*

Proof. (Sketch) We consider the worst (smallest area) triangle in T, assume that no two vertices of it are adjacent in P. Then we consider the three bordering triangles of the worst triangle in T. We use Property 1 to derive a contradiction by enlarging the three bordering triangles so that they form a triangle in which the worst triangle is inscribed. □

Let us denote the vertices of the polygon P by $v_1, v_2, \ldots, v_n$. Further, we shall assume, for the rest of this paper that $v_{i+kn} = v_i$, that is the vertices of the polygon are enumerated modulo n and that the order of the vertices from v_1 to v_n is their clockwise order.

Here we recall another classical result about convex polygons.

Property 2. (Distance unimodality [5]): *The distance between the line through an edge of a convex polygon and the vertices of the polygon in clockwise (or counterclockwise) order along its boundary is unimodal.*

The above property implies that the area of the triangles with a given base edge in a convex polygon is also unimodal. Another way of looking at this property (the unimodality of area) is by defining the threshold lines. For each value τ of the threshold there is a line parallel to the edge and such that the points on that line form triangles with area equal to τ. We will concentrate only on the threshold line that lies on the same side of the edge as the polygon itself. Then, another way of describing the unimodality with relation to the threshold lines is to say that if the function is unimodal, then its threshold curve cuts out of the polygon a contiguous piece (or does not intersect the polygon at all). If the function is not unimodal, the pieces that lie inside or outside the threshold curve may alternate more than once as we go along the boundary of the polygon in a chosen direction.

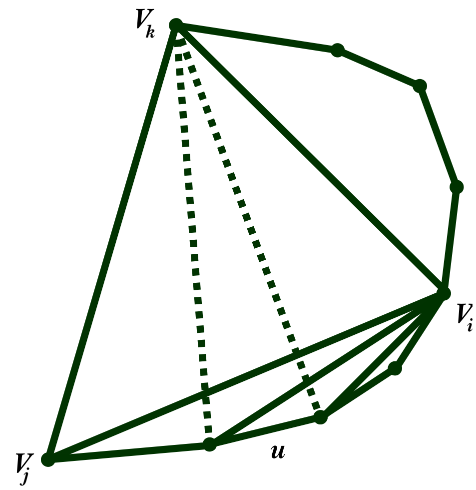

Figure 1: Retriangulation in a 2-zone polygon.

Definition 3. *Given convex polygon P, for an edge v_iv_j we will denote by $Top(v_iv_j)$ the vertex of P that is farthest from the line through the edge v_iv_j along the boundary of P. If there are two such vertices, we use the leftmost one (preceding the other in the clockwise order from v_i to v_j) as $Top(v_iv_j)$.*

The value of the function Top for all the diagonals in P can be computed in $O(n^2)$ time by rotating calipers. This approach is due to Toussaint in [6].

Definition 4. (Zonality): *Given a convex polygon P in the plane and a clockwise ordering of its vertices, consider the subpolygon P_{ij}, containing vertices v_i through v_j: P_{ij} is a k-zone subpolygon, $k \in \{0, 1, 2, 3, 4\}$ if and only if $k = \left\lceil \frac{2(\angle v_{i+1}v_iv_j + \angle v_{j-1}v_jv_i)}{\pi} \right\rceil$.*

Definition 5. (Complementary subpolygons): *We call the subpolygons P_{ij} and P_{ji}* **complementary**. *The union of P_{ij} and P_{ji} is P.*

Definition 6. (Zonality function): *Let $z(P_{ij})$ be a function defined over the subpolygons of the convex polygon P in the plane and having its values in the set $\{0, 1, 2, 3, 4\}$, such that: $z(P_{ii}) = 0$, $Z(P_{ij}) = \alpha$ iff P_{ij} is α-zone subpolygon.*

Property 3. $z(P_{ij}) + z(P_{ji}) \leq 5$.

Property 4. *Let v_i, v_j and v_k be three vertices of P in the same clockwise order. Then $z(P_{ij}) + z(P_{jk}) + z(P_{ki}) \leq 6$.*

Proof. (Sketch) The proofs of Properties 3 and 4 involve case analysis and use of basic geometric facts such as sum of the angles of a polygon, sum of two angles at the intersection of two lines, etc. □

We now study area triangulations in 2-zone subpolygons.

Lemma 2. (2-zone polygons): *Let P_{ij} be a 2-zone polygon. Let μ represent the minimum area of a triangle in a triangulation. Given a threshold τ, if there exists a triangulation T of P_{ij} such that $\mu(T) \geq \tau$, then there exists a triangulation T' of P_{ij} such that $\mu(T') \geq \tau$, and the triangulation T' contains one of the triangles $\triangle v_i v_{i+1} v_j$ or $\triangle v_i v_{j-1} v_j$.*

Proof. The triangulation T contains a triangle $\triangle v_i v_k v_j$ for some $i+1 \leq k \leq j-1$, as shown in Figure 1. If $k = i+1$ or $k = j-1$ we are done. Otherwise, the vertex v_k will be either between $Top(v_i v_j)$ and v_j or between v_i and $Top(v_i v_j)$. Assume that v_k is between $Top(v_i v_j)$ and v_j. We can show that for the vertices of the chain between $Top(v_i v_j)$ and v_j, the farthest vertex of P_{ij} is v_i.

Now, we will construct T' from T by connecting all vertices $v_{k+1} \dots v_{j-1}$ to v_i. The part of T inside the subpolygon P_{ik} remains unchanged in T'. □

Note that the proof of Lemma 2 automatically implies that the same is true for all 1-zone subpolygons.

Lemma 3. *In every triangulation T of a convex polygon P, there exists a triangle $\triangle v_i v_j v_k$, such that $z(P_{ij}) \leq 2$, $z(P_{jk}) \leq 2$, $z(P_{ki}) \leq 2$.*

Proof. (Sketch) We start at an ear triangle and traverse the triangulation until we hit an edge that divides the polygon into a 2- and 3-zone subpolygons. We then continue into the 3-zone subpolygon. □

Definition 7. *Let P_{ji} be a subpolygon of P such that $z(P_{ji}) \leq 2$. We will denote by $MaxCW(v_i)$ the last (in clockwise order from v_i) vertex of P such that $z(P_{i,MaxCW(v_i)}) \leq 2$. In other words, in the series of the subpolygons $P_{i,i+1}, P_{i,i+2}, \dots, P_{i,MaxCW(v_i)}$ all the subpolygons have zonality of 2 or less and $z(P_{i,MaxCW(v_i)+1}) \geq 3$. Analogously, we will define $MaxCCW(v_i)$ to be the last vertex in counterclockwise order from v_i, such that $z(P_{MaxCCW(v_i),i}) \leq 2$.*

Property 5. *$MaxCW(v_i) = Top(v_i v_{i+1})$ or $Top(v_i v_{i+1}) + 1$ and $MaxCCW(v_i) = Top(v_{i-1} v_i)$.*

Proof. Obvious, given the definitions of Top, $MaxCW$ and $MaxCCW$. □

Lemma 4. (Intervals of admissibility): *Let P_{ji} be a subpolygon of P such that $z(P_{ji}) \leq 2$. Consider the interval of the vertices of P between $MaxCCW(v_j)$ and $Top(v_{MaxCCW(v_j)} v_j)$. Given a quality measure μ representing area and a threshold τ, if there exist two vertices k_1 and k_2 in the interval such that there exist triangulations T_1 of $P_{k_1 j}$ and T_2 of $P_{k_2 j}$, respectively such that $\mu(T_1) \geq \tau$ and $\mu(T_2) \geq \tau$, then for each vertex k in the interval between k_1 and k_2 there exists a triangulation T of P_{kj} such that $\mu(T) \geq \tau$.*

Proof. The idea is similar to that of Lemma 2. We will show how to obtain T from T_1 and T_2. Please, refer to Figure 2. The subpolygon between $MaxCCW(v_j)$ and v_j is 2-zone by definition. This means that for all the vertices between $MaxCCW(v_j)$ and $Top(v_{MaxCCW(v_j)} v_j)$, the vertex v_j is the farthest vertex. Then, we can obtain T by adding to T_2 a fan from v_j to the vertices in the chain between k and k_2. Because of the existence of T_1, we know that all these edges (in the chain between k and k_2) can be connected in a way that the threshold condition is satisfied, in T we connect them (possibly) farther, but not closer to the vertex that they were connected in T_1. □

Thus, we have a solid basis for checking all possible triangles satisfying the premises of Lemma 3.

4 The algorithm

Algorithm 1 (MaxMin Area Triangulation):
Input: Convex polygon P represented by the list of its n vertices in sorted clockwise order.
Output: A triangulation T of P such that the minimum area triangle in T has the maximum possible area over all triangulations of P.

(1) Compute areas of all the triangles that have a boundary edge of P as its edge.

(2) Store the values in an array $Triangles[\ ]$. Sort $Triangles[\ ]$ in ascending order.

(3) Compute for each of the $O(n^2)$ edges and diagonals of P, the most distant vertex of P and the zonality of the subpolygon(s) induced by this edge/diagonal, store the results in the arrays $Top[\ ]$ and $Z[\ ]$. Compute the arrays $MaxCW[\ ]$ and $MaxCCW[\ ]$ from $Top[\ ]$.

(4) Find the largest value of τ in $Triangles[\ ]$ such that $\exists T : \mu(T) \geq \tau$ by binary search in $Triangles[\ ]$.

(5) Construct the triangulation T corresponding to the optimal τ found in step (4).

Function $M(\tau)$ returns *true* if there is a triangulation T of P such that $\mu(T) \geq \tau$ and *false* otherwise.

$M(\tau)$ uses an $O(n^2)$ array $Table[\ ]$. In the entry $Table[i,j]$ we store the index k of the vertex v_k that is connected to the edge $v_i v_j$ in a triangulation (if such triangulation exists) of 2-zone P_{ij} that satisfies the threshold, or zero otherwise. The algorithm will set to zero all entries in $Table[\ ]$ that correspond to subproblems of zonality three or higher. The information in $Table[\ ]$ is duplicated in two arrays of priority queues R and C. The array R encodes the rows of $Table[\ ]$ and the array C encodes the columns of $Table[\ ]$. For example $R[i]$ represents the i-th row of $Table[\ ]$. All nonzero entries in the i-th row of $Table[\ ]$ are elements of the priority

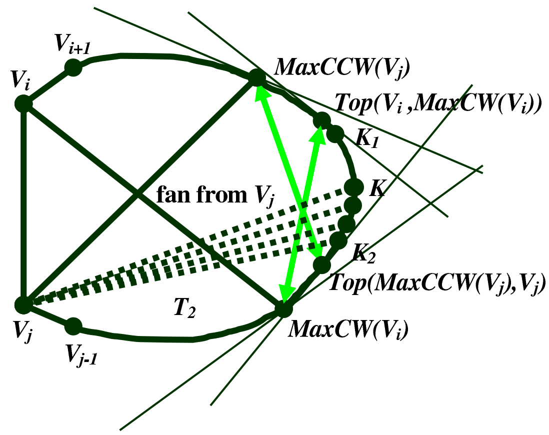

Figure 2: Lemma 4, intervals and retriangulation.

queue $R[i]$, the zero entries are not represented. To be more clear, in $R[i]$ we insert all indices j that correspond to an element $Table[i, j] > 0$. Similarly, in the priority queue $C[j]$ we insert all indices i that correspond to an element $Table[i, j] > 0$. Again, the zero entries in the j-th column of $Table[\,]$ are not represented in $C[j]$.

(a) Initialize $Table[\,]$ with zeros and R and C with empty queues.

(b) For every possible pair (i, j), if $z(P_{ij}) \leq 2$, check whether an admissible triangulation exists and if so – update $Table[i, j]$, $R[i]$ and $C[j]$ accordingly. Use the method of Lemma 2.

(c) Check for each entry in the $Table[\,]$, whether both $Table[i, j]$ and $Table[j, i]$ are non-zero, if so – return *true*.

(d) Check for every non-zero entry $Table[j, i]$, whether there exists $k \in (i, j)$ such that $Table[i, k]$ and $Table[k, j]$ are non-zero and $A_{\triangle v_i v_k v_j} \geq \tau$, if so – return *true*. Lemma 3 allows us to search only for a k that yields 2-zone subproblems and Lemma 4 allows us to do this efficiently.

(e) Return *false*.

Algorithm and data structures explanation

Each individual entry in the arrays R and C is a van Emde-Boas priority queue. These priority queues are central part of the efficiency of the algorithm. Van Emde-Boas priority queues [8, 7] are data structures that operate over a universe of keys (usually integers) of size N in a way that the operations insertion, deletion, membership testing, finding predecessor and successor are performed in $O(\log \log N)$ time. The size of the priority queue is $O(N)$. In our algorithm the universe of keys is the set of indices of the vertices of the polygon P, i.e. $[1 \ldots n]$. Thus, the size of each individual priority queue is $O(n)$ and the overall size of the arrays R and C is $O(n^2)$.

The algorithm performs several preprocessing tasks. First in (1) it computes the areas of all triangles that could eventually be the worst. This takes $O(n^2)$ time since there are $O(n^2)$ such. Then, in (2), the array $Triangles[\,]$ is sorted, which takes $O(n^2 \log n)$ time. In (3), we compute the value of $Top[\,]$ for each of $O(n^2)$ edges. This can be done in $O(n^2)$ time. The idea again is to use the rotating calipers [6]. But here, instead of considering edges on the boundary of P one after another and move the calipers accordingly, we consider the fan of edges that are incident to one particular vertex and move the calipers to compute the $Top[\,]$ of the edges in this fan. This is done in $O(n)$ time since the calipers do not make more than one full rotation around the boundary of P, and we compute $Top[\,]$ for the $O(n)$ edges in the fan in this pass. We have n vertices, hence n such passes and the task is completed in $O(n^2)$ time. The binary search in step (4) will take time of $O(\log n) \times time(M(\tau))$. Finally, the triangulation can be recovered in step (5) from the data in $Table[\,]$ in linear time, $O(n)$, because it has linear complexity and each time we look up in $Table[\,]$ we add at least one (and at most two) new edge to the triangulation.

Now, we have to analyze the timing of the computation of the function $M(\tau)$. Initializations of $Table[\,]$ and the arrays R and C in (a) is done in $O(n^2)$ time. After that, in (b), we have two nested loops that compute the values of $Table[\,]$ for all subproblems of zonality two or less. There are $O(n^2)$ iterations through this program segment. In every iteration we perform a constant number of checks and logical operations and at most two insertions in the priority queues, if the subproblem has an admissible triangulation. Therefore, the total time for the execution of the two nested loops is $O(n^2 \log \log n)$. If the subproblem is a triangle, we only need to check whether the area of this triangle is larger than the threshold value τ. If so, we reflect this in the queues R and C and in the $Table[\,]$. If the subproblem is larger in size than a triangle but has zonality of two or less, Lemma 2 imples that we only need to check the two triangles that are immediately adjacent to the base edge, if any of them is larger in area than the threshold value τ and the rest of the subpolygon has admissible triangulation, we have to record the admissible triangulation of the subproblem in the queues R and C and in the $Table[\,]$. When the algorithm has gone through all subproblems and has identified all subproblems that have admissible triangulations, we have to obtain the answer for the polygon as a whole. This is done in (c) and (d). In (c) we have two nested loops, both of n iterations, giving us $O(n^2)$ overall iterations. In each iteration we check one of the induced subproblems. If both the subproblem and its complementary subproblem have been found to have admissible triangulation we can return *true* and thus exit the computation of the function $M(\tau)$. So, the time for (c) is $O(n^2)$. In (d) we have to look for the possibility of the current edge, v_iv_j, being a base of a triangle with the property of Lemma 3. To check this, we use the results and procedure of Lemma 4. Namely, if such a triangle exists, with the given edge as a base, we only have to check certain intervals. We know, by Lemma 4, that admissible subproblems that start from v_i and end between $Top(v_iv_{MaxCW(v_i)})$ and $MaxCW(v_i)$, if any, form an interval. Thus, we can look in the priority queue $R[i]$ for these subproblems. This is done by insertion of the two ends of the interval - $Top(v_iv_{MaxCW(v_i)})$ and $MaxCW(v_i)$ in $R[i]$ and then querying $R[i]$ about their respective successor and predecessor. If those exist, we denote them by k_1 and k_2 and use them to find an admissible subproblem that starts between k_1 and k_2 and ends at v_j. Again this is done efficiently by inserting k_1 and k_2 in the priority queue $C[j]$. We also insert $Top(v_iv_j)$ there, and then query the queue $C[j]$ about the successor or/and predecessor of $Top(v_iv_j)$ in the interval $[k_1, k_2]$. If either of these vertices is found to give admissible triangulation, we have to record it in the queues R and C and in the $Table[\,]$ and return *true* for $M(\tau)$. If we cannot find such a vertex, we have to symmetrically try to find a vertex in the admissible interval $[k_3, k_4]$ of subproblems ending at v_j, such that the triangulation of P is admissible. If such a vertex exists we return *true* for $M(\tau)$. Otherwise, we exit the iteration. If none of the iterations has resulted in the return of *true* for $M(\tau)$, we have to conclude that a triangulation of P given this threshold is impossible, and return *false* for $M(\tau)$. As we have mentioned, there are $O(n^2)$ overall iterations. Within each iteration we only have a constant number of constant time operations or van Emde-Boas priority queue operations that are performed in $O(\log\log n)$ time. Thus, the total time complexity of the fragment in (d) and therefore of the computation of $M(\tau)$ is $O(n^2 \log\log n)$. The space that data structures, local to $M(\tau)$ use is $O(n^2)$ and they are reusable, i.e. each call to $M(\tau)$ uses the same memory space. Thus the space complexity of the local data structures is the same as the space used of the global data structures, namely $O(n^2)$.

From the analysis in this section we can conclude the following:

Theorem 1. *Algorithm 1 computes the MaxMin Area triangulation of a convex polygon P with n vertices in $O(n^2 \log\log n)$ time and $O(n^2)$ space.*

5 Conclusions

The future research that stems from the work on this problem has two main directions. First, there are some other optimal triangulations that might be attacked using the apparatus outlined here in the case of a convex polygon, MinMax Area, MaxMin Inradius and MaxMin Inradius/Circumradius. Second, the question arises about the computability of MaxMin Area triangulation in case of a simple but not convex polygon, and in the case when the point set is in general position. Both the question about the existence of a subcubic time algorithm for convex polygon and the question about the existence of a polynomial time algorithm for general set of points with respect to the three other quality measures mentioned above remain open.

The authors of the paper improved the claimed running time from $O(n^2 \log n \log\log n)$ to $O(n^2 \log n)$ within the same space bound of $O(n^2)$ between the first submission of the paper and the submission of its final version.

References

[1] T. J. Baker and P. P. Pebay. Comparison of triangle quality measures. In *Proceedings of the 10th International Meshing Roundtable*, pages 327–340. Sandia National Laboratories, October 2001.

[2] M. Bern, H. Edelsbrunner, D. Eppstein, S. Mitchell, and T. S. Tan. Edge insertion for optimal triangulations. *Discrete and Computational Geometry*, 10:47–65, 1993.

[3] O. Bottema, R. Ž. Djordjevič, R. R. Janič, D. S. Mitrinovič, and P. M. Vasič. *Geometric Inequalities*. Wolters-Noordhoff Publishing, Groningen, The Netherlands, 1969.

[4] G. T. Klincsek. Minimal triangulations of polygonal domains. *Anals of Discrete Mathematics*, 9:121–123, 1980.

[5] G. Toussaint. Complexity, convexity and unimodality. Technical Report SOCS 81.22, McGill University, Montreal, July 1981.

[6] G. Toussaint. Solving geometric problems with the rotating calipers. In *Proceedings of IEEE MELECON'83*, pages A10.02/1–4, Athens, Greece, May 1983.

[7] P. van Emde-Boas. Preserving order in a forest in less than logarithmic time and linear space. *Information Processing Letters*, 6(3):80–82, June 1977.

[8] P. van Emde-Boas, R. Kaas, and E. Zijlstra. Design and implementation of an efficient priority queue. *Mathematical Systems Theory*, 10:99–127, 1977.