Optimal Empty Pseudo-Triangles in a Point Set*

Hee-Kap Ahn† Sang Won Bae† Iris Reinbacher†

Abstract

Given n points in the plane, we study three optimization problems of computing an empty pseudo-triangle: we consider minimizing the perimeter, maximizing the area, and minimizing the longest maximal concave chain. We consider two versions of the problem: First, we assume that the three convex vertices of the pseudo-triangle are given. Let n denote the number of points that lie inside the convex hull of the three given vertices, we can compute the minimum perimeter or maximum area pseudo-triangle in $O(n^3)$ time. We can compute the pseudo-triangle with minimum longest concave chain in $O(n^2 \log n)$ time. If the convex vertices are not given, we achieve running times of $O(n \log n)$ for minimum perimeter, $O(n^6)$ for maximum area, and $O(n^2 \log n)$ for minimum longest concave chain. In any case, we use only linear space.

1 Introduction

A *pseudo-triangle* is a simply connected region of $\mathbb{R}^2$with exactly three convex vertices such that the boundary curves connecting pairs of these convex vertices must be concave. When all three boundary curves are polygonal, a pseudo-triangle is a simple polygon with exactly three convex vertices, that is, all the other vertices are concave (we consider a vertex with internal angle π to be concave). By definition, any triangle is a pseudo-triangle. Moreover, the convex hull of any pseudo-triangle is a triangle.

Pseudo-triangles were introduced in the context of computing visibility relations among convex obstacles in $\mathbb{R}^2$ [8, 7]. Later, a number of different optimization problems of *pseudo-triangulations*, partitionings of a region into polygonal pseudo-triangles, have been studied [1, 2, 5, 6, 9]. For an overview of pseudo-triangulations, we refer to the survey by Rote et al. [10].

Here, we are interested in an "empty" pseudo-triangle in the sense that, given a finite set of points in $\mathbb{R}^2$, it contains no points from the set in its interior. In particular, we consider the following problems: (1) either minimizing the perimeter or maximizing the area of the empty pseudo-triangle, and (2) minimizing the longest maximal concave curve of the empty pseudo-triangle. We first study these problems when the three convex vertices are given. We will later consider optimizations over all possible pseudo-triangles in a point set.

*This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD, Basic Research Promotion Fund) (KRF-2007-331-D00372).

†Department of Computer Science and Engineering, POSTECH, {heekap, swbae, irisrein}@postech.ac.kr

2 Empty Pseudo-Triangles with Given Corners

In this section, we consider finding empty pseudo-triangles in a given triangle that contains a set P of n points in its interior.

2.1 Preliminary Observations

Observation 1 *Every empty pseudo-triangle partitions P into three subsets.*

It follows that the convex hull of each subset is enclosed by the convex hull of each concave curve. An empty pseudo-triangle that either minimizes the perimeter or maximizes the area, is a simple polygon with concave vertices taken from P such that there are no points of P in its interior. The longest maximal concave curve is minimized by a concave polygonal chain on vertices of P. Therefore, it suffices to find an empty polygonal pseudo-triangle for each of the three optimization problems, and in the remainder of the paper we only consider such polygonal pseudo-triangles.

Let T, L, and R be the three corners of the input triangle. W.l.o.g. we assume that the segment connecting L and R is horizontal with L to the left of R, and that T lies above the segment. Let P be a set of n points inside the triangle. We let $L = t_0, t_1, \ldots, t_n, t_{n+1} = R$ be the sequence of points in $P \cup \{L, R\}$ sorted in counter-clockwise order around T. Similarly, let $R = l_0, l_1, \ldots, l_n, l_{n+1} = T$ be the sorted sequence around L, and let $T = r_0, r_1, \ldots, r_n, r_{n+1} = L$ be the sorted sequence around R in counter-clockwise order.

Assume that we are given an empty pseudo-triangle. At each corner, there are two concave chains, *left* and *right*. Let l_i, r_j, and t_k be the first vertices encountered when we follow the left concave chain from L, R, and T, respectively. Figure 1 shows these vertices and three lines, each induced by a corner and its corresponding vertex. Consider the gray region bounded by the lines $\ell(L, l_i)$ and $\ell(T, t_k)$. All points from P inside the gray region must be enclosed by the concave chain connecting L and T. Therefore the polygonal chain enclosing the

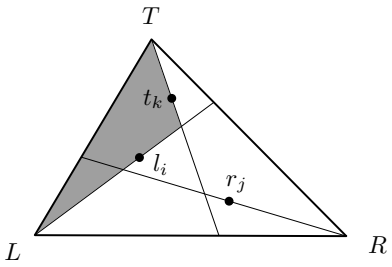

Figure 1: Every empty pseudo-triangle can be uniquely determined by a tuple (i, j, k) of indices.

points in the gray region is uniquely defined by l_i and t_k, and we denote the chain by $C_L(i, k)$. Analogously, the concave chain $C_T(k, j)$ connecting T and R is uniquely defined by t_k and r_j, and the concave chain $C_R(j, i)$ connecting R and L is uniquely defined by r_j and l_i.

Lemma 1 *Every empty pseudo-triangle is uniquely determined by a tuple* (i, j, k).

Let $\tau(i, j, k)$ be the empty pseudo-triangle determined by the tuple (i, j, k). Note that not every tuple determines an empty pseudo-triangle.

For a polygonal chain C, let $|C|$ be the total sum of the edge lengths in the chain. Then $|\tau(i, j, k)| = |C_L(i, k)| + |C_T(k, j)| + |C_R(j, i)|$. So we can formulate the problem of finding the minimum perimeter pseudo-triangle as follows.

$$\min\{|\tau(i, j, k)| \mid 1 \le i, j, k \le n, \tau(i, j, k) \text{ is empty}\}$$

Note that the problem of finding the maximum area pseudo-triangle can be formulated likewise, except that the area must be maximized.

2.2 Minimizing the Perimeter or Maximizing the Area

Recently, van Kreveld and Speckmann showed that given a triangle with n points inside it, there can be $\Theta(n^3)$ different empty polygonal pseudo-triangles [11]. By Lemma 1, each of them is uniquely determined by a tuple (i, j, k). A brute-force (n^4)-time algorithm can be designed as follows: for every pair of indices (i, j), iterate k from 1 to n and compute the two concave chains, $C_L(i, k)$ and $C_T(k, j)$. During the iteration on the index k, we can compute $C_L(i, k + 1)$ from $C_L(i, k)$, and $C_T(k + 1, j)$ from $C_T(k, j)$ in linear time.

To reduce the time complexity, we propose a faster way to compute the chains $C_L(i, k)$ and $C_T(k, j)$. For an index i, we define

$$C_L(i) := \bigcup_{k\in[1,n+1]} C_L(i, k).$$

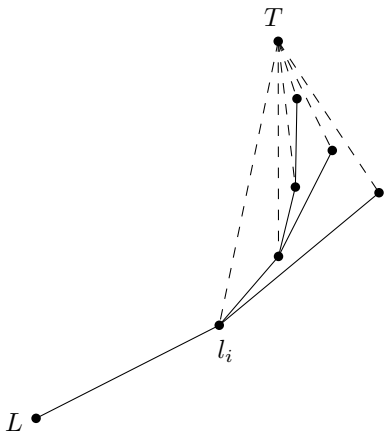

Figure 2: $C_L(i)$ consists of a star-graph $S_L(i)$(dotted segments) and a tree $U_L(i)$(solid segments).

By definition, l_i is the first vertex along $C_L(i, k)$, therefore the vertices of $C_L(i, k)$ must lie in a wedge (or on its boundary) bounded by the lines $\ell(L, l_i)$ and $\ell(T, l_i)$ and lying above l_i as in Figure 2. Let $P_L(i)$ be the set of points from P that lie in the wedge, and let $K_L(i) := \{k \mid 1 \le k \le n, t_k \in P_L(i)\}$.

Lemma 2 $C_L(i)$ *consists of* $2|K_L(i)|$ *noncrossing segments.*

For $C_L(i)$, let $S_L(i)$ denote the set of segments connecting T and t_k with $k \in K_L(i)$, and let $U_L(i)$ denote the set of segments in $C_L(i) \setminus S_L(i)$. Then $U_L(i)$ is a tree with root L that spans all vertices in $P_L(i) \setminus \{T\}$.

Similarly, we define $C_T(j) := \bigcup_{k\in[1,n+1]} C_T(k, j)$, and the set $P_R(j) \subset P$ of all points that lie above r_j and in the wedge bounded by the lines $\ell(T, r_j)$ and $\ell(R, r_j)$. Then $C_T(j)$ also consists of $2|K_T(j)|$ noncrossing segments, where $K_T(j) := \{k \mid 1 \le k \le n, t_k \in P_T(j)\}$.

2.2.1 Minimizing the perimeter

The following algorithm shows how to find for a pair of indices (i, j) the index k that minimizes $|C_L(i, k)| + |C_T(k, j)|$.

Algorithm *ShortestPseudo-triangle*
Input: A set P of n points inside a triangle
Output: The empty pseudo-triangle with minimum perimeter

```
1.  Sort P in angular order around L, R, and T
2.  for 1 ≤ i, j ≤ n + 1
3.      Compute C_R(j, i)
4.      Build C_L(i) and C_T(j)
5.      for each k ∈ K_L(i) ∩ K_T(j)
6.          Traverse U_L(i) and U_T(j), evaluate
            |C_L(i, k)| and |C_T(k, j)|
7.          Maintain the smallest |τ(i, j, k)|
```

8. **return** minimum perimeter empty pseudo-triangle

For a fixed pair (i, j) of indices, $C_R(j, i)$ can be computed in $O(n \log n)$ time by identifying all the points in P lying below the lines $\ell(L, l_i)$ and $\ell(R, r_j)$ in linear time and computing the convex hull of them and $\{L, R\}$. Once we build $C_L(i)$, we traverse the vertices v in $U_L(i)$ and store the length of the path from L to v at v. Let $\pi(v)$ denote the path length stored at v. For index $k' = \max\{k'' \in K_L(i) \mid k'' < k\}$, $|C_L(i, k)| = |Tt_{k'}| + \pi(t_{k'})$. Similarly, we can compute $|C_R(k, j)|$ once we build $C_R(j)$. Therefore, once we have $C_L(i)$ and $C_R(j)$, we can compute in linear time both $|C_L(i, k)|$ and $|C_R(k, j)|$ for all $k \in K_L(i) \cap K_R(j)$.

It remains to show how to build $C_L(i)$ and $C_R(j)$ in linear time. Since $S_L(i)$ can be computed in linear time, we only explain the linear-time construction of $U_L(i)$ in detail. Note that the path from L to any node in $U_L(i)$ is a convex curve. As illustrated in Figure 2, for each point t_k with $k \in K_L(i)$ we can find its parent in $U_L(i)$ in the angular (clockwise) order of points around T.

Algorithm *RootedTree*
Input: A corner L and index i
Output: A plane graph $U_L(i)$
1. $U \leftarrow (l_i, L)$
2. $p \leftarrow l_i$ and $q \leftarrow$ `NULL`
3. **for** each $k \in K_L(i)$ in increasing order
4. **do** Set $q \leftarrow p$ and $p \leftarrow$ the parent of p in U
5. Let γ be the ray from p towards q
6. **while** t_k is not to the right of γ
7. $U \leftarrow U \cup \{(t_k, q)\}$
8. $p \leftarrow t_k$ and $q \leftarrow$ `NULL`
9. **return** U as $U_L(i)$

Lemma 3 *Algorithm RootedTree computes $U_L(i)$ in time $O(|K_L(i)|)$.*

Proof. By induction on the index $k \in K_L(i)$. Consider the points t_k for $k \in K_L(i)$ in increasing order, and assume that we have computed the tree U' rooted at L up to k'. Then the algorithm sets $p = t_{k'}$ and $q =$`NULL`. Let k be the index next to k' in $K_L(i)$ and let U be the tree rooted at L up to k. By assumption, $C_L(i, k)$ consists of the segment connecting $t_{k'}$ and T, and the convex chain from $t_{k'}$ to L. Since t_k is not contained in the convex hull of $C_L(i, k)$, U must consist of U' and the segment connecting t_k and a vertex in $C_L(i, k)$.

Consider a vertex $v \in U_L(i)$. In the algorithm, we can see that v is traversed as many times as its degree in $U_L(i)$: once the pointer p moves from v to its parent node, it never points to v again. This is because the path from L to any node in U is a convex curve, and once v is enclosed by the convex hull of of such a convex curve of U and T, p will never point at v again. Therefore, the algorithm *RootedTree* takes $O(|K_L(i)|)$ time. □

Theorem 4 *Given n points inside a triangle, the empty pseudo-triangle with minimum perimeter can be computed in $O(n^3)$ time using linear space.*

2.2.2 Maximizing the area

For a set C we denote its area by $||C||$. Then the problem of maximizing the area of the empty pseudo-triangle is equivalent to the problem of finding indices i, j and k that minimize $||\tau(i, j, k)|| = ||\text{conv}(C_L(i, k))|| + ||\text{conv}(C_T(k, j))|| + ||\text{conv}(C_R(j, i))||$.

Hence, we can solve this problem using the algorithms of the previous section, except that instead of computing the length of each concave chain we compute the area of the convex hull of each convace chain. To do so, consider the arrangement of $C_L(i)$ in the triangle. Since $C_L(i)$ is a plane graph consisting of $O(n)$ segments, there are $O(n)$ cells in the arrangement. We can compute the areas of all the cells in linear time by traversing the tree $U_L(i)$.

Theorem 5 *Given n points inside a triangle, the empty pseudo-triangle with maximum area can be computed in $O(n^3)$ time using linear space.*

2.3 Minimizing the longest concave chain

For a pair of indices (i, j), we define a function h as follows.

$$h(i, j) := \min_{k \in [1, n+1]} \max\{|C_L(i, k)|, |C_T(k, j)|\}.$$

Then the problem of minimizing the longest chain is to find a pair of indices (i, j) that minimizes $\max\{|C_R(j, i)|, h(i, j)\}$.

Lemma 6 *For i, j with $1 \le i, j \le n$, the following inequalities hold.*

$$|C_B(j, i)| \le |C_B(j, i+1)|, \qquad |C_B(j, i)| \le |C_B(j+1, i)|;$$
$$h(i, j) \ge h(i+1, j), \qquad h(i, j) \ge h(i, j+1).$$

Let M be an $(n+1) \times (n+1)$ matrix with $M(i, j) = \max\{|C_R(j, i)|, h(i, j)\}$. Since both $|C_R|$ and h are monotone in both indices i and j, the matrix M is unimodal in columns and rows. As we did in the previous section, we can fill an entry $M(i, j)$ in linear time. Since M is unimodal, for each column we do binary search on the column, compute only the entries encountered during the search, and find the best indices. Since there are $n + 1$ columns, we can find the global optimal indices by computing only $O(n \log n)$ entries [4].

Theorem 7 *Given n points inside a triangle, the empty pseudo-triangle that minimizes the longest concave chain can be computed in $O(n^2 \log n)$ time using linear space.*

3 Empty Pseudo-Triangles in Point Sets

We now discuss the case where the three convex vertices are not given in advance. In this case, van Kreveld and Speckmann showed that the maximum possible number of empty pseudo-triangles is $\Theta(n^6)$ [11]. This leads to a straightforward adaptation of our algorithms from above to this case: Simply take all possible triples of points of P as the convex vertices of the pseudo-triangles and apply the appropriate algorithm. This obviously increases the running time by a factor of $O(n^3)$ in each case, without influencing the space requirement.

However, with a simple observation we can drastically reduce the running times for the two minimization problems. For the maximum area problem we get:

Theorem 8 *Given n points in $\mathbb{R}^2$, the empty pseudo-triangle with maximum area can be computed in $O(n^6)$ time using linear space.*

3.1 Minimizing the perimeter

Lemma 9 *Given n points in $\mathbb{R}^2$, the empty pseudo-triangle with minimum perimeter is an empty triangle.*

The proof uses the triangle inequality. We can use a divide-and-conquer approach similar to finding the closest pair of points to find the closest triple of points. We refer to the book by Cormen et.al. [3] for further details of the algorithm.

Theorem 10 *Given n points in $\mathbb{R}^2$, the empty pseudo-triangle with minimum perimeter can be computed in $O(n \log n)$ time using linear space.*

3.2 Minimizing the longest concave chain

Lemma 11 *Given n points in $\mathbb{R}^2$, the empty pseudo-triangle that minimizes the longest concave chain is an empty triangle.*

Using this lemma, we can devise a simple greedy approach: We sort all possible edges on the points of P by increasing length and insert them into P one by one. The first triangle that is created has the minimum longest edge over all possible triangles. This approach takes $O(n^2 \log n)$ preprocessing time and $O(n^2)$ time to find the first triangle, and $O(n^2)$ space. By using a simple observation we can improve the space requirement:

Observation 2 *When inserting edges by length, the first triangle is closed before the first crossing occurs.*

As there can be no crosssings, it follows directly that we need to take only the shortest $3n-6$ edges into account for the insertion. We start with an empty array M that can hold up to $3n-6$ entries. For each point in P, we do the following: Sort the edges to all other points in order of increasing length in an array H, and merge it with M. All edges that do not fit into M can be discarded directly. When we have finished the merge for all points, M contains the $3n-6$ shortest edges in P, and we can now proceed to consecutively insert the edges of M by increasing length into P until we close the first triangle. As we can reuse H for each new point, and we use only M during the run time, we have reduced the space complexity to $O(n)$.

Theorem 12 *Given n points in $\mathbb{R}^2$, the empty pseudo-triangle minimizing the longest concave chain can be computed in $O(n^2 \log n)$ time using linear space.*

Acknowledgements We would like to thank Marc van Kreveld for introducing us to the problem.

References

[1] O. Aichholzer, F. Aurenhammer, H. Krasser, and B. Speckmann. Convexity minimizes pseudo-triangulations. *Comp. Geom.: Th. Appl.*, 28(1):3–10, 2004.

[2] O. Aichholzer, D. Orden, F. Santos, and B. Speckmann. On the number of pseudo-triangulations of certain point sets. *J. Comb. Th. A*, 115(2):254–278, 2008.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2nd edition, 2001.

[4] E. Demaine and S. Langerman. Optimizing a 2d function satisfying unimodality properties. In *Proc. 13th Ann. Europ. Symp. Alg. (ESA 2005)*, pages 887–898, 2005.

[5] J. Gudmundsson and C. Levcopoulos. Minimum weight pseudo-triangulations. *Comp. Geom.: Th. Appl*, 38(3):139–153, 2007.

[6] M. Pocchiola and G. Vegter. Pseudo-triangulations: Theory and applications. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 291–300, 1996.

[7] M. Pocchiola and G. Vegter. Topologically sweeping visibility complexes via pseudo-triangulations. *Discrete Comput. Geom.*, 16:419–453, Dec. 1996.

[8] M. Pocchiola and G. Vegter. The visibility complex. *Internat. J. Comput. Geom. Appl.*, 6(3):279–308, 1996.

[9] G. Rote, F. Santos, and I. Streinu. Expansive motions and the polytope of pointed pseudo-triangulations. *Disc. Comp. Geom.*, The Goodman-Pollack Festschrift:699–736, 2003.

[10] G. Rote, F. Santos, and I. Streinu. Pseudo-triangulations–a survey. In J. E. Goodman, J. Pach, and R. Pollack, editors, *Surveys on Discrete and Computational Geometry-Twenty Years Later*, volume 453 of *Contemporary Mathematics*, pages 343–410. American Mathematical Society, 2008.

[11] M. van Kreveld and B. Speckmann. On the number of empty pseudo-triangles in point sets. In *Proc. 19th Can. Conf. Comp. Geom (CCCG 2007)*, pages 37–40, 2007.