On the Maximum Triangle Problem

Afrouz Jabalameli

Hamid Zarrabi-Zadeh

Abstract

Given a set \( P \) of \( n \) points in the plane, the maximum triangle problem asks for finding a triangle with three vertices on \( P \) enclosing a maximum number of points of \( P \). While the problem is easily solvable in \( O(n^3) \) time, it has been open whether a subcubic solution is possible. In this paper, we show that the problem can be solved in \( o(n^3) \) time, settling this open problem. We also improve the runtime of some of the previous approximation algorithms available for the problem.

1 Introduction

Let \( P \) be a set of \( n \) points in the plane. In the maximum triangle problem, the objective is to find a triangle with three vertices on \( P \), so that the number of points of \( P \) enclosed by the triangle is maximum (see Figure 1 for an illustration). Eppstein et al. [4] showed that the problem can be solved in \( O(n^3) \) time. They indeed solved a more general problem of finding a convex \( k \)-gon enclosing a maximum (or minimum) number of points in \( O(kn^2) \) time. They left this question open whether the problem can be solved faster.

Doué et al. [3] revisited the maximum triangle problem, and presented several subcubic approximation algorithms for it. They again posed finding an \( o(n^3) \)-time exact algorithm as an open problem.

In this paper, we settle this open problem in affirmative by showing that an \( o(n^3) \)-time exact algorithm is indeed possible, using a reduction to the min-plus matrix multiplication, for which slightly subcubic algorithms are already known [1, 2, 5, 6]. The min-plus matrix multiplication (also known as distance product) has recently attracted considerable attention due to its connection to several fundamental problems such as all-pairs shortest paths, minimum cycles, replacement paths, metricity, etc. [7]. The current best time complexity for computing the min-plus product is \( n^3/2^{\Omega(\sqrt{\log n})} \) [2, 6].

We also consider approximation algorithms for the maximum triangle problem, and improve the runtime of several algorithms proposed by Doué et al. [3] for the problem. Table 1 shows a summary of our results. In this table, \( h \) denotes the size of the convex hull of \( P \).

Figure 1: An example of a maximum triangle.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Previous Runtime</th>
<th>New Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact</td>
<td>( O(n^3) )</td>
<td>( n^3/2^{\Omega(\sqrt{\log n})} )</td>
</tr>
<tr>
<td>3-approx</td>
<td>( O(nh^2 \log n) )</td>
<td>( O(nh \log n + nh^2) )</td>
</tr>
<tr>
<td>4-approx</td>
<td>( O(n^2 \log h) )</td>
<td>( O(nh \log h + h^3) )</td>
</tr>
<tr>
<td>4-approx</td>
<td>( O(n \log^2 n) )</td>
<td>( O(n \log n \log h) )</td>
</tr>
</tbody>
</table>

Table 1: Summary of the results.

2 Preliminaries

Let \( P \) be a set of \( n \) points in the plane. Throughout this paper, we assume that the points are in general position, i.e., no three points are co-linear, and no two points have the same \( x \)-coordinates.

Given three points \( p, q, r \in P \), we call \( \triangle pqr \) a triangle in \( P \), and denote by \( |\triangle pqr| \) the number of points of \( P \) enclosed by \( \triangle pqr \). A triangle \( \triangle pqr \) with maximum \( |\triangle pqr| \) is called a maximum triangle of \( P \), or in short, an optimal triangle.

3 A Subcubic Exact Algorithm

In this section, we show how the maximum triangle problem can be solved in \( o(n^3) \) time, using matrix multiplication over the \((\min, +)\)-semiring, for which slightly subcubic algorithms are available. Recall that the \( \min-plus \) product of two \( n \times n \) matrices \( A \) and \( B \) is defined as
\[
(A \oplus B)_{i,j} = \min_{1 \leq k \leq n} \{ A_{i,k} + B_{k,j} \}.
\]

Theorem 1 Let \( P \) be a set of \( n \) points in the plane. A maximum triangle of \( P \) can be found in \( O(T(n)) \) time, where \( T(n) \) is the time needed for computing the min-sum product of two \( n \times n \) matrices, the best current algorithm for which has \( n^3/2^{\Omega(\sqrt{\log n})} \) runtime.
proposed several subcubic approximation algorithms for the maximum triangle problem. The main idea behind their algorithms is to reduce the number of triangles enumerated by fixing 1, 2, or 3 vertices of the optimal triangle on the convex hull of the points. They also used this observation that if the surface of an optimal triangle is covered by $c$ triangles (for a constant $c \geq 1$), then one of these triangles is a $c$-approximation of the optimal triangle.

In this section, we improve the runtime of the approximation algorithms proposed by Douieb et al. [3], using faster methods for counting the number of points in the enumerated triangles.

Proof. For each pair of points $p, q \in P$, we denote by $n_{pq}$ the number of points of $P$ in the vertical slab below the line segment $pq$. The value of $n_{pq}$ for all pairs $p, q \in P$ can be computed in $O(n^2)$ time [4]. For any two points $p, q \in P$, we set $n_{pq} = n_{qp}$ if the vector $pq$ is directed from left to right, and set $n_{pq} = -n_{qp}$ otherwise.

Now, for any three points $p, q, r \in P$ in clockwise order, the number of points in the triangle $\triangle pqr$ can be written as:

$$|\triangle pqr| = n_{pq} + n_{qr} + n_{rp},$$

(see Figure 2 for an illustration). For points in counterclockwise order, we have $|\triangle pqr| = -(n_{pq} + n_{qr} + n_{rp}).$

Let $A$ be a $n \times n$ matrix with $A_{pq} = n_{pq}$, and let $B = A \oplus (A \oplus A)$. By the definition of the min-plus product, we have

$$B_{p,p} = \min_{q, r \in P} \{A_{p,q} + A_{q,r} + A_{r,p}\},$$

for all $p \in P$. Therefore, to obtain a maximum triangle, we just need to check the $n$ values on the main diagonal of the matrix $B$ for the smallest (negative) number, whose absolute value corresponds to the number of points in a maximum triangle. The optimal triangle itself can be easily found in $O(n^2)$ time by enumerating all $O(n^2)$ triangles with one vertex on the point realizing the smallest value in the diagonal. The whole runtime of the algorithm is therefore bounded by that of computing the min-plus product. \hfill $\square$

4 Improved Approximation Algorithms

Douieb et al. [3] proposed several subcubic approximation algorithms for the maximum triangle problem. The main idea behind their algorithms is to reduce the number of triangles enumerated by fixing 1, 2, or 3 vertices of the optimal triangle on the convex hull of the points. They also used this observation that if the surface of an optimal triangle is covered by $c$ triangles (for a constant $c \geq 1$), then one of these triangles is a $c$-approximation of the optimal triangle.

In this section, we improve the runtime of the approximation algorithms proposed by Douieb et al. [3], using faster methods for counting the number of points in the enumerated triangles.

Proof. For each pair of points $p, q \in P$, we denote by $n_{pq}$ the number of points of $P$ in the vertical slab below the line segment $pq$. The value of $n_{pq}$ for all pairs $p, q \in P$ can be computed in $O(n^2)$ time [4]. For any two points $p, q \in P$, we set $n_{pq} = n_{qp}$ if the vector $pq$ is directed from left to right, and set $n_{pq} = -n_{qp}$ otherwise.

Now, for any three points $p, q, r \in P$ in clockwise order, the number of points in the triangle $\triangle pqr$ can be written as:

$$|\triangle pqr| = n_{pq} + n_{qr} + n_{rp},$$

(see Figure 2 for an illustration). For points in counterclockwise order, we have $|\triangle pqr| = -(n_{pq} + n_{qr} + n_{rp}).$

Let $A$ be a $n \times n$ matrix with $A_{pq} = n_{pq}$, and let $B = A \oplus (A \oplus A)$. By the definition of the min-plus product, we have

$$B_{p,p} = \min_{q, r \in P} \{A_{p,q} + A_{q,r} + A_{r,p}\},$$

for all $p \in P$. Therefore, to obtain a maximum triangle, we just need to check the $n$ values on the main diagonal of the matrix $B$ for the smallest (negative) number, whose absolute value corresponds to the number of points in a maximum triangle. The optimal triangle itself can be easily found in $O(n^2)$ time by enumerating all $O(n^2)$ triangles with one vertex on the point realizing the smallest value in the diagonal. The whole runtime of the algorithm is therefore bounded by that of computing the min-plus product. \hfill $\square$

In this remaining of this section, we assume that $P$ is a set of $n$ points in the plane, $H$ is the convex hull of $P$, and $h = |H|$. We will use the following two auxiliary results from Douieb et al. [3].

Lemma 2 ([3]) Among all triangles in $P$ with $k$ vertices on the convex hull ($1 \leq k \leq 3$), there exists a triangle that $(k+1)$-approximates an optimal triangle.

Lemma 3 ([3]) Given two points $p, q \in H$, the value of $|\triangle pqr|$ for all $r \in P$ can be computed in $O(n \log n)$ time. Furthermore, $|\triangle pqr|$ for all $r \in H$ can be computed in $O(n \log h)$ time.

The following is a direct corollary of Lemma 3.

Lemma 4 Given a point $p \in H$, the value of $|\triangle pqr|$ for all $q, r \in H$ can be computed in $O(n \log h)$ time. Furthermore, $|\triangle pqr|$ for all $q \in P$ and $r \in H$ can be computed in $O(nh \log n)$ time.

Proof. Fix a point $q$ on $H$. By Lemma 3, $|\triangle pqr|$ for all $r \in H$ can be computed in $O(n \log h)$ time. Since there are $h - 1$ option for choosing $q$, computing $|\triangle pqr|$ for all $q, r \in H$ takes $O(nh \log h)$ time in total. Similarly, if we fix $q \in P$, the algorithm takes $O(nh \log n)$ time by Lemma 3. \hfill $\square$

Now, we prove two lemmas which are the main ingredients of our improved algorithms.

Lemma 5 The value of $|\triangle pqr|$ for all $p, q, r \in H$ can be computed in $O(nh \log h + h^3)$ time.

Proof. Let $p, q, r, s$ be four points on $H$ in clockwise order. The value of $|\triangle pqr|$ can be written as $|\triangle spq| + |\triangle sqr| - |\triangle spr|$ (see Figure 3). By Lemma 4 we can compute the number of points enclosed by all triangles on $H$ whose one vertex is fixed on $s$ in $O(nh \log n)$ time. Therefore, after this preprocess step, we can compute the value of $|\triangle pqr|$ for each $p, q, r \in H$ in $O(1)$ time. Since there are $O(h^3)$ such triangles, the whole process takes $O(nh \log h + h^3)$ time in total. \hfill $\square$
Lemma 6 For all \( p, q \in H \) and \( r \in P \), the value of \(|\triangle pqr|\) can be computed in \( O(nh \log n + nh^2) \) total time.

Proof. For a fixed point \( s \) on \( H \), we compute the number of points enclosed by all triangles with one vertex on \( s \), and the other two vertices freely chosen one from \( P \) and the other from \( H \) in \( O(nh \log n) \) time using Lemma 4. Now, for any triangle \( \triangle pqr \) with \( p, q \in H \) and \( r \in P \), we compute \(|\triangle pqr|\) as follows.

(i) If \( r \) lies inside \( \triangle pqs \), then \(|\triangle pqr| = |\triangle pqs| - |\triangle prs| - |\triangle qrs|\).

(ii) If \( \overrightarrow{pq} \) crosses \( \overrightarrow{sr} \), then \(|\triangle pqr| = |\triangle pqs| + |\triangle qrs| - |\triangle prs|\).

(iii) If \( \overrightarrow{qs} \) crosses \( \overrightarrow{pr} \), then \(|\triangle pqr| = |\triangle pqs| + |\triangle qrs| - |\triangle prs|\).

(iv) If \( \overrightarrow{ps} \) crosses \( \overrightarrow{qr} \), then \(|\triangle pqr| = |\triangle prs| + |\triangle qrs| - |\triangle pqs|\).

In any of the above cases, \(|\triangle pqr|\) can be computed in \( O(1) \) time. Since there are \( O(nh^2) \) different triangles \( \triangle pqr \) with \( p, q \in H \) and \( r \in P \), we can compute \(|\triangle pqr|\) for all such triangles in \( O(nh \log n + nh^2) \) total time. \( \square \)

Now, Lemmas 5 and 6 together with Lemma 2 yield the following theorem.

Theorem 7 A 3-approximation of an optimal triangle can be found in \( O(nh \log n + nh^2) \) time. Furthermore, a 4-approximation of an optimal triangle can be found in \( O(nh \log h + h^3) \) time.

Remark. Eppstein et al. [4] proved that \( P \) can be preprocessed in \( O(n^2) \) time, so that for any query triangle \( \triangle pqr \) in \( P \), \(|\triangle pqr|\) can be reported in \( O(1) \) time. Using this as an alternative way for counting the number of points in the enumerated triangles, we can rewrite the time bounds in Theorem 1 as \( O(\min(n^2 + nh^2, nh \log n + nh^2)) \) for the 3-approximation, and \( O(\min(n^2 + h^3, nh \log h + h^3)) \) for the 4-approximation algorithm.

In the following theorem, we present an alternative 4-approximation algorithm for the problem.

Theorem 8 A 4-approximation of an optimal triangle can be found in \( O(n \log n \log h) \) time.

Proof. Let \( t_1, t_2, \ldots, t_h \) be the vertices of \( H \) in clockwise order, and let \( m = \lceil h/2 \rceil + 1 \). We partition \( H \) into two convex polygons \( H_1 = t_1, t_2, \ldots, t_m \) and \( H_2 = t_m, t_{m+1}, \ldots, t_h, t_1 \). Let \( P_1 \) and \( P_2 \) be the points of \( P \) enclosed by \( H_1 \) and \( H_2 \), respectively. We use Lemma 3 to compute \(|\triangle t_1 t_m n|\) for all \( p \in P \) in \( O(n \log n) \) time. We then recur on \( P_1 \) and \( P_2 \), and return a triangle found containing a maximum number of points.

To prove correctness, we first recall that there exists a triangle \( \triangle t_1 t_m p \) with \( p \in P \) that 2-approximates an optimal triangle [3]. If \( t_1 t_m \) crosses \( pq \), then the two triangles \( \triangle t_1 t_m p \) and \( \triangle t_1 t_m q \) cover \( \triangle t_1 p q \), and hence, one of them is a 2-approximation of \( \triangle t_1 p q \), which is in turn, a 4-approximation of an optimal triangle. On the other hand, if \( pq \) lies in one side of \( t_1 t_m \), the recursive call on that side returns a 2-approximation.

Let \( T(n, h) \) be the time required by the algorithm on a point set of size \( n \) whose convex hull has size \( h \). Then, \( T(n, h) = T(n_1, h_1) + T(n_2, h_2) + O(n \log n) \), where \( n_1 = n + 2, h_1 = \lceil h/2 \rceil + 1 \), and \( h_2 = \lceil h/2 \rceil + 1 \). The recurrence tree for this relation has height \( O(\log h) \), and yields \( T(n, h) = O(n \log n \log h) \).

5 Conclusions

In this paper, we presented a slightly subcubic algorithm for the maximum triangle problem, and improved the runtime of several approximation algorithms available for the problem. A main question that remains open is whether a truly subcubic algorithm with \( O(n^{3-c}) \) time is possible for the problem. It is also interesting to study the generalized maximum \( k \)-gon problem, for \( k \geq 4 \).

Acknowledgments The authors would like to thank Mohammad-Reza Maleki, Hamed Valizadeh, and Hamed Saleh for their helpful discussions during the early stages of this work.

References