**Linear-time Algorithms for Largest Inscribed Quadrilateral**

Keikha, Vahideh
2020

**Institute of Computer Science**
**The Czech Academy of Sciences**

# Linear-time Algorithms for Largest Inscribed Quadrilateral

Vahideh Keikha

Technical report No. V-1283

December 2020

Pod Vodárenskou věží 2, 182 07 Prague 8, phone: +420 266 053 290, fax: +420 286 585 789,
e-mail: keikha@cs.cas.cz

**Institute of Computer Science**
**The Czech Academy of Sciences**

# Linear-time Algorithms for Largest Inscribed Quadrilateral

Vahideh Keikha[1]

Technical report No. V-1283

December 2020

Abstract:

Let P be a convex polygon of n vertices. We present a linear-time algorithm for the problem of computing the largest-area inscribed quadrilateral of P.
We also design the parallel version of the algorithm with O(log n) time and O(n) work in CREW PRAM model, which is quite work optimal. Our parallel algorithm also computes all the antipodal pairs of a convex polygon with O(log n) time and O($log^2 n + s$) work, where s is the number of antipodal pairs, that we hope is of independent interest.
We also discuss several approximation algorithms (both constant factor and approximation scheme) for computing the largest-inscribed k-gons for constant values of k, in both area and perimeter measures.

Keywords:
Maximum-area quadrilateral, extreme area k-gon

[1]Institute of Computer Science, The Czech Academy of Sciences, Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic, E-mail: keikha@cs.cas.cz.

# Linear-time Algorithms for Largest Inscribed Quadrilateral

Vahideh Keikha

Institute of Computer Science, the Czech Academy of Sciences, keikha@cs.cas.cz

June 2019

**Abstract**

Let $P$ be a convex polygon of $n$ vertices. We present a linear-time algorithm for the problem of computing the largest-area inscribed quadrilateral of $P$. We also design the parallel version of the algorithm with $O(\log n)$ time and $O(n)$ work in CREW PRAM model, which is quite work optimal. Our parallel algorithm also computes all the antipodal pairs of a convex polygon with $O(\log n)$ time and $O(\log^2 n + s)$ work, where $s$ is the number of antipodal pairs, that we hope is of independent interest. We also discuss several approximation algorithms (both constant factor and approximation scheme) for computing the largest-inscribed $k$-gons for constant values of $k$, in both area and perimeter measures.

## 1 Introduction

Let $P$ be a simple polygon. One of the classic problems in computational geometry concerns on computing a subset $Q$ of $P$, where the constructed polygon on $Q$ entirely is located inside $P$, and it is optimal, in some sense. Let $k$ denote the cardinality of $Q$.

For the case where $k = 2$, several algorithms are presented during the past 40 years. For instance, when $P$ is convex, the problem reduces to computing the *diameter* of $P$ that realizes the maximum pairwise distance of the vertices of $P$. But when $P$ is not convex, such measure is called *stick* and cannot be computed faster than $O(n \log n)$ time, since it does not use the vertices of $P$ necessarily. There exist several exact and approximation algorithms for either case, see e.g., [10] and the references therein.

For the case where $k > 2$, there exist several studies, which compute a subset of $k$ vertices that optimize the area or the perimeter of $Q$. The best known algorithms work in $O(kn+n \log n)$ time [11, 6, 8, 2]. For the special case of $k = 3$, there exists a linear time algorithm for computing the largest-area inscribed triangle [7] of a convex polygon. See also [12] and the references therein. For the case of $k = 4$, independent of this work, a linear-time algorithm is presented by Günter Rote [13]. The author also shows that this problem is closely related to the problem of computing the smallest-area inscribing parallelogram of $P$; any algorithm that solves one of these problems can solve the other problem, with a slight modification.

But to date, neither a linear time algorithm nor a lower bound exists for the problem of computing the largest-perimeter inscribed triangle of $P$. Already two $O(n \log n)$ time algorithms are presented to compute the largest-perimeter inscribed triangle of $P$ [6, 11], where both algorithms are based on the fact that the largest-perimeter triangle rooted at a given vertex $r$ can be computed in linear time [6]. One might be interested to have a near-optimum solution computed by a constant factor approximation algorithm for this

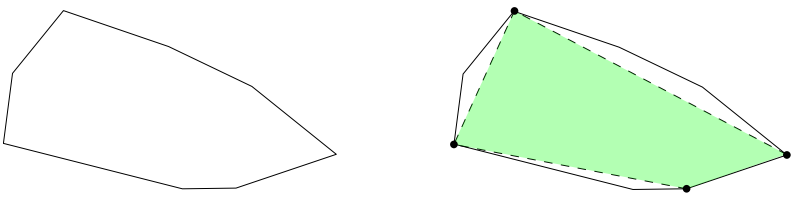

Figure 1: Problem definition and its optimal solution. (left) A convex polygon $P$ of 8 vertices. (right) The largest-area inscribed quadrilateral of $P$.

problem, but ideally computing the exact solution or an approximation scheme which it's running time has a linear dependency to $n$ is more desired. We also address this problem.

While there is a very extensive literature on "parallel algorithms", it invariably emphasizes the allocation of attention to see is there any strategy to develop parallel computational methods that run very fast. A closely related problem is the all-farthest neighbors problem in a simple polygon $P$ [9] that seeks the furthest neighbor of each vertex, where the distance is measured by the shortest path between them constrained to lie inside $P$. This problem can be solved in $O(\log^2 n)$ time and $O(n \log^2 n)$ work on CREW PRAM model that provides a shared memory with concurrent-read and exclusive-write. We will solve this problem much more efficiently for convex polygons, where the furthest neighbor of each vertex is the anchored diameter of $P$ at that vertex. The Largest-area empty rectangle can also be computed in $O(\log^2 n)$ time and $O(n \log^3 n)$ work on CREW PRAM model [3]. The largest-area inscribed and inscribing triangles of $P$ can also be computed in $O(\log n)$ time and $O(n)$ work on CREW PRAM model [7]. See also the references therein.

### 1.1 Contribution

Let $P$ be a convex polygon of $n$ vertices, and let $k \leq n$ be a positive integer. We present several approximation algorithms for computing the largest-inscribed $k$-gons of $P$, for constant values of $k$, and in both area and perimeter measures (Section 2). We then show that for the problem of computing the largest-area inscribed quadrilateral of $P$ it suffices to find the antipodal pairs of the convex polygon, and we present a linear-time algorithm for this problem (Section 3, Section 4). An illustration of the problem is shown in Figure 1. We also design the parallel version of the algorithm with $O(\log n)$ time and $O(n)$ work in CREW PRAM model, which is quite work optimal. Our parallel algorithm also computes the diameter and all the antipodal pairs of a convex polygon with $O(\log n)$ time and $O(\log^2 n + s)$ work, where $s$ is the number of antipodal pairs.

## 2 Approximating the Largest $k$-gon

We start with a simple constant factor approximation algorithm for the case where $k = 3$..

**Observation 1.** *There exists a 2/3-approximation algorithm for computing the largest-perimeter triangle inscribed in a convex polygon $P$ which runs in linear time.*

*Proof.* Let $d$, $d_1$ and $d_2$, respectively, denote the diameter, second and third longest chord of the polygon. We compute three largest perimeter triangle $P_d$, $P_1$ and $P_2$ with a base fixed at $d$, $d_1$ and $d_2$, respectively. Let $\Lambda_P^3$ denote the optimal triangle. Obviously $\Lambda_P^3 \leq d + d_1 + d_2$, and the largest triangle among $P_d$, $P_1$ and $P_2$ is always greater than $2d$. Let $P_{max} = max(P_d, P_1, P_2)$. Since $d + d_1 + d_2 < 1/2(P_d + P_1 + P_2)$, we have $2/3\Lambda_P^3 \leq P_{max}$. □

We also can adapt the above algorithm to approximate any largest-perimeter $k$-gon for any value of $k$, and within the factor of $2/k$. We first compute all antipodal pairs of the polygon $P$ in linear time, then we compute the $k$-th largest chord among them. Let $\zeta$ denote the length of such a chord. Then we can partition the list of the antipodal pairs into two pars; the list of the $k$ chords which have a larger length than $\zeta$, and those who do not. Let $P_\zeta$ denote the first list. Since we still have $d + \ldots + d_k < 1/2(P + \ldots + P_k)$, in which each $P_i$ for $i = 1, \ldots, k$ is any convex $k$-gon constructed on the chord $d_i$, then the largest one among all approximates the largest-perimeter $k$-gon within a factor of $2/k$.

**Corollary 2.** *There exists a $2/k$-approximation algorithm for computing the largest-perimeter inscribed $k$-gon of a convex polygon which runs in linear time.*

### 2.1 An Approximation Scheme

It is well known that an arbitrary convex polygon $P$ of $n$ vertices can be approximated by another convex polygon $Q$, where $Q$ can be computed in $O(n+1/\epsilon^{-1})$ time and has at most $\epsilon^{-1/2}$ vertices, and the Hausdorff distance between $P$ and $Q$ is at most $\epsilon \cdot \text{diam}(P)$ [5]. We will use this result to design a simple approximation algorithm for our problems.

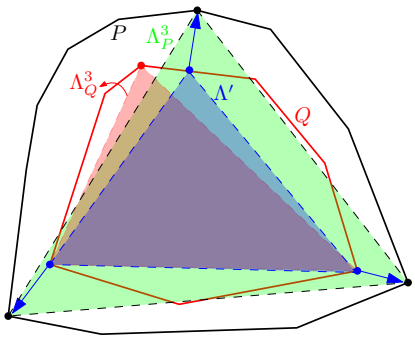


Figure 2: Illustration of the relation between $\Lambda_P^3, \Lambda_Q^3$ and $\Lambda'$. The distance of $P$ and $Q$ are agglomerated for clarity. In reality, the Hausdorff distance of $P$ and $Q$ equals $0 < \epsilon < 1$.

Let $\Lambda_P^k$ denote the largest-perimeter inscribed $k$-gon of $P$. Also, let $\Lambda'$ denote the inscribed $k$-gon of $Q$ which its vertices are in the direction that the normal of the edges of $Q$ passes through the vertices of $\Lambda_P^k$. First suppose $k = 3$. An example is illustrated in Figure 2. Obviously $\Lambda' \leq \Lambda_Q^3$. Also $\Lambda_P^3 \leq \Lambda' + 6\epsilon \cdot diam(P)$, since each vertex of $\Lambda_P^3$ is at most a $\epsilon \cdot$ diam$(P)$ further than a vertex of $\Lambda'$. Also $\Lambda_P^3 > 2diam(P)$. Altogether $\Lambda_P^3(1 - 3\epsilon) \leq \Lambda_Q^3$. Since $Q$ can be computed in $O(n + 1/\epsilon^{-1})$ time, obviously the approximate largest perimeter triangle can also be computed in the mentioned time. This method works for computing any largest perimeter inscribed polygon with $k > 3$ sides, where $k$ is a constant, so that $\Lambda_P^k(1 - k\epsilon) \leq \Lambda_Q^k$. Notice that $k < \epsilon^{-1}$.

In the following, we show that we can design an approximation scheme that can also approximate the largest-area inscribed $k$-gon.

**Lemma 3.** *Let $P$ be a given set of $n$ points in the plane. There exists an $\epsilon$-kernel of size $O(\sqrt{\epsilon^{-1}})$ for the maximum area/perimeter $k$-gon $Q$ with a constant $k \leq \epsilon^{-1/2}$, where all the vertices of $Q$ are selected from $P$.*

*Proof.* Agarwal *et al.* [1] proved that for any point set in $d$-dimensional space, there is an $\epsilon$-kernel of size $O(1/\epsilon^{(d-1)/2})$ and it is also worst case optimal. Let $P_k(P)$ (resp. $A_k(P)$) denote a maximal perimeter (resp. area) $k$-gon with vertices of $P$. Let $opt(P)$ denote the optimal solution to the maximum area/perimeter $k$-gon to be constructed on $P$. Then there exists an $\epsilon$-kernel $Q_{opt}$ for $opt(P)$, that is $opt(P) \leq (1 + \epsilon)P_k(Q_{opt})$. Also we have $P_k(Q_{opt}) \leq opt(Q_{opt})$. Then $opt(P) \leq (1 + \epsilon)opt(Q_{opt})$. The lemma follows. □

Note that in above lemma we do not need the optimal $k$-gon to be inscribed in a convex polygon, since if $|CH(P)| < k$, then the largest-area $k$-gon which selects its vertices from $P$ is not necessarily inscribed in $CH(P)$. Also above lemma implies that we can approximate an optimal $k$-gon for larger values of $k$, but only by reducing the accuracy of the approximation, where $k$ can be any bounded big constant.

For a given convex polygon $P$ of $n$ vertices, the largest area/perimeter inscribed $k$-gon can be computed in $O(kn \log n)$ time [6]. However we do not know the $\epsilon$-kernel, we still can use its size to design a PTAS which has a linear dependency to $n$. By considering all subsets of size $O(\epsilon^{-1/2})$ of $P$, we achieve the following results.

**Lemma 4.** *Let $P$ be a convex polygon of $n$ vertices. There exists a PTAS for the problem of computing the largest area or largest perimeter $k$-gon inscribed in $P$ which runs in $O(kn^{\epsilon^{-1/2}}\epsilon^{-1/2} \log \epsilon^{-1/2})$ time.*

It is an interesting open question to design an approximation scheme for computing a $k$-gon with maximum area or perimeter which works in linear time for arbitrary values of $k$.

# 3 An $O(n)$ Time Sequential Algorithm for the Largest Quadrilateral

Suppose $P = \{p_1, \ldots, p_n\}$ is a convex polygon specified by the clockwise ordered list of its vertices along with their Cartesian coordinates. If for any fixed vertex $p_i$ of $P$, the distance of $p_i$ to the remaining vertices traversed in the given order makes a unimodal function, then $P$ is a unimodal polygon. An antipodal pair in $P$ is a pair of two points admitting two parallel lines of support being tangent to both points included

in the antipodal pair, without crossing the interior of $P$. We suppose general position assumption for the vertices of $P$ so that no two edges of $P$ are parallel. For any vertex $p_i \in P$, we denote by $p_{i-1}$ (resp. $p_{i+1}$) the previous (resp. next) neighbor of $p_i$ on the boundary of $P$ in clockwise (all indexes are in module $n$).

Boyce *et al.* [6] define a *rooted* polygon $Q$ with root $p_r \in P$ to be any polygon which its set of vertices is a subset of the vertices of $P$ and includes $p_r$. Dobkin and Snyder [8] define a *stable* triangle to be a rooted triangle $T = p_i p_j p_r$, where $p_r$ is the root, such that any other triangle $p_{\{i+1,i-1\}} p_j p_r$ or $p_i p_{\{j+1,j-1\}} p_r$ has area smaller than $T$. Such triangles were refereed as *2-stable* in [11], and $p_i$ and $p_j$ are called stable vertices. We also define a *3-stable* 4-gon $p_i p_j p_l p_r$ as a quadrilateral to be rooted at $p_r$, with three stable vertices $p_i, p_j$ and $p_l$.

We start stating our results by providing some Lemmas which we need in the following.

**Lemma 5.** *Let $P$ be a unimodal convex polygon without any parallel edges. Any vertex of $P$ is an antipodal pair's point.*

*Proof.* Since the polygon $P$ is unimodal and convex, for any arbitrary vertex $p_a$ there always exists a vertex $p_b$ with maximum distance from $p_a$. Obviously $p_a$ and $p_b$ always admit parallel lines of support for $P$. Consequently, any vertex $p_a \in P$ is always an antipodal pair's point. □

**Lemma 6.** *Let $P$ be a unimodal convex polygon without any parallel edges. Let $p_s$ and $p_e$ be the first and last antipodal pair's point of any vertex $p_r$ on the cyclic ordering of $P$. Then all the vertices of $P$ which are located between $p_s$ and $p_e$ are also antipodal pair's point of $p_r$.*

*Proof.* Suppose the lemma is false. Then there exists at least a veretx $p_x$ between $p_s$ and $p_e$ on the cyclic ordering of $P$, such that $p_x$ is not an antipodal pair's point for $p_r$. Since $p_{x-1}$ and $p_{x+1}$ are the antipodal pair's point of $p_r$, both $p_x p_{x-1}$ and $p_x p_{x+1}$ with their corresponding parallel lines through $p_r$ are supporting $P$, if not either $p_x$ is a reflex vertex, or $p_{x-1} p_{x+1}$ is an edge of $P$. Both gives contradiction.

□

**Lemma 7.** *Let $P$ be a convex polygon without any parallel edges. The chords which are constructed on the antipodal pairs are pairwise intersecting.*

*Proof.* Suppose $P$ is ordered clockwise. Each antipodal pair $p_a$ and $p_b$ decomposes $P$ into two (Gaussian shaped) convex chains. Suppose the lemma is false, then at least there exists another antipodal pair $p_c, p_d$, where $p_a p_b$ and $p_c p_d$ are completely disjoint. But then, either $P$ is not convex at some vertices either between $p_b$ and $p_c$, or $p_d$ and $p_a$, or $p_b p_c$ and $p_d p_a$ are two parallel sides in $P$. Both gives contradiction. □

**Lemma 8.** *Let $p_x$ be the first antipodal pairs point of $p_r$ along the boundary of the convex polygon, in clockwise order. Then, for point $p_{r+1}$, the first antipodal pair's point of $p_{r+1}$ must only be located on or after $p_x$.*

*Proof.* Suppose the lemma is false, then there exists an antipodal pair's point of $p_{r+1}$ which is located strictly before $p_x$. But then there exist two chords on antipodal pairs point of $p_r$ and $p_{r+1}$ which are not intersecting. Contradiction with Lemma 7. □

**Lemma 9.** *The number of antipodal pair's points of each pair in a convex polygon is constant in amortize.*

*Proof.* It is the case because the number of antipodal pairs in a convex polygon is linear in total. □

**Lemma 10.** *Let $\Lambda_P^4 = p_a p_b p_c p_d$ be the largest-area quadrilateral of $P$ in clockwise order. Then $p_a$ is the furthest vertex among all the vertices of $P$ from segment $p_b p_d$, and the same argument is hold for all the other vertices of $\Lambda_P^4$.*

*Proof.* If that is not the case, then there must be at least another vertex $p_x$ that is further than $p_a$ from $p_b p_d$. Then obviously $p_x$ could be substituted for $p_a$ to give a larger area quadrilateral, contradiction. □

Throughout the paper, we always assume that the vertices of any convex polygon are ordered clockwise. From the above lemma we can conclude:

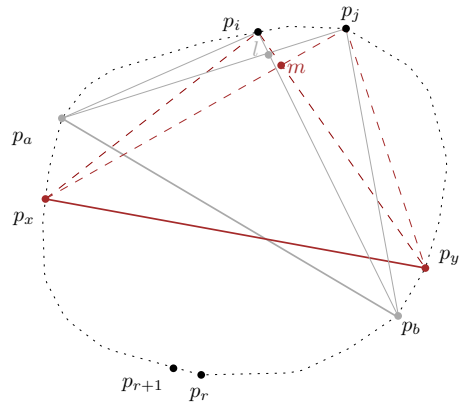


Figure 3: Illustration of Lemma 14.

**Corollary 11.** *Let $\Lambda_P^4 = p_a p_b p_c p_d$ be the largest-area quadrilateral of $P$, then each pair of vertices $p_a, p_c$, and $p_b, p_d$ admit parallel lines of support, where they are parallel to the lines passing through $p_b p_d$ and $p_a p_c$, respectively.*

**Corollary 12.** *All the vertices of any $\Lambda_P^k$ for all values of $k$ are stable.*

If that is not the case, by the same argument of Lemma 10 we still can increase the area of $\Lambda_P^4$.

**Lemma 13.** *The number of 3-stable 4-gons of a unimodal polygon is bounded by $O(n)$.*

*Proof.* Let $Q = p_a p_b p_c p_d$ be any 3-stable 4-gon in clockwise order. W.l.o.g let $p_a$ be the root. As discussed above, any 3-stable 4-gon admits parallel lines of support, so that one line is parallel to the line passing through $p_b p_d$ and supports $Q$ at $p_c$, and two lines are parallel to the line passing through $p_a p_c$ and support $Q$ at $p_b$ and $p_d$.

We consider all the 4-gons that can be rooted at $p_a$. They obviously will select the opposite vertex of $p_a$ one of those vertices that are an antipodal pairs' point of $p_a$, since otherwise we still can increase the area of $Q$ (and it would no longer be stable at that vertex).

Two other vertices $p_b$ and $p_d$ are the furthest vertices from $p_a p_c$ in both sides. These vertices may need to be updated for each antipodal's pairs point of $p_a$, however, the total number of the pairs $p_b$ , $p_d$ is bounded by the total number of antipodal pair's point of $p_a$ anyway, which can only be constant in amortizing. (Notice that after computing $p_b$ and $p_d$, $p_c$ may also need to be updated, but the substitution candidate of $p_c$ can only be chosen from $S_{p_a}$. Then again $p_b$ and $p_d$ may also move clockwise, but the whole procedure can be repeated at most $O(|S_{p_a}|)$ times which is constant in amortizing.)

Notice that no other vertex of $P$ can be a candidate for $p_b$ or $p_d$, since otherwise we still can increase the area of $Q$, and $Q$ would no longer be stable. Thus the total number of the rooted 3-stables 4-gon on each vertex $p_a$ is bounded by the number of antipodal's pairs point of $p_a$. We already know that they can at most be a linear number of antipodal pairs in a convex polygon [14] (and they all can be computed in linear time). Altogether, the total number of 3-stables 4-gons is bounded by $O(n)$. □

**Lemma 14.** *Let $p_x p_y$ and $p_a p_b$ be two chords of any two 3-stable 4-gons rooted at two consecutive roots $p_r$ and $p_{r+1}$ in a convex polygon $P$, so that $a > x$ and $b > y$. See Figure 3. If $p_j$ is further than $p_i$ from $p_x p_y$, then $p_j$ is further than $p_i$ from $p_a p_b$.*

*Proof.* Let $l = p_i p_b \cap p_j p_a$ and $m = p_j p_x \cap p_i p_y$. The triangle $p_x p_y m$ is involved in both triangles $p_x p_y p_j$ and $p_x p_y p_i$. Since $p_j$ is further than $p_i$ from $p_x p_y$, $p_x p_y p_j$ is larger than $p_x p_y p_i$. Thus $|p_i p_x| < |p_j p_y|$. Because of the stability of the 4-gons and convexity of $P$, by considering the half-planes to the left of the lines passing through each of $p_a p_b$ and $p_x p_y$, $|p_i p_a| < |p_i p_x|$, and also $|p_j p_y| < |p_j p_b|$. Since the polygon admits parallel line of supports at $p_a$ and $p_b$, for any vertex $p_z \in P$ between $p_a$ and $p_b$, the distances of all vertices of $P$ which are located between $p_z$ and $p_b$ must only be increased from $p_z$, where keep moving clockwise from $p_z$. Also since $p_j$ is further than $p_i$ from $p_x p_y$, and since $i < j$ and $a > x$, $|p_i p_a| < |p_i p_b|$.

Like above, the triangle $p_a p_b l$ is involved in both triangles $p_a p_b p_j$ and $p_a p_b p_i$. Using the similarity of triangles and knowing $|p_i p_a| < |p_i p_b|$, $p_a p_b p_i < p_a p_b p_j$. Consequently $p_j$ is further than $p_i$ from $p_a p_b$. □

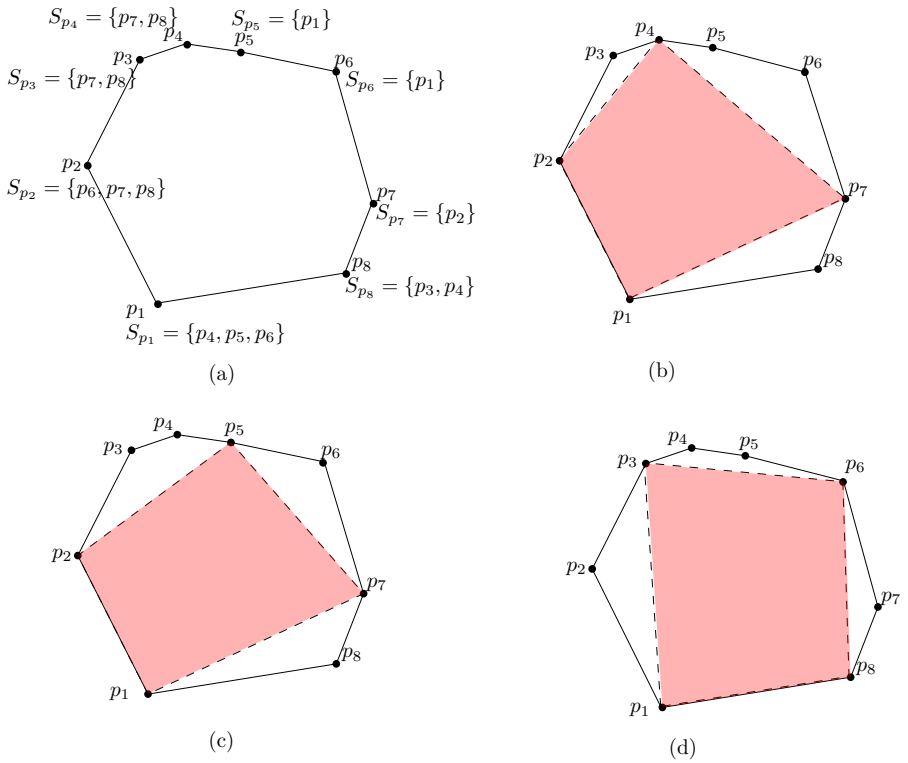


Figure 4: (a) Illustration of the definition of $S_{p_i}$. (b,c,d) The first step of Algorithm 1.

Notice that the relation between the triangles refers to their area. Also notice that for the case where $p_a$ coincides with $p_x$, or $p_y$ coincides with $p_b$, above lemma is still correct.

### 3.1 Algorithm

The idea of the algorithm is to compute all the rooted 3-stable 4-gons. The maximum area quadrilateral would be the largest one among them (which of course has four stable vertices). We first compute all the antipodal pairs of $P$ by a rotating caliper technique and memorize them. There are at most a linear number of such pairs and they all can be computed in linear time [14]. At the same time, for each point $p_i$, we make a set $S_{p_i}$ which includes the labels of the antipodal's pairs points of $p_i$ according to their clockwise order on the boundary of $P$. See Figure 4(a). Thus for each point $p_i$, we also know the antipodals' pairs points of $p_i$ start at and end with which points.

In the first step of the algorithm, we start from $p_1$ and the first vertex in $S_{p_1}$. Let $p_x$ denote this vertex. Then we find the furthest vertex of $p_1p_x$ in both sides of $p_1p_x$ in $O(\log n)$ time (using a binary search on the boundary of $P$). Let $p_b$ and $p_d$ denote them. If after computing $p_b$ and $p_d$, $p_x$ is no longer the furthest vertex from $p_bp_d$, then we move it forward among the vertices of $S_{p_1}$ to find the furthest one. Then after computing the furthest, we may need to update $p_b$ and $p_d$ again, where they can only move in clockwise order according to Lemma 14, and because the chords on antipodal pairs are pairwise intersecting (Lemma 7). But since $S_{p_1}$ has a constant size in amortizing, repeating this procedure for all the vertices of $P$ will take linear time.

Then, for $p_{x+1}$ (the second element of $S_{p_1}$ if it exists), according to Lemma 14, two furthest vertices of $P$ from $p_1p_{x+1}$ can only move in clockwise direction along the boundary of $P$, starting from $p_b$ and $p_d$ (the first position which were computed by binary search in previous step). After computing their optimal positions, we memorize them for the case where we consider $p_{x+2}$. We repeat this procedure for all the elements of the set $S_{p_1}$. Since no other vertex before $p_x$ or after the last element of $S_{p_1}$ can admit parallel lines of support with respect to $p_1$, in above procedure we can find all potential 3-stable 4-gons rooted at $p_1$. See Figure 4(b,c,d). We will remember the largest-area 4-gon rooted at $p_1$.

In the second step of the algorithm, we will repeat the above procedure for $p_2$. Then two cases can

happen, either the first element of $S_{p_2}$, $p_y$ was also included in $S_{p_1}$, or not. We first consider the first case. Let $p_y$ be the first point (in cw order) along the boundary of $P$ in $S_{p_1} \cap S_{p_2}$. In which case, the furthest vertex of chords $p_2p_y$ can only be on or after the furthest vertex of $p_1p_y$, in the clockwise order (according to Lemma 14). Since we already have computed the furthest vertex of $p_1p_y$, the furthest vertex of $p_2p_y$ can also be computed in amortize constant time.

In the second case, according to Lemma 8, $y$ is greater than the index of last element of $S_{p_1}$, and thus, according to Lemma 14, the furthest vertex of $p_2p_y$ must be located on or after the last furthest vertex which we have computed for $p_1$ and the last element of $S_{p_1}$. Let $p_z$ be the last point of $S_{p_1}$. Then obviously we can find the optimal position of the furthest vertex of $p_2p_y$ by moving clockwise along the boundary of $P$, where the start is from the furthest vertex of $p_1p_z$, which we already have computed. Thus in this case again the furthest vertex of $p_2p_y$ can also be computed in amortized constant time.

We will repeat the above procedure until we return to $p_1$, and in each iteration, we update the maximum area 4-gon, if it is necessary. Since we only wrap around the boundary of $P$ in the clockwise direction, and we never go back, the whole algorithm runs in linear time. This algorithm is outlined in Algorithm 1.

**Algorithm 1:** Quadrilateral algorithm

```
Input P: a convex unimodal polygon P = {p_1, ..., p_n}
Output Λ_P^4: the largest-area quadrilateral
Legend Operation next means the next vertex in clockwise order of P
for i = 1 to n
    S_{p_i} = all the antipodal's pair's points of p_i in clockwise order
max, i = 1
while True do
    p_x=the first element of S_{p_i}
    while there is an unprocessed p_x ∈ S_{p_i} do
        p_b, p_d=furthest neighbours of p_i p_x in bothe sides
        while each of p_x, p_b and p_d is not stable do
            if p_x is no longer the furthest from p_b p_f then
                x = x + 1 (if p_{x+1} ∈ S_{p_i})
            end
            if p_b is not the furthest vertex to the left of p_i p_x then
                update p_b by the new furthest by moving clockwise along P
            end
            if p_d is not the furthest vertex to the right of p_x p_i then
                update p_d by the new furthest by moving clockwise along P
            end
        end
        max=max(max, p_1 p_b p_x p_d)
    end
    p_i = next(p_i)
    if p_i=p_1 then
        return max
    end
end
```

## Correctness

According to Lemma 5 on each vertex $p_i$ we always can decide about the path of the algorithm. Also according to Lemma 6, we can always consider all the possible 3-stable 4-gons rooted at each vertex of the polygon. The correctness of the algorithm comes from the fact that we do not skip over 3-stable 4-gons and we indeed compute all the possible 3-stable 4-gons rooted on each vertex of $P$.

**Theorem 1.** *For any unimodal convex polygon $P$, the largest-area quadrilateral inscribed in $P$ can be computed in linear time.*

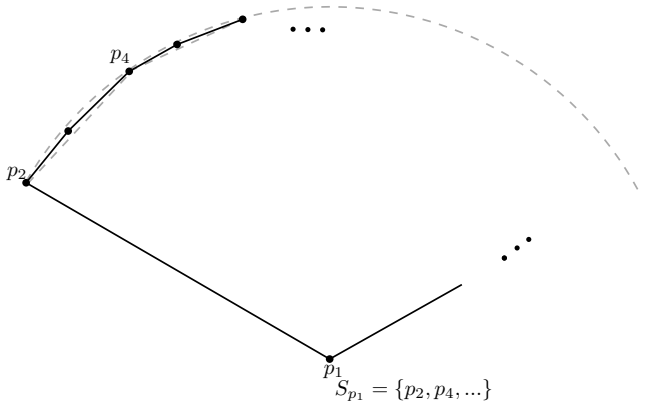


Figure 5: Illustration of the reason of not directly applicability of Algorithm 1 to a non-unimodal convex polygon [4]. In the figure, $p_1$ is located on the center of the circle, and $S_{p_1}$ includes non-consecutive vertices (it may happen for several vertices in a non-unimodal convex polygon). But since still Lemma 8 holds, we can manage the situation.

# 4 Extension to Any Convex Polygon

Regardless of the unimodality of the convex polygon, the largest-area inscribed quadrilateral has four stable vertices and admits parallel lines of support at the stable vertices. Also regardless of the unimodality, any convex polygon has only linear number antipodal pairs, however, for each vertex, the antipodal pair's points are not consecutive necessarily. It is easy to observe that Lemmas 5 and 6 break down for general convex polygons without parallel edges. But since still Lemma 8 holds, we can manage the situation. If a vertex $p_i$ does not admit an antipodal pair, we can simply omit it, since there is no vertex on the opposite side of $p_i$ to admit parallel line of support and to be a stable vertex. Also observe that Lemma 13 is still correct, since the number of the antipodal pair's point of each point is still linear in amortizing, no matter that they are not consecutive.

For a point $p_1$, (suppose it has at least an antipodal pair's point) the first step flows as before, and we still remember which points determine the first position of each of $p_b$ and $p_d$. Then for the next vertices of $S_{p_1}$ we need to move $p_b$ and $p_d$ forward in the clockwise direction, as Lemma 14 still works. At each determined 3-stable 4-gon we update the area of $\Lambda_P^4$ if it is necessary.

In the second step, let $p_y$ denote the first element of $S_{p_2}$, then according to Lemma 14 the furthest vertex of $p_2p_y$ in both sides can only be located on or after $p_b$ and $p_d$. We can find the optimal positions of furthest points of $p_2p_y$ by moving clockwise starting from $p_b$ and $p_d$. We will remember them for the next step. As before, we repeat this procedure until we return to $p_1$.

It is remained to discuss the correctness of the algorithm. Notice that at each point $p_i$ as the root, we never can have a stable vertex at a vertex which is not included in $S_{p_i}$, since they cannot admit parallel lines of support. Thus we do not skip over any 3-stable 4-gon as before, and the algorithm still works in linear time.

# 5 Parallel Algorithm

Let $P$ be a convex polygon of $n$ vertices. We design a parallel algorithm in the context of CREW PRAM model. We first provide a parallel algorithm for computing the antipodal pairs of a convex polygon. The difficulty of this problem is that the rotating caliper technique is essentially sequential.

We introduce a parallel algorithm that computes the antipodal pairs of a convex polygon in $O(\log n)$ time.

In each step of the algorithm, we compute two antipodal pairs, where the first one is anchored at an arbitrary vertex of the sub-problem. This antipodal pairs will decompose the sub-problem vertices into two contiguous intervals. The other antipodal pair is anchored at the median vertex of the largest interval. These two antipodal pairs decompose the vertices of the sub-problem into four intervals, however, according to Lemma 7, only one of the two intervals that are incident to the median vertex can contain a vertex of the next antipodal pairs in the next step, and consequently, two other sub-problems will be generated. In the last step of the recursion, only two vertices remain that we simply return. In the following lemma, we prove that this method will reduce the size of the induced sub-problems by a factor $\frac{3}{4}$.

**Lemma 15.** *Let $P'$ be a convex polygon with $m$ vertices in step (i). The two sub-problems induced by $P'$ have size at most $\frac{3}{4}(m+1)$.*

*Proof.* Omitting the antipodal pairs vertices of step (i), the interval that contains $m$ has a size at least $\frac{1}{2}(m-4)$, however only one of the sub-intervals that are incident to $m$ can contain a vertex of an antipodal pairs, otherwise the constructed antipodal pairs no longer intersecting both antipodal pairs of this sub-problem (contradicting Lemma 7). Thus in the next step we have two sub-problems with at most $m - \frac{1}{4}(m-4) = \frac{3}{4}(m+1)$ vertices. □

From Lemma 15 we write the recursion as $T(n) \leq T(\frac{3}{4}n) + O(\log n)$, which implies that $T(n)$ is $O(\log n)$.

In each step of the algorithm, we increase the number of sub-problems by a factor 2. In step (i) there may be up to $2^i$ separate sub-problems. Thus the total number of the sub-problems can be computed as $\sum_{i=1}^{\log n} 2^i$, which implies that if we allocate $P(n) = O(\frac{n}{\log n})$ processors, in $O(\log n)$ steps we can solve all the (previously computed) sub-problems in parallel, which is quite work optimal. Note that the coefficient for the $T(\frac{3}{4}n)$ is 1 because all these sub-problems were processed simultaneously. Also, from above recursion we can write the recursive equation of the required processors as $P(n) \leq 2P(\frac{3n}{4})$, which is $O(\log n)$.

The maximum of all the computed values can be reported in $O(\log n)$ time in CREW model. Thus the diameter of the convex polygon can be computed at the same time.

**Corollary 16.** *All the antipodal pairs of a convex polygon of $n$ vertices and its diameter can be computed in $O(\log n)$ time with $O(\log^2 n + s)$ work in PRAM CREW model, where $s$ is the number of the antipodal pairs.*

Then we divide the list of vertices of the polygon into $O(\log n)$ contiguous sub-lists of size $O(\frac{n}{\log n})$ each, and compute the optimal solution of each sub-list by allocating one processor as in the sequential case. One can observe that this step requires no more than $O(n)$ work.

According to [13] (Theorem 5), each 3-stable quadrilateral introduces a unique parallelogram that $P$ is contained in it, and such parallelograms would be the candidates of being the smallest area parallelograms inscribing the polygon. Consequently, The smallest area parallelogram inscribing the convex polygon can be computed at the same time.

**Theorem 2.** *The largest-area inscribed quadrilateral and the smallest area inscribing parallelogram of a convex polygon of $n$ vertices can be computed in $O(\log n)$ time and $O(n)$ operations in PRAM CREW model.*

# 6 Concluding Remarks

It is remained open whether the largest-area inscribed $k$-gon for constant values of $k$ can be computed in linear time. However, we would like to make the conjecture that it can. Also, the existence of a linear-time algorithm for the problem of computing the largest-perimeter inscribed triangle of a convex polygon, as well as the problem of computing the largest-perimeter inscribed $k$-gon for constant values of $k$, remained open.


## Acknowledgment

We would like to thank Maarten Löffler and Günter Rote for several discussions on the problem. We also would like to thank Marc van Kreveld for suggesting the approximate version of the problems.


# References


[1] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *Journal of the ACM (JACM)*, 51(4):606–635, 2004.

[2] A. Aggarwal, M. Klawe, S. Moran, P. Shor, and R. Wilber. Geometric applications of a matrix searching algorithm. In *Proceedings of the second annual symposium on Computational geometry*, pages 285–292. ACM, 1986.

[3] A. Aggarwal, D. Kravets, J. K. Park, and S. Sen. Parallel searching in generalized monge arrays. *Algorithmica*, 19(3):291–317, 1997.

[4] D. Avis, G. T. Toussaint, and B. K. Bhattacharya. On the multimodality of distances in convex polygons. *Computers & Mathematics with Applications*, 8(2):153 – 156, 1982.

[5] J. L. Bentley, F. P. Preparata, and M. G. Faust. Approximation algorithms for convex hulls. *Commun. ACM*, 25(1):64–68, Jan. 1982.

[6] J. E. Boyce, D. P. Dobkin, R. L. S. Drysdale, and L. J. Guibas. Finding extremal polygons. *SIAM Journal on Computing*, 14(1):134–147, 1985.

[7] S. Chandran and D. M. Mount. A parallel algorithm for enclosed and enclosing triangles. *International Journal of Computational Geometry & Applications*, 2(02):191–214, 1992.

[8] D. P. Dobkin and L. Snyder. On a general method for maximizing and minimizing among certain geometric problems. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pages 9–17, Oct 1979.

[9] S. Guha. Parallel computation of internal and external farthest neighbors in simple polygons. *International Journal of Computational Geometry & Applications*, 2(02):175–190, 1992.

[10] O. Hall-Holt, M. J. Katz, P. Kumar, J. S. Mitchell, and A. Sityon. Finding large sticks and potatoes in polygons. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 474–483. Society for Industrial and Applied Mathematics, 2006.

[11] V. Keikha, M. Löffler, A. Mohades, J. Urhausen, and I. van der Hoog. Maximum-area triangle in a convex polygon, revisited. *CoRR*, abs/1705.11035, 2017.

[12] G. Rote. The largest inscribed triangle and the smallest circumscribed triangle of a convex polygon: An overview of linear-time algorithms. *Class notes, June*, 2019.

[13] G. Rote. The largest quadrilateral in a convex polygon. *CoRR*, abs/1905.11203, 2019.

[14] M. I. Shamos. Computational geometry. *Ph. D. thesis, Yale University*, 1978.