社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－AL－120 (6)
2008／11／7

凸多角形に対する包含多角形列の計算

大西 建輔 *, 星 守 †

* 東海大学理学部情報数理学科
〒259-1292 神奈川県平塚市北金目 1117
E-mail: KensukeOnishi@acm.org

† 電気通信大学大学院情報システム学研究科
〒182-8585 東京都調布市調布ヶ丘 1-5-1

あらまし: 凸多角形 P に対し, P を含む k 角形の列 $\{P_k\}_{k\geq 3}$ を計算する問題を扱う. 本稿では, この問題を解く 2 つの手法を提案する. 最初の手法は, k ごとに最小面積の包含 k 角形を計算する手法で, **最適法**と呼ぶ. もう一方の手法は, 出力された k 角形にその時点での最小面積の三角形を加え, $(k-1)$ 角形を逐次的に構成する手法である. これを**貪欲法**と呼ぶ.
これら 2 つの手法を実装し, 凸多角形に対し計算機実験を行った. その結果, $k=3$ の場合を除き, 面積という観点で, 貪欲法が最適法と同程度の面積を持つ包含 k 角形列を出力することがわかった.

Computing a Sequence of Circumscribing Polygons for Convex Polygon

Kensuke Onishi*, Mamoru Hoshi†

*Department of Mathematical Sciences, Tokai University. 1119 Kitakaname, Hiratsuka, Kanagawa, 259-1292, Japan
Email: KensukeOnishi@acm.org

†Graduate School of Information Systems, The University of Electro-Communications. 1-5-1, Chofugaoka, Chofu, Tokyo, 182-8585, Japan.

Abstract: For a given convex polygon P, compute a sequence of polygons $\{P_k\}_{k\geq 3}$ circumscribing P such that each P_k with k vertices circumscribes P. For this problem we propose two algorithms: one is to compute optimal k-gons, called *optimal algorithm* and another is to compute k-gons by adding smallest triangle, called *greedy algorithm.*
We implemented these two algorithms and executed computational experiments for some convex polygons. The results show that the greedy algorithm works well from the viewpoint of area, except $k=3$ (triangle).

1 Introduction

Polygon is a fundamental object in computational geometry. Many kinds of problems about polygon have been investigated, and collected in [4].

Consider "similarity" between polygons for *retrieval of shape.* Many similarities have been already proposed [5]. For example, Hausdorff distance is often used as a similarity between polygons and its error bounds are analyzed in some cases [3]. Since to calculate these similarities we need to match polygons, or vertices, the computation of similarity is time consuming. If the load due to the matching is none, or less, the computation of similarity is fast. Boxer *et al.* proposed an algorithm to reduce edges of general polygon for approximation [2]. When the problem below is solved, the load of matching is reduced.

We investigate the following problem in this paper:

> **Problem** Given a convex polygon P with n vertices. Compute a sequence of polygons P_k with k vertices $(k=3,\ldots,n-1)$, where P_k circumscribes P.

This problem can be solved by Aggrawal's algorithm [1] for computing a minimum area circumscribing k-gon. Applying the algorithm to P, we have a sequence $\{P_k\}$ such that each k-gon P_k has a minimum area among all k-gons circumscribing P.

We propose another algorithm, called *greedy algorithm*: first, compute a $(n-1)$-gon P_{n-1} from the n-gon $P(=P_n)$ and, next compute $(n-2)$-gon P_{n-2} from the P_{n-1}. This step is repeated until we have a 3-gon. The sequence obtained by the greedy method is also an answer of the problem above. Moreover each polygon P_k of the sequence circumscribes P_{k+1} $(k=3,\ldots,n-1)$.

Algorithm 1 Optimal Algorithm

Input P : polygon with n vertices;
Output $\{P_k\}$: each P_k has k vertices, $P_k \supset P$;

for $k := 3$ **to** $n-1$ **do**
  Compute optimal k-gon by Aggrawal's algorithm.

We consider a situation where we search an object with *a special shape*, like a knife, in videos taken by some stream cameras. Currently, such a check is done by a watchdog. He has to pay attention to many monitors in his business hours.

Consider a support system for the search. The system do the following tasks on a server/servers:

1. divide the videos into many pictures;
2. check objects in each picture;
3. report to watchdog if doubtful objects are found;

If every cameras send all video data to the server, the system works very slowly, and we have to buy so expensive computers with high performance as the server. Or, we ask the cameras work harder, i.e., execute step 1 and step 2 to reduce the load on the server. Since the cameras have only low computing power, the cameras can execute only simple algorithms with small memory, like a *greedy algorithm.*

The rest of this paper is organized as follows: In section 2, we explain two algorithms: optimal and greedy. In section 3, we describe our computational experiment and its results. And we discuss the experimental results in section 4.

2 Algorithm

In this section we explain two algorithms: optimal and greedy. Their time and space complexities are analyzed.

2.1 Optimal algorithm

We explain an optimal algorithm for computing a sequence of k-gons of a given convex polygon P.

For a given P and a positive integer $k(=3, \ldots, n-1)$, a minimum area k-gon is computed in $O(kn + n\log n)$ time and $O(kn)$ space by Aggrawal's algorithm [1]. We use Aggrawal's algorithm in the optimal algorithm repeatedly (Algorithm 1).

The output convex polygon with k vertices by Aggrawal's algorithm is called *optimal k-gon*, or *optimal polygon* in this paper.

The time and space complexities of the optimal algorithm are given by those of Aggrawal's algorithm.

Theorem 1 A sequence of k-gons circumscribing P with n vertices can be computed in $O(n^3)$ time and $O(n^2)$ space by the optimal algorithm. Moreover each k-gon has minimum area among all k-gons circumscribing P.

Proof: In the optimal algorithm, we use Aggrawal's algorithm $(n-3)$ times. Since the time complexity of Aggrawal's algorithm is $O(kn + n\log n)$, the time complexity of the optimal algorithm is $\sum_{k=3}^{n-1} O(kn + n\log n) = O(n^3)$.

Since each execution is independent, used memory can be reset in each step. Then the space complexity is $O(n^2)$ in any step of the algorithm. □

2.2 Greedy algorithm

We propose a greedy algorithm for the problem. The algorithm is based on the following lemma.

Lemma 1 Let $P = (p_0, p_1, \ldots, p_{n-1})$ be a convex polygon with n vertices. Let l_i be the line through p_i, p_{i+1}. The minimum area flush[*1] $(n-1)$-gon circumscribing P is $(p_0, \ldots, p_{i-1}, q_i, p_{i+2}, \ldots, p_{n-1})$, where q_i is the cross point between two lines l_{i-1}, l_{i+1} and the area of $\triangle q_i p_{i+1} p_{i-1}$ is the smallest for $i = 0, \ldots, n-1$.

Proof: Consider four consecutive vertices p_{i-1}, p_i, p_{i+1}, p_{i+2} of P (see Figure 1). When l_{i-1} and l_{i+1} intersects at q_i in the half-plane determined by l_i and in the opposite side of P, consider a flush polygon Q_i obtained from P by replacing the points p_i, p_{i+1} with q_i. The triangle $\triangle q_i p_{i+1} p_i$ is called *added triangle* T_i. The area of Q_i is determined by the area of T_i.

Running the i from 1 to n, we have a minimum flush polygon with $(n-1)$ vertices. □

[Remark] Non-flush $(n-1)$-gons may have smaller area than flush $(n-1)$-gon. In this subsection we deal with only flush polygon.

From lemma 1, we have a greedy algorithm (Algorithm 2).

The output convex polygon with k vertices by Algorithm 2 is called *greedy k-gon*, or *greedy polygon* in this paper.

The first step in Algorithm 2 needs to $O(n)$ time and $O(n)$ space. The pairs (q_i, S_i) can be

[*1] A polygon Q is *flush* for P if every edge of Q contains a edge of P.

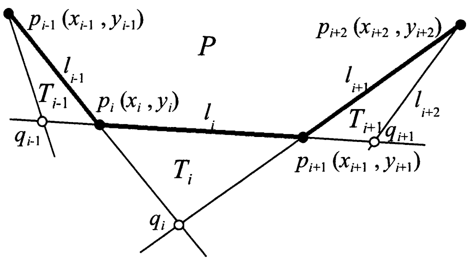

Fig. 1 cross point q_i and added triangle T_i

Algorithm 2 Greedy Algorithm

Input P : polygon with n vertices;
Output $\{P_k\}$: each P_k has k vertices, $P_k \supset P$;

1. **for** $i := 0$ **to** $n-1$ **do**
   (1) Compute the cross point q_i and the area S_i of added triangle T_i, if exists.
   (2) Add the pair (q_i, S_i) with key S_i to heap H.
2. **for** $k := n-1$ **downto** 3 **do**
   (1) Get a pair from H (S_i of the pair is smallest in H). Let i be the index of the S_i.
   (2) Output k-gon P_k computed from P_{k+1} by replacing the points p_i, p_{i+1} with q_i.
   (3) Modify (q_{i-1}, S_{i-1}), (q_{i+1}, S_{i+1}) on H.

---

managed by heap in $O(n)$ space. The construction of heap is $O(n)$ time. A modification is $O(\log n)$ time. Then, Algorithm 2 is done with $O(n \log n)$ time and $O(n)$ space except output of k-gons. In the output, we report $(n-3)$ polygons each of which has at most $(n-1)$ vertices, then we need $O(n^2)$ time.

Theorem 2 A sequence of k-gons circumscribing P with n vertices can be computed in $O(n^2)$ time and $O(n)$ space by greedy algorithm. Moreover each greedy k-gon circumscribes greedy $(k+1)$-gon.

Since P_k is obtained from P_{k+1} by replacing the points p_i, p_{i+1} with q_i, it is sufficient to output the sequence $\{q_i$ and their subscript$\}$ instead of the sequence $\{P_k\}$. So the output is only $O(n)$, the computation time for output is reduced to $O(n \log n)$.

3 Experimental results

In this section we show experimental results of optimal algorithm and greedy algorithm.
[**Implement**] We implemented two methods: optimal algorithm and greedy algorithm.

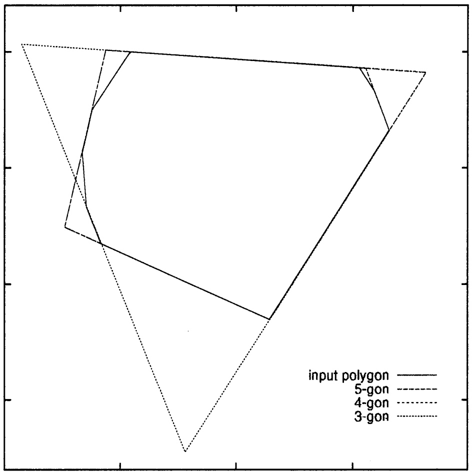

Fig. 2 Result of optimal algorithm

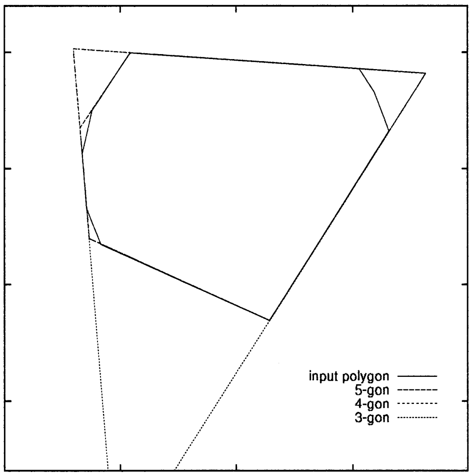

Fig. 3 Result of greedy algorithm

Figure 2 shows optimal k-gons circumscribing an input convex polygon. Figure 3 shows greedy k-gons circumscribing the input convex polygon. These k-gons are not optimal, but circumscribing $(k+1)$-gons.
[**Experiment**] We experimented on sets of polygons with two algorithms as follows.

1. fix n $(= 10, 20, 30, 40, 50)$;
2. generate ten polygons with n vertices on a circle;
3. for each polygon
   (1) compute $\{P_k\}$ by the algorithms and compute the area of P_k;
   (2) calculate the ratio of area of greedy k-gon to that of optimal k-gon;
4. compute the average of the ratio over ten polygons;

Since we generated points on a circle in the experiment, all the points are vertices of convex polygon, called *random set* in this paper. A polygon computed from random set is called *random polygon*. The input polygon in Figure 2 is a random polygon with 10 vertices.

The x-axis and the y-axis in Figure 4 are the number of vertices of output polygon and the average ratio of the area of the greedy polygon to that of optimal, respectively.

We also generated other sets of polygons for experiment. We selected j points on a circle and for each selected point p, generated new points in the vicinity of the p. The set has j clusters on a circle, and is called *skew set* in this paper. A polygon computed from the skew set is called *skew polygon*. In the experiment we used polygons with 30 vertices and $2, 3, 4, 5, 6$ clusters.

Axes in Figure 5 are the same as the axes in Figure 4.

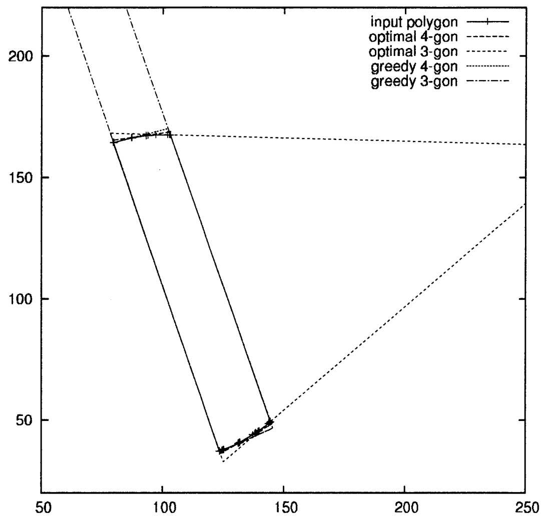

Fig. 6 Results of optimal and greedy algorithms for a skew data (2 clusters)

4 Discussion

In this section we discuss the experimental results.

Comparison of Figure 2 and Figure 3

Figure 2 and Figure 3 show the results of optimal algorithm and greedy algorithm for *one* convex polygon with 10 vertices, respectively.

Compare the optimal 5-gon and the greedy 5-gon in the figures. These 5-gons are similar except only one part and the area ratio is 1.00390. The optimal k-gon is the same as greedy k-gon for $k = 6, 7, 8, 9$. On other hand, 3-gons and 4-gons show much difference and the area ratios are 1.12228 and 1.01094, respectively.

Area ratio for random polygon

In Figure 4, we show the average area ratio of random polygon with n vertices ($n = 10, 20, 30, 40, 50$). For each n-gon, the average ratio of k-gon is almost 1.00 for $k \geq 5$. The average ratios are less than 1.07 and 1.63 for $k = 4$ and $k = 3$, respectively. The maximum ratios are less than 1.17 and 3.87 for $k = 4$ and $k = 3$, respectively. These results show that greedy k-gon is a good approximation of optimal k-gon for $k = 4, 5, \ldots, n-1$. The optimal and greedy 3-gons have much difference for many polygons.

The average area ratios of 3-gons are 1.04700, 1.34554, 1.32365, 1.38190, 1.62466 for $n = 10, 20, 30, 40, 50$, respectively. This suggests that the ratio increases with increasing number of vertices of convex polygon.

Area ratio for skew polygon

Figure 6 shows optimal and greedy k-gons ($k = 3, 4$) for a convex polygon the vertices of which cluster two regions. The greedy 4-gon is similar to the optimal 4-gon. The area ratio of 4-gon is about 1.00585, and that of 3-gon is about 3.62299, which is the worst case in the experiments for skew polygons.

The average area ratio of 3-gon are 1.46956, 1.05859, 1.04731, 1.14039, 1.17446 for the number of cluster $j = 2, 3, 4, 5, 6$, respectively. These ratios are smaller than the ratio 1.32365 of random polygon with 30 vertices (except $j = 2$).

When there are j clusters in the set, the points in a cluster are near to each other and the area of added triangle consisting of the near points is small. If the distance between clusters is sufficiently far, a cluster consisting of vertices $p_i, p_{i+1}, \ldots, p_j$ becomes an angle $\angle p_i q p_j$ where q

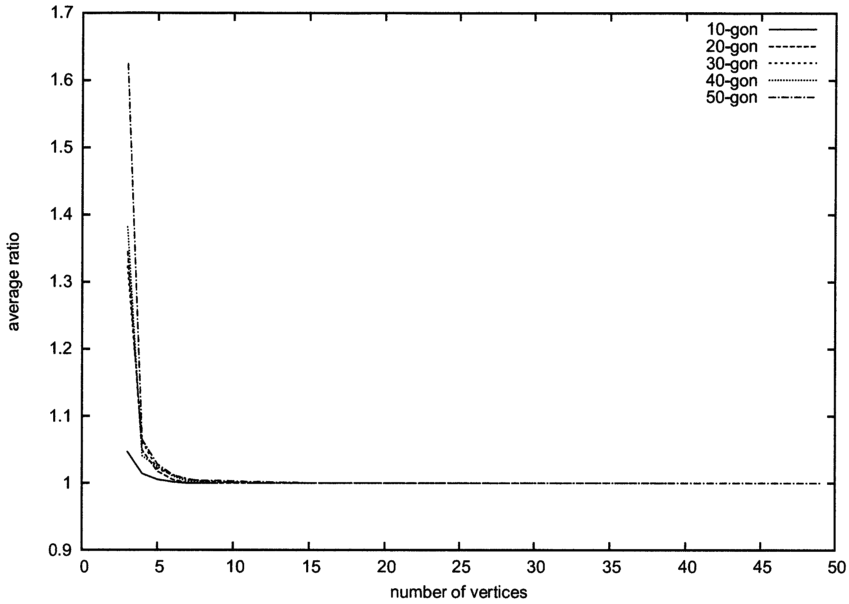

Fig. 4 Results for random polygons with n vertices ($n = 10, 20, 30, 40, 50$)

is the cross point between l_i and l_{j-1} (see Figure 7). Suppose we have an l-gon when all clusters become angles, then $p_i q$ and $q p_j$ are the edges of the l-gon. Since we have j clusters, $2j$ edges are found in the l-gon. On the other hand, there are at most j edges connecting the angles. The l-gon, therefore, has at most $2j + j = 3j$ edges, i.e., $l \leq 3j$. The l depends on the number of clusters j, not the number of vertices n. The l-gon obtained is regarded as random polygon with $l(< n)$ vertices.

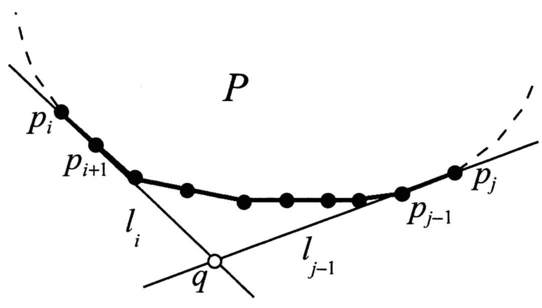

Fig. 7 a cluster $p_i, \ldots, p_j$ in a skew data and the angle $\angle p_i q p_j$

In our experiments for skew polygons the ratio is smaller than that of random 30-gons. The average and the maximum of the area ratios of 4-gon are less than 1.02 and 1.03, respectively. From the discussion above, the area ratio of skew polygon is smaller than that of random polygon.

5 Concluding remark

In this paper we dealt with the algorithms for computing a sequence of k-gons $\{P_k\}$ ($P_k \supset P$) for a given convex polygon P. We proposed two algorithms: optimal and greedy and implemented these algorithms. We experimented on two type polygons: random polygon and skew polygon. The experimental results show that greedy algorithm works as well as the optimal algorithm do except for the 3-gon.

Note that the problem of this paper can be dealt in the dual space. In the dual space, the problem is formulated as follows:

> **Problem** Given a convex polygon P with n vertices. Compute a sequence of polygons P_k with k vertices ($k = 3, \ldots n-1$), where P_k inscribes P.

This problem is also solved by optimal and greedy

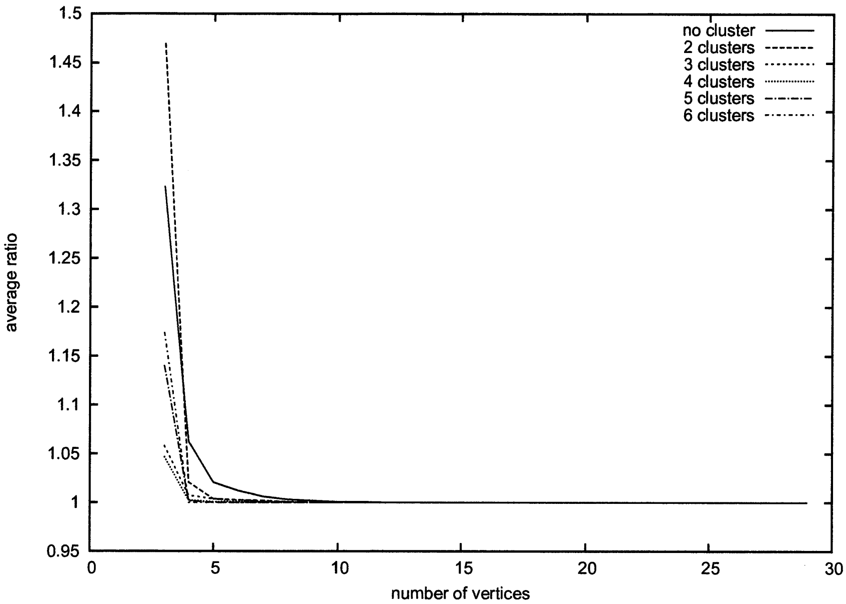

Fig. 5 Results for skew polygons with j clusters ($j = 2, 3, 4, 5, 6$) and for random polygons with 30 vertices

algorithms.

References

[1] A. Aggrawal and J.K. Pach. Note on searching in multidimensional monotone arrays. In *Proc. 29th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 479-512, 1988.

[2] L. Boxer, C.-S. Chang, R. Miller and A. Rau-Chaplin. Polygonal approximation by boundary reduction, *Pattern Rec. Letters* **14**, pages 111-119, 1993.

[3] P. Brass. On the approximation of polygons by subpolygons. In *Proc. European Workshop Comput. Geom. (EuroCG)*, pages 59-61, 2000.

[4] J.E. Goodman and J. O'Rourke ed. Handbook of Discrete and Computational Geometry, second edition, Chapman & Hall, 2004.

[5] P.M. Gruber. Approximation of Convex Bodies. In *Handbook of Convex Geometry*, Vol. A (P.M. Gruber and J.M. Wills, ed.), North-Holland, pages 319-345, 1993.

Appendix A Calculation of cross point and of area of added triangle

In this section we explain the computation of the cross point q and area of added triangle for four consecutive vertices $p_{i-1}, p_i, p_{i+1}, p_{i+2}$ on a convex polygon (see Figure 8). Let (x_i, y_i) be the coordinate of vertex p_i.

The line l_i through p_i, p_{i+1} is expressed by

$$l_i : (y_{i+1} - y_i)\,x - (x_{i+1} - x_i)\,y + x_{i+1}y_i - x_i y_{i+1} = 0$$

The coordinate of cross point q between l_{i-1} and l_{i+1} is

$$\left(\frac{\begin{array}{c} -(x_{i+2} - x_{i+1})(x_i y_{i-1} - x_{i-1} y_i) \\ +(x_i - x_{i-1})(x_{i+2} y_{i+1} - x_{i+1} y_{i+2}) \end{array}}{\begin{array}{c} (x_{i+2} - x_{i+1})(y_i - y_{i-1}) \\ -(x_i - x_{i-1})(y_{i+2} - y_{i+1}) \end{array}}, \right.$$
$$\left. \frac{\begin{array}{c} -(y_{i+2} - y_{i+1})(x_i y_{i-1} - x_{i-1} y_i) \\ +(y_i - y_{i-1})(x_{i+2} y_{i+1} - x_{i+1} y_{i+2}) \end{array}}{\begin{array}{c} (x_{i+2} - x_{i+1})(y_i - y_{i-1}) \\ -(x_i - x_{i-1})(y_{i+2} - y_{i+1}) \end{array}} \right).$$

We show a relation among areas of triangles. Let S, S_1, S_2 and S_3 be the area of the trian-

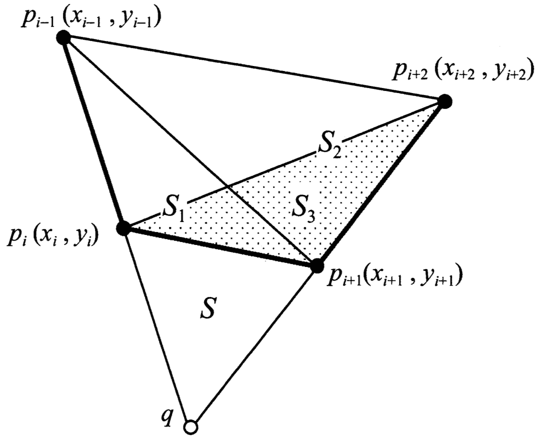

Fig. 8 coordinate of cross point q

gles $\triangle p_{i+1}p_iq$, $\triangle p_{i-1}p_ip_{i+1}$, $\triangle p_{i-1}p_{i+1}p_{i+2}$ and $\triangle p_ip_{i+1}p_{i+2}$, respectively.

Consider $\triangle p_{i+2}p_iq$. The triangle is divided into two triangles by the line segment $\overline{p_ip_{i+1}}$ and let their areas be S and S_3, respectively. Then the ratio of $\|\overline{qp_{i+1}}\|$ (the length of $\overline{qp_{i+1}}$) to $\|\overline{p_{i+1}p_{i+2}}\|$ is that of S to S_3. Similarly, the ratio of $\|\overline{qp_i}\|$ to $\|\overline{p_ip_{i-1}}\|$ is that of S to S_1. So the ratio of the area of $\triangle p_{i+2}p_{i-1}q$ to that of $\triangle p_{i+2}p_iq$ is $S/(S+S_1)$ and the ratio of area of $\triangle p_{i+1}p_iq$ to that of $\triangle p_{i+2}p_iq$ is $S/(S+S_3)$. Then, the ratio of area of $\triangle p_{i+2}p_{i-1}q$ to that of $\triangle p_{i+1}p_iq$ is

$$\frac{S}{S+S_1} \cdot \frac{S}{S+S_3}.$$

Since the area of $\triangle p_{i+2}p_{i-1}q$ is $S+S_1+S_2$, then the ratio above is equal to $S/(S+S_1+S_2)$. So, we have the following equation:

$$\frac{S}{S+S_1+S_2} = \frac{S}{S+S_1} \cdot \frac{S}{S+S_3}.$$

We solve the equation for S:

$$S = \frac{S_1 \cdot S_3}{S_2 - S_3} \; (S_2 \neq S_3).$$

When $\overline{p_{i-1}p_i}$, $\overline{p_{i+1}p_{i+2}}$ are parallel, then $S_2 = S_3$ and q does not exist. When q is in the interior of convex polygon, we have $S_2 < S_3$. So, we only compute S when q is in the outside of convex polygon, i.e., $S_2 > S_3$. Moreover, q is calculated only when S is the smallest among added triangles.