Minimum area circumscribing Polygons

Article *in* The Visual Computer · August 1985

DOI: 10.1007/BF01898354 · Source: dx.doi.org

3 authors, including:

Chee Keng Yap
New York University
261 PUBLICATIONS **6,443** CITATIONS

SEE PROFILE

The Visual Computer

Minimum area circumscribing Polygons

Alok Aggarwal[1], J.S. Chang[2] and Chee K. Yap[2]

[1] IBM T.J. Watson Center, P.O. Box 218, Yorktown Heights, New York, 10598, USA
[2] Courant Institute of Mathematical Sciences, 251 Mercer Street, New York, New York 10012, USA

We show that the smallest k-gon circumscribing a convex n-gon can be computed in $O(n^2 \log n \log k)$ time.

Key words: Geometric algorithms – Circumscribing – Optimization – Convex polygons – Minimum area

VARIOUS RESEARCHERS HAVE PROPOSED FAST algorithms for circumscribing and inscribing a given polygon with minimum or maximum convex k-gons. For instance, see [1, 3, 6, 7, 8, 9, 10, 12]. Potential applications of these algorithms include areas such as robotics [2] and computer aided design [8]. If the given polygon has n sides, then Klee and Laskowski [10] have provided an $O(n \log^2 n)$ algorithm that computes the minimum circumscribing triangle. O'Rourke et al. [12] improved this to a linear time algorithm. (Here, and throughout the paper, minimum or maximum is with respect to area.) Chang and Yap [3] used a dynamic programming approach to compute a minimum circumscribing convex k-gon in $O(n^3 \log k)$ time[1]. DePano and Aggarwal [5] restricted the problem and provided efficient algorithms for computing a minimum equiangular as well as a regular k-gon that circumscribes a given n-gon. In this paper, we improve the results of Chang and Yap and obtain an $O(n^2 \log n \log k)$ algorithm for solving this problem.

This paper has seven sections. In section II, we establish some basic concepts. In section III we present a key lemma about minimum circumscribing k-gons. This lemma is exploited in sections IV and V to solve two subproblems which together yield an $O(kn^2 \log n)$ algorithm for computing the minimum circumscribing k-gon. In section VI, we improve this to an $O(n^2 \log k \log n)$ algorithm. We conclude in section VII with some open problems.

II. Preliminaries

Throughout this paper, we will use the following notation. We consider the polygon to be a closed polygonal path. The convex n-gon to be enclosed is P. The edges of P are indexed in clockwise order as $0, \ldots, {}^{n-1}$, and we refer to these edges by their indices. Thus, we often say, "edges i, $i+1, \ldots, k$ of P." If x and y are two points of P, then (x, y) indicates the *open polygonal chain* from x to y (it is open since we omit the end points x and y). Let Q denote a k-gon enclosing P. Since we are interested in minimum area, we assume without loss of generality, that each edge of Q intersects P. Also, every edge of P or Q is assumed to consist of the open edge segment and its preceeding vertex only.

[1] Dori and Ben Bessat [8] provided a linear time algorithm for finding the minimum area circumscribing k-gon for a given n-gon. However, their optimality proof is faulty and O'Rourke [11] has provided some counterexamples

The Visual Computer (1985) 1:112–117

We say that a side s of Q is *flush* with an edge e of P (or simply s is flush with P) if $e \subseteq s$. Let e be an edge of Q and v a vertex of P. Then, we say that e *impinges* on v if (i) e touches v and (ii) either e is not a flushed edge or e is flush with the edge of P containing v. Now, clearly, every edge of Q impinges on a unique vertex of P. Following Klee and Laskowski [10], a k-gon Q that encloses P is a *local minimum* if there exists some $\varepsilon > 0$ such that the area of Q' is at least the area of Q, for every k-gon Q' that encloses P and is at a Hausdorff distance less than ε from Q'. Note that although there may be infinitely many local minima, Klee and Laskowski's ideas can be generalized to classify these minima into finitely many equivalence classes.

If the midpoint of an edge of Q touches P then this edge is called a *balanced edge*. We now quote a result of DePano [4] that will simplify the derivation of our algorithm:

Lemma 1. (DePano) *For $k \geqq 4$, there exists a globally minimum k-gon* Q *that has at least $k-1$ edges flush with* P. *Furthermore, all the edges of this k-gon are balanced.*

Define a *P-anchored* k-gon as a circumscribing k-gon which has at least $k-1$ edges flush with P. Then Lemma 1 implies that in looking for a minimum k-gon, we can restrict our search to the set of P-anchored k-gons.

Let i and j denote any two distinct edges of P and let v and v' denote the unique vertices associated with these edges. We define an *h-sided anchored* (i, j)-chain (or simply, an anchored chain, if h, i and j are understood) to be any polygonal chain $C = (e_0, e_1, e_2, \ldots, e_{h+1})$ if it satisfies the following:

(i) Edge e_0 is flushed with edge i.
(ii) Edge e_{h+1} is flushed with edge j.
(iii) For every l between 1 and h, edge e_l touches a vertex (which is unique by our convention) in the polygonal chain $(v, v') \subseteq P$.

The *extra area* of such a chain C is the area of the bounded (but perhaps disconnected) region determined by C and the chain $(v, v') \subseteq P$. The anchored chain C is *flush* if all its edges are flush with P. Let $W_h(i, j)$ denote the set of all h-sided anchored (i, j)-chains and let $W^*{}_h(i, j)$ denote the set of all h-sided flush (i, j)-chains. Then, $W^*_h(i, j) \subseteq W_h(i, j)$ and $W_h(i, j)$ is empty if there are at most $h-1$ edges between i and j. A chain in $W_h(i, j)$ is said to be *optimal* for $W_h(i, j)$ if it has the minimum extra area among all chains in $W_h(i, j)$. Let $C_h(i, j)$ and $\alpha_h(i, j)$ denote such a chain and its extra area, respectively. Similarly, let $C^*_h(i, j)$ denote an optimal flush chain in $W^*_h(i, j)$ and let its extra area be denoted by $\alpha^*_h(i, j)$. (Fig. 1.) If $i_1, \ldots, i_m$ $(m \geqq 3)$, is any sequence of edges, that occurs in the listed order, in a clockwise traversal around P, we write

$$i_1 \leqq i_2 \leqq \ldots \leqq i_m \text{ (mod P)}.$$

If all $i_1, \ldots, i_m$ are distinct, then we write $i_1 < i_2 \ldots < i_m$ (mod P). Clearly, this notion also extends to vertices: if $u_1, u_2, \ldots, u_m$ are vertices of P in the clockwise order then we write $u_1 \leqq u_2 \leqq \ldots \leqq u_m$ (mod P).

Suppose Q* is a minimum k-gon that encloses P. By DePano's lemma, we may assume that there are two edges i and j of P that touch Q* and decompose Q* into two chains C_1 and C_2 such that the following two conditions hold:

(i) C_1 is a *one*-sided chain that is optimal for $W_1(i, j)$, i.e., $C_1 = C_1(i, j)$.
(ii) C_2 is a flush $(k-3)$-sided chain that is optimal for $W^*_{k-3}(j, i)$, i.e., $C_2 = C^*_{k-3}(j, i)$.

Hence, if we compute $C_1(i, j)$ and $C^*_{k-3}(i, j)$ for all $i \neq j$, then Q* can be obtained as the polygon that achieves the minimum value:

$$\min_{i \neq j} \alpha^*_{k-3}(j, i) + \alpha_1(i, j).$$

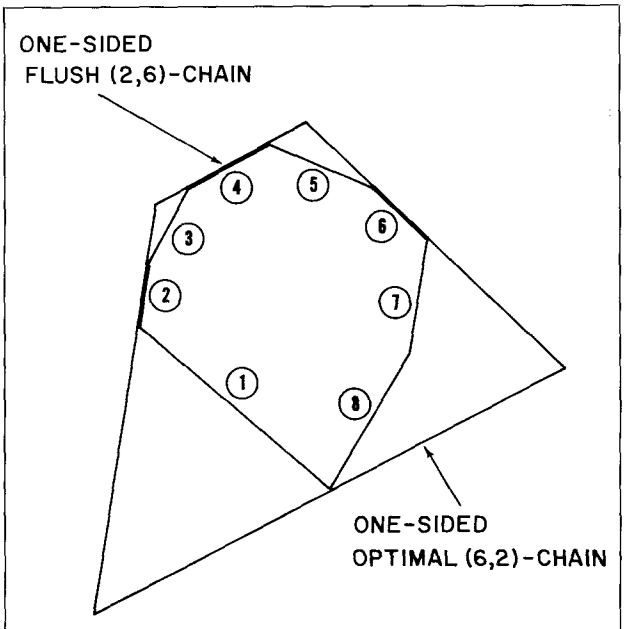

Fig. 1. Minimum circumscribing polygon

Consequently, if two problems (P1) and (P2) given below are solved, then Q* can be obtained in $O(n^2)$ additional steps. As it turns out, all our solutions to (P1) and (P2) require $\Omega(n^2)$ time, so the complexity of all our algorithms is determined by that of (P1) and (P2).

(*P1*): For all $i \neq j$, compute the optimal chain $C_1(i, j)$ and optimal area $\alpha_1(i, j)$.

(*P2*): For all $i \neq j$ compute the optimal chain $C^*_{k-3}(i, j)$ and optimal area $\alpha^*_{k-3}(i, j)$.

In the rest of this paper, we will show how to compute (P1) and (P2) efficiently.

III. The interspersing Lemma and an $O(kn^2 \log n)$ algorithm

Let $i<j<j'$ (mod P), $C \in W_h(i, j)$, and $C' \in W_h(i, j')$. Furthermore, let the h edges of C touch the vertices $u_1, \ldots, u_h$ of P in the clockwise order. Similarly, let the edges of C' touch the vertices $u'_1, \ldots, u'_h$ of P. Then, C' *intersperses* C if $u_1 \leqq u'_1 \leqq u_2 \leqq u'_2 \leqq \ldots \leqq u_h \leqq u'_h$ (mod P).

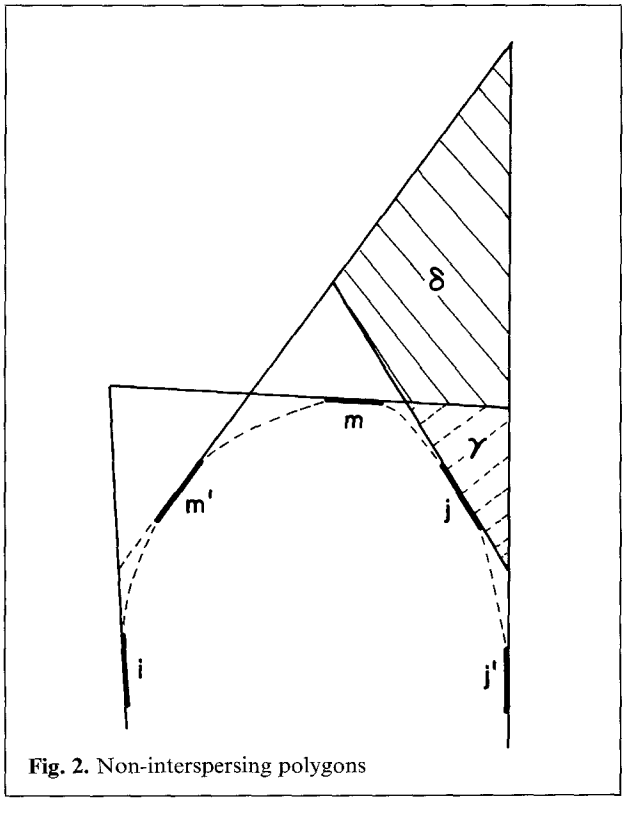

Fig. 2. Non-interspersing polygons

Lemma 2. (Interspersing Lemma). *Let* $i<j<j'$ (mod P), $C \in W_h(i, j)$, *and* $C' \in W_h(i, j')$. *If* $C = C_h(i, j)$ *and if* $C' = C_h(i, j')$ *then* C' *intersperses* C.

Proof. Let $C = C_h(i, j)$ and $C' = C_h(i, j')$ such that $i<j<j'$ (mod P). Then, using induction on h, we show that C' intersperses C.

Basis ($h=1$). Let C touch edges i, m, j of P and let C' touch edges i, m', j' of P. Note that C' intersperses C is equivalent to $i<m \leqq m'<j'$ (mod P). Assume otherwise, that $i<m'<m<j'$ (mod P). (Fig. 2.) Denote the extra areas of C' and C by $area(i, m', j')$ and $area(i, m, j)$ respectively. Let e be the edge of C that intersects (i.e. touches or overlaps) edge m and, similarly let e' be the edge of C' that intersects edge m'. Consider the unique chain C'' in $W_1(i, j')$ whose edge overlaps e and refer to its extra area as $area(i, m, j')$. Similarly, denote the extra area of the chain C''' in $W_1(i, j)$ whose edge overlaps e' by $area(i, m', j)$. (Since C''' and C'' use the edges of C and C', we say C'' and C''' are formed of the "edge interchanges" of C and C'.) Now, it is easy to see from Fig. 2:

$$area(i, m, j') = \gamma + area(i, m, j).$$

Hence

$$\begin{aligned} area(i, m, j') &< \gamma + \delta + area(i, m, j) \\ &\leqq \gamma + \delta + area(i, m', j) \\ &\quad \text{(by optimality of } C) \\ &= area(i, m', j'). \end{aligned}$$

But this contradicts the optimality of C'. This argument which is used to establish the basis will be called the "edge exchange argument".

Induction ($h>1$). Let m be the last edge of P that C intersects before j. Similarly, let m' be the last edge before j' that intersects C'. If $m \leqq m' \leqq j'$ (mod P) then by the inductive hypothesis, the $(h-1)$-sided subchain of C' from i to m' intersperses the $(h-1)$-sided subchain of C from i to m. Hence C' intersperses C as desired. Now, assume $m' < m \leqq j'$ (mod P) and note that an argument similar to the one used for the basis case, can be used to complete the proof. (The argument in the basis case essentially goes through word for word.) □

Observe that the proof of the interspersing lemma is valid even if $W^*_h(i, j)$ is used instead of $W_h(i, j)$:

Lemma 3. *If $C_h^*(i,j)$ is optimal for $W_h^*(i,j')$, $C_h^*(i,j')$ is optimal for $W_h^*(i,j)$, and $i<j<j'$* (mod P) *then $C_h^*(i,j')$ intersperses $C_h^*(i,j)$.*

In fact, the proof is valid for any subsets $\tilde{W}_h(i,j') \subseteq W_h(i,j')$ and $\tilde{W}_h(i,j) \subseteq W_h(i,j)$ which are closed under "edge interchange operations." It is perhaps possible to exploit this to obtain more efficient algorithms. However, since we will not use this observation, we omit its formulation. In O'Rourke et al. [12] there is another form of interspersing lemma for minimum triangles enclosing a polygon. The next two sections exploit our interspersing lemma.

IV. Computing optimal one sided chains

We consider the problem of computing $O(n^2)$ values

$$\{M_1(i,j), \alpha_1(i,j): i \neq j\}$$

where $M_1(i,j)$ is the unique edge of P that intersects $C_1(i,j)$ and lies between $i+1$ and $j-1$. We can view (P1) as the problem of computing the entries of two $n \times n$ arrays α_1 and M_1 such that $\alpha_1(i,j)$ and $M_1(i,j)$ represent the (i,j)-th entries of these two arrays. If $|i-j| \leqq 1$, we conventionally let $W_1(i,j) = \Phi$ (some special symbol) and assume $\alpha_1(i,j) = \infty$. Now, to solve (P1) in $O(n^2 \log n)$ time, it suffices to show that the i-th row of both arrays can be computed in $O(n \log n)$ time.

Let i be fixed in the following discussion. Suppose for some j' and j'' where $i<j'<j''$ (mod P), we want to fill in the entries:

$$\{M_1(i,j), \alpha_1(i,j): j = j'+1, \ldots, j''-1\}.$$

Inductively, assume that $\alpha_1(i,j')$, $M_1(i,j')$, $\alpha_1(i,j'')$, and $M_1(i,j'')$ have already been computed. (Initially, we choose $j' = i-1$ (mod n) and $j'' = i-1$ (mod n) so that $M_1(i,i+1)$, $M_1(i,i-1)$, $\alpha_1(i,i+1)$ and $\alpha_1(i,i-1)$ are computed in $O(n)$ time.) Let r denote the number of edges of P between $M_1(i,j')$ and $M_1(i,j'')$ and let $T(j',j'',r)$ denote the time spent in computing the entries

$$\{M_1(i,j), \alpha_1(i,j): j = j'+1, \ldots, j''-1\}.$$

Also, let $j_1 = \lfloor (j'+j'')/2 \rfloor$. Then, we can compute $M_1(i,j_1)$ in $O(r)$ time as follows: By the interspersing lemma, we note that $M_1(i,j_1)$ lies in between $M_1(i,j')$ and $M_1(i,j'')$. For each edge e among the r edges between $M_1(i,j')$ and $M_1(i,j'')$, we can determine in $O(1)$ time if the balanced one-sided chain the intersects e exists, and, if so, compute its extra area. The maximum of these extra areas is $\alpha_1(i,j_1)$, hence $M_1(i,j_1)$ can also be determined. Consequently, the recursion for computing the entries from j' to j'' in the i-th row is given by:

$$T(j',j'',r) \leqq T(j',j_1,r_1) + T(j_1,j'',r-r_1) + O(r)$$

where r_1 is the number of edges between $M_1(i,j')$ and $M_1(i,j_1)$. If there are s edges from $j'+1$ to $j''-1$ then it is easy to verify that $T(j',j'',r) = O(r \log s)$. Since r and s are both less than n, this proves our claim that $O(n \log n)$ time is sufficient to compute all the entries of the i-th row of α_1 and M_1.

V. Computing optimal flush chains

In considering optimal flush chains of length h, where $h = 1, \ldots, k-3$, we define $M_h^*(i,j)$ to be the *middle* edge of the chain $C_h^*(i,j)$, so that if the edges of $C_h^*(i,j)$ are $e_1, \ldots, e_h$ then $M_h^*(i,j)$ is the edge $e_{\lfloor h/2 \rfloor}$. Again, we regard M_h^* as an $n \times n$ array with $M_h^*(i,j) = \Phi$ whenever there are at most $h-1$ edges in the polygonal chain from i to j. Observe that if the arrays $M_1^*, M_2^*, \ldots, M_h^*$ are available then the optimal chain $C_h^*(i,j)$ can be easily reconstructed for each $i \neq j$. Hence, we can view (P2) as the problem of computing the entries of $n \times n$ arrays α_h^* and M_h^* for all h between 1 and $k-3$. To show that (P2) can be solved in $O(kn^2 \log n)$ time, it suffices to show that for any given h, arrays $M_h^*(i,j)$ and $\alpha_h^*(i,j)$ can be computed in $O(n^2 \log n)$ time. Our procedure for computing these arrays is similar to the one given for (P1); using the interspersing lemma, we show below that each row of these two arrays can be filled in $O(n \log n)$ time.

Let $h \geqq 2$ be fixed and assume inductively that the arrays M_l^* and α_l^* for $l = 1, \ldots, h-1$ have been computed. (To begin the induction on h, we can use the method of the previous section and compute the arrays M_1^* and α_1^* in $O(n^2 \log n)$ time.) Also, let i be fixed in the following discussion and suppose for some j' and j'', where $i<j'<j''$ (mod P), we want to fill in the entries:

$$\{M_h^*(i,j), \alpha_h^*(i,j): j = j'+1, \ldots, j'-1\}.$$

Assume inductively that $\alpha_h^*(i,j')$, $M_h^*(i,j')$, $\alpha_h^*(i,j'')$, and $M_h^*(i,j'')$ have already been computed. (Initially, we choose $j' = i+1$ (mod n) and $j'' = i-1$ (mod n) so that $M_h^*(i,i+1)$, $M_h^*(i,i-1)$,

$\alpha_h^*(i, i+1)$ and $\alpha_h^*(i, i-1)$ are computed in $O(n)$ time.) Let r denote the number of edges of P which are in between $M_h^*(i, j')$ and $M_h^*(i, j'')$ and let $j_1 = \lfloor (j'+j'')/2 \rfloor$. Furthermore, let $T(j', j'', r)$ be the time spent in computing the entries

$$\{M_h^*(i, j), \alpha_h^*(i, j): j = j'+1, \ldots, j''-1\}.$$

The interspersing lemma tells us that $M_h^*(i, j_1)$ lies between $M_h^*(i, j')$ and $M_h^*(i, j'')$. For each candidate edge e between $M_h^*(i, j')$ and $M_h^*(i, j'')$, we can compute in $O(1)$ time the value

$$\alpha^*_{\lceil (h-1)/2 \rceil}(i, e) + \alpha^*_{\lceil (h-1)/2 \rceil}(e, j_1)$$

since the arrays α_l^* for $l < h$ are available. The minimum of these values is $\alpha_h^*(i, j_1)$. Hence, $M_h^*(i, j_1)$ can be computed in $O(r)$ time steps. The recursion for computing the entries from $j'+1$ to $j''-1$ in the i-th row is given by:

$$T(j', j'', r) \leqq T(j', j_1, r_2) + T(j_1, j'', r - r_1) + O(r)$$

where r_1 is the number of edges between $M_h^*(i, j')$ and $M_h^*(i, j_1)$. If there are s edges from $j'+1$ to $j''-1$, then it is easy to verify that $T(j', j'', r) = O(r \log s)$. Since r and s are both less than n, $O(n \log n)$ time is sufficient to compute all the entries of the i-th row of α_h^* and M_h^*. This concludes our proof that problem (P2) can be computed in $O(kn^2 \log n)$ time.

VI. The $O(n^2 \log k \log n)$ algorithm

We will improve the previous solution by using an observation that was first used by Chang and Yap [3]. For all h between 1 and $k-3$, the above procedure computed all h-sided optimal flush chains. However, we now show that to compute the arrays α_{k-3}^* and M_{k-3}^*, we only need to compute α_h and M_h for at most $2 \log k$ values of h. To show this, it is sufficient to prove that for any $h \geqq 2$, we can compute the arrays

$$M_h^*, M_{h+1}^*, \alpha_h^*, \alpha_{h+1}^*$$

in $O(n^2 \log n)$ time from the entries of

$$M_m^*, M_{m+1}^*, \alpha_m^*, \alpha_{m+1}^* \qquad (1)$$

where m equals $\lceil (h-1)/2 \rceil$. The method of section V clearly shows that M_h^*, α_h^* can be obtained from (1). To see that M_{h+1}^*, α_{h+1}^* can also be obtained from (1), we observe that

$$\{\lceil h/2 \rceil, \lfloor h/2 \rfloor\} \subseteq \{m, m+1\}.$$

This concludes our demonstration of the main result:

Theorem. *For any given n-gon P, the minimum area circumscribing k-gon can be computed in $O(n^2 \log k \log n)$ time.*

VII. Concluding remarks

We have presented a geometric characterization of the minimum area k-gons which enabled us to demonstrate an $O(n^2 \log n \log k)$ algorithm for solving the same problem. However, the optimality of our algorithm remains to be determined. In fact, it seems that to achieve a subquadratic algorithm even for a minimum quadrilateral, some new geometric properties would be required.

The algorithm as presented in this paper depends on Lemma 1 [4]. However, it can be shown that the same time complexity can also be obtained from the following weakening of DePano's result: There exists a globally minimum k-gon that encloses a given n-gon P such that all its sides are balanced and at least $(k-2)$ sides are flush with P. This result can be deduced from the lemmas in [3]. It is not hard to further show that the two possibly non-flushed edges (called "V-edges" in [3] and "contracting edges" in [4] must be adjacent to each other. Hence this reduces the problem to computing $C_{k-4}^*(i, j)$ and $C_2^*(i, j)$ for $i \neq j$ so that in addition to the steps of the above algorithm, we also need to compute the two-sided optimal (possibly non-flush) chains. Using the interspersing lemma, this latter computation can also be achieved in $O(n^2 \log n)$ time and the modified algorithm still takes $O(n^2 \log n \log k)$ time.

Acknowledgements. The first author thanks Joseph O'Rourke and Adlai DePano of the Johns Hopkins University for many useful discussions.

References

1. Boyce JE, Dobkin DP, Drysdale RL, Guibas LJ (1985) Finding extremal polygons. SIAM J Comp 14:134–147
2. Chazelle BM (1985) Approximation and decomposition of shapes, In: Yap CK, Schwartz J (eds) Advances in Robotics. Vol 1. Lawrence O'Erlbaum Inc.,
3. Chang JS, Yap CK (1984) A polynomial solution for potato-peeling and other polygon inclusion and enclosure problems. Proc. of the 25-th Annual Symposium on the Foundations of Computer Science. pp 408–419

4. DePano A (1984) On k-envelopes and shared edges. Technical Report. Department of Electric Engineering and Computer Science. The Johns Hopkins University
5. DePano A, Aggarwal A (1984) Finding restricted K-envelopes for convex polygons. Proc. 22nd Allerton Conf. on Comm. Control and Computing
6. Dobkin DP, Snyder L (1979) On a general method for maximizing and minimizing among certain geometric problems. Proceedings of the Twentieth Annual Symposium on the Foundations of Computer Science. pp 7–19
7. Dobkin DP, Drysdale RL, Guibas LJ (1983) Finding smallest polygons In: Preparata FP (ed) Advances in Computing Research. Jai Press. pp 181–214
8. Dori D, Ben-Bessat M (1983) Circumscribing a convex polygon with polygon of fewer sides with minimal area addition. Comput Vision Graph Image Proc 24:131–159
9. Freeman H, Shapira R (1975) Determining the minimum area encasing a package. CACM 18:409–413
10. Klee V, Laskowski MC (1985) Finding the smallest triangles containing the given polygon' J Algorithms
11. O'Rourke J (1984) Counterexamples to a minimal circumscription algorithm. Manuscript, The Johns Hopkins University
12. O'Rourke J, Aggarwal A, Madilla S, Baldwin M (1985) An optimal algorithm for finding minimal enclosing triangles. J Algorithms