An Implementation of a Linear Time Algorithm for Computing the Minimum Perimeter Triangle Enclosing a Convex Polygon

Anna Medvedeva
School of Computer Science
University of Windsor
Windsor, Canada

Asish Mukhopadhyay
School of Computer Science
University of Windsor
Windsor, Canada

Abstract

In this paper, we discuss an efficient and robust implementation of a linear time algorithm due to [1] for computing the minimum perimeter triangle that circumscribes a convex n-gon. Our implementation is in C++, and utilizes the OpenGL graphics library for visualization and animation. The proposed implementation is efficient in the sense that it complies with the algorithm's linear time complexity while achieving a small constant factor. The implementation is robust in the sense that it will work for all input instances.

1 Introduction

The problem of computing a minimum perimeter triangle enclosing a convex n-gon in linear time has been a long-standing problem in the field of geometric optimization. De Pano [6] proposed an $O(n^3)$ algorithm for this problem. Later, it was improved to $O(n^2)$ by Chang and Yap [5]. Aggarwal and Park [2] used the powerful matrix searching technique to improve the complexity to $O(n \log n)$. Recently, Bhattacharya and Mukhopadhyay [1] proposed a linear time algorithm for solving the minimum perimeter triangle problem. In this paper, we describe an implementation of this algorithm.

2 Overview of the Algorithm

It was established in [6] that a minimum perimeter triangle that circumscribes a convex polygon (polygon, for short) is flush with at least one of its edges.

The main (and simple) idea of the algorithm is to consider each edge in turn and compute a minimum perimeter triangle that is flush with this edge. This computation is built on a novel scheme of *circle-fitting* and *wedge-flipping*. The former consists of fitting the smallest circle into a wedge that contains the given polygon as shown in Fig. 1. Once such a circle is determined, we also have a new wedge that contains the polygon as shown in Fig. 2, and we repeat the previous step with this new wedge. This is *wedge-flipping*.

It has been proved in [1] that in each iteration, we reduce the perimeter of the enclosing triangle. We stop when we obtain an enclosing triangle $\triangle ABC$ such that $BP = CQ$, where P is the point where the circle fitted into the wedge $W(CA, CB)$ touches AB, and Q is the point where the circle fitted into the wedge $W(BA, BC)$ touches AC (See Fig. 3).

The circle-fitting procedure makes use of a solution to the following basic problem.

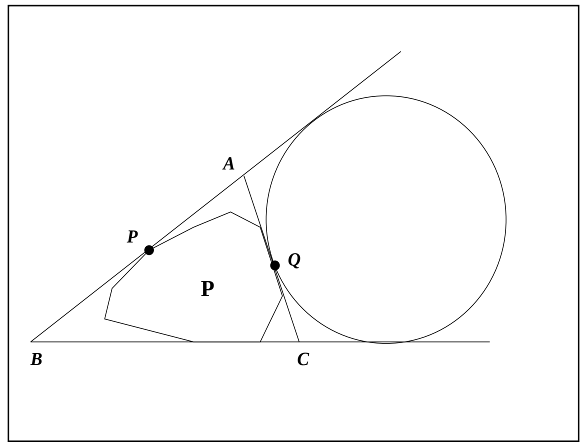

Figure 1: Fitting a circle into a wedge from the right

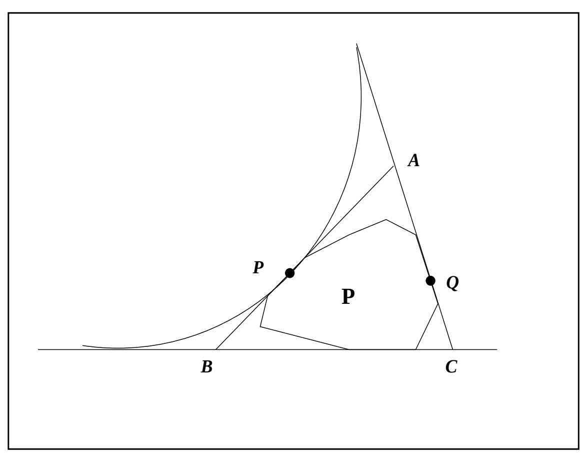

Figure 2: Fitting a circle into a wedge from the left

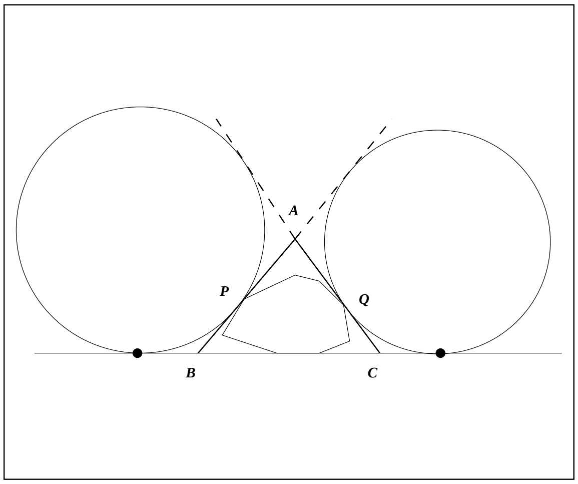

Figure 3: A minimum perimeter configuration for a given polygon edge

Problem 1 *Given a wedge $W(BA, BC)$ and a point Q contained in it, find a triangle of minimum perimeter that has two of its sides along the arms of the wedge and the third side incident on the given point Q.*

When the circle-fitting and wedge-flipping procedure terminates, we have a triangle of minimum perimeter. The proof of the former makes fundamental use of the fact that the solution to the following problem (Problem 2) is a unique one and can also be found by circle-fitting and wedge-flipping.

Problem 2 *Given two points P and Q at heights h_P and h_Q respectively, $h_P \geq h_Q$, above a line L, P to the left of Q, find a triangle $\triangle ABC$ of minimum perimeter such that the side BC is incident on L, while the points P and Q are interior to the sides AB and AC respectively.*

Finally, the linearity of the algorithm crucially depends on the fact that the following *left-interspersing lemma* holds. Its chief implication is that we do not have to backtrack as we move from one edge to the next in an anticlockwise order.

Lemma 1 *The search for a minimum perimeter circumscribing triangle never backtracks as we traverse the polygon $\mathcal{P}$ in an anticlockwise order.*

The polygon needs to be traversed in a clockwise order too, as shown in the paper [1], and a similar *right-interspersing lemma* underlies the linearity of this search.

3 The Implementation

3.1 Details and Efficiency Issues

Our implementation follows the algorithm closely. We find a minimum perimeter triangle for each edge of an n-gon. The polygon edges are considered in clockwise and anticlockwise order with both implementations available to the user. One of the sides in the minimum enclosing triangle is always flush with the corresponding polygon edge, a property proved by De Pano [6] and utilized in the implementation. The remaining two triangle sides are found by using the wedge-switching and circle-fitting technique, described in the previous section.

The algorithm uses several geometric operations as it computes the enclosing triangles. In particular, it relies on the operations such as finding an antipodal point, finding an excircle for a triangle, computing a point of tangency of a circle and a line, finding a circle inscribed in a wedge, computing a cross point of two lines, and so on. To achieve maximum efficiency for our implementation, we mathematically derived optimal solutions for all operations specified or implied by the algorithm. All geometric primitives were determined through the use of Euclidian orthogonal coordinates.

Once the code for all geometric operations is written, we follow the algorithm in invoking the necessary functions to accomplish the tasks specified by the algorithm. As we compute the geometric primitives during the progression of our algorithm (intersection points, antipodal lines, inscribed circles, etc.), we graphically display the intermediate results using OpenGL.

3.2 Robustness Issues

All coordinates and geometric equation coefficients for this implementation are floating-point numbers. Since we use finite-precision arithmetic, all comparison tests are carried out with respect to an input precision. For instance, for many algorithmic operations, the program computes lines through their algebraic equations, and checks whether two lines are parallel. In this case, we compare the floating-point coefficients of the line equations not exactly, but rather with certain precision, specified by the user. This precision identifies an acceptable error limit beyond which the values being compared are considered indistinguishable. Accordingly, if the absolute value of the difference is greater than the specified precision, the values are considered distinct. Our typical precisions used for most calculations are 0.1 and 0.01.

The same precision argument applies to the key comparison operation in the algorithm, which determines the number of wedge-switching and circle-fitting iterations necessary to either terminate the iterations and output the minimum perimeter triangle for the edge under consideration (when $BP = CQ$) or conclude that the current edge cannot generate a minimum perimeter triangle (when the height of point P becomes less than that of point Q).

3.3 Experimental Results

The key factor in characterizing the efficiency of our implementation is the number of iterations computed for each polygon edge in the process of finding an enclosing riangle of a minimum perimeter corresponding to that edge. We performed the following experiments in order to empirically determine whether there is any dependency of the number of such iterations on n. We executed our program for various arbitrary n-gons, where n varied from 4 to 30. Each polygon was specified by the floating-point Euclidian coordinates of its vertices. Table 1 shows our results, where "Average" and "Minimum" have the following meaning. We recorded the average of the number of iterations, "Average", that were needed to compute a minimum perimeter triangle for each polygon edge (the average was computed over all polygon edges). We also recorded the number of iterations, "Minimum", that corresponded to the edge that produced the minimum perimeter triangle for the polygon. We performed both, anticlockwise and clockwise searches of the polygon edges. The averages and the minimum values were computed for both of the edge searches. The number of polygon vertices n and the polygon vertex coordinates were chosen arbitrarily.

Our experiments indicate that the number of iterations necessary to compute a minimum perimeter triangle corresponding to a polygon edge is at most a fraction of the number of edges n, and the overall number of iterations for all edges of the polygon is on the order of n. This result agrees with the algorithm's linear time complexity and shows that this complexity can be achieved experimentally. The number of iterations is determined by the geometry of the input polygon and by the desired calculation precision.

Our results also show that the algorithm often finds the same minimum perimeter triangle whether we consider the polygon edges in anticlockwise or clockwise order. When both searches are performed, the resulting enclosing triangle is not dependent on whether we first consider the polygon edges in clockwise or anticlockwise order. The number of iterations may slightly differ for anticlockwise and clockwise solutions, which is particularly true for polygons with no geometric symmetry. We have also observed that the final minimum perimeter triangle for the polygon is often (in our experiments) flush with the longest polygon side, particularly for larger n. Finally, we have demonstrated that the resulting minimum perimeter triangle for the polygon is not dependent on our choice of precision.

n	Anticlockwise Search				Clockwise Search			
	Average		Minimum		Average		Minimum	
	0.1	0.01	0.1	0.01	0.1	0.01	0.1	0.01
4	2.75	2.75	2	2	2.00	2.00	2	2
4	1.75	1.75	2	2	1.75	1.75	2	2
4	11.0	14.0	11	14	11.0	14.0	11	14
5	3.00	3.00	3	3	5.00	5.60	3	3
5	4.40	5.40	2	2	3.40	4.20	2	2
5	2.40	2.40	3	3	3.20	3.20	6	6
8	3.25	3.25	3	3	3.25	3.25	4	4
8	2.88	2.88	3	3	6.00	6.88	4	4
8	6.50	8.00	11	14	3.38	3.63	10	12
10	2.90	2.90	3	3	4.20	4.50	3	3
10	5.30	6.30	4	4	4.60	5.40	4	4
10	4.20	4.80	9	12	3.80	4.00	8	10
13	4.08	4.31	3	3	4.08	4.08	4	4
13	3.31	3.69	3	3	4.08	4.77	4	4
13	4.62	5.08	4	4	5.31	6.15	3	3
15	4.33	4.93	3	3	4.93	5.67	4	4
15	4.67	5.20	5	8	4.93	5.60	7	7
15	5.67	6.67	9	12	5.00	5.67	10	13
18	4.56	5.22	5	5	3.44	3.78	4	4
18	5.89	6.83	4	4	4.17	4.33	3	3
18	4.22	4.56	6	6	3.06	3.22	2	2
20	5.20	5.60	4	4	4.95	5.45	3	3
20	4.30	4.60	3	3	3.95	4.25	4	4
20	4.65	5.00	4	4	4.95	5.65	3	3
23	3.91	4.43	3	3	4.57	4.91	4	4
23	4.52	5.17	4	4	4.43	4.91	4	4
23	4.57	5.13	3	3	4.61	5.17	4	4
25	4.92	6.32	3	3	4.96	5.96	3	3
25	4.88	5.72	3	3	3.76	4.16	4	4
25	5.76	6.52	4	4	4.20	4.76	4	4
28	5.07	5.79	9	11	4.29	4.75	6	7
28	5.32	6.04	3	3	4.75	5.25	4	4
28	4.54	4.75	6	6	5.79	6.43	4	4
30	5.07	5.77	4	4	4.80	5.37	4	4
30	5.13	5.83	11	14	5.43	6.17	10	13
30	5.03	5.60	10	13	5.37	6.13	4	4

Table 1: Experimental results for different n-gons

4 Conclusion

We have given an implementation of a linear time algorithm for computing the minimum perimeter triangle enclosing a convex polygon. This implementation has been shown to be reliable for various input instances. Asymptotically, our implementation is more efficient than previous solutions for accomplishing the same task. Our program also achieves flexibility by allowing the user to specify the precision to which arithmetic comparison tests are to be performed. It appears that the precision factor determines the number of iterations needed for every polygon edge while computing its corresponding minimum perimeter triangle candidate, if one exists. Thus, our implementation controls its own constant factor by means of specifying the precision variable. We have demonstrated that the average number of iterations is small and rarely exceeds 10 for precisions of 0.1 and 0.01 and for polygons with up to 30 edges. This once again illustrates that the number of iterations for each edge is a fraction of n with $O(n)$ iterations for all polygon edges combined. We have shown that the resulting minimum perimeter triangle is not dependent on the order in which the polygon edges are considered, being it clockwise or anticlockwise search first. Both searches often produce the same results; however, in general, it is necessary to perform both searches. Finally, we have demonstrated that the resulting minimum perimeter triangle is not dependent on our choice of the precision value.

Our use of OpenGL makes the implementation graphical and interactive.

References

[1] B. Bhattacharya and A. Mukhopadhyay, *On minimum perimeter triangle enclosing a convex polygon*, Japan Conference on Discrete and Computational Geometry, December 2002.

[2] A. Aggarwal and J. K. Park, *Notes on searching in multi-dimensional monotone arrays*, FOCS, pp. 497–512, 1988.

[3] P. K. Agarwal and M. Sharir, *Efficient Algorithms for geometric optimization*, ACM Computing Surveys, Vol. 30, pp. 412–458, 1998.

[4] J. E. Boyce, D. P. Dobkin, R. L. Drysdale, and L. Guibas, *Finding extremal polygons*, SIAM J. Comput., Vol. 14, pp. 134–147, 1985.

[5] J. S. Chang and C. K. Yap, *A polynomial solution for potato-peeling and other polygon inclusion and enclosure problems*, In 25th Annual Sym posium on Foundations of Computer Science, IEEE, Singer Island, Florida, pp. 408–416, 24–26 October 1984.

[6] N. A. A. De Pano, *Polygon approximation with optimized polygonal enclosures: applications and algorithms*, Ph. D. Thesis, 1987.

[7] D. P. Dobkin and L. Snyder,*On a general method for maximizing and minimizing among certain geometric problems*, Proc. IEEE Symp. FOCS, pp. 9–17, 1979.

[8] D. Dori and M. Ben-Bassat, *Circumscribing a convex polygon by a polygon of fewer sides with minimal area addition*, Computer Vision Graphics and Image Processing, Vol. 24, pp. 131–159, 1983.

[9] H. Freeman and R. Shapira, *Determining the minimum area enclosing rectangle for an arbitrary closed curve*, CACM, Vol. 18, pp. 409–413, 1975.

[10] P. M. Gruber, *Approximation of convex bodies*, In Convexity and its Applications, Editor P. M. Gruber, Birkhauser, 1983.

[11] V. Klee and M. C. Laskowski, *Finding the smallest triangles containing a given convex polygon*, J. Algorithms, Vol. 6, pp. 359–375, 1985.

[12] J. O'Rourke, A. Aggarwal, S. Madilla, and M. Baldwin, *An optimal algorithm for finding minimal enclosing triangles*, Journal of Algorithms, Vol. 7, pp. 258–269, 1986.

[13] A. W. Roberts and D. E. Varberg, *Convex functions*, Academic Press, 1973.

[14] G. T. Toussaint, *Solving geometric problems with the "rotating calipers"*, Proceedings IEEE MELECON '83, Athens, Greece, 1983.