International Journal of Computational Geometry & Applications

COMPUTING THE CENTER OF AREA OF A CONVEX POLYGON*

Peter Braß
Department of Computer Science R8/206, City College CUNY, 138th Street at Convent Ave.
New York NY 10031, USA
peter@cs.ccny.cuny.edu

Laura Heinrich-Litan
Institut für Informatik, Freie Universität Berlin
D-14195 Berlin, Germany
litan@inf.fu-berlin.de

Pat Morin
School of Computer Science, Carleton University, 1125 Colonel By Drive
Ottawa, Ontario, K1S 5B6, Canada
morin@cs.carleton.ca

Received September 18, 2002
Revised July 25, 2003
Communicated by Pankaj K. Agarwal

The center of area of a convex planar set X is the point p for which the minimum area of X intersected by any halfplane containing p is maximized. We describe a simple randomized linear-time algorithm for computing the center of area of a convex n-gon.

Keywords: Geometric optimization; center of area, Tukey center

1. Introduction

Let X be a convex planar set with unit area. The *center of area* of X is a point p^* that maximizes the *cut off area function*

$$f(p) = \min\{\text{area}(h \cap X) \mid h \text{ is a halfplane that contains } p\} \ ,$$

and the value $\delta^* = f(p^*)$ is known as *Winternitz's measure of symmetry.*[14] The δ-level Γ_δ of X is defined as

$$\Gamma_\delta = \{p \mid f(p) = \delta\} \ .$$

*This research was partly funded by Deutsche Forschungsgemeinschaft (DFG), grant BR 1465/5-2, and by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract Number IST-2000-26473 (ECG - Effective Computational Geometry for Curves and Surfaces).

It is known that Γ_δ is a closed convex curve and that Γ_{δ_1} is strictly contained in Γ_{δ_2} if $\delta_1 > \delta_2$. From this it follows that p^* is unique.

There is a long history of work on the center of area of convex sets. A classical result of Winternitz,[3] which has been rediscovered many times,[12,16,18,19,21] is that $f(g) \geq 4/9$ where g is the centroid of X, with equality if and only if X is a triangle. (In d dimensions, Ehrhart[11] showed that $f(g) \geq d^d/(d+1)^d$ with equality if and only if X is a pyramid on any (d-1)-dimensional convex base.) For centrally symmetric sets, $f(g) = 1/2$, since any line through the point of symmetry cuts X into two pieces of equal area. Thus, $4/9 \leq f(g) \leq 1/2$ with $f(g) = 4/9$ for triangles and $f(g)$ close to $1/2$ for highly symmetric sets.

Although much is known about the center of area, it is quite nontrivial to determine the center of area for a given convex set. In a series of papers, Díaz and O'Rourke[7,8,9] develop an $O(n^6 \log^2 n)$ time algorithm for finding the center of area of a convex n-gon. The same authors give an approximation algorithm that runs in $O(GK(n + K))$ time, where G is the bit-precision of the input polygon P and K is the output bit-precision of the point p^*. Braß and Heinrich-Litan[4] describe an $O(n^2 \log^3 n\alpha(n))$ time algorithm for computing the center of area of a convex n-gon. As an application of tools for searching in arrangments of lines, Langerman and Steiger[15] present an $O(n \log^3 n)$ time algorithm for finding the center of area of a convex n-gon. All of these algorithms are deterministic.

In this paper we give a simple randomized linear-time algorithm for finding the center of area of a convex n-gon P, which also computes Winternitz's measure of symmetry for P. We proceed by first giving a linear-time algorithm for the decision problem: Does there exist a point p such that $f(p) > \delta$? We then apply a randomized technique due to Chan[5] to turn this decision algorithm into a linear-time optimization algorithm. For convenience, our model of computation is the real RAM,[20] though we do not use any functions that are specific to this model. We require only that it is possible to to compute the exact area of a convex polygon.

The remainder of the paper is organized as follows. Section 2 describes our algorithm for the decision problem and Section 3 shows how to convert this decision algorithm into an optimization algorithm. Section 4 summarizes and concludes with directions for future research.

2. The Decision Algorithm

In this section, we give an $O(n)$ time algorithm for the following decision problem: Is there a point p such that $f(p) \geq \delta$? An alternative statement of this problem is: is Γ_δ non-empty? In what follows, we show that Γ_δ can be computed in $O(n)$ time.

A δ-cut of P is a directed line segment uv with endpoints u and v on the boundary of P such that the area of P to the right of uv is at most δ. Note that, for any δ-cut uv, the point p cannot be to the right of uv. On the other hand, if there is no δ-cut uv with p on its right, then $f(p) \geq \delta$. Therefore, each δ-cut defines a linear constraint on the location of p, which we call a δ-constraint. The answer to the decision problem is affirmative if and only if there is a point p that simultaneously satisfies all δ-constraints. If such a point p exists, we call the constraints feasible, otherwise we call them infeasible.

Unfortunately, every polygon has an infinite number of δ-cuts and hence an infinite number of δ-constraints. However, we will show that all constraints imposed by these δ-cuts can be represented succinctly as $O(n)$ non-linear (but convex) constraints that can be computed in $O(n)$ time.

To generate a representation of all δ-constraints, we begin by choosing a point u on the boundary of P and finding the unique point v so that uv is a δ-cut. Next, we sweep the points u and v counterclockwise along the boundary of P maintaining the invariant that uv has an area of exactly δ to its right. We continue this process until we return to the original points u and v.

Observe that, as long as u and v do not cross a vertex of P, the intersection of all δ-constraints belonging to an edge pair is a convex region whose boundary consists of at most 2 linear pieces and 1 non-linear piece. (See Figure 1.) In fact, this non-linear piece is a hyperbolic arc. This is due to the well known fact that a line tangent to a hyperbola forms a triangle of constant area with the asymptotes of the hyperbola. Furthermore, the description complexity of these pieces is constant, since they are defined by a four-tuple of vertices of P. Thus, the intersection of all these δ-constraints can be computed explicitly in constant time. Since u and v sweep over each vertex exactly once, we obtain $2n$ such convex constraints whose intersection is equal to the intersection of all δ-constraints.

Therefore, the decision problem reduces to determining if the intersection of $2n$ convex regions is empty. We can compute an explicit representation of this intersection in $O(n)$ time, as follows: Separately compute the intersection of all δ-constraints that contain the point $(0, +\infty)$ and all δ-constraints that contain the point $(0, -\infty)$ and then compute the intersection of the two resulting convex regions. Since the δ-constraints are generated sorted by slope, the first step is easily done in $O(n)$ time using an algorithm similar to Graham's Scan.[1,13] Since the two boundaries of the two resulting regions are x-monotone and upwards, respectively downwards, convex, their intersection (step two) can be computed in $O(n)$ time using a left-to-right plane sweep.[2]

We have just proven:

Theorem 1. *Let P be a convex n-gon and $\delta > 0$ a real parameter. Then there exists an $O(n)$ time algorithm for the decision problem: Does there exist a point p such that $f(p) \geq \delta$?*

3. The Optimization Algorithm

In this section, we show how to use the decision algorithm of the previous section along with a technique of Chan[5] to solve the optimization problem: What is the largest value of δ such that Γ_δ is non-empty? Chan's technique requires only that we be able to (1) solve the decision problem in $D(n) = \Omega(n^\epsilon)$ time, $\epsilon > 0$ and

(2) generate a set of $r > 1$ subproblems each of size αn, $\alpha < 1$, such that the solution to the original problem is the minimum (or maximum) of the solutions to the subproblems. Under these conditions, the optimization problem can be solved by a randomized algorithm in $O(D(n))$ expected time.

To apply Chan's technique, we need a suitable definition of subproblem. Let S be a subset of edges of P. The S-induced δ-constraints are the set of all δ-constraints uv, where u and v are both on edges of S. The type of subproblems we consider are those of determining for a given set S and a value δ whether or not the S-induced δ-constraints feasible. To obtain a linear-time algorithm, we must be able to solve such subproblems in $O(|S|)$ time.

For a given set S, computing a representation of the S-induced δ-constraints, can be done using a modification of the sweep algorithm from the previous section so that it only considers δ-cuts uv where u and v are on elements of S. The only technical tool required for this modification is a data structure that, given two points u and v on elements of S (the boundary of P) tells us the area of P to the right of uv in constant time. This data structure is provided by Czyzowicz *et al*[6] who show that any convex n-gon can be preprocessed in $O(n)$ time so that the area of the polygon to the right of any chord uv can be computed in $O(1)$ time. Using this data structure, it is straightforward to generate a representation of S-induced δ-constraints in $O(|S|)$ time. Once we have computed these constraints, we can test if they are feasible in $O(|S|)$ time. Thus, Condition 1 required to use Chan's technique is satisfied with $D(n) = \Theta(n)$.

Next, we observe that Helly's theorem in the plane (c.f., Eckhoff[10]) implies that for any $\delta > \delta^*$ there exists a set of three δ-constraints whose intersection is empty. In our context, this means that P contains 6 edges such that, for any $\delta > \delta^*$, the δ-constraints induced by those edges are infeasible. Therefore, if a set S contains those 6 edges, then the S-induced δ-constraints are feasible if and only if $\delta \leq \delta^*$.

Therefore, we can solve our maximization problem as follows: Partition the edges of P in 7 groups, $E_1, \ldots, E_7$, each of size approximately $n/7$. Next, generate subsets $S_1, \ldots, S_7$, by taking all 7 6-tuples of $E_1, \ldots, E_7$. Note that, for each S_i, the S_i-induced δ-constraints are satisfiable if $\delta \leq \delta^*$, since they are just a subset of the original constraints. On the other hand, for the set S_j that contains the 6 edges guaranteed by Helly's theorem, the S_j-induced δ-constraints are not satisfiable for any $\delta > \delta^*$. Therefore,

$$\delta^* = \min \{\max \{\delta \mid S_i\text{-induced } \delta\text{-constraints are satisfiable}\} \mid 1 \leq i \leq 7\} \quad .$$

Finally, observe that each S_i is of size at most αn, for $\alpha = 6/7$. Therefore, we have satisfied the second condition required to apply Chan's optimization technique. This completes the proof of:

Theorem 2. *There exists a randomized $O(n)$ expected time algorithm for the optimization problem: What is the largest value δ^* for which Γ_{δ^*} is non-empty?*

Of course, once δ^* is known, an explicit representation of Γ_{δ^*} can be computed in $O(n)$ time. Alternatively, Chan's technique can also be made to output a point $p^* \in \Gamma_{\delta^*}$.[5]

4. Conclusions

We have given a randomized linear-time algorithm for determining the center of area of a convex n-gon. This algorithm is simple, implementable, and is asymptotically faster than any previously known algorithm.

Although our algorithm is simple and easy to implement, the constants hidden in the O-notation are enormous. A close examination of Chan's technique reveals that the number of subproblems generated in our application is actually $r \geq \binom{k}{6}$, where k is an integer that satisfies $\ln \binom{k}{6} + 1 < k/6$. The smallest such value of k is 146, which leads to $r = \binom{146}{6} = 12\,122\,560\,164$ subproblems. Reducing this constant while maintaining the $O(n)$ asymptotic running time remains an open problem. One possible approach is to treat the problem as an LP-type problem and try to use the Matoušek-Sharir-Welzl algorithm.[17] The difficulty with this approach is that the underlying LP-type problems consists of as many as $\binom{n}{2}$ constraints (though only $O(n)$ apply to any given value of δ). A linear-time deterministic algorithm is also an open problem. The current fastest deterministic algorithm runs in $O(n \log^3 n)$ time.[15]

Finally, we have not considered the problem of computing the center of area of a non-convex polygon. There are two different versions of this problem, depending on whether a cut is defined as a chord of P, which partitions P into two polygons, or a line which may partition P into many polygons. Approximation algorithms for the second case are considered by Díaz and O'Rourke.[7] To the best of our knowledge, there are no exact algorithms for either version.

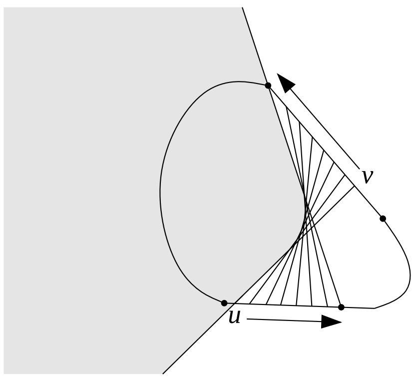
Acknowledgements

The authors would like to thank Stefan Langerman for helpful discussions and for bringing his paper[15] to our attention.

References

1. A. M. Andrew. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9(5):216–219, 1979.
2. J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, C-28:643–647, 1979.
3. W. Blaschke. *Vorlesungen über Differentialgeometrie. II, Affine Differentialgeometrie.* Springer, Berlin, 1923.
4. P. Braß and L. Heinrich-Litan. Computing the center of area of a convex polygon. Technical Report B 02-10, Freie Universität Berlin, Fachbereich Mathematik und Informatik, March 2002.
5. T. Chan. Geometric applications of a randomized optimization technique. *Discrete & Computational Geometry*, 22:547–567, 1999.

6. J. Czyzowicz, F. Contreras-Alcalá, and J. Urrutia. On measuring areas of polygons. In *Proceedings of the 10th Canadian Conference on Computational Geometry (CCCG'98)*, 1998.
7. M. Díaz and J. O'Rourke. Computing the center of area of a polygon. In F. Dehne, J.-R. Sack, and N. Santoro, editors, *Workshop on Algorithms and Data Structures (WADS '89)*, volume 382 of *Lecture Notes in Computer Science*, pages 171–182, Ottawa, Canada, 1989. Springer-Verlag.
8. M. Díaz and J. O'Rourke. Chord centers for convex polygons. *Contemporary Mathematics*, 119:29–43, 1991. Proceedings of the AMS Special Session on Vision Geometry 1989.
9. M. Díaz and J. O'Rourke. Algorithms for computing the center of area of a convex polygon. *The Visual Computer*, 10:432–442, 1994.
10. J. Eckhoff. Helly, Radon, and Caratheodory type theorems. In P. M. Gruber and J. M. Wills, editors, *Handbook of Convex Geometry*, chapter 2.1, pages 389–448. North-Holland, Amsterdam, Netherlands, 1993.
11. E. Ehrhart. Sur les ovales and les ovoides. *C. R. Acad. Sci. Paris*, 240:583–585, 1955.
12. E. Ehrhart. Une généralisation du théorème de Minkowski. *C. R. Acad. Sci. Paris*, 240:483–485, 1955.
13. R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132–133, June 1972.
14. B. Grünbaum. Measures of symmetry for convex sets. In V. Klee, editor, *Convexity: AMS Proceedings of the Symposium on Pure Mathematics VII*, pages 233–270, 1963.
15. S. Langerman and W. Steiger. Optimization in arrangements. Technical Report SOCS-02.7, School of Computer Science, McGill University, 2002.
16. M. A. Lavrent'ev and L. A. Lyusternik. *Elements of the Calculus of Variations*, volume I, part II. Moscow, 1935. (in Russian).
17. J. Matousek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. In *Proc. Eighth Ann. ACM Symp. on Computational Geometry*, pages 1–8, Berlin, Germany, June 1992.
18. B. H. Neumann. On an invariant of plane regions and mass distributions. *Journal of the London Mathematical Society*, pages 226–237, 1945.
19. D. J. Newman. Partitioning of areas by straight lines. *Notices of the American Mathematical Society*, 5:510, 1958.
20. F. P. Preparata and M. I. Shamos. *Computational Geometry*. Springer-Verlag, 1985.
21. I. M. Yaglom and V. G. Boltyanskii. *Convex Figures*. Holt, Rinehart and Winston, 1961. (Translated from Russian).


v
u