Computing Maximal Islands

C. Bautista-Santiago[*] J.M. Díaz-Báñez[†] D. Lara[‡] P. Pérez-Lantero [§] J. Urrutia[¶] I. Ventura[||]

Abstract

Given a set S of red and blue points on the plane in general position, an island $I(S,C) \subseteq S$ is the intersection of S and a convex region C. We study the problem of finding a maximal island according to certain criterium. For instance, a largest monochromatic island $I(S,C)$ is a maximum cardinality subset of S, such that every point of $I(S,C)$ belongs to the same chromatic class. An $O(n^3)$-time and $O(n^2)$-space algorithm is proposed to find one such subset. Our approach can be adapted to find monochromatic islands which maximize parameters such as the area, perimeter.

1 Introduction

In this paper, S will always denote a set of n points on the plane in general position such that its elements are classified into two classes or *colors*, say *red* and *blue.* A subset $\mathcal{I}$ of S is called an island of S if there is a convex set C such that $\mathcal{I} = S \cap C$. We say that an island of S is a blue (resp. red) island of S if all of its elements are blue (resp. red).

In this paper we study the problem of finding islands of S that are optimal according to some parameters, e.g. find a blue island of S with maximum cardinality.

The problem of finding the largest monochromatic island of a point set S (*LMIP*) was studied in [17], where an $O(n^3 \log n)$ time and $O(n^2)$-space algorithm to solve this problem is given. We present here an $O(n^3)$ time and $O(n^2)$-space algorithm to solve the LMIP problem which also solves all the problems studied in [17]. Our algorithms can also handle the weighted version of our main problem in which the elements of S are assigned weights (usually integral values).

We observe that when the labels of the elements of S are chosen carefully, we can solve problems apparently unrelated. For instance, if we label all the blue points of S with 1, and the red points with -1, a maximum weight island is an island with maximum discrepancy (absolute difference between blue and red points) [11]. If we label the blue points of S with 1, and the red points of S with $-\infty$, the maximum weight island is the largest monochromatic island. Our algorithm can also be easily adapted to solve the following problems: Find the monochromatic island $\mathcal{I}$ of S maximizing the number of vertices on the convex hull of $\mathcal{I}$; find a monochromatic island $\mathcal{I}$ of S that maximizes the area, or the perimeter of the convex hull $Conv(\mathcal{I})$ of $\mathcal{I}$, discrepancy and others.

An outline of the paper is as follows. In Section 2 we present the algorithm in detail. A generalization for solving other optimization problems is stated in Section 3. Finally, in Section 4 we show an application of our algorithm to give an heuristic method for a red-blue set covering problem.

1.1 Related work

A key problem in computational geometry is the identification of subsets of a point set having particular properties. Erdös and Szekeres [16] asked whether there was a value $f(k)$ such that all sets of at least $f(k)$ points in general position in the plane determine a convex k-gon. They give upper and lower bounds for $f(k)$.

Several papers have also studied the algorithmic aspects of problems of this kind, e.g. see [4, 10]. Algorithms are also known for finding subsets of points with k elements that minimize parameters such as diameter, or perimeter of the convex hull of these subsets [2, 15].

Our motivation to study the LMIP problem arises from applications in data mining, statistical clustering, pattern recognition or data compression. In data mining and classification problems, a natural method for analyzing data is to select prototypes representing different data classes. A standard technique for achieving this is to perform cluster analysis on the training data [12]. The clusters can be obtained using simple geometric shapes. Aronov and Har-Peled [3] and Eckstein et al. [13] considered circles and axis-

[*]Instituto de Matemáticas, Universidad Nacional Autónoma de México, crevel@uxmcc2.iimas.unam.mx

[†]Departamento Matemática Aplicada II, Universidad de Sevilla, dbanez@us.es, partially supported by grant FEDER-MTM2006-03909.

[‡]Instituto de Matemáticas, Universidad Nacional Autónoma de México, dlara@uxmcc2.iimas.unam.mx

[§]Departamento de Computación, Universidad de La Habana, pablo@matcom.uh.cu, supported by MAEC-AECI

[¶]Instituto de Matemáticas, Universidad Nacional Autónoma de México, urrutia@matem.unam.mx, partially supported by grant FEDER-MTM2006-03909.

[||]Departamento Matemática Aplicada II, Universidad de Sevilla, iventura@us.es, partially supported by grant FEDER-MTM2006-03909.

aligned boxes. In this paper we consider a convex set for the selection problem.

2 The largest monochromatic island

In this section we present an $O(n^3)$ time algorithm to solve the LMIP problem. Our algorithm is based on Fisher's $O(n^3 \log n)$ algorithm to solve the LMIP problem [17]. We improve Fisher's running time algorithm by using efficient data structures.

We proceed now to give some definitions that will be needed in the rest of this paper. $\triangle(p, e)$ will denote the triangle having a line segment e as one of its sides and the point p as the vertex opposite to e; $p_i \to p_j$ will denote the directed line segment starting at p_i and ending at p_j.

Let p_0 be a blue point in S, and S_0 the set of blue points in S below p_0. From now on we will assume that the elements of S_0 are labeled $\{p_1, \ldots, p_k\}$ in the counterclockwise order with respect to p_0.

Let $\mathcal{B}$ be a blue island of S, and p_0 the point of $\mathcal{B}$ with the largest y-coordinate (w.l.o.g. we will assume that such a point is unique). The point p_0 will be called the *anchor* of $\mathcal{B}$. The weight of $\mathcal{B}$ is the cardinality of $\mathcal{B}$.

Our approach to solve the LMIP is the following: For each blue point $p_0 \in S$, find the largest monochromatic island anchored in p_0.

Let $\mathcal{B}$ be a blue island anchored at p_0, and $p_0, p_{\sigma_1}, \ldots, p_{\sigma_k}$ be the vertices of the convex hull of $\mathcal{B}$ labeled in the counterclockwise order around the boundary of the convex hull of $\mathcal{B}$, starting at p_0. We call the edge $p_{\sigma_{k-1}} - p_{\sigma_k}$ joining $p_{\sigma_{k-1}}$ to p_{σ_k} the *last edge* of the convex hull of $\mathcal{B}$, and sometimes we will say that $\mathcal{B}$ *ends* at $p_{\sigma_{k-1}} - p_{\sigma_k}$. We denote as $\text{Blue}(P)$ the number of blue points contained in the convex set P. We will also assign a weight $w(p_{\sigma_{k-1}} - p_{\sigma_k})$ to the edge $p_{\sigma_{k-1}} - p_{\sigma_k}$ equal to the weight of the largest blue island of S anchored at p_0 that ends at $p_{\sigma_{k-1}} - p_{\sigma_k}$. If the triangle $\triangle(p_0, p_{\sigma_{k-1}} - p_{\sigma_k})$ contains at least one red point, $w(p_{\sigma_{k-1}} - p_{\sigma_k}) = 0$.

Let $\text{Blue}(p_{\sigma_i})$ denote the number of blue points inside or on the convex polygon with vertices $p_0, p_{\sigma_1}, \ldots, p_{\sigma_i}$, $(1 \le i \le k)$.

Our algorithm is based on the following crucial observation that suggests a dynamic programming approach to solve the problem:

Observation 1. $\text{Blue}(p_{\sigma i+1}) = \text{Blue}(p_{\sigma_i}) + \text{Blue}(\triangle(p_0, p_{\sigma_i} - p_{\sigma i+1})) - 2$.

The additive property in Observation 1, allows us to solve the problem of finding the largest monochromatic island anchored at a point p_0 by performing a radial sweep of the blue points below p_0 in the counterclockwise order around p_0 by joining sets of so-called *good triangles* with respect to p_0 [17]; that is sets of triangles with blue vertices (one of which is p_0), such that they have disjoint interiors, do not contain red points, and their union forms a convex polygon anchored at p_0, see Figure 1. The bottleneck of the sweeping approach proposed in [17] is the joining process. In the next section, we will show how to compute the LMIP for an anchored island in $O(n^2)$ time and space, thus obtaining an $O(n^3)$-time algorithm to solve the LMIP.

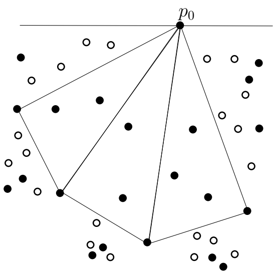

Figure 1: We add good triangles from left to right.

2.1 The algorithm

The algorithm does a preprocessing stage: First, we calculate *for all blue points* $p \in S$ the radial ordering of the blue points in S below p in overall $O(n^2)$ time and space [14] . Second, we preprocess S such that for each triangle with vertices in S, the number of red and the number of blue points in the triangle can be determined in constant time. We use a simple modification of the algorithm proposed in [15]. The preprocessing phase takes $O(n^2)$-time and space.

Let p_0 be a blue point in S, and relabel the blue points in S below p_0 by $\{p_1, \ldots, p_k\}$ such that the line segment joining p_0 to p_{i+1} is to the right of that joining p_0 to p_i. This labeling can be obtained in $O(n)$ time during the preprocessing stage. The algorithm scans $\{p_1, \ldots, p_k\}$ from p_1 to p_k such that for each two points p_i and p_j $(1 \le i < j \le k)$ we calculate the weight $w(p_i - p_j)$ to the edge joining p_i to p_j equal to the size of the largest blue island $\mathcal{B}$ (if any) of S anchored at p_0, and ending at the edge $p_i - p_j$. If the triangle $\triangle(p, p_i - p_j)$ contains at least one red point of S, $w(p_i - p_j)$ equals 0, see Figure 2. For the sake of clarity if $i < j$, we will orient the edge $p_i - p_j$ with the orientation $p_i \to p_j$, thus obtaining an oriented acyclic graph $G(p_0)$ with vertex set $\{p_1, \ldots, p_k\}$.

Observation 1 will allow us to calculate the weights $w(p_i \to p_j) = w(p_i - p_j)$ of the edges of $G(p_0)$ inductively as follows:

In the initial stage, when $i = 1$, we assign to all edges $p_1 \to p_j$ the weight $\text{Blue}(\Delta(p_0, p_1 - p_j))$, and $prev(p_1 \to p_j) = null$. Suppose that we have assigned $w(\cdot)$ and $prev(\cdot)$ to all edges $p_i \to p_j$ $(1 \le j < k)$. We now show how to set the weights of all edges $p_j \to p_l$ $(j < l \le k)$.

Assume that all incoming and outgoing edges to p_j are labelled $L_a = a_1, \ldots, a_q$ and $L_b = b_1, \ldots, b_r$

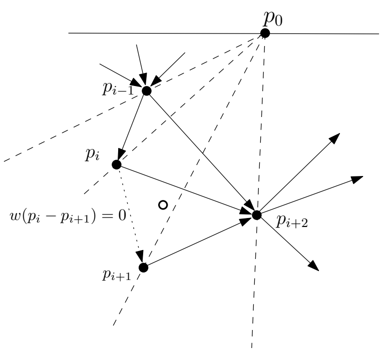

Figure 2: The weight of edge $p_i \rightarrow p_{i+1}$ is 0 because $\triangle(p_0, p_i \rightarrow p_j)$ contains a red point.

respectively such that L_a (resp. L_b) is sorted with respect to p_j, see Figure 3. During the preprocessing stage both L_a and L_b can be obtained in linear time. Given an edge a_r in L_a and an edge b_s in L_b, we say that a_r *is compatible* with b_s if the union of $\Delta(p_0, a_r)$ and $\Delta(p_0, b_s)$ is a convex polygon.

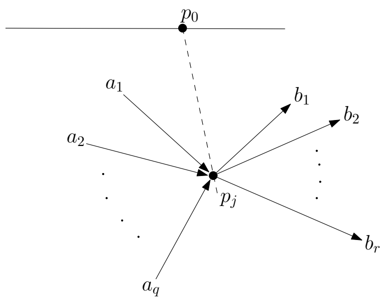

Figure 3: Ordering of the edges of p_j

Now, for every edge b_s in L_b such that triangle $\triangle(p_0, b_s)$ contains no red points in S, we want to find the edge a_r in L_a such that a_r is *compatible* with b_s and has maximum weight (if $\triangle(p_0, b_s)$ contains red points, then $w(b_s) = 0$). It is easy to see that if a_r exists then $w(b_s) = w(a_r) + \text{Blue}(\Delta(p_0, b_s)) - 2$. The index r will be $prev(b_s)$

To obtain r we proceed as follows: We label each a_i $(1 \leq i \leq q)$ with $max(a_i)$ which is a pointer to the edge a_h $(1 \leq h \leq i)$ such that $w(a_h)$ is maximum among $w(a_1), \ldots, w(a_i)$. This can be done in $O(q)$ time by doing $max(a_1) = a_1$ and for $i = 2 \ldots q$ applying the following formula:

$$max(a_i) = \begin{cases} a_i & \text{if } w(a_i) \geq w(max(a_{i-1})) \\ max(a_{i-1}) & \text{if } w(a_i) < w(max(a_{i-1})) \end{cases}$$

The following procedure computes $w(\cdot)$ and $prev(\cdot)$ for all b in L_b:

Start by setting $i = q$. For $m = 1, \ldots, r$ find the first index t from i to 1 such that a_t is *compatible* with b_m. If such position t exists set $w(b_m) = w(max(a_t)) + \text{Blue}(\Delta(p_0, b_m)) - 2$ and $prev(b_m) = t$, otherwise set $t = 0$, $w(b_m) = \text{Blue}(\Delta(p_0, b_m))$ and $prev(b_m) = null$. After this make $i = t$ and continue the iteration of m.

Clearly above procedure runs in $O(n)$ time thus the complete assignment of $w(\cdot)$ and $prev(\cdot)$ is done in $O(n^2)$ time. Therefore we have proved:

Lemma 1 *Assigning $w(.)$ and $prev(.)$ to all edge of $G(p_0)$ can be done in $O(n^2)$ time.*

The largest blue island $\mathcal{B}$ anchored at p_0 corresponds to the edge e of $G(p_0)$ with maximum weight. Said in another way, $\mathcal{B}$ *ends* at e. The convex hull, and thus $\mathcal{B}$, can now be computed by using the pointers $prev(e)$ recursively.

By repeating the same procedure with the red points of S, we obtain the following result:

Theorem 2 *Let S be a bichromatic point set in general position on the plane. The largest monochromatic island can be found in $O(n^3)$ time, by using a preprocessing of $O(n^2)$ time and space.*

3 Generalizations

The algorithm presented in the previous section can be used to solve a collection of optimization problems. Suppose we have a function $f : \mathcal{P} \rightarrow I\!R$, where $\mathcal{P}$ is a set of convex polygons.

Definition 1 *We say that a function f on $\mathcal{P}$ [15] is decomposable iff for any polygon $P = \{p_1, p_2, \ldots, p_k\}$ and any index $2 < i < k$,*

$$f(P) = g(f(\{p_1, \ldots, p_i\}), f(\{p_1, p_i, \ldots, p_k\}), p_1, p_i)$$

where g can be calculated in constant time.

Notice that if f counts the number of points of S contained within or on the boundary of a convex polygon P, then $g(x, y, p, q) = x + y - 2$.

Other decomposable functions are the perimeter, area, number of points in the convex hull, discrepancy, sum of weights of the points, etc. It is easy to see that our main algorithm can b e easily modified to minimizing or maximizing these functions. We can also modify the method for solving the problem of finding maximal or minimal empty polygons. Another interesting variant is the following:

The empty ordered heterochromatic island problem: *Given n points in the plane colored with colors $1, \ldots, k$, compute the largest empty convex polygon P with k vertices such that the colors on the vertices of P appear in the order $1, \ldots, k$.*

See [9] for a related paper. Note that the modification we have to do to our main algorithm is to build the graph $G(p_0)$ following colors in order $1, \ldots, k$. The next result follows.

Theorem 3 *Let S be a set of points in the plane colored with k colors and let f be a monotone decomposable function. The ordered heterochromatic island that minimizes or maximizes f can be computed in $O(n^3)$ time and $O(n^2)$ space.*

4 The Class Cover Problem with Convex Sets

Given a set S of red and blue points, a classical problem is that known as the *Class Cover Problem* [7]. It consists in finding a set of circles with minimum cardinality such that every blue point in S is contained in at least one of the circles, and no circle contains a red point. We consider a variant of this problem in which we want to cover the blue points by using (non-necessarily) disjoint convex polygons. In [1] the problem of covering a point set with the smallest number of pairwise disjoint triangles is studied. It is proved by using a reduction from planar 3SAT-problem that this problem is NP-hard . We can use the same construction as in [1] to reduce any instance of the planar 3SAT to an instance of the Class Cover Problem with convex sets in which the only possible solutions are formed by sets of pairwise disjoint triangles. Hence our problem is also NP-hard.

The $O(\log n)$-approximation greedy approach for the more general *Set Cover Problem* [18] can easily be applied to our problem. It works as follows: recursively compute the maximum blue island I of S, remove it and repeat until there are no more blue points left in S. This approach produces a $O(n^4)$-time $O(\log n)$-approximation algorithm.

The techniques of [6] to obtain $o(\log n)$-approximation algorithms work for systems with constant VC-dimension. Notice that the VC-dimension of our system is not constant, specifically it can be $\Omega(n)$ (take a set of n points in convex position).

References

[1] P. K. Agarwal and S. Suri. *Surface Approximation and Geometric Partitions* Proc. 5th annual ACM-SIAM Symp. on Discrete Algorithms. ACM Press, 24-33.

[2] A. Aggarwal, H. Imai, N. Katoh, and S. Suri. *Finding k points with minimum diameter and related problems.* Proc. 5th ACM Symp. on Computational Geometry, 283-291, 1989.

[3] B. Aronov and S. Har-Peled. On approximating the depth and related problems. *Proceedings 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2005.

[4] D. Avis and D. Rappaport. *Computing the largest empty convex subset of a set of points.* In SCG 85: Proceedings of the first annual symposium on Computational geometry, 161167, ACM, 1985.

[5] J.E. Boyce, D.P. Dobkin, I. Robert L.(Scot) Drysdale, and L.J. Guibas. *Finding extremal polygons.* In STOC 82: Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, 282289, ACM, 1982.

[6] H. Brönnimann, M.T. Goodrich. *Almost optimal set covers in finite VC-dimension.* Discrete and Computational Geometry, 14, 463-479, 1995.

[7] A.H. Cannon and L.J. Cowen. *Approximation algorithms for the class cover problem.* Annals of Mathematics and Artificial Intelligence, 40(3-4):215-223, 2004.

[8] T. H. Cormen, C. E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* Second Edition. MIT Press, McGraw-Hill, 2001.

[9] J.M. Díaz-Báñez, G. Hernandez, D. Oliveros, A. Ramirez-Vigueras, J.A. Sellarès, J. Urrutia, I. Ventura, Computing Shortest Heterochromatic Monotone Routes Operational Research Letters, Volume 36, 684-687, 2008.

[10] D.P. Dobkin, H. Edelsbrunner, and M.H. Overmars. *Searching for empty convex polygons* In SCG 88: Proceedings of the fourth annual symposium on Computational geometry, 224228, ACM, 1988.

[11] D. P. Dobkin, D. Gunopulos, and W. Maass. Computing the maximum bichromatic discrepancy, with applications to computer graphics and machine learning. *J. Computer and Systems Sciences*, 52(3) (1996) 453470.

[12] R. Duda, P. Hart, and D. Stork. *Pattern classification. John Wiley and Sons, Inc.*, 2001.

[13] J. Eckstein, P. L. Hammer, Y. Liu, M. Nediak, and B. Simeone. The maximum box problem and its applications to data analysis. *Comput. Optim. Appl.* 23 (2002) 285–298.

[14] H. Edelsbrunner, J. O'Rourke, and R. Seidel. *Constructing arrangements of lines and hyperplanes with applications.* SIAM J. Comput. 15, 341-363, 1986.

[15] D. Eppstein, M. Overmars, G. Rote, and G. Woeginger. *Finding minimum area k-gons.* Discrete and Computational Geometry, 7, 4558, 1992.

[16] P. Erdös and G. Szekeres. *On some extremum problems in elementary geometry.* Ann. Univ. Sci. Budapest, (3-4), 5362, 1960/1.

[17] P. Fischer. *Sequential and parallel algorithms for finding a maximum convex polygon.* Computational Geometry: Theory and Applications, 7, 187200, 1997.

[18] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, New York, NY. 1979.