Computing Optimal Islands

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

16th March 2010

Abstract

Let S be a bicolored set of n points on the plane. A subset $\mathcal{I} \subseteq \mathcal{S}$ is called an island of S, if $\mathcal{I}$ is the intersection of S and a convex set C. In this paper we give an $O(n^3)$-time algorithm to find a monochromatic island of maximum cardinality. Our approach also optimizes other parameters and gives an approximation to the class cover problem.

Keywords: Maximum Convex Polygon, Classification, Computational Geometry, Algorithms.

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

[†]Corresponding author. José Miguel Díaz-Báñez. Postal address: Departamento Matemática Aplicada II. Escuela Técnica Superior de Ingenieros. Camino de los Descubrimientos, s/n, 41092, Sevilla, Spain, `dbanez@us.es`

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

[§]Departamento de Computación, Universidad de La Habana, `pablo@matcom.uh.cu`

[¶]Universidad Nacional Autónoma de México, `urrutia@matem.unam.mx`

[||]Departamento Matemática Aplicada II, Universidad de Sevilla, `iventura@us.es`

1 Introduction

Given a set of points S on the plane, the convex hull of S, denoted as $CH(S)$, is the smallest convex set of the plane containing S. Let S be 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 if there is a convex set C such that $\mathcal{I} = S \cap C$. An island of S is monochromatic if all of its elements have the same color. A monochromatic island is called red or blue depending on the color of its elements.

In this paper we study the problem of finding a *largest monochromatic island* of a bicolored point set S, that is a monochromatic island of S with maximum cardinality. We will refer to this problem as the *LMI-problem*. An $O(n^3)$-time algorithm to solve the LMI-problem is presented, improving on the $O(n^3 \log n)$-time algorithm presented in [14]. Our algorithm is simple and easy to implement and it can be easily generalized to solve a collection of maximization or minimization problems involving so-called decomposable functions. In the rest of this paper, S will always denote a bicolored point set, and will assume without loss of generality that the largest monochromatic island of S is blue. By running our algorithm twice, first finding the largest blue island, and then the largest red island we will obtain the optimal solution to our problem. Thus from now on, we consider only the problem of finding the largest blue island of S.

With minor modifications, our algorithms can solve weighted versions of our main problem. In these versions, the elements of S have been assigned weights (usually integral values), and our objective is that of finding islands of S with maximum weight. We observe that when the labels or weights of the elements of S are chosen carefully, we can solve problems apparently unrelated. For example, finding an island of S with *maximum discrepancy*, that is, an island in which the absolute value of the difference between the number of blue points and the number of red points is maximized [9], can be obtained by solving two instances of the maximum weight problem as follows: First assign weight 1 (resp. -1) to all blue points (resp. red points), and find an island of maximum weight. Then assign weight -1 (resp. 1) to all blue points of S (resp. red points in S), and find an island of maximum weight. The solution with maximum weight to both problems will produce the island of S with maximum discrepancy.

1.1 Related work

In recent years, there has been a lot of work on geometric and algorithmic problems on bicolored point sets on the plane. Two of the first problems studied here concern the existence of simple alternating paths in bicolored point sets [3], and that of finding monochromatic spanning trees with few intersections [19]. Since then, different problems on bicolored point sets have been studied. Many of these problems can be cast as partitioning problems of point sets into sets of disjoint islands with specific properties, e.g. if a point set S contains kn red and n blue points, partition S into a set of n disjoint islands such that together they cover all of S, and each island contains k red and one blue point. The interested reader can find a good survey on some of these problems in [16].

Several papers have studied the algorithmic aspects of problems of this kind, for example in [4] it is studied the problem of finding a *largest empty convex subset* of a point set, that is, a largest subset of a point set P, such that its elements are the vertices of a convex polygon Q containing no element of P in its interior. Algorithms are also known for finding subsets of points with k elements that minimize parameters such as the diameter, the perimeter, or the number of vertices of their convex hull. For relevant results see [2, 13].

Our motivation to study the LMI-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 [10]. In this paper we propose the use of convex polygons. In pattern recognition, the convex hulls have been considered to measure the separability among classes [17]. Indeed, the relationship between those convex hulls and support vector machines (SVMs) have been well studied [5].

2 The largest monochromatic island

In this section we present an $O(n^3)$-time algorithm to solve the LMI-problem. To start, we note that there are configurations of red and blue points for which there are an exponential number of solutions to the LMI-problem. For example take a regular k-gon P_k with vertices $v_1, \ldots, v_k$. For each vertex v_i of P_k place a set S_i with $2k$ points on a convex curve C such that every second point on C is blue, see Figure 1. It is easy to see that any largest monochromatic blue island of the point set thus obtained, has exactly k elements. Moreover these islands can be obtained by choosing a blue point from each S_i, or all the blue points of some S_i. Thus there are $k^k + k$ of such islands.

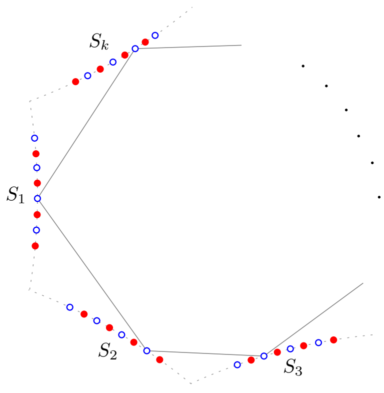

Figure 1: There are k groups $S_1, \ldots, S_k$ of $2k$ points each. There are $k^k + k$ solutions.

We give now some terminology and definitions that will be useful to us. From now on we will assume, without loss of generality, that no two elements of S have the same y-coordinate. Denote by $p - q$ the line segment joining the points p and q. Given a line segment $p - q$ and a point r not in $p - q$, $\Delta(r, p - q)$ will denote the triangle whose vertices are p, q, and r. For a set X, Blue(X) denotes the number of blue points in X. Thus Blue$(\Delta(r, p - q))$ will denote the number of blue points in the triangle $\Delta(r, p - q)$.

Given a point p and two segments $e = q - r$ and $e' = r - s$ with blue endpoints, we call e and e' *p-compatible* if the following conditions hold:

1. $\Delta(p, e)$ and $\Delta(p, e')$ contain no red points in their interiors,
2. $\Delta(p, e)$ and $\Delta(p, e')$ have disjoint interiors, and
3. $\Delta(p, e) \cup \Delta(p, e')$ is a convex polygon, see Figure 2 *a*).

Let $\mathcal{P}$ be a convex polygon with vertices in S, and p the vertex of $\mathcal{P}$ with the largest y-coordinate. We call p the anchor of $\mathcal{P}$, and say that $\mathcal{P}$ is anchored at p. Our objective is now to find, for each blue point $p \in S$ the largest blue island $\mathcal{B}$ of S such that the anchor of the convex polygon determined by the convex hull of $\mathcal{B}$ is p, that is *the largest blue island of S anchored at p.*

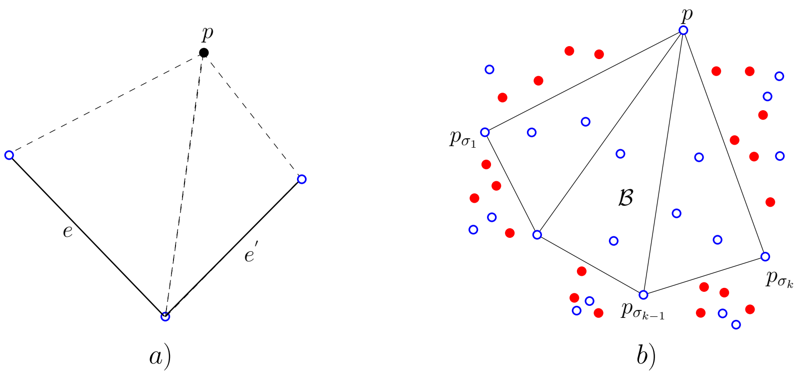

Figure 2: $a)$ The edges e and e' are p-compatible. $b)$ $CH(\mathcal{B})$ is the union of blue triangles anchored at p.

Let $\mathcal{B}$ be a blue island of S anchored at p. Assume that the vertices on $CH(\mathcal{B})$ are labelled $p, p_{\sigma_1}, \ldots, p_{\sigma_k}$ in the counterclockwise order along the boundary of $CH(\mathcal{B})$. We will say that $\mathcal{B}$ *ends* at $p_{\sigma_{k-1}} - p_{\sigma_k}$.

Let h_p be the horizontal line through p, and p_i and p_j two blue points of S lying below h_p. Observe that if $\Delta(p, p_i - p_j)$ contains red elements of S, $p_i - p_j$ will never be an edge of a blue island anchored at p. Thus in the following, we will deal only with segments $p_i - p_j$ such that $\Delta(p, p_i - p_j)$ contains no red points. We associate a weight $w(p_i - p_j)$ to edge $p_i - p_j$ as follows: *$w(p_i - p_j)$ is equal to the weight $Blue(\mathcal{B})$ of the largest blue island $\mathcal{B}$ of S anchored at p that ends at $p_i - p_j$.*

Thus finding a maximum blue island of S anchored at p reduces to finding an edge $p_i - p_j$ with maximum weight. The following observation suggests a dynamic programming approach to solve LMI-problem:

Observation 1. Let $\mathcal{B}$ be a blue island anchored at p, and let $p, p_{\sigma_1}, \ldots, p_{\sigma_{k-1}}, p_{\sigma_k}$ be the vertices of $CH(\mathcal{B})$ labeled in counterclockwise order. Let $\mathcal{B}^{(i)}$ $(1 \leq i \leq k)$ be the island such that the vertices of $CH(\mathcal{B}^{(i)})$ are $p, p_{\sigma_1}, \ldots, p_{\sigma_i}$. Observe that $p_{\sigma_i} - p_{\sigma_{i+1}}$ and $p_{\sigma_{i+1}} - p_{\sigma_{i+2}}$ are p-compatible, $i = 1, \ldots k-2$, and Blue$(\mathcal{B}^{(k)})$ satisfies the following formula:

$$\text{Blue}(\mathcal{B}^{(i)}) = \begin{cases} 2 & \text{if } i = 1 \\ \text{Blue}(\mathcal{B}^{(i-1)}) + \text{Blue}(\Delta(p, p_{\sigma_{i-1}} - p_{\sigma_i})) - 2 & \text{if } 1 < i \leq k. \end{cases}$$

The additive property in Observation 1, allows us to solve the problem of finding the largest monochromatic island anchored at a point p by performing a radial sweep of the blue points below h_p in the counterclockwise order around p by joining sets of p-compatible edges, i.e. sets of triangles with blue vertices (one of which is p) such that they have disjoint interiors, do not contain red points, and their union forms a convex polygon anchored at p, see Figure 2 b). The bottleneck of the sweeping approach proposed in [14] is the joining process. The *prefix-maximum data structure* used in that paper does not avoid performing a binary search to select the best solution in each step. In the next section, we give a simple data structure that allows to compute the largest blue island anchored at a point p in $O(n^2)$ time and space, thus obtaining an overall $O(n^3)$-time algorithm to solve the LMI-problem.

2.1 Computing the weights of edges $p_i - p_j$

The following result presented in [13] will be useful to us:

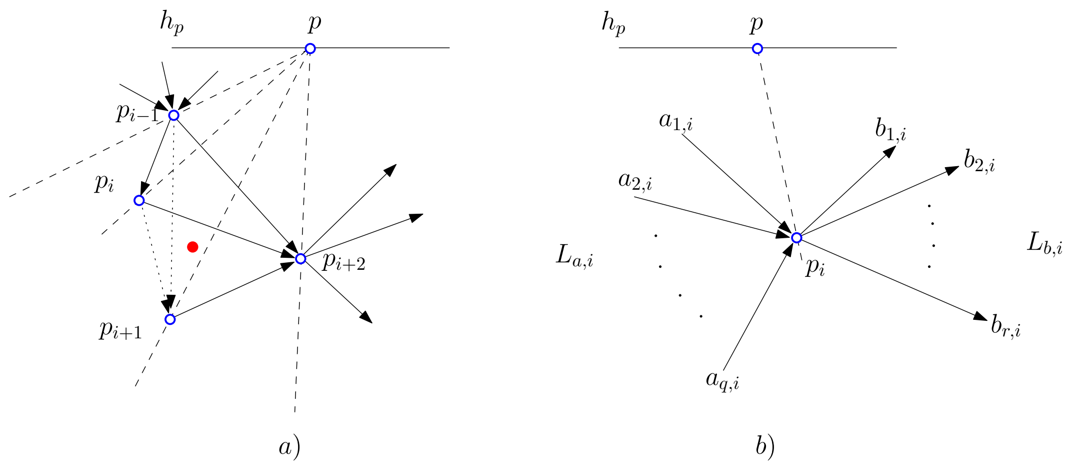

Figure 3: a) $\Delta(p, p_i - p_{i+1})$ contains a red point, and thus edge $p_i - p_{i+1}$ is discarded. b) Ordering of the edges of p_i.

Theorem 1 *Let P be a set of n points in the plane in general position. Then it is possible to preprocess P in $O(n^2)$ time and space such that for any triangle T with vertices in P we can, in constant time, determine the number of points of P in T.*

Straightforward modifications to their algorithm can be used to solve the following problems:

- For each triangle T with vertices in P, find the number of red and the number of blue points contained in T in constant time.
- If the elements of P have weights assigned to them, calculate the sum of the elements of P contained in T in constant time.

Let S_p be the set of blue points in S below h_p. Suppose that the elements of S_p are labeled $p_1, \ldots, p_k$ from left to right according to the slope of the line segments joining them to p. By Theorem 1, we can discard, in constant time per edge, all edges $p_i - p_j$ such that $\Delta(p, p_i - p_j)$ contains at least one red point, see Figure 3 a). We process all remaining edges as follows:

First, and 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 with vertex set S_p, $1 \le i < j \le k$. For every $1 < i \le k$ relabel the sets of incoming and outgoing edges of p_i with $L_{a,i} = \{a_{1,i}, \ldots, a_{q,i}\}$ and $L_{b,i} = \{b_{1,i}, \ldots, b_{r,i}\}$ respectively, such that $L_{a,i}$ and $L_{b,i}$ are radially sorted with respect to p_i as shown in Figure 3 b). For all the points $p \in S$, the corresponding ordering $p_1, \ldots, p_k$, as well as the sorted sets $L_{a,i}$ and $L_{b,i}$, $i = 1, \ldots, k$ can be obtained overall in quadratic time and space [12].

We show next how to calculate $w(p_i - p_j)$ recursively for all $1 \le i < j \le k$. Recall that for the calculation of $w(b_{m,i})$, $1 \le m \le r$, we must find an edge $a_{s,i}$ which is p-compatible with $b_{m,i}$ such that $w(a_{s,i})$ is as large as possible. The idea is to handle the lists $L_{a,i}$ and $L_{b,i}$ in such a way all outgoing edges can be weighted in linear time. We elaborate on this:

We assign to all edges $p_i - p_j$ a pointer $\text{prev}(p_i - p_j)$ initially set to *null*. Each edge $p_1 - p_j$ has now assigned the weight $w(p_1 - p_j) = \text{Blue}(\Delta(p, p_1 - p_j))$. Suppose that all edges $p_\alpha - p_\beta$ have been assigned weights, $1 \le \alpha < \beta \le k$, $\alpha < i < k$. We now show how to assign weights to all edges $p_i - p_j$, $i < j \le k$.

For $1 \le \ell \le q$, let $h(\ell)$ be the smallest integer such that $w(a_{h(\ell),i}) = \max\{w(a_{1,i}), \ldots, w(a_{\ell,i})\}$. The values $h(1), \ldots, h(q)$ can be calculated in $O(q)$ time as follows: $h(1) = 1$ and for $\ell = 2, \ldots, q$ applying the following formula:

$$h(\ell) = \begin{cases} \ell & \text{if } w(a_{\ell,i}) > w(a_{h(\ell-1),i}) \\ h(\ell-1) & \text{if } w(a_{\ell,i}) \leq w(a_{h(\ell-1),i}) \end{cases}$$

The following observation will be useful:

Observation 2: Let s be the largest index such that $a_{s,i}$ is p-compatible with $b_{m,i}$, then $a_{h(s),i}$ is p-compatible with $b_{m,i}$, and has maximum weight over all the edges in $L_{a,i}$ compatible with $b_{m,i}$.

Therefore we set $w(b_{m,i}) = w(a_{h(s),i}) + \text{Blue}(\Delta(p, b_{m,i})) - 2$ and $\text{prev}(b_{m,i}) = a_{h(s),i}$.

The following procedure computes $w(\cdot)$ and $\text{prev}(\cdot)$ for all $b_{m,i}$ in the list $L_{b,i}$:

For $m = 1, \ldots, r$ find the largest index s_m such that $a_{s_m,i}$ and $b_{m,i}$ are p-compatible. If no incoming edge to p_i is p-compatible with $b_{m,i}$, set $s_m = 0$. If $s_m = 0$ then $w(b_{m,i}) = \text{Blue}(\Delta(p, b_{m,i}))$, else $w(b_{m,i}) = w(a_{h(s_m),i}) + \text{Blue}(\Delta(p, b_{m,i})) - 2$. Since $s_1 \geq s_2 \geq \ldots \geq s_r$, it follows that we can compute the weights and $\text{prev}(\cdot)$ of all of the elements of $L_{b,i}$ in a single pass over $L_{a,i}$ and $L_{b,i}$. It is clear that the procedure above runs in $O(n)$ time. Thus we have:

Lemma 1 *The weights of all edges $p_i - p_j$ lying below h_p can be calculated in $O(n^2)$ time.*

To obtain a largest blue island $\mathcal{B}$ anchored at p, find an edge $p_i - p_j$ with maximum weight, and to calculate the convex hull of $\mathcal{B}$, simply follow the pointers $\text{prev}(\cdot)$ recursively. By repeating the above procedure for all the blue elements of S we obtain:

Theorem 2 *Let S be a bichromatic point set in general position on the plane. The largest monochromatic blue 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. To this end suppose we have a function $f : \mathcal{P} \to \mathbb{R}$, where $\mathcal{P}$ is a set of convex polygons.

Definition 1 [13] *We say that a function f on $\mathcal{P}$ is decomposable iff for any polygon $P = \{p_1, p_2, \ldots, p_k\} \in \mathcal{P}$ 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.

Roughly speaking, a function f is decomposable if, when a polygon P is cut into two subpolygons P_1 and P_2 along a diagonal e joining vertices p_1 and p_j of P, $f(P)$ can be calculated in constant time from $f(P_1)$, $f(P_2)$, and some information on e. For example, the function $\mathcal{H}$ that counts the number of points of S contained within or on the boundary of a convex polygon P is decomposable, $\mathcal{H}(P) = g(x, y, p_1, p_i) = x + y - 2$, where $\mathcal{H}(P_1) = x$ and $\mathcal{H}(P_2) = y$.

A key observation is that if we change the function $\text{Blue}(\cdot)$ in Observation 2, by any decomposable function f, the method we developed to calculate the weights of the edges $p_i - p_j$, will instead calculate (within the same complexity) the function f for islands ending at $p_i - p_j$.
Since functions such as the area or perimeter of a convex polygon are decomposable, it follows that by changing $\text{Blue}(\cdot)$ by above functions, we can calculate in $O(n^3)$ time a blue island of S with maximum area, or perimeter.

Suppose now that we assign weights to the elements of S, for our current purposes usually integral values. Observe that the function that calculates the sum of the elements of S within a polygon is decomposable, and this allows us to calculate islands of maximum weight. As mentioned in the introduction of this paper, if we choose the weights of the elements of S carefully, we can solve problems such as that of finding an island of S with maximum discrepancy. Moreover if we label the blue points with 1, and the red points with $-\infty$, the maximum weight island is the largest monochromatic blue island.

We conclude this section by mentioning some generalizations of our problems on bicolored point sets, to sets of points whose elements are colored with k colors. To this end, let S' be a k-colored point set. Let P be a convex polygon with vertices in S'. We say that P is a *hole* of S', if P contains no element of S' in its interior. Straightforward modifications of our algorithm allow us to solve, in $O(n^3)$ time, the problem of finding a monochromatic hole P of S' with the largest number of vertices, or with maximum area or perimeter.

An interesting open problem in this scenario is that of finding, if it exists, a *heterochromatic* island of S' with k elements, that is, an island of S' with k elements such that all of its elements have different color. Another open problem is that of finding, if it exists, a convex heterochromatic polygonal chain, or a monotone heterochromatic polygonal chain of minimum length, see [8]. However, if we restrict the elements of our convex chain, or monotone path to appear in a predetermined order, e.g. the colors appear in order $1, 2, \ldots, k$, then the monotone chain can be computed in $O(n \log^2 n)$ time [8] and the convex chain, using the procedure of this paper, in $O(n^3)$ time.

4 The Class Cover Problem with Convex Sets

In this section we propose the use of the Largest Monochromatic Island's algorithm to approximate a class region by a small collection of convex sets. Several geometric objects as rectangles or circles have been used in machine learning and recognition to separate a bicolored set of points [10, 11, 9]. However, as pointed out in [17], the number of convex hulls needed to approximate a class region is less than, for instance, that of rectangles needed to approximate the same class region. Thus, the use of a set of islands instead of a set of rectangles to approximate a class region seems adequate.

Given a set S of two classes, say red and blue points, a problem with application to classification is the so-called the *Class Cover problem.* This problem is the one of finding a small number of sets covering (containing) points from one class without covering any points from a second class. The problem was introduced in [6] and consists of computing a set of circles $\mathcal{C}$ of equal size, and with minimum cardinality such that the elements of $\mathcal{C}$ contain no red point, and every blue element of S is contained in at least one element of $\mathcal{C}$. We consider here a variant of this problem, named the *Convex Polygon Class Cover problem*, in which we want to cover the blue points by using (non-necessarily) disjoint convex polygons. Firstly, we show the hardness of this problem and then an approximation solution is proposed. In [1], it is proved the NP-hardness of the following problem: given n red and m blue points in the plane, find a minimum number of pairwise disjoint triangles such that each blue point is covered by some triangle and no red point lies in any of the triangles. The reduction is from the planar 3SAT-problem [15]. It is straightforward that we can use the same construction as in [1]. In fact, we can reduce any instance of the planar 3SAT-problem to an instance of the Convex Polygon Class Cover Problem in which the only possible solutions consist of sets of pairwise disjoint triangles. Hence our problem is also NP-hard. Next we show an approximation algorithm based on the Largest Monochromatic Island problem.

Let us consider the $O(\log n)$-approximation greedy approach for the more general *Set Cover Problem* [15]. It can easily be applied to our problem and works as follows: recursively compute the maximum blue island of S, remove it and repeat until there are no more blue points left in S.

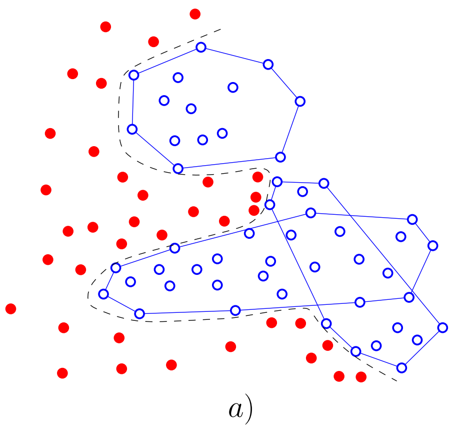

Figure 4: a) Separating the classes by using big islands. b) Disjoint and non-disjoint covering set.

Observe that the convex hulls of the blue islands obtained by the process may intersect. This approach requires $O(n)$ iterations in the worst case and this will happen when all resulting islands have constant cardinality. Thus the complete algorithm takes $O(n^4)$ time and produces a collection of convex sets with cardinality within a $O(\log n)$ factor from the optimal one. An illustrative example is given in Figure 4 a). Notice that a more efficient $O(n^3)$-time randomized greedy heuristic can be useful in practical applications. In order to do this, we compute for each iteration the maximal blue island anchored at a random blue point.

We note that in data mining and machine learning applications, the main goal is to separate blue from red points. Thus it is possible that overlapping islands may appear and the above method can be applied. In other applications, as visualization and computer graphics, the computation of disjoint pieces is of interest. A similar greedy method works to approximate a class region by using pairwise-disjoint convex sets. Once a blue island has been obtained, simply recolor all of its elements red, and proceed with the next iteration. With this process it may occur that the cardinality of the solution obtained could be arbitrarily large compared with the obtained by allowing the islands to intersect. For example, let S be the point set with $2k$ points (k blue and k red) as shown in Figure 4 b). We observe that k blue points, $k \geq 4$, can be covered with at most three islands which intersect (two islands if k is even). However, if we require blue islands with disjoint convex hulls, we cannot cover the k blue points with less than $\lfloor \frac{k}{3} \rfloor + 1$ disjoint islands. From above discussion we get the following.

Theorem 3 *There is an $O(n^4)$-time algorithm with approximation ratio $\log n$ for the Convex Polygon Class Cover problem both for disjoint or non-disjoint convex sets.*

Acknowledgements

Authors Díaz-Báñez, Pérez-Lantero, Urrutia and Ventura were supported by Spanish MEC grants MTM2006-03909 and MTM2009-08625.

References

[1] P. K. Agarwal and S. Suri. *Surface Approximation and Geometric Partitions* SODA '94: Proc. 5th annual ACM-SIAM Symp. on Discrete Algorithms. ACM Press, 24-33, 1994.

[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] J. Akiyama and J. Urrutia. *Simple alternating path problem*, Discrete Math. 84,(1990), 101-103

[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, 161-167, ACM, 1985.

[5] K. P. Bennett and E. J. Bredensteiner. Duality and geometry in SVM classifiers. In Proc. 17th International Conf. on Machine Learning, pages 5764. Morgan Kaufmann, San Francisco, CA, 2000.

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

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

[8] 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, (2008), 684-687.

[9] 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), 453-470.

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

[11] 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.

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

[13] D. Eppstein, M. Overmars, G. Rote, and G. Woeginger. *Finding minimum area k-gons.* Discrete and Computational Geometry, 7, (1992), 45-58.

[14] P. Fischer. *Sequential and parallel algorithms for finding a maximum convex polygon.* Computational Geometry: Theory and Applications, 7, (1997), 187-200.

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

[16] A. Kaneko and M. Kano. *Discrete geometry on red and blue points in the plane - A survey.* Discrete and Computational Geometry Algorithms Combin., 25, Springer, (2003), 551-570.

[17] M. Kudo, Y. Torii, Y. Mori, and M. Shimbo. Approximation of class regions by quasi convex hulls. Pattern Recognition Letters, 19, (1998), 777-786.

[18] M. Kudo, A. Nakamura, I. Takigawa. Classification by Reflective Convex Hulls. In 19th International Conference on Pattern Recognition (ICPR 2008), Florida, USA. IEEE 2008.

[19] S. Tokunaga. *On a straight-line embedding problem of graphs.* Discrete Math. 150, (1996), 371-378.