Optimal Line Bipartitions of Point Sets *

Olivier Devillers † Matthew J. Katz ‡

Abstract

Let S be a set of n points in the plane. We study the following problem: Partition S by a line into two subsets S_a and S_b such that $max\{f(S_a), f(S_b)\}$ is minimal, where f is any monotone function defined over 2^S. We first present a solution to the case where the points in S are the vertices of some convex polygon and apply it to some common cases — $f(S')$ is the perimeter, area, or width of the convex hull of $S' \subseteq S$ — to obtain linear solutions (or $O(n \log n)$ solutions if the convex hull of S is not given) to the corresponding problems. This solution is based on an efficient procedure for finding a minimal entry in matrices of some special type, which we believe is of independent interest. For the general case we present a linear space solution which is in some sense output sensitive. It yields solutions to the perimeter and area cases that are never slower and often faster than the best previous solutions.

1 Introduction

Let S be a set of n points in the plane. We wish to partition S by a line into two subsets S_a and S_b, such that $max\{f(S_a), f(S_b)\}$ is minimal, where f is some real-valued monotone function that is defined over the collection of subsets of S. (f is *monotone* if $S_1 \subseteq S_2 \Rightarrow f(S_1) \leq f(S_2)$, for $S_1, S_2 \subseteq S$.) The problem of efficiently finding an optimal bipartition in respect to some criterion is the simplest form of the more general k-partition problem, and it has been studied extensively, see e.g. [1, 2, 3, 6, 7, 8, 9, 10, 11, 13]. The problem considered here was studied by Mitchell and Wynters [12] who present solutions for two instances of the problem, in which $f(S')$ is either the perimeter or area of the convex hull of S'. Their solutions require $O(n^3)$ time and $O(n)$ space. They also present an $O(n^2)$ time $O(n^2)$ space solution to the perimeter case. (In their paper they also solve the *minsum* versions of these cases, i.e., find a line bipartition for which $f(S_a) + f(S_b)$ is minimal.) Rokne et al. [16] present an $O(n^2 \log n)$ time and $O(n)$ space algorithm for the four cases considered in [12]. Both Mitchell and Wynters and Rokne et al. claim that an optimal bipartition is necessarily a line bipartition. However, Glozman et al. [5] show that this claim is false except for the minsum perimeter case.

*Part of this work was done while the second author was visiting INRIA Sophia-Antipolis.

†INRIA, BP 93, 06902 Sophia-Antipolis cedex (France), E-mail: Olivier.Devillers@sophia.inria.fr.

‡Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands, E-mail: matya@cs.ruu.nl. Supported by the Dutch Organization for Scientific Research (N.W.O.).

Here, we first consider the restricted problem where the points in S are in convex position, i.e., they are the vertices of some convex polygon P. For a portion P' of (the boundary of) P whose corresponding set of points is S', we often write $f(P')$ instead of $f(S')$. For the restricted problem we present a solution of complexity $O(g(n) + n \cdot h(n))$, where $g(n)$ is the complexity of the preprocessing (if required), and $h(n)$ is the complexity of maintaining the value $f(P')$ under insertions and deletions of extreme vertices to/from P', where P' is a portion of (the boundary of) P. We apply this solution to some common cases, and obtain the following results. (The width case is especially interesting since it is not obvious how to maintain the width of P' in constant time per operation.)

Theorem 1 *Let S be a set of n points in convex position in the plane, and assume that the convex hull of S is known. It is possible to compute an optimal line bipartition of S with respect to the perimeter, area, or width functions in linear time. For the width function, an optimal line bipartition is also an optimal (general) bipartition.*

Our solution is based on a procedure for searching for a minimal entry in matrices M of the following kind, which we believe is of independent interest. Let $A = (a_{i,j}), B = (b_{i,j})$ be two $m \times n$ matrices of real values, such that the rows and columns of A (resp. B) define increasing (resp. decreasing) sequences. The corresponding matrix M is a $m \times n$ matrix $M = (m_{i,j})$ where $m_{i,j} = max\{a_{i,j}, b_{i,j}\}$. If A and B are, for example, discrete representations of two surface patches in 3-D lying above a rectangular region in the plane, then the procedure computes the lowest point lying on the upper envelope of these surfaces. In [17], Sharir considers this problem for 0-1 matrices A and B. His solution examines $O((m+n)\log(m+n))$ entries of M, while our solution examines only $O(m+n)$ entries.

We now return to the non-restricted problem. Clearly it is enough to consider all the lines passing through a pair of points a, b of S. Every such line l defines two possible bipartitions (a belongs to the right half plane and b to the left half plane, or vice versa). We present a solution to the non-restricted problem which is sensitive to some value k defined in Section 5. The problem of establishing tight bounds for k is still open; k is clearly at most quadratic but the authors believe that k is subquadratic. In any case, in practical situations k is almost always linear or nearly linear. (For some very specialized point sets k is larger, but still far from quadratic.) More precisely the complexity of the algorithm is $O((n+k)h(n))$, where $h(n)$ is typically a polylogarithmic factor. We apply this solution to some common cases, and obtain the following results.

Theorem 2 *Let S be a set of n points in the plane. It is possible to compute an optimal line bipartition of S with respect to the perimeter or area functions in $O((n+k)\log^2 n)$ time (or in randomized $O((n+k)\log n)$ time) and linear space, where k is as above.*

Thus, whenever k is less than $O(n^2)/\log n$ (or $O(n^2)$ for the randomized version), our algorithm is more efficient than the previous algorithms. (As mentioned, it is possible that k is always less than $O(n^2)$; we were unable to prove or disprove this claim. However, in many examples k is clearly less than $O(n^2)$.)

The paper is organized as follows. In Section 2 we describe the matrix searching procedure. In Section 3 we present the (general) solution to the restricted problem, that is based on the procedure of Section 2, and in Section 4 we apply the solution of Section 3 to some common functions. The solution to the non-restricted problem is presented in Section 5.

2 A Matrix Searching Procedure

Let $A = (a_{i,j}), B = (b_{i,j})$ be two $m \times n$ matrices of real values, such that the rows and columns of A (resp. B) define increasing (resp. decreasing) sequences. Define a third $m \times n$ matrix $M = (m_{i,j})$ by $m_{i,j} = max\{a_{i,j}, b_{i,j}\}$.

In this section we describe an efficient procedure for finding an entry $m_{k,l}$ of M such that $m_{k,l} \leq m_{i,j}$, for $1 \leq i \leq m$ and $1 \leq j \leq n$. The procedure finds such an entry of M after inspecting only $O(m+n)$ entries of M. The sequence of entries that are inspected by the procedure forms a continuous path in M. This latter property is crucial for some of our applications.

We first observe that for any $1 \leq j \leq n$, the j'th column satisfies exactly one of the following conditions.

$a_{1,j} \geq b_{1,j}$: The "best" entry in this column (i.e., the entry for which the maximum between the a-value and the b-value is minimal) is therefore $m_{1,j}$. (Since the sequence $a_{1,j}, \ldots, a_{m,j}$ is increasing while the sequence $b_{1,j}, \ldots, b_{m,j}$ is decreasing.) We set i_j to 0.

$a_{m,j} \leq b_{m,j}$: The best entry in this column is therefore $m_{m,j}$. We set i_j to m.

$a_{1,j} < b_{1,j}$ and $a_{m,j} > b_{m,j}$: The best entry in this column is therefore either the entry corresponding to the meeting point of the curves defined by the sequences $a_{1,j}, \ldots, a_{m,j}$ and $b_{1,j}, \ldots, b_{m,j}$, if such an entry exists (i.e., if the meeting point corresponds to a sample point), or one of the two entries adjacent to the meeting point of these curves. We set i_j to the index such that $m_{i_j,j} = b_{i_j,j}$ and $m_{i_j+1,j} = a_{i_j+1,j}$; the best entry is thus either $m_{i_j,j}$ or $m_{i_j+1,j}$ (see Figure 1).

Thus, if we begin our search for the best entry of the j'th column by inspecting some arbitrary entries $a_{i,j}$, $b_{i,j}$ of the j'th columns, then we can determine easily whether $m_{i,j}$ is the desired entry (possibly by inspecting one of its vertically adjacent entries). If $m_{i,j}$ is not the desired entry, we can immediately say which direction (i.e., upwards or downwards) leads to the desired entry.

We next observe that

Lemma 3 *i_j is decreasing.*

Proof. We will prove that $i_j \geq i_{j+1}$. The claim is trivial if $i_j = 0$ or $i_{j+1} = m$. (Assume, for example, that $i_j = 0$. Then $m_{1,j} = a_{1,j} \geq b_{1,j}$, and since $a_{1,j+1} \geq a_{1,j}$ and $b_{1,j} \geq b_{1,j+1}$,

$$A = \begin{pmatrix} a_{1,1} & \leq \ldots \leq & a_{1,n} \\ \wedge & & \wedge \\ \vdots & & \vdots \\ \wedge & & \wedge \\ a_{m,1} & \leq \ldots \leq & a_{m,n} \end{pmatrix} \qquad B = \begin{pmatrix} b_{1,1} & \geq \ldots \geq & b_{1,n} \\ \vee & & \vee \\ \vdots & & \vdots \\ \vee & & \vee \\ b_{m,1} & \geq \ldots \geq & b_{m,n} \end{pmatrix}$$

$$M = \begin{pmatrix} b_{1,1} & \geq \ldots \geq & b_{1,j} & \ldots & \\ \vee & & \vee & & \\ \vdots & & \vdots & & \\ \vee & & \vee & & \\ b_{i_j,1} & \geq \ldots \geq & b_{i_j,j} & & \\ & & a_{i_j+1,j} & \leq \ldots \leq & a_{i_j+1,n} \\ & & \wedge & & \wedge \\ & & \vdots & & \vdots \\ & & \wedge & & \wedge \\ & \ldots & a_{m,j} & \leq \ldots \leq & a_{m,n} \end{pmatrix}$$

Figure 1: The matrix M

we have $a_{1,j+1} \geq b_{1,j+1}$, and therefore $i_{j+1} = 0$.) Assume therefore that $1 \leq i_j < m$ and thus $m_{i_j,j} = b_{i_j,j}$ and $m_{i_j+1,j} = a_{i_j+1,j}$. The monotonicity of row $i_j + 1$ of matrices A and B ensures that $m_{i_j+1,j+1} = a_{i_j+1,j+1}$, since $a_{i_j+1,j+1} \geq a_{i_j+1,j} \geq b_{i_j+1,j} \geq b_{i_j+1,j+1}$. Thus the transition between the b-values and the a-values in column $j+1$ cannot appear below this transition in the j-th column, which implies that $i_j \geq i_{j+1}$. $\square$

We are now ready to describe the procedure for finding a best entry of M. First compute the best entry $m_{i,1}$ of the first column of M, by inspecting the entries of the first column one by one beginning at the bottom entry and stopping once the best entry is encountered. According to the remark just above Lemma 3 this involves at most one inspection above the best entry. Now we inspect the entry $m_{i,2}$ and its adjacent entries in the second column. If either $m_{i,2}$ or $m_{i+1,2}$ is the best entry of the second column, then we move on to the third column (to $m_{i,3}$). Otherwise, we know by Lemma 3 that the best entry of the second column is above $m_{i,2}$, and we search for it (beginning at $m_{i,2}$ and moving upwards in the second column). We continue in this way until we reach the top entry of the last column. The best entry of M is the best of the column-best entries that were found.

The following theorem summarizes the result of this section.

Theorem 4 *A minimal entry of M can be found in $O(m+n)$ time. The search for such an entry requires only $O(m+n)$ entry inspections, and the sequence of entries that are inspected by the procedure forms a continuous path in M.*

Remark: For our purposes we are only interested in the case where $m = n$, and, by the theorem above, the desired entry of M can be found in linear time in this case. However,

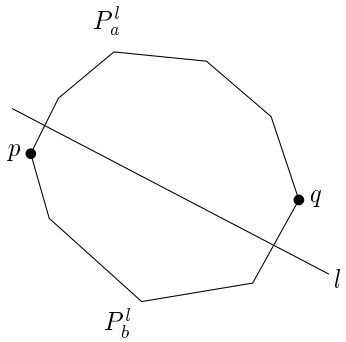

Figure 2: The line bipartition defined by p and q

in the general case where, say, $m >> n$, it is still possible to compute the desired entry of M efficiently; more precisely in $O(n \log m)$ time. This follows from the remark just above Lemma 3 which implies that the best entry of the j'th column can be found in $O(\log m)$ time.

3 The Solution to the Restricted Problem

In this section we assume that the points in S are the vertices of some convex polygon P. A pair of distinct points $p, q \in S$ define a line bipartition of S as follows (see Figure 2). Consider a line l passing through a point on (the boundary of) P immediately following p in clockwise direction and a point on P immediately following q. Then l partitions P into two portions and S into two subsets. Denote by P_a^l (resp. P_b^l) the portion of (the boundary of) P lying above (resp. below) l. Denote by S_a^l (resp. S_b^l) the subset of points of S lying above (resp. below) l. Clearly every bipartition of S by a line into two non empty subsets is obtained by a pair of points of S in this way. Thus the total number of such bipartitions is $\binom{n}{2}$, and we want to find such a bipartition $\{S_a, S_b\}$ such that $max\{f(S_a), f(S_b)\}$ is minimal, where f is a monotone function defined over 2^S.

First we observe that if $f(P_a^l) \geq f(P_b^l)$ for some line l, then any line that cuts P below l defines a worse partition. Thus the best line bipartition is either defined by a line crossing l inside P, by l itself, or by a line that cuts P above l. The latter case is treated by the algorithm below by recursively exploring portions of P, and the first case is treated by searching in an appropriate matrix. We now describe the j-th stage of our algorithm which consists of $O(\log n)$ stages. After the j-th stage we are left with a portion P_j of P of size (number of vertices) $n/2^j$, such that the bipartitions that we still need to consider at this point are only those that are defined by a line that intersect P_j at two points.

The j-th stage
Let P_{j-1} be the portion of P that is left after the $(j-1)$'th stage ($P_0 = P$), and assume the vertices of P_{j-1} in clockwise order are $p_1, \ldots, p_{n/2^{j-1}}$. The j-th stage consists of two steps.

Step 1

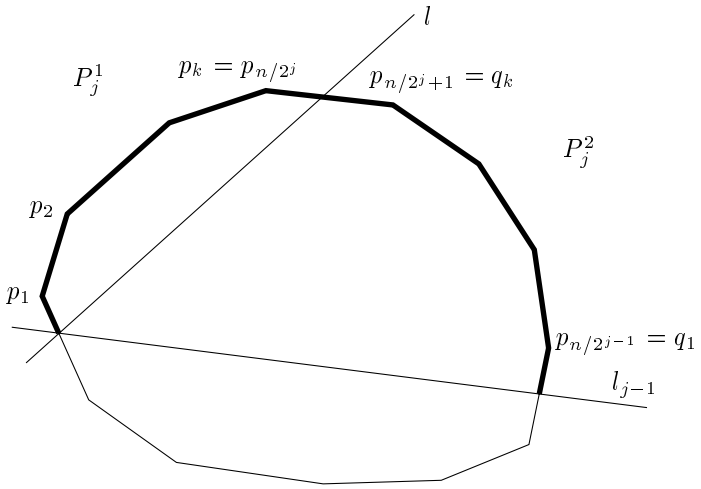

Figure 3: Step 1 - the recursive division of P_{j-1}

Consider any bipartition of S that splits P_{j-1} into two portions P_j^1 and P_j^2, such that the vertices of P_j^1 are $p_1, \ldots, p_{n/2^j}$ and those of P_j^2 are $p_{n/2^j+1}, \ldots, p_{n/2^{j-1}}$ (see Figure 3). Let l be the line defining this partition. Compute the values $f(S_1)$ and $f(S_2)$, where S_1 (resp. S_2) is the subset of S defined by l containing the vertices of P_j^1 (resp. P_j^2). If $f(S_1) \geq f(S_2)$, then for any bipartition of S by a line l' that intersects P_j^2 at two points, we have $f(S_1') \geq f(S_1)$, where S_1' is the subset of the bipartition defined by l' that contains S_1. Thus we can discard all these partitions from further consideration. An analogue argument applies when $f(S_2) \geq f(S_1)$. As the line l we take one of the two lines intersecting P between the vertices $p_{n/2^j}$ and $p_{n/2^j+1}$ and at the left or right endpoint of P_{j-1}. The appropriate values of f for these two lines can be computed in $O(n/2^j h(n))$ time from the values associated with the partition defined by l_{j-1}, where $h(n)$ is the complexity of maintaining the value $f(P')$, for a portion P' of (the boundary of) P, under insertions and deletions of extreme vertices to/from P'. Assume that we take the left line as l. Now, if $f(S_1) \geq f(S_2)$, then we still have to consider all the partitions that are defined by a line intersecting P_j^1 at two points, and if $f(S_1) < f(S_2)$, then we still have to consider all the partitions that are defined by a line intersecting P_j^2 at two points. In both cases we also have to consider the partitions that are defined by a line intersecting both P_j^1 and P_j^2. In the former case, we set l_j to the line l, i.e, the line that is defined by the endpoints of P_j^1 and we set P_j to P_j^1, and in the latter case, we set l_j to the line which was the alternative choice for l, i.e., the line that is defined by the endpoints of P_j^2 and we set P_j to P_j^2.

Step 2

In this step we consider the partitions that are defined by a line intersecting both P_j^1 and P_j^2. Let $p_1, \ldots, p_k$ be the points in P_j^1 in clockwise order, and let $q_1, \ldots, q_k$ be the points in P_j^2 in counterclockwise order ($k = n/2^j$). We define a $k \times k$ matrix $M = (m_{i,j})$ as follows. Consider the bipartition of S that is defined by the points p_i and q_j. The entry $m_{i,j}$ is the maximum between the pair of real values $a_{i,j}, b_{i,j}$, where $a_{i,j}$ is the value of f for the subset of the bipartition that has vertices of P that do not belong to P_{j-1}, and $b_{i,j}$ is the value

of f for the second subset. At this point, we would like to use the procedure described in the previous section. For this we need to check that the corresponding matrices A and B have the required properties. Indeed, it is easy to see that the rows and columns of A form increasing sequences while the rows and columns of B form decreasing sequences. For example, if we fix i and let j change from 1 to k, then the subset of the bipartition that has vertices of P that do not belong to P_{j-1} only grows, and thus the value of f only grows. Applying the procedure of the previous section to the matrix M yields the best partition that is defined by a mixed pair of points. The cost of applying the procedure is $O(kh(n)) = O(n/2^j h(n))$. Note that we do not need to compute the entire matrix M, we only need to compute $O(k)$ entries as required by the procedure of the previous section. Moreover, the property of the procedure concerning the sequence of entries that is generated allows us to update the values that are computed dynamically, since the partition of the current entry is obtained from the previous one by moving a single extreme point from one of the sets to the other.

Thus at the end of this stage we are left with the set P_j whose size is only $n/2^j$. The total running time of the j-th stage is $O(n/2^j h(n))$, and thus the total running time of the solution is $O(g(n)+n\cdot h(n))$. The term $g(n)$ is the cost of the preprocessing that is required for the dynamic updates.

The following theorem summarizes the result of this section.

Theorem 5 *Let S be a set of n points in the plane, and assume that the points in S are the vertices of some convex polygon P. Let f be some monotone function that is defined over the collection of subsets of S. A line bipartition for which the maximal value (between the two corresponding values of f) is minimal can be computed in $O(g(n)+n\cdot h(n))$, where $g(n)$ is the cost of the preprocessing that is required, and $h(n)$ is the complexity of maintaining the value $f(P')$ under insertions and deletions of extreme vertices to/from P', where P' is a portion of (the boundary of) P.*

4 Some Common Cases

In this section we show how the convex hull can be maintained in total time that is linear in n, for the sequence of insertions and deletions. Perimeter, area, and width can be maintained within the same time bound.

Let $P = \{p_1 \dots p_k, q_k, \dots q_1\}$ be a convex polygon and $P_{i,j} = \{p_1 \dots p_i, q_j, \dots q_1\}$ be a subset of P. During the algorithm of the previous section, $P_{i,j}$ can be transformed into $P_{i+1,j}$ (by insertion of p_{i+1}) or into $P_{i,j-1}$ (by deletion of q_j). Both operations are straightforward and the area and perimeter can be easily updated, since the transformation consists of adding or removing a single triangle to $P_{i,j}$.

The maintenance of the width requires some additional effort. The width is defined by antipodal pairs [15], thus to update the width it is enough to examine the new antipodal pairs. We define p' as the first antipodal point of p_i turning clockwise around P beginning at p_i, q' the last antipodal point of q_j, and r' the point antipodal to the line segment p_i, q_j

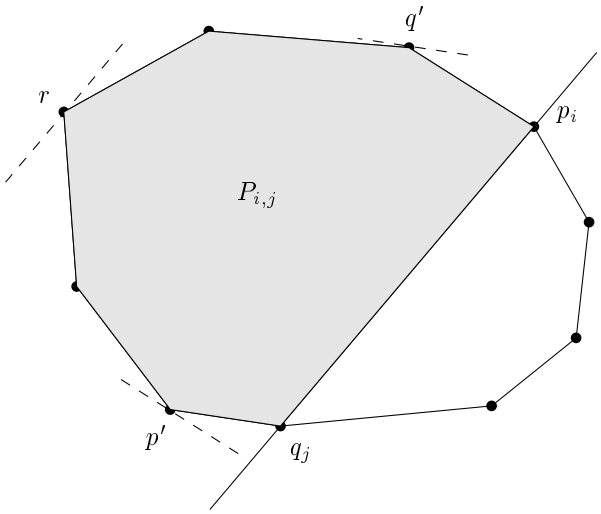

Figure 4: The interesting antipodal pairs of $P_{i,j}$

(see Figure 4). In polygon $P_{i,j}$ the vertices antipodal to p_i are the ones on the chain from p' to r', and the vertices antipodal to q_j are on the chain from r' to q'. Now, we just have to notice that if p_{i+1} is inserted q' and r' are also moving clockwise, and if p_j is deleted p' and r' are moving clockwise. Thus during any insertion-deletion sequence going from $P_{1,k}$ to $P_{k,1}$ the total number of antipodal pairs considered is linear. We thus obtain the first theorem stated in the Introduction.

Theorem 1 *Let S be a set of n points in convex position in the plane, and assume that the convex hull of S is known. It is possible to compute an optimal line bipartition of S with respect to the perimeter, area, or width functions in linear time. For the width function, an optimal line bipartition is also an optimal (general) bipartition.*

Remark: It is easy to see that the theorem above also holds for the diameter function, and that an optimal line bipartition in this case is also an optimal (general) bipartition. Thus we obtain an alternative solution, with the same bounds, to the one given by Asano et al. [2] for the problem of computing an optimal bipartition of S as above with respect to the diameter function. Asano et al. also solve this problem for an arbitrary set of points in $O(n \log n)$ time.

5 The Solution to the Non-Restricted Problem

In this section, we remove the assumption that the points of S are in convex position. Nevertheless, there exists some similarity between this part of the paper and the previous part; here too we will search in some matrix whose rows are bitonic (first decreasing and then increasing).

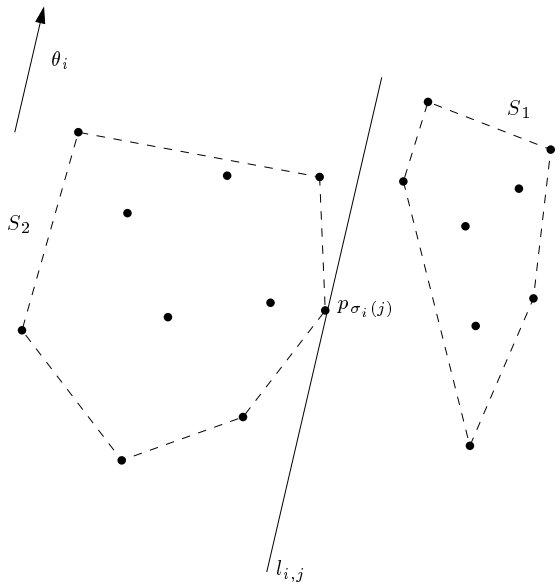

Figure 5: Definition of $l_{i,j}$

It is clear that any line bipartition can also be defined by a line passing through a pair of points of S, so we may restrict ourselves to the set of such lines. However, we prefer to search in a larger set of lines consisting of the lines that pass through a point of S and are parallel to a line that passes through two points of S.

Denote by $N = \binom{n}{2}$ the number of pairs of points of S, and let $\theta_1 < \theta_2 < \ldots < \theta_N$ be the slopes defined by these pairs. For a fixed slope θ_i, denote by σ_i the permutation of the points of S corresponding to the direction $\theta_i - \frac{\pi}{2}$, that is, a line of slope θ_i sweeping S from left to right will first encounter $p_{\sigma_i(1)}$ and then $p_{\sigma_i(2)}, p_{\sigma_i(3)} \cdots p_{\sigma_i(n)}$. Finally, denote by $l_{i,j}$ the line of slope θ_i passing through point $p_{\sigma_i(j)}$, and by $m_{i,j}$ the maximum between the two values of f for the bipartition defined by $l_{i,j}$ (see Figure 5). ($l_{i,0}$ is to the left of S, and $p_{\sigma_i(j)}$ is considered a member of the left subset.)

We will now examine some properties of the $N \times (n+1)$ matrix $M = (m_{i,j})$.

1. Line i of M is bitonic. When the line of direction θ_i sweeps S, it splits S into a subset S_1 which is decreasing from S to $\emptyset$ and a subset S_2 which is increasing from $\emptyset$ to S. Thus $m_{i,j} = \max(f(S_1), f(S_2))$, $j = 0, \ldots, n$, is bitonic.

2. The difference between σ_i and σ_{i+1} is small. More precisely, σ_{i+1} is obtained from σ_i by a single swap, that is applied to a pair of adjacent elements in σ_i (the pair of elements defining the slope θ_{i+1}).

3. If the best partition of row i is defined by $l_{i,j}$, and $p_{\sigma_i(j')} = p_{\sigma_{i+1}(j')}$ for $j' = j$ and $j' = j+1$ (i.e., the swap that is applied to σ_i does not involve the j-th or the $(j+1)$-th point), then the best partition of row $i+1$ is defined by $l_{i+1,j}$.

The reason is fairly simple. If the swap does not involve one of these points, then $m_{i,j'} = m_{i+1,j'}$ for $j' \in \{j-1, j, j+1\}$, since the corresponding partitions are identical. Thus $m_{i+1,j}$ is also the minimal value of the bitonic sequence of row $i+1$.

4. If the best partition of row i is defined by $l_{i,j}$, and $p_{\sigma_i(j')} \neq p_{\sigma_{i+1}(j')}$ for $j' = j$ or $j' = j+1$, then the best partition of row $i+1$ is defined by a line $l_{i+1,j'}$. where $j-2 \leq j' \leq j+2$.

   Assume that $p_{\sigma_{i+1}(j)} = p_{\sigma_i(j-1)}$ and $p_{\sigma_{i+1}(j-1)} = p_{\sigma_i(j)}$, then the only difference between row i and row $i+1$ is in the $(j-1)$-th column. We know that the sequence $m_{i+1,0}, \ldots, m_{i+1,j-2}$ is decreasing, and that the sequence $m_{i+1,j}, \ldots, m_{i+1,n}$ is increasing. Thus the minimal value of the bitonic sequence of row $i+1$ is $m_{i+1,j-2}$, $m_{i+1,j-1}$, or $m_{i+1,j}$.

We will show how to locate the minimal value of the (implicit) matrix M in an efficient way. The idea here is to jump directly from row i whose optimal value corresponds to a partition by line $l_{i,j}$, to the next row i', $i' > i$, where a swap involving $p_{\sigma_i(j)}$ or $p_{\sigma_i(j+1)}$ occurs. In other words, if the optimal value of the i-th row is in the j-th column, we would like to jump directly to the first row i' whose optimal value may change. We denote by k the number of such jumps in the algorithm.

We first fix the notation (see Figure 6). Assume that for some row i the optimal bipartition splits S into S_1 and S_2 with $S_2 = \{p_{\sigma_i(1)} \cdots p_{\sigma_i(j_0)}\}$. γ_1 is the slope of the edge following p_{j_0+1} on (the convex hull) of S_1, γ_2 is the slope of the edge preceding p_{j_0} on S_2, and γ_t is the slope of the common tangent of S_1 and S_2 whose slope is greater than θ_i. Clearly, the next slope for which the swap involves either p_{j_0} or p_{j_0+1} is defined by one of these three orientations. γ_1 corresponds to the swap of p_{j_0+1} and p_{j_0+2}, γ_2 corresponds to the swap of p_{j_0} and p_{j_0-1}, and γ_t corresponds to the swap of p_{j_0} and p_{j_0+1}. Thus the next relevant slope and the next row in the matrix that we need to examine, is row i' such that $\theta_{i'} = \min\{\gamma_1, \gamma_2, \gamma_t\}$.

Thus we can compute the minimal entry of M using an incremental algorithm that computes at the i-th row the slopes $\gamma_2, \gamma_t, \gamma_1$, and then, depending on whether γ_2, γ_t, or γ_1 is minimal, computes $m_{i',j}$ for j respectively in $[j_0-2, j_0]$, $[j_0-1, j_0+1]$ or $[j_0, j_0+2]$. These three values correspond to bipartitions that are obtained from the optimal bipartition of row i by adding and deleting a constant number of points. Finally, the minimal value of row i' is the minimal value of the three.

The complexity of the algorithm clearly depends on k, but unfortunately no tight bound for k has been established. However, notice that (i) $N = O(n^2)$ is a trivial upper bound for k, and therefore the *proved* complexity of our algorithm is similar to the best previous results, and (ii) in many examples k is usually much smaller. Indeed, in a dual setting (where points are replaced by lines and vice-versa), the line separating between S_1 and S_2 maps into a point belonging to a monotone path in an arrangement of lines. k is the number of arrangement vertices traversed by this monotone path, and for many arrangements of lines the longest monotone path (in this sense) is much smaller than the entire arrangement. Thus the sensitivity to k of the algorithm is a real advantage, except maybe for some very special cases.

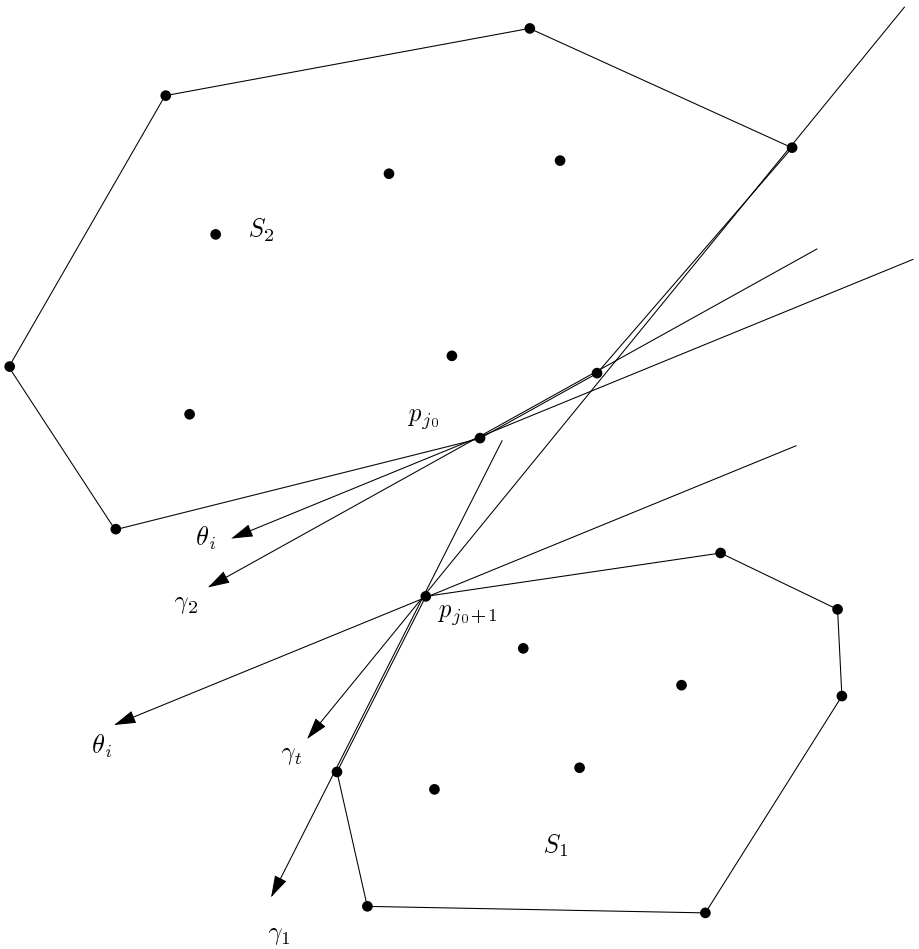

Figure 6: The next slope where the optimal bipartition might change

The following theorem summarizes the result of this section.

Theorem 6 *Let S be a set of n points in the plane and let f be some monotone function that is defined over the collection of subsets of S. A line bipartition for which the maximal value (between the two corresponding values of f) is minimal can be computed in $O((n+k)h(n))$, where $h(n)$ is the complexity of maintaining the convex hull of S' and the value $f(S')$, where S' is a subset of S, under insertions and deletions to/from S', and of some basic queries concerning the convex hulls of the two sets consisting of the current partition, and $k = O(n^2)$ is as above.*

Perimeter and area cases

If $f(S')$ is the perimeter or the area of the convex hull of S', then the basic problem is the maintenance of the convex hull under insertions and deletions. After the convex hull has been updated, it is easy to update f. The convex hull can be maintained dynamically in $O(\log^2 n)$ time [14, 15] or in randomized $O(\log n)$ time [4] per operation. Both methods allow the maintenance of the perimeter and the area in logarithmic time, and also the common tangents of the two convex polygons can be computed in logarithmic time. We thus obtain the second theorem stated in the Introduction.

Theorem 2 *Let S be a set of n points in the plane. It is possible to compute an optimal line bipartition of S with respect to the perimeter or area functions in $O((n+k)\log^2 n)$ time (or in randomized $O((n+k)\log n)$ time) and linear space, where k is as above.*

References

[1] Pankaj K. Agarwal and M. Sharir. Planar geometric location problems. *Algorithmica*, 11:185–195, 1994.

[2] Te. Asano, B. Bhattacharya, J. M. Keil, and F. Yao. Clustering algorithms based on minimum and maximum spanning trees. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 252–257, 1988.

[3] D. Avis. Diameter partitioning. *Discrete Comput. Geom.*, 1:265–276, 1986.

[4] K. L. Clarkson, K. Mehlhorn, and R. Seidel. Four results on randomized incremental constructions. *Comput. Geom. Theory Appl.*, 3(4):185–212, 1993.

[5] A. Glozman, K. Kedem, and G. Shpitalnik. Finding optimal bipartitions of points, 1994. manuscript.

[6] Alex Glozman, Klara Kedem, and Gregory Shpitalnik. On some geometric selection and optimization problems via sorted matrices. In *Proc. 4th Workshop Algorithms Data Struct.*, volume 955 of *Lecture Notes in Computer Science*, pages 26–37. Springer-Verlag, 1995.

[7] J. Hagauer and G. Rote. Three-clustering of points in the plane. In T. Lengauer, editor, *Proc. 1st Annu. European Sympos. Algorithms (ESA '93)*, volume 726 of *Lecture Notes in Computer Science*, pages 192–199. Springer-Verlag, 1993.

[8] J. Hershberger and S. Suri. Finding tailored partitions. *J. Algorithms*, 12:431–463, 1991.

[9] J. W. Jaromczyk and M. Kowaluk. An efficient algorithm for the Euclidean two-center problem. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 303–311, 1994.

[10] M. J. Katz. Improved algorithms in geometric optimization via expanders. In *Proc. 3rd Israel Symposium on Theory of Computing and Systems*, pages 78–87, 1995.

[11] M. J. Katz and M. Sharir. An expander-based approach to geometric optimization. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 198–207, 1993.

[12] J. S. B. Mitchell and E. L. Wynters. Finding optimal bipartitions of points and polygons. In *Proc. 2nd Workshop Algorithms Data Struct.*, volume 519 of *Lecture Notes in Computer Science*, pages 202–213. Springer-Verlag, 1991.

[13] C. Monma and S. Suri. Partitioning points and graphs to minimize the maximum or the sum of diameters. In *Graph Theory, Combinatorics and Applications (Proc. 6th Internat. Conf. Theory Appl. Graphs)*, volume 2, pages 899–912, New York, NY, 1991. Wiley.

[14] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.*, 23:166–204, 1981.

[15] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, NY, 1985.

[16] J. Rokne, S. Wang, and X. Wu. Optimal bipartitions of point sets. In *Proc. 4th Canad. Conf. Comput. Geom.*, pages 11–16, 1992.

[17] Micha Sharir. A near-linear algorithm for the planar 2-center problem. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 106–112, 1996.