Optimal Bipartitions of Point Sets (Extended Abstract)

Jon Rokne *, Shangzhi Wang †, Xiaolin Wu ‡

Summary Optimal bipartition of a set of points in the plane under various measures is studied. Several improvements in time and space complexities are made over existing algorithms.

1 Introduction

Bipartition of a point set is the simplest form of the K-clustering problem, i.e., when $K = 2$. It has been extensively studied in the framework of computational geometry [1, 2, 3, 9, 6, 11, 12]. Let S be a set of N points in the plane, and $C(S)$ be a dissimilarity measure of the points in S. The common choices of $C(S)$ in clustering practice are: the variance of S (the sum of the squared distances of all pairs of points in S divided by the population $|S|$), the weighted variance of S (variance timed by $|S|$), the perimeter, the diameter, and the area of the convex hull of S. In the sequel the terms perimeter, diameter and area will be used in the context of a point set with the implication that these quantities are defined on the convex hull of the point set. The goal of optimal bipartition of a point set is to split S into two subsets S_1 and S_2 such that a cost function $F(C(S_1), C(S_2))$ is minimized. In this extended abstract we consider two natural cost functions: $F = C(S_1) + C(S_2)$ and $F = \max\{C(S_1), C(S_2)\}$. Thus we have two optimization criteria: minimum sum and minimum maximum, and shortened to minsum and minmax in the remainder of the text.

For the minmax criterion and the measure of set diameter, a very efficient $O(N \log N)$ algorithm based on the maximum spanning tree was proposed by Asano, Bhattacharya, Keil and Yao [2], which was an improvement over an earlier algorithm due to Avis [3]. An $O(N^2)$ algorithm was given by Monma and Suri to minimize the sum of two set diameters. Recently Mitchell and Wynters [12] proposed an $O(N^3)$ time and $O(N)$ space algorithm to compute optimal bipartitions of point sets for minisum and minmax criteria and for the measures of set perimeter and area. They also gave an $O(N^2)$ time, but $O(N^2)$ space algorithm for minsum perimeter, minsum area, and minmax perimeter bipartitions. In this extended abstract we present a compromise between the time and space complexities for the problem. An $O(N^2 \log N)$ time and $O(N)$ space algorithm is devised for both minsum and minmax criteria and for cluster dissimilarity measures of set perimeter and area. We will also give an $O(N^2)$ time and $O(N)$ space algorithm for optimal bipartitions under the variance-based cluster measures and an $O(N^2)$ time and $O(N^2)$ space algorithm for the convex-hull-based cluster measures. The proposed algorithms are simple both conceptually and structurally,

*Department of Computer Science, University of Calgary, Calgary, Alberta. Supported by the NSERC.
†Department of Mathematics, Beijing Teacher's College.
‡Department of Computer Science, University of Western Ontario, London, Ontario. Supported by the NSERC.

and hence can be implemented with ease.

2 Optimal Bipartition Algorithms

All our algorithms will be developed under the following important geometric property of optimal bipartition of a point set.

LEMMA 1 *For variance-based and convex-hull-based measures C and under the minmax or minsum criterion, optimal bipartition of S must yield linearly separable S_1 and S_2.*

Proof. Trivial and omitted. □

Due to the above lemma, we only need to examine all possible bipartitions formed by a line separator. Since the line connecting any pair of two points in S defines a bipartition, there are $O(N^2)$ different bipartitions. Then a naive algorithm can evaluate the cost function for each bipartition and find the minimum. This requires $O(N^2M)$ time where M is the number of operations to evaluate the cost function. Clearly, $M = O(N)$ for variance-based cluster measures and $M = O(N \log N)$ for convex-hull-based cluster measures per F evaluation, if the $O(N^2)$ F evaluations required by the bipartition enumeration are not organized. To improve the efficiency we need an incremental scheme of bipartition enumeration. If only $O(1)$ points change their cluster memberships between two consecutive bipartitions, then the cost function F can be updated in $O(1)$ time for variance-based measures, and in $O(\log^2 N)$ time for convex-hull-based measures by using Overmars and van Leeuwen's dynamic convex hull maintenance technique [13].

2.1 Topological sweep

We use geometric duality to map points $p_i \in S$ into lines L_i. Then the bipartition given by the line p_ip_j in the original plane corresponds to the intersection point I_{ij} of L_i and L_j in the dual plane. The N dual lines form $O(N^2)$ intersections which are $O(N^2)$ bipartitions. After the above transform, we can use the topological sweeping technique [7] to sweep the arrangement of the $O(N)$ lines in the dual plane in $O(N^2)$ time and $O(N)$ space. Effectively we enumerate all bipartitions in the original plane by the sweep. A beauty of the topological sweep is that the sweeping topological line marches in so-called elementary step. The elementary step advances from I_{ij} to either I_{it} or I_{jt}. This means that a new bipartition is visited by rotating the line separator anchored at either p_i or p_j. Consequently, either p_j or p_i changes its cluster membership in such a step. $O(1)$ time suffices to update the cost function from the previous bipartition to the current one. Using topological sweeping we achieve an incremental enumeration of all bipartitions without explicit sorting, and thus conclude the following.

THEOREM 1 *Under the criteria of minsum variance, minsum weighted variance, minmax variance and minmax weighted variance, the optimal bipartition can be found in $O(N^2)$ time using $O(N)$ space.*

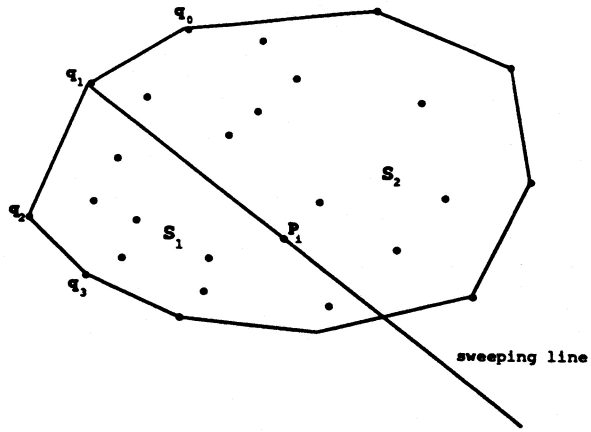

Figure 1: Sweep from an edge on a convex layer to enforce monotonic growth of a subset.

It is possible to sweep all faces of an arrangement of N hyperplanes in $\Re^d$ in $O(N^d)$ time [7]. This means that the above algorithm can be generalized to d-dimensional space with a time complexity of $O(dN^d)$ to compute the optimal bipartition of a set of N d-dimensional points. The coefficient d is there because $O(d)$ operations per update are required in $\Re^d$.

Unfortunately, the above technique cannot improve the angular sweeping algorithm for the convex-hull-based measures since the $O(\log^2 N)$ cost per point for dynamic convex hull maintenance overrides the $O(\log N)$ cost per point for sorting. But the improvement can be made by another sweep arrangement.

2.2 Layerwise angular sweep

The previous angular sweep and topological sweep generate an arbitrary sequence of insertion and deletion of points into and from the convex hulls of S_1 and S_2. This necessitates the use of hull tree data structures to dynamically maintain the convex hulls. However, on a second reflection, we can enforce S_1 and S_2 to grow (shrink) monotonically, thus make a uniform sequence of point insertions. First pick an edge q_1q_2 of the convex hull of S and sort all other points $p_i \in S$ by the angles $\angle p_iq_1q_2$ (see Figure 1). Then we sweep the line separator counterclockwise from q_1q_2 to q_1q_0, where as marked in Figure 1 q_0 is the vertex of the convex hull of S to the right of q_1. Call the point set to the left of the sweeping line S_1 and the remaining points belong to S_2. Clearly, S_1 monotonically grows and S_2 monotonically shrinks during the sweep.

Now consider how to dynamically maintain the convex hull of S_1 and incrementally compute the area and the perimeter of S_1. Figure 2 is a snap shot of the angular sweep starting from q_1q_2, where q_1a is the previous line separator and q_1b is the current line separator. Points a and b are adjacent in the sorted list of angles of p_i with respect to q_1q_2. It is clear from Figure 2 that q_1b must be an edge of the new convex hull of S_1 which just absorbed b, i.e., q_1 is always the support of the convex hull of previous S_1 with respect to the current point. In order to compute the convex hull of the new S_1, we need to find the other support c on the previous hull. To this end we walk along the old hull of S_1 clockwise from point a until the support c is found. Then we delete the chain $q_1a \cdots c$

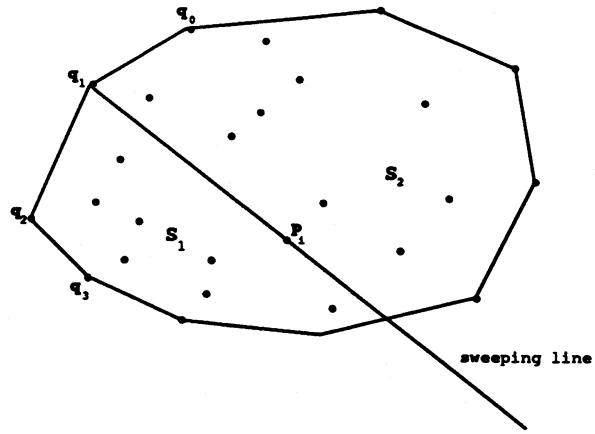

Figure 2: Dynamic convex hull updating and measuring.

from the old hull and replace it by q_1bc. In doing so we can compute the length of the chain $q_1a\cdots c$ if perimeter measure is used or the area of the concave region $q_1a\cdots cbq_1$ if the area measure is used for the purpose of updating the cost function. The cost of maintaining the convex hull of S_1 and measuring the hull at the current step is proportional to the length of deleted chain $q_1a\cdots c$. We can then charge the cost to the deleted S_1 hull points. Since the deleted points can never resurface in the sweep, the total cost of dynamic S_1 hull maintenance and measurement for marching the line separator from q_1q_2 to q_1q_0 is only $O(N)$. So despite that we may spend $O(N)$ time at a single step the amortized cost is still $O(1)$ per step.

The above process and analysis also apply to S_2 if we sweep the point set from q_1q_0 to q_1q_2 clockwise. By combining the results of the two sweeps of opposite directions we can find the best bipartition among all possible bipartitions formed by the line separator q_1p_i, $1 \leq i \leq N$. After this done we should remove q_1 from further considerations.

We will march on the convex hull of S, choosing the hull vertices q_2, q_3, ... as the new pivot in that order, and for each pivot carry out the same sweeps as described above for q_1. After all the vertices on the convex hull of S are so processed and removed from further considerations, we will move to the second convex layer of S and sweep angularly the remaining points of S from consecutive edges of the second convex layer in the same way as we did to the first convex layer (the convex hull) of S, and then move to the third, fourth, ... convex layers until all points are processed. In summary, the proposed algorithm sweeps convex layers inward, and then sweeps edges on a given convex layer. Starting from a given edge on a give convex layer the algorithm enumerates bipartitions radially.

All together $2N$ angular sweeps are carried out: N counterclockwise and N clockwise sweeps for S_1 and S_2 respectively with each $p_i \in S$ being a pivot of angular sweeps once. Thus $O(N^2 \log N)$ time is required to construct N sorted lists of angles. As we argued before, the amortized cost for incremental evaluation of the cost function during an angular sweep is $O(1)$ per step, or $O(N)$ per sweep, and it amounts to $O(N^2)$ for N sweeps. The convex layers can be constructed in $O(N \log N)$ time [5]. Thus the cost of sorting dominates and we reach the following conclusion.

THEOREM 2 *Under the criteria of minsum set perimeter, minsum set area, minmax set perimeter and minmax set area, the optimal bipartition can be found in $O(N^2 \log N)$ time using $O(N)$ space.*

Compared with the $O(N^2)$ time and $O(N^2)$ space algorithms in [12] which are based on visibility graphs, our layerwise angular sweeping algorithm is simple both conceptually and structurally. No complicated data structures are required. In fact, the dynamical maintenance of the convex hulls of S_1 and S_2 for our sweeping algorithm is simpler and faster than Preparata's on-line convex hull algorithm [14]. The former needs $O(1)$ time per update (after amortization) while the latter needs $O(\log N)$ per update. The former can simply represent the convex hull as a linear list while the latter needs a concatenable queue.

2.3 Layerwise sweep without sorting

As we saw above if the sweep is organized as such that a subset grows monotonically, then the dynamic convex hull maintenance needs to handle insertions only. As the result, an amortized $O(1)$ cost per update is achieved for both dynamic hull maintenance and measuring (computing the area or diameter of each subset). So the enumeration of $O(N^2)$ bipartitions needs only $O(N^2)$ time. The additional $O(\log N)$ factor is incurred by the angular sortings, a preprocessing to facilitate monotonic growth of subsets. If we can avoid explicit sorting while still having monotonic growth of subsets, then the time complexity of optimal bipartition for convex-hull-based measures can be reduced to $O(N^2)$. This is indeed possible if we are willing to use more space.

As in subsection 2.1 we can employ geometric duality to transform points p_i into lines L_i, and lines p_ip_j to points I_{ij} which correspond to different bipartitions. When we march along an arbitrary line L_i in the dual plane across points $I_{ij}, I_{ik}, I_{il} \cdots$, effectively we rotate a line separator anchored at p_i to pass $p_j, p_k, p_l \cdots$ and in that order. However, points $p_j, p_k, p_l \cdots$ appear, in general, alternatingly at two different sides of p_j, corresponding to an arbitrary schedule of insertions and deletions in and from a subset. But if we march along a line L_q whose dual is a vertex q on the convex hull of S in the original plane, then we can guarantee that all the points on L_q so passed are inserted into a same subset (equivalently deleted from the other subset). The order of insertions is the same as the order of points on L_q. This means that the line arrangement in the dual space implicitly contains the order information we need.

Now we are ready to present an $O(N^2)$ algorithm for optimal bipartition for convex-hull-based measures. We compute and save the line arrangement of L_i in $O(N^2)$ time and $O(N^2)$ space using any of the algorithms in [10, 8]. We also construct the convex layers of p_i in the original plane, and traverse the first layer (the convex hull of S) by consecutive edges and then the second, third, ... layers. Suppose that q_1q_2 is the current edge on the current convex layer. We march in the dual plane from the intersection point of L_{q_1} and L_{q_2} and along the line L_{q_1}. Then the sequence of points on the line L_{q_1} in the dual plane so visited gives us the exact sequence of bipartitions generated by the angular sweep as described in subsection 2.2, only this time no explicit sorting is required. Note that we need to march along L_{q_1} twice in two opposite directions, corresponding counterclockwise

and clockwise angular sweeps. In this way we achieve monotonic growth of S_1 or S_2, thus $O(1)$ cost per update suffices. Once L_{q_1} is finished, i.e., after all bipartitions given by the line q_1p_i are considered, we delete L_{q_1} from the line arrangement. Then the algorithm proceeds to the next edge on the current convex layer, and to the next convex layer, and so forth. Clearly, the deletion of line L_q from the line arrangement takes $O(N)$ time. The cost of angular sweeps in the original plane is the same as in subsection 2.2, thus the following theorem.

THEOREM 3 *Under the criteria of minsum set perimeter, minsum set area, minmax set perimeter and minmax set area, the optimal bipartition can be found in $O(N^2)$ time using $O(N^2)$ space.*

Acknowledgement

A constructive suggestion from Professor Derick Wood is deeply appreciated.

References

[1] A. Aggarwal, H. Imai, N. Katoh, and S. Suri, "Finding k points with minimum diameter and related problems", *J. of Algorithms*, vol. 12, pp. 38-56, 1991.

[2] T. Asano, B. Bhattacharya, M. Keil, and F. Yao, "Clustering algorithms based on minimum and maximum spanning trees", *Proc. 4th ACM Symp. Computational Geometry*, pp. 252-257, 1988.

[3] D. Avis, "Diameter partitioning", *Disc. and Comput. Geom.*, vol. 1, no. 3, p. 265-276, 1986.

[4] E. Boros and P. L. Hammer, "On clustering problems with connected optima in Euclidean spaces", *Discrete Math.*, vol. 75, pp. 81-88, 1989.

[5] B. M. Chazelle, "Optimal algorithms for computing depths and layers", *Proc. 21st Allerton Conf. on Comm. Control and Comput.*, pp. 427-436, 1983.

[6] F. Dehne and H. Noltemeier, "A computational geometry approach to clustering problems", *Proc. 1st ACM Symp. on Computational Geometry*, pp. 245-250, 1985.

[7] H. Edelsbrunner and L. J. Guibas, "Topologically sweeping an arrangement" *J. of Comput. Syst. Sci.*, vol. 38, pp. 165-194, 1989.

[8] H. Edelsbrunner, J. O'Rourke, and R. Seidel, "Constructing arrangements of lines and hyperplanes with applications", *SIAM J. Comput.* vol. 15, pp. 317-340, 1986.

[9] V. Capoyleas, G. Rote and G. Woeginger, "Geometric clustering," *J. Algorithm*, vol. 12, pp. 341-356, 1991.

[10] B. Chazelle, L. J. Guibas, and D. T. Lee, "The power of geometric duality," *Bit*, vol. 25, pp. 76-90, 1985.

[11] J. Hershberger and S. Suri, "Finding tailored partitions", *J. of Algorithms*, vol. 12, pp. 431-463, 1991.

[12] J. S. B. Mitchell and E. L. Wynters, "Finding optimal bipartitions of points and polygons," *Lecture Notes in Computer Science* no. 519, also in *Proc. of 2nd Workshop on Algorithms and Data Structures* pp. 202-213, 1991.

[13] M. H. Overmars and J. van Leeuwen, "Maintenance of configurations in the plane," *J. Comput. and Syst. Sci.*, vol. 23, pp. 166-204, 1981.

[14] F. P. Preparata, "An optimal real time algorithm for planar convex hulls," *Comm. ACM*, vol. 22, pp. 402-405, 1979.