Optimal Point Set Partitioning using Rigid Motion Star Placement *

Prosenjit Bose† Jason Morrison‡

Abstract

We consider the problem of determining the placement of a *star* R on a set $\mathcal{P}$ of n points in the plane such that a given objective function is maximized. A *star* R is a set of m rays $\{r_1, \ldots, r_m\}$ in $\mathbb{R}^2$, emanating from a point p such that the angle between two consecutive rays is $\frac{2\pi}{m}$. A cone defined by two consecutive rays is c-occupied if it contains at least c points of $\mathcal{P}$. Our main result is an $O(n^3m^2)$ expected time (and $O(n^3m^2 \log nm)$ deterministic time) algorithm to find the rigid motion placement of R that maximizes the number of c-occupied cones. We then show how this technique can be extended to solve several other optimization problems.

1 Introduction

Given a set of n points in the plane, how do you partition the plane into m equal sized regions such that the number of points in each region (or some other criterion) is optimized? In this paper, we show how to find a point in the plane such that equal angle subdivisions around it induce a partitioning of the points that maximizes the minimum number of points in any given subdivision. Our algorithm runs in $O(n^3m^2)$ expected time (and $O(n^3m^2 \log mn)$ deterministic time). Furthermore the solution technique applies to a variety of other optimization criteria specified in Section 2.

The inspiration for describing and solving this family of problems comes from other problems that partition pointsets in the plane using equal sized buckets. Two such problems include the uniform projection problem involves partitioning a point set with m equal width parallel strips and its extension to optimally partition using a rectangular grid of two overlapping, orthogonal sets of strips. Results culminated in subquadratic time algorithms by Agarwal *et al.* [1] with algorithms for specific values of m and output sensitivity.

The problems solved here generalize to m partitions work done by Bose *et al.* [4], which partitions with $m = 3$ using a structure referred to as "claws". The running time of their algorithm was later improved from $O(n \log n)$ to linear by Steiger and Streinu [10]. The idea of claws was also used by Bespamyatnikh *et al.*[3] who studied a generalization of ham-sandwich type problems. Our solutions to the general problem lack some of the simplicity that exists in solutions for small values of m, but this may be a reflection of the underlying complexity of the problem for general m.

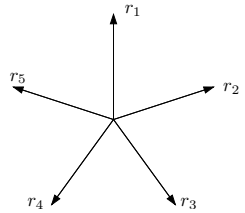

Figure 1: A star with 5 rays

2 Problem Specification

Rigid motion placements of a geometric object are defined by Chazelle [6] to be the placements of a geometric object using any translation or rotation. The problems we analyze here involve the partitioning of a point set $\mathcal{P}$ by a rigid motion placement of a star R, hereafter referred to simply as placements. The star R is a collection of m rays $\{r_1, r_2, \ldots, r_m\}$ sharing a common endpoint, called its *origin*, and ordered in a clockwise direction around the origin (see Fig. 1). For simplicity we assume that the rays are spaced at equal angles, but with minor modifications all results can be applied to the general case. Given a placement of a star R we define a set of m *cones* $\{C_1, C_2, \ldots, C_m\}$ such that C_i is the region bounded by (r_i, r_{i+1}) and includes r_i (note that $r_{m+1} = r_1$). The *occupancy* of R is the number of cones in R that contain at least one point, and the *c-occupancy* of R is the number of cones containing at least c points. Assuming the previous definitions this paper solves the following problems:

Problem 1 *Find a placement of R that maximizes the occupancy of R.*

Problem 2 *Given a fixed integer c, find a placement of R that minimizes or maximizes the c-occupancy of R.*

Problem 3 *Given a fixed integer c find a placement of R (if one exists) such that no cone of R is c-occupied.*

*Research supported in part by NSERC

†School of Computer Science, Carleton University, Ottawa, Canada, jit@scs.carleton.ca

‡Biosystems Engineering, University of Manitoba, Winnipeg, Canada, jason_morrison@umanitoba.ca

Problem 4 *Find the smallest integer c such that there exists a placement of R where no cone is c-occupied.*

Problem 5 *Given a fixed integer k, find the largest integer c such that there exists a placement of R with c-occupancy at least k.*

The solutions presented here extend the translation only placement of Bose and Morrison [5], which uses entirely different techniques. As this paper is an extended abstract any ommitted details are available in [8].

3 Stable Placements

Observe that two or more placements of R on $\mathcal{P}$ may result in the same partitioning of $\mathcal{P}$. For example, many placements of R can be translated by an arbitrarily small distance $\epsilon > 0$, such that no point will cross any ray r_i. We define two placements as being *combinatorially equivalent* if and only if they result in combinatorially equivalent partitionings of $\mathcal{P}$ and *combinatorially distinct* otherwise. Thus when considering possible placements, any objective function that measures only the resulting partitions need only examine one representative placement for each set of combinatorially equivalent placements.

Chazelle [6] defines *stable placements* of generic geometric structures to be placements of the structure such that any three points (p'_0, p'_1, p'_2) of $\mathcal{P}$ coincide with any three boundaries (r'_0, r'_1, r'_2) of the structure. We use stable placements to examine only a finite number of star placements. The proof that there exist a finite number of stable placements and that examining these placements is sufficient for solving our problems lies in Lemmata 1 & 2, proven by Chazelle in [6] [1]. Lemma 1 allows us to bound the number of stable placements and the time to compute them. While Lemma 2 establishes that examining the stable placements is sufficient to examine all combinatorially distinct placements.

Lemma 1 (Chazelle 1) *Given any points (p'_0, p'_1, p'_2) and any rays (r'_0, r'_1, r'_2) there are at most two stable placements with p'_0 on r'_0, p'_1 on r'_1 and p'_2 on r'_2. Moreover, these placements can be computed in $O(1)$ time.*

Lemma 2 (Chazelle 2) *If a given subset $\mathcal{P}_i$ can be contained within a cone of R then there exists a stable placement for which $\mathcal{P}_i$ and only $\mathcal{P}_i$ is contained within the same cone of R.*

4 Brute-Force Solution & Complexity

The first algorithm we develop is a brute force solution to Problems 1 to 5. Essentially, we record necessary occupancy information for the combinatorially distinct placements of R and find an optimal placement for the specific problem being solved. We begin by asking: How many combinatorially distinct placements are there?

Chazelle proved that Lemma 1 holds for a Real RAM model of computation with unit time calculable trigonometric functions. This lemma provides us with a way of calculating an upper bound on the number of combinatorially distinct placements. Since at most each triple of points (p'_0, p'_1, p'_2) can be matched to $O(m^3)$ triples of rays (r'_0, r'_1, r'_2) each possibly producing a constant number of stable placements, there are at most $O(n^3m^3)$ different stable placements. This is an overestimate since by symmetry we can shave a factor of m which yields an $O(n^3m^2)$ bound on the number of different placements. For each placement all partition sizes are computed in $O(n)$ time and the cone with maximal occupancy in a further $O(m)$ time and space. Thus the algorithm runs in $O(n^3m^2(n+m))$ time and $O(n+m)$ space.

5 Star Rotation Diagrams Properties

Note that the stable placements are not examined in any specific order by the brute force algorithm necessitating the constant reassessment of the size of all partitions. We address this observation by adapting Dickerson and Scharstein's rotation diagrams [7]. The key components to the solution of our problems is the development of a rotation diagram for stars with Property 1. Using rotation diagrams, we examine the stable placements in an order that minimizes the differences between the partitionings induced by successive stable placements.

Property 1 *A rotation diagram for a star R, a point set $\mathcal{P}$ and a fixed point p_0 is an arrangement of curves and line segments in which each vertex represents a stable placement of R on $\mathcal{P}$ with p_0 on a ray of R.*

Since the rotation diagram is an embedded graph, we can examine the vertices representing stable placements in a depth-first ordering. We assume that the arrangement is represented as a vertical decomposition and that additional vertices and segments are only examined to connect otherwise separated components. Theorem 3 quantifies the amount of possible change between two directly connected vertices. Proofs of the following theorems can be found in the full version of the paper.

Theorem 3 *Given two directly connected vertices u and v in a rotation diagram, the total number of points that can be in different sub-sets of the induced partitioning is at most $O(\mathsf{deg}(u) + \mathsf{deg}(v))$.*

Our technique constructs a rotation diagram such that each curve and line segment is associated with a point p_i and the two partitions of P that it changes

[1] Chazelle's lemmata are stated here in a manner specific to problems addressed in this paper

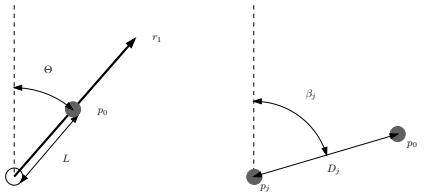

Figure 2: Characterization of star placement

between. Thus, given any curve or line segment of the rotation diagram we can determine in $O(1)$ time the partition changes associated with that segment. The rotation diagram's vertices are examined in depth first search order while the information on how the points are partitioned is updated using lists of points to represent the up to m partitions of P. Since each point only appears once in all of the lists, we can represent all points in $O(n)$ space and each list can be represented with additional $O(1)$ space per possible partition thereby enabling maintenance of the size of each partition. This implies that as each vertex of the rotation diagram is examined the partitioning is changed appropriately. Since the outlined structure requires at most $O(1)$ time per partition change we only need to determine the number of changes to the partitioning to determine the time used to examine each stable placement.

Theorem 4 *For points in general position the time spent updating partition information while examining all of the vertices of the rotation diagram is $O(k)$ where k is the number of vertices in the rotation diagram.*

6 Diagram Construction & Problem Solutions

This section describes a rotation diagram for stars that conforms to Property 1 and its algorithmic use to solve Problems 1 - 5. Similar to Dickerson and Scharstein [7] we construct n rotation diagrams. Each of our diagrams corresponds to a point p_0 being on ray r_1. To construct a single rotation diagram we compute all placements that keep p_j on r_i while p_0 is on r_1, defining $D_j = ||p_0 p_j||$, and β_j to be the cw angle between vertical and the vector from p_j to p_0 (see Fig. 2).

We characterize all rigid motions that force $p_0 \in r_1$ using the two variables (Θ, L) where Θ is the cw angle of r_1 from the vertical and L is the distance from p_0 to the origin (see Fig. 2). A rotation diagram for the star R consists of plotting the values of L and Θ that place each point $p_j \in r_i$ (e.g. Fig. 3) .

Figure 3) is based on two points being unit distance apart with the point p_j directly above p_0 (*i.e.*, $D_j = 1$ and $\beta_j = \pi$). Intuitively, if p_j is at the center of the star and p_0 is on r_1 then $\Theta = \beta_j$, $L = D_j$ and p_j is on all rays of the star. In all rotation diagrams, m arc segments

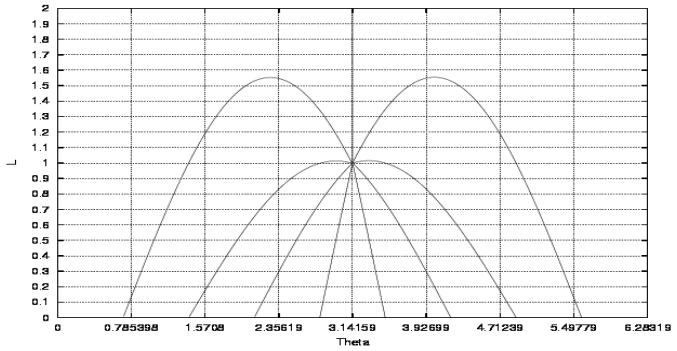

Figure 3: Star rotation diag. ($D_j = 1$, $\beta_j = \pi$, $m = 9$)

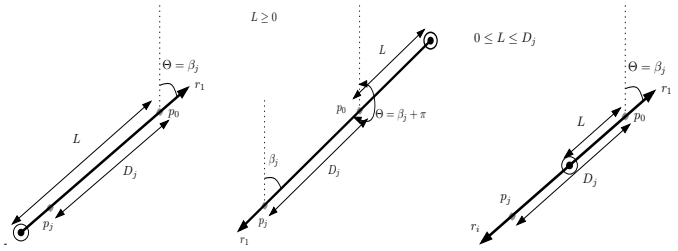

Figure 4: a) & b) $\Psi_i = 0$ c) $\Psi_i = \pi$

emanate from such a point. Each arc segment represents the simultaneous translation and rotation necessary to move p_j from the center of the star as far as possible along a specific ray r_i while keeping p_0 on r_1. The limit of this motion is typically when p_0 reaches the center of the star ($D_j = 0$). The analysis of placing p_0 on r_1 and p_j on r_i breaks into four simple cases all depending on the angle Ψ_i, which is the cw angle between r_1 and r_i.

Case 1: $\Psi_i = 0$ There exist two values of Θ where $p_j \in \{r_i = r_1\}$. Given $L \geq 0$, then either $\Theta = \beta_j$ or $\Theta = \beta_j + \pi$ (see Figs. 4a-b). When $\Theta = \beta_j$, the point $p_j \in r_i$ when $D_j \leq L$. Similarly if $\Theta = \pi + \beta_j$ then $p_j \in r_i$ for all $L \geq 0$.

Case 2: $\Psi_i = \pi$ Note that p_j is on r_i iff $\Theta = \beta_j$ and $0 \leq L \leq D_j$. Proof is by inspection since $L \geq 0$ (see Fig. 4c).

Case 3: $0 < \Psi_i < \pi$ Two restrictions must be true: $\alpha \geq 0$ and $\gamma \geq 0$. These restrictions decide the range of values of θ and positive L for which $p_j \in r_i$ (see Fig. 5a). Computing α and γ and use these values yields the following bounds:

$$\beta_j \leq \Theta \leq \pi - \Psi_i + \beta_j \qquad (1)$$

For values of Θ in the range described by Eq. 1 we analyze L using the Sine Law and Fig. 5a.

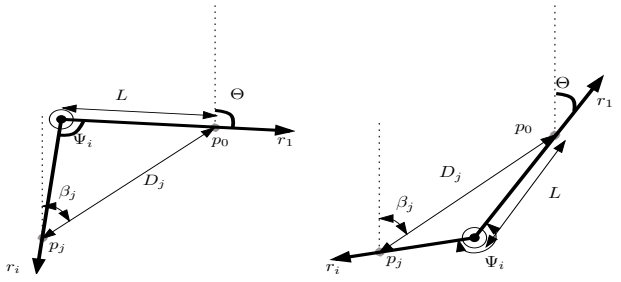

Figure 5: a) $0 < \Psi_i < \pi$ b) $\pi < \Psi_i < 2\pi$

$$L = D_j \frac{\sin\gamma}{\sin\Psi_i} = D_j \frac{\sin(\Theta + \Psi_i - \beta_j)}{\sin\Psi_i} \qquad (2)$$

Case 4: $\pi < \Psi_i < 2\pi$ Using Fig. 5b and proceeding as in Case 3 yields Eq. 2 over the range of Θ not covered in previous cases.

Given the case analysis above, it is possible to construct a diagram where p_0' is on r_1 with Θ as the x-axis and L as the y-axis. For each point p_j and ray r_i a curve is plotted and the total arrangement of these curves can be computed. Since there are $O(nm)$ monotonic curve segments it is possible to compute this arrangement in $O(nm\log(nm)+k)$ expected time using Mulmuley's algorithm [9] or $O((nm+k)\log nm)$ deterministic time using Bentley and Ottmann's algorithm [2] where k is the complexity of the arrangement.

For a given diagram any intersection of two curves represents a stable placement of the star on the point set. This is proven by noting that each curve represents a point being on a ray and the diagram forces some point p_0' to be on r_1 which implies an upper bound of $k = O(n^2m^2)$.

The algorithm proceeds as follows: choosing a point, calculate the rotation diagram; calculate the partitioning of points into cones for one stable point; perform a depth first search traversal of the arrangement of the diagram, updating the partitioning at each vertex; record the best partitioning seen and repeat the process for all possible points. There are at most $O(m)$ sub-sets in any partitioning and each curve segment in the arrangement can contain a pointer to the point changing and which partitions are involved. Thus each partitioning can be created and the optimization information kept in $O(1)$ time per stable placement. Thus the algorithm's runtime is bounded by the construction of the rotation diagram. Given that n diagrams must be calculated, Problems 1 to 5 can be solved in in $O(n^3m^2\log nm)$ deterministic time or $O(n^3m^2)$ expected time. Thus we have achieved a $O(n^3m^2)$ time complexity, which is optimal for algorithms that examine all stable placements.

7 Conclusions and Future Work

We have presented a new technique for partitioning planar point sets. The algorithm runs in $O(n^3m^2)$ expected time or $O(n^3m^2\log nm)$ deterministic time using $O(n^2m^2)$ space (the maximum size of any rotation diagram). This expected time is optimal for algorithms which consider all stable placements of a Star R on a point set $\mathcal{P}$ with sizes m and n respectively. This paper leaves open the question of lower bounds on this problem and if/when the number of placements examined can be lowered while still guaranteeing optimality.

References

[1] Agarwal, P., Bhattacharya, B., and Sen, S. Improved algorithms for uniform partitions of points. *Algorithmica 32*, 4 (2002), 521–539.

[2] Bentley, J., and Ottmann, T. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computing* (1979), 643–647.

[3] Bespamyatnikh, S., Kirkpatrick, D., and Snoeyink, J. Generalizing ham sandwich cuts to equitable subdivisions. *Discrete and Computational Geometry 24*, 4 (2000), 605–622.

[4] Bose, P., Guibas, L., Lubiw, A., Overmars, M., Souvaine, D., and Urrutia, J. The floodlight problem. *The International Journal of Computational Geometry 7*, 1,2 (1997), 153–163.

[5] Bose, P., and Morrison, J. Optimally placing a star on a point set. In *Proceedings of the Canadian Conference on Computational Geometry* (2005).

[6] Chazelle, B. The polygon placement problem. In *Advances in Computing Research*, F. Preparata, Ed., vol. 1. JAI Press, 1983, pp. 1–34.

[7] Dickerson, M., and Scharstein, D. Optimal placement of convex polygons to maximize point containment. *Computational Geometry: Theory and Applications 11*, 1 (Aug. 1998), 1–16.

[8] Morrison, J. *Geometric placement problems.* PhD thesis, School of Computer Science, Carleton University, 2002.

[9] Mulmuley, K. A fast planar partition algorithm. *International Journal on Symbolic Computation 10*, 3–4 (1990), 253–280.

[10] Steiger, W. L., and Streinu, I. Illumination by floodlights. *Computational Geometry: Theory and Applications 10*, 1 (Apr. 1998), 57–70.