In 7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'96), pp. 114–121

Chapter 14
Optimal Placement of Convex Polygons to Maximize Point Containment

Matthew Dickerson[*] Daniel Scharstein[†]

Abstract

Given a convex polygon P with m vertices and a set S of n points in the plane, we consider the problem of finding a placement of P that contains the maximum number of points in S. We allow both translation and rotation. Our algorithm is self-contained and utilizes the geometric properties of the *containing regions* in the parameter space of transformations. The algorithm requires $O(nk^2m^2\log(mk))$ time and $O(n+m)$ space, where k is the maximum number of points contained. This provides a linear improvement over the best previously known algorithm when k is large ($\Theta(n)$) and a cubic improvement when k is small. We also show that the algorithm can be extended to solve bichromatic and general weighted variants of the problem.

1 Introduction

A planar *rigid motion* ρ is an affine transformation of the plane that preserves distance (and therefore angles and area also). We say that a polygon P *contains* a set S of points if every point in S lies on P or in the interior of P. In this paper, we examine the following problem:

PROBLEM 1. (OPTIMAL POLYGON PLACEMENT) *Given a convex polygon P and a planar point set S, find a rigid motion ρ that maximizes the number of points contained by $\rho(P)$. Report ρ and the subset of S contained by $\rho(P)$.*

1.1 Background

Finding a transformation of a region such that it contains a given point set or subset is a problem that has received considerable attention [2, 4, 5, 6, 7, 8, 10, 11]. An interesting variant of the problem is to find an optimal placement of a given polygon that *maximizes* the number of points contained. Several authors have proposed algorithms for different versions of this problem, constraining either the shape of the polygon or the kind of allowed transformation.

Eppstein and Erickson [6], as a substep of their algorithm to find the minimum L_∞ diameter k-subset of a given set S, note that an algorithm of Overmars and Yap [9] can be modified to find the maximum depth of an arrangement of axis-aligned rectangles. This approach solves in $O(n\log n)$ time the problem of finding an optimal translation of a rectangle to cover the maximum sized subset of S. That is, it solves Problem 1 in $O(n\log n)$ time in the special case when the polygon is a rectangle and placement is restricted to translation only. Efrat, Sharir, and Ziv [5] as a substep in their algorithm for finding the smallest k-enclosing homothetic copy of an m-vertex polygon, claim an *oracle* solving the translation version for general m-vertex convex polygons. They suggest a line-sweep technique for their oracle, yielding an algorithm running in $O(nk\log n\log m)$ time. Barequet, Dickerson, and Pau recently showed that the optimizing translation of a convex polygon can be found in $O(nk\log(mk))$ time and $O(m+n)$ space [1] using anchored sweeps around points in S.

For Problem 1, the more general problem allowing rigid motion of a convex polygon, less is known. A brute force approach would be: for each triple of points in S, compute all stable placements of P, and for each placement count the number of points contained (see Section 2.1 for a definition of stable placements). Chazelle [3] has shown that all stable placements of an m-vertex polygon P on three points can be computed in $O(m^2)$ time. This leads to an $O(n^4m^2\log m)$ time algorithm for Problem 1. Eppstein has pointed out in personal communication that the problem can be solved in $O(n^2k^3m^2\log k)$ time using techniques of [6]. He noted that the set covered by the optimal polygon placement is a subset of the $O(k)$ nearest points in vertical distance to a segment pq, where p and q are the maximum and minimum points (with respect to x-coordinate) of the optimal set. This reduces the problem to searching $O(n^2)$ subsets of $O(k)$ points each. Though k may be as large as n, making this an $\Omega(n^5)$ algorithm in the worst case, it does show that the dependence on n can be reduced to quadratic. In the case where k is small, this method provides the best previously known algorithm.

[*]Department of Math and Computer Science, Middlebury College, Middlebury, VT 05753; dickerso@middlebury.edu. Supported by NSF grant CCR-9301714.

[†]Department of Computer Science, Cornell University, Ithaca, NY 14853; schar@cs.cornell.edu. Supported by NSF grant IRI-9057928.

1.2 Outline of our Approach

In this paper we provide a new algorithm for Problem 1 requiring only $O(nk^2m^2\log(km))$ time and $O(m+n)$ space. The basic idea of the algorithm is as follows:

For each point $q_i \in S$, we identify all transformations ρ of the polygon P that keep q_i on its boundary. We capture these transformations geometrically in the *rotation diagram*, a compact geometric characterization of a two-dimensional subset of the parameter space. Each other point q_j yields a *containing region* in the rotation diagram which can be decomposed into $O(m^2)$ subregions of constant complexity, and can be computed efficiently by considering the intersections of two rotating copies of P. We search the arrangement of all containing regions using a line sweep to find the *translation-stable* placement yielding maximum point containment.

We reduce our search space by using a bucketing approach to eliminate points that are too far from q_i to be contained by a placement of P with q_i on the boundary. This yields an output-sensitive time bound for our algorithm, i.e., a time bound dependent on the number k of points contained by the optimal placement.

The rest of the paper is organized as follows: In Section 2 we present some definitions and geometric lemmas upon which our algorithm is based. In Section 3 we present our algorithm along with an analysis and proof of correctness. In Section 4, we generalize our algorithm to solve the bichromatic variant of our problem, as well as a general weighted variant. Section 5 provides our summary remarks and some open questions including comments on possible extensions to some related problems.

2 Geometric and Algorithmic Preliminaries

We now present some geometric results necessary for our algorithm. We begin with some definitions and notation that will be used throughout the paper.

We use q_i to represent the i^{th} point in our input set S. We assume that the polygon P is represented as a list of its vertices $v_1, \ldots, v_m$ given in clockwise order with v_1 located at the origin. We use the standard notation ∂P to represent the boundary of the polygon P; that is, the union of the edges and vertices of P. Likewise, $\partial\rho(P)$ is the boundary of the polygon $\rho(P)$.

2.1 Stable Placements

Stable placements play a fundamental role in our algorithm, allowing us to consider only a finite subset of the infinite number of all possible placements. Barequet, et al. [1] gave the following definition of translation stable placement of a polygon with respect to a point set.

Definition 2.1. *Let $\tau(P)$ be a translation of polygon P containing a set of points S. We say that $\tau(P)$ is in* translation stable placement *if at least 2 points in S are on ∂P.*

Chazelle [3] gave a definition of stable placement of two polygons P and Q with P containing Q under general rigid motion (translation and rotation). He showed that if a polygon P contains a polygon Q, then there exists a stable placement of P and Q with P still containing Q.

In our algorithm, we use a simplified version of this result:

Lemma 2.1. *Let S be a planar point set and P a convex polygon. If there is a rigid motion ρ such that $\rho(P)$ contains $k \geq 2$ points in S, then there exists a rigid motion ρ^* such that $\rho^*(P)$ contains at least k points in S with at least one point in S on the boundary $\partial\rho^*(P)$.*

We also use the following lemma (which was stated and proven in a slightly different form in [1]) relating intersections of polygons and translation stable placements.

Lemma 2.2. (Barequet, Dickerson, Pau) *Let P be a convex polygon, q_1, q_2 points, and τ_1 and τ_2 the translations mapping the origin to points q_1 and q_2 respectively. For any point x on $\partial\tau_1(P)$, define $\tau_x = q_2 - x$ as the translation that maps x to q_2. Then x is a point of intersection between $\partial\tau_1(P)$ and $\partial\tau_2(P)$ if and only if q_1 is on $\partial\tau_x(\tau_1(P))$. (Note that q_2 must be on $\partial\tau_x(\tau_1(P))$ by the definition of τ_x.)*

What the previous lemma tells us is that the points of intersection between two translated copies of a convex polygon P determine all translations of P that are in translation stable placement with respect to these points. Specifically, $\tau_x(\tau_1(P))$ is in translation stable placement with q_1 and q_2 on $\partial\tau_x(\tau_1(P))$ if x is a point of intersection between $\partial\tau_1(P)$ and $\partial\tau_2(P)$. See Figure 1 for illustration. The proof of this lemma follows from elementary geometry and vector arithmetic. In fact, the lemma easily generalizes to the following lemma on *containment* of two points q_1 and q_2:

Lemma 2.3. *Let P be a convex polygon, q_1, q_2 points, and τ_1 and τ_2 the translations mapping the origin to points q_1 and q_2 respectively. For any point x, define $\tau_x = q_2 - x$ as the translation that maps x to q_2. Then $x \in (\tau_1(P) \cap \tau_2(P))$ if and only if $\tau_x(\tau_1(P))$ contains both q_1 and q_2.*

To solve the problem of maximal point containment for translation only, the algorithm of [1] uses the information provided by these lemmas to perform an anchored sweep of the polygon P around each point in S. That is, for a given $q_i \in S$, and for every other point $q_j \in S$, it determines which translations keep q_i

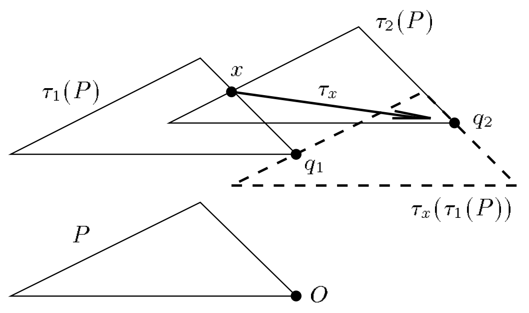

Figure 1: Illustration of Lemma 2.2: $\tau_x = q_2 - x$ yields a translation stable placement $\tau_x(\tau_1(P))$ with respect to q_1 and q_2 *iff* $x \in (\partial\tau_1(P) \cap \partial\tau_2(P))$.

on the boundary and contain q_j. These translations can be considered as an *arrangement* of regions along the boundary of P, where the boundary of P has been parameterized from 0 to the circumference of P. The algorithm then performs a sweep on this 1-D arrangement to find the deepest level. The discrete events in this sweep are the points where both q_i and some point q_j are on the boundary, which are determined by the intersections of two copies of P translated by q_i and q_j.

Unfortunately, this method for computing the anchored sweep cannot be used for the more general problem allowing translation and rotation, as there are an infinite number of possible rotations. To solve the more general problem, we must substantially modify the approach of [1]. Nonetheless, though Problem 1 allows general rigid motion—translation and rotation—we will make use of Lemmas 2.2 and 2.3 as follows: for a *discrete* set of rotations of a polygon around a point $q_i \in S$, we compute the translations of that polygon (with angle of rotation fixed) that contain other points in S.

2.2 General Strategy for Rigid Motion

We now describe our general strategy and give some necessary lemmas, beginning with a description of the basic new conceptual object used in our solution: the rotation diagram.

2.2.1 Rotation Diagrams

For each point $q_i \in S$, we want to find the rigid motion ρ such that q_i is on $\partial\rho(P)$, and $\rho(P)$ contains a maximal number of points. This involves rotating the polygon P around q_i, keeping vertex v_1 on q_i, and for each angle of rotation computing the arrangement of translations that keep q_i on the boundary. We map out this two-dimensional parameter space of all rotations and all translations that keep q_i on the boundary of P using a *rotation diagram*. The angles of rotation, ranging from 0 to 2π, form the horizontal axis of the diagram. We can also parameterize all translations by identifying them with the points on the boundary of P. This yields the vertical axis, ranging from 0 to C_P, the circumference of P. We call the resulting two-dimensional space the *rotation diagram* R_{P,q_i} Note that both axes of R_{P,q_i} "wrap around", since 0 and 2π, and also 0 and C_P are identified. (In terms of topology, the rotation diagram forms the surface of a torus.) Each point (θ, t) in R_{P,q_i} corresponds to the placement of P rotated by θ and translated such that the point at position t along P's circumference is identified with q_i.

Given other points in S, we can then map out their *containing regions* in the rotation diagram, allowing us to geometrically capture the optimal placement of P. We illustrate this idea in the following section for only one extra point, and give a number of the lemmas necessary for both our proof of correctness and run-time analysis.

2.2.2 Pairs of Rotation Polygons

Consider placing two copies P_1 and P_2 of a polygon P on points $q_1, q_2 \in S$. Specifically, we place vertex v_1 of P on each of the two points. We now rotate the two polygons in tandem and examine what happens. By Lemma 2.2, at any given angle of rotation the intersections between the two polygons determine the translations of P which would place both q_1 and q_2 on the boundary of P. As we rotate the pair of polygons, these translations trace out curves in the rotation diagram R_{P,q_1} for P and q_1. By Lemma 2.3, these curves are the boundary of a (not necessarily connected) *containing region* in R_{P,q_1} corresponding to all placements of P that contain q_2 (and have q_1 on the boundary). Let us call this region A_{q_2}. We will now study the properties of A_{q_2}.

At each angle θ we can imagine a vertical line in R_{P,q_1}, representing the "unfolded" boundary of the polygon P rotated by θ. Note that such a line can intersect at most two of the bounding curves, since two copies of the same convex polygon at the same orientation can intersect in at most two distinct points (or, in the degenerate case, along a consecutive part of the boundary). In between these two intersection points, the line intersects the containing region A_{q_2}, corresponding to the part of the boundary of P_1 that is covered by P_2. We now decompose A_{q_2} into smaller regions bounded on the left and right by certain *critical* angles: namely those angles at which a vertex of P_1 sweeps through ∂P_2 or a vertex of P_2 sweeps through ∂P_1 (see Figure 3). Each decomposed region has four sides, and it is clear by definition that the left and right sides are vertical lines at the critical angles. In the case that at a critical angle the polygons start (or stop)

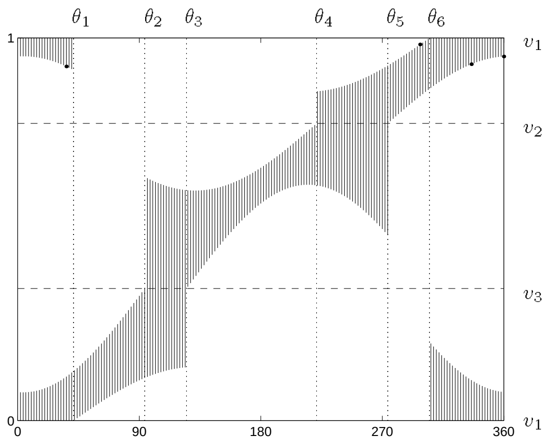

Figure 2: An example of a containing region A_{q_2} in the rotation diagram R_{P,q_1} for a triangle P. The critical angles are marked θ_1 through θ_6.

being in contact with each other, a vertical bounding line might degenerate to a point. Figure 2 shows an example of a containing region in a rotation diagram for a triangle. The containing region A_{q_2} is traced out by vertical lines; the critical angles are shown by dotted lines. Remember that the diagram wraps around.

The obvious next question to ask is: What kind of curves form the top and bottom boundaries of the containing region? Lemma 2.4 answers this question, and Lemmas 2.5 and 2.6 deal with the size and number of the decomposed regions.

LEMMA 2.4. *The upper and lower curves bounding the decomposed regions of A_{q_2} are sine curves of the form $c_1 + c_2 \sin(\theta + c_3)$.*

Proof. The key observation for this proof is that rotating two polygons in tandem is equivalent to keeping one of the polygons fixed, and *translating* the other polygon on a circle around it. No rotation needs to take place, since the relative orientation of the two polygons to each other remains constant. Specifically, if we keep P_1 in place, every vertex of P_2 describes a circle around the corresponding vertex of P_1 (see Figure 3). All circles have the same radius r, which is equal to the distance between the points q_1 and q_2. Between two critical angles, a bounding curve of A_{q_2} is created by some edge e_2 of P_2 intersecting an edge e_1 of P_1. The y-coordinate in the rotation diagram is the position of the intersection point on the circumference of P_1, i.e., the position of the intersection point on e_1 plus the length l of all previous edges of P_1. To show that the position of the point of intersection on an edge with another edge sweeping out a circle is a multiple of a sine function, let us reorient the coordinate system by a translation that puts the center

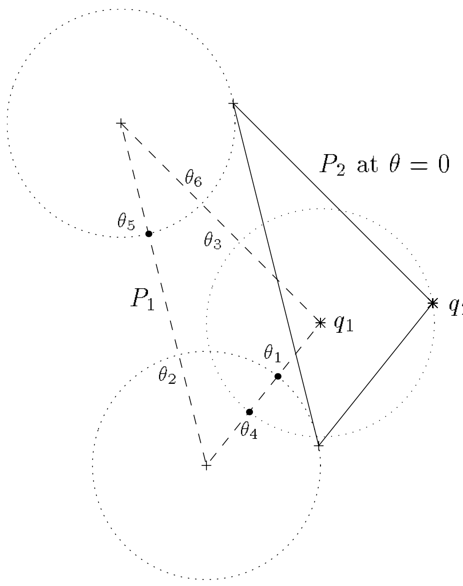

Figure 3: The underlying geometry for the rotation diagram in Figure 2. The critical angles correspond to intersections of circles with edges.

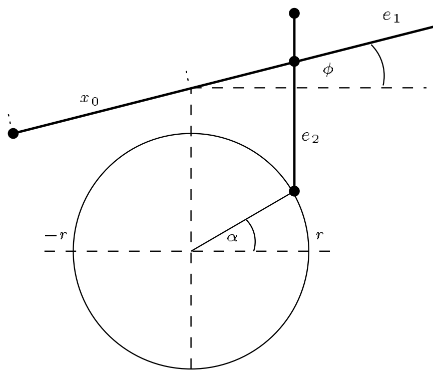

Figure 4: Illustration of Lemma 2.4: Edge e_2 translating on a circle intersects edge e_1 at position $x_0 + \frac{r}{\cos\phi}\sin\alpha$.

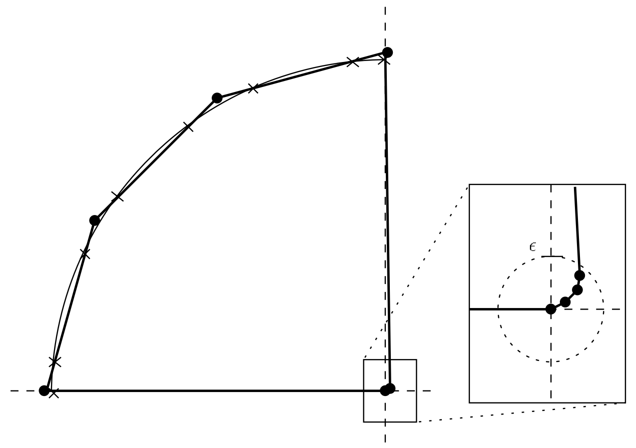

Figure 5: A polygon yielding m^2 critical angles ($m = 8$ in the figure). Half of the vertices form a quarter of a regular polygon that intersects the unit circle m times, the other half are closely spaced next to the origin.

of rotation at the origin, and by an angle of rotation γ such that e_2 is now vertical. Now, e_2 intersects the x-axis at $r \sin \alpha$, where $\alpha = \theta - \gamma$. Let ϕ be the orientation of e_1 in our new coordinate system, and let x_0 be the length of the part of e_1 that lies to the left of the y-axis (see Figure 4). Then, the intersection point on the circumference of P_1 is at $l + x_0 + \frac{r}{\cos \phi} \sin(\theta - \gamma)$, or $c_1 + c_2 \sin(\theta + c_3)$. □

LEMMA 2.5. *Each decomposed region is at most π wide.*

Proof. A decomposed region is bounded by two critical angles, which are created by vertices sweeping through edges of the other polygon. The maximum width of the regions is bounded by the maximal part of a circle—centered on a vertex—that lies inside the polygon, that is, by the biggest angle of the polygon. Since the polygon is convex, the biggest possible angle is π. □

LEMMA 2.6. *A_{q_2} is decomposed into at most $O(m^2)$ regions.*

Proof. Polygon P has m vertices and edges. Each vertex can sweep through an edge of the other polygon at most twice. □

LEMMA 2.7. *The bound in Lemma 2.6 is tight, i.e., there are cases yielding $\Omega(m^2)$ decomposed regions.*

Proof. We construct an example resulting in m^2 critical angles. See Figure 5 for illustration. We construct a polygon by placing $m/2$ vertices such that they form a quarter of a regular $4(\frac{m}{2} - 1)$-gon, intersecting the unit circle m times (the intersections are marked with crosses). Let ϵ be the maximum variation of the radius of the circle still yielding m intersections. Place the other $m/2$ vertices such that they form a rounded corner less than ϵ away from the origin. Now, for two points q_1, q_2 of distance 1, the rotation diagram has m^2 critical angles as the $m/2$ closely spaced vertices of each copy of the polygon intersect m times edges of the other copy of the polygon as θ ranges from 0 to 2π. □

2.2.3 Generalization to Multiple Points

For a given point q_i, we can compute the containing regions A_{q_j} in R_{P,q_i} for all other points $q_j, j \neq i$. For points q_j and q_k, the regions A_{q_j} and A_{q_k} may intersect, indicating a placement of P containing q_i, q_j and q_k. The deepest point in the arrangement of all containing regions gives us the rotation and translations keeping q_i on the boundary and containing the maximum number of other points in S. We compute these arrangements of containing regions for all rotation diagrams R_{P,q_i} and compute the deepest arrangement in all n diagrams. See Figure 6 for an example of an arrangement of 4 containing regions.

Besides the problem considered in this paper, rotation diagrams also have applications in robot motion planning: Note that the rotation diagram $R_{P,q}$ represents the configuration space of a convex polygonal robot P that must stay in contact with a point q (say, a power source). A path in $R_{P,q}$ that does not cross any boundaries of containing regions corresponds to a collision-free motion of P avoiding (or containing) a given set of points while keeping q on the boundary of P. The tools to compute and search arrangements of containing regions in rotation diagrams developed in this paper can also be used to compute such collision-free paths efficiently.

3 The Algorithm

Figure 7 presents the algorithm for the solution to Problem 1. The correctness of this algorithm follows from the lemmas and discussion of Section 2. In particular, Lemma 2.1 tells us that we only have to consider translation stable placements. From Lemma 2.3 we see that the regions in the rotation diagrams correctly correspond to placements (rotations and translation) containing the specified points. The trick is to compute these containing regions, and to find the maximum depth. The methods for these tasks are described in the following sections.

3.1 Computing the Regions

Given polygons P_i and P_j—that is, given the polygon P translated to points q_i and q_j—we describe how

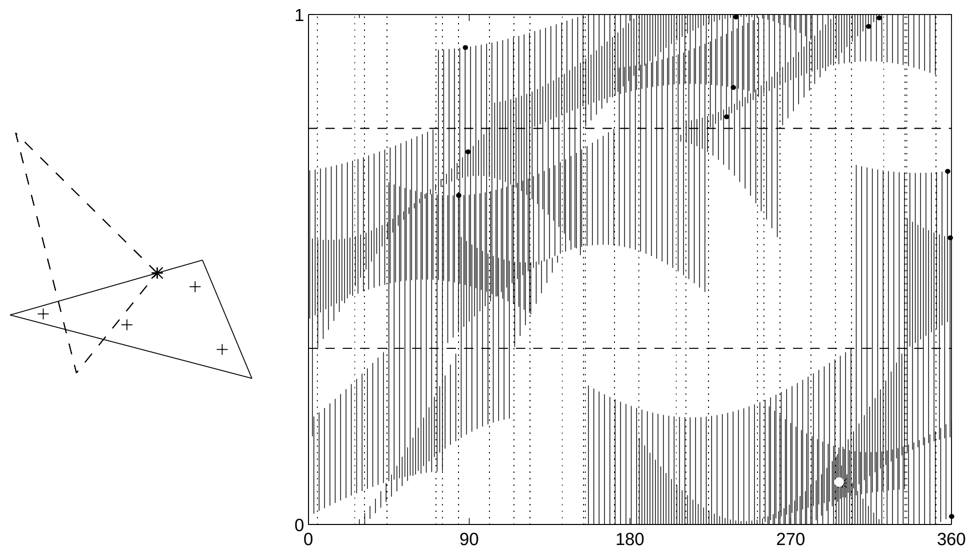

Figure 6: An arrangement of four containing regions. The deepest point in the arrangement is marked with a white dot. The corresponding transformation of the polygon (yielding containment of all four points) is drawn on the left. The tranformation corresponding to the origin of the rotation diagram is shown dashed.

```
I. Preprocessing:
Preprocess points into buckets. For all points
$q_i \in S$, we define $B_i$ as the set of buckets that could
intersect a placement of polygon $P$ containing point
$q_i$.

II. Iteration:
1. Set max := 0
2. FOR each point $q_i \in S$ DO BEGIN
3.        Initialize rotation diagram $R_{P,q_i}$
4.        FOR each $j \neq i$ and $q_j \in B_i$ DO
              Compute containing region $A_{q_j}$ in $R_{P,q_i}$
5.        Use line sweep to compute point $x$ of
          maximum depth $d$ in the arrangement
          of all containing regions $A_{q_j}$
6.        IF $d >$ max THEN
              Set max := $d$; Store $x$
7. END FOR
```

Figure 7: Algorithm 1

to compute the containing region A_{q_j} in the rotation diagram R_{P,q_i}.

We saw in our proof of Lemma 2.4 that rotating the two polygons in tandem is equivalent to sweeping polygon P_j in a circle with the orientation remaining fixed. Specifically, each vertex in P_j moves around the corresponding vertex in P_i on a circle whose radius is the distance from q_i to q_j. We see that a given vertex of P_j can pass through a given edge of P_i at most two times. Likewise, a given vertex of P_i can be swept through by a given edge of P_j at most two times. For each vertex-edge or edge-vertex pair, we can explicitly compute the angles of rotation causing an intersection in constant time by intersecting a circle with a segment. We have a total of m^2 work to compute these angles. We sort these in $O(m^2 \log m)$ time, and use them to compute at most m^2 decomposed regions of A_{q_j}.

3.2 Complexity of the Arrangements via Line Sweep

Given the rotation diagram R_{P,q_i} for a point q_i, we now show how the depth of the arrangement of all containing regions A_{q_j}, $j \neq i$, in R_{P,q_i} can be found. We use a line sweep with a vertical line through all angles of rotation θ. The events in the line sweep consist of the discrete set of angles marking the left and right boundaries of the decomposed regions, and the intersections between different sine curves forming the top and bottom boundaries of the decomposed regions. A segment tree, modified for sine curves, can be used as the data structure.

Note first that in rotation diagram R_{P,q_i}, there are $n-1$ containing regions A_{q_j}, each decomposed into at most $O(m^2)$ subregions. At any given angle, there are at most $2n-2$ sine curves to be stored in the segment tree. This data structure therefore requires $O(n)$ space and $O(\log n)$ time per operation. How many queue events are there? We can consider the decomposed regions of each A_{q_j} as disjoint. Given two containing regions A_{q_j} and A_{q_k}, note that there are at most $O(m^2)$ pairs of intersecting decomposed regions, since both sets of decomposed regions are ordered linearly as we sweep through angles from 0 to 2π. But each such intersecting pair has a bounded number of intersections since the left and right sides of the region are vertical lines, and the top and bottom of the regions are sections of sine curves of less than π according to Lemmas 2.4 and 2.5. Therefore there are at most $O(m^2)$ intersections between the two sets of decomposed regions for each pair A_{q_j} and A_{q_k}.

Since there are at most $\binom{n}{2}$ pairs of containing regions (A_{q_j}, A_{q_k}), there are a total of $O(n^2m^2)$ events in the event queue. The following section shows that we can tighten this bound to $O(k^2m^2)$ with bucketing, using $O(k)$ space and $O(\log k)$ time for the line sweep structure.

3.3 Bucketing

Let D be the diameter of the polygon P. If C is a chord of P of length D, let W be the width of P perpendicular to C. Our bucketing strategy is to bucket space into squares of size $D \times D$, and to place each point in its correct bucket. The following lemma is straightforward and not difficult to prove.

LEMMA 3.1. *Any bucket can be tiled by a fixed number c of copies of P, with c dependent only on the ratio D/W. Furthermore, if any bucket contains K points, then there is a placement of P containing $\Omega(K/c)$ points.*

A polygon containing a point q_i in bucket b can contain only points in b and the 8 buckets neighboring b. We call this group of 9 buckets B_i. In our construction of the rotation diagram R_{P,q_i} for polygon P and point q_i we consider only points in B_i.

3.4 Overall Analysis

The analysis of the algorithm follows from the previous sections. For each rotation diagram R_{P,q_i}, it follows from Lemma 3.1 that the number of events in the line sweep to compute the depth is $O(k^2m^2)$ and thus the sweep can be performed in time $O(k^2m^2\log(km))$ time. As there are n diagrams to test, the total running time is $O(nk^2m^2\log(km))$.

3.5 A Bucketless Approach

For "skinny" polygons with a large ratio $c = D/W$, the bucketing approach may be inefficient: by Lemma 3.1, the number k of points covered is proportional to K/c where K is the number of points in the densest bucket. Thus, the exact running time is given by $O(nK^2m^2\log(km)) = O(c^2nk^2m^2\log(km))$. The following theorem uses the fact that each intersection event in our line sweep corresponds to a stable placement with three points on the boundary of the polygon, and states that even without bucketing there is still an output sensitive time bound better than $O(n^3m^2\log(nm))$.

THEOREM 3.1. *Given a convex polygon P and points q_i and q_j, let k be the number of points $q' \in S$ such that q' is at least as close to q_i as q_j, and there is a stable placement of P with q_i, q_j, and q' on the boundary. Then there is a placement of P covering $\Omega(k)$ points in S.*

This theorem tells us that for any rotation diagram R_{P,q_i}, the number of intersections between pairs of containing regions is bounded by $O(nk)$ where k is the depth of the maximum arrangement. This leads to the following theorem.

THEOREM 3.2. *When no bucketing is used, Algorithm 1 runs in $O(n^2km^2\log(nm))$ time.*

This analysis is independent of the ratio D/W and shows that the algorithm even without bucketing is still asymptotically faster than those based on the results of [3] and [6] by factors of n^2/k and k^2 respectively.

4 Bichromatic and Weighted Variants

The algorithm presented here can easily be extended to a more general version of the problem. Consider a set S where each point q_i is given a weight $W(q_i)$. Instead of maximizing the number of points covered, we want to maximize the total weights of all points contained. If all weights are greater than or equal to zero, then Lemma 2.1 still applies where k becomes the total weight of the contained points rather than the number of contained points. The algorithm runs with no modifications.

However, if $W(q_i) < 0$ for some points in the set, then it is possible that there is no *stable* placement that maximizes the total weight of the points covered. (It is not difficult to give an example where the only stable placement covering the same points as that covered by the maximal placement also covers an additional negatively weighted point.) In particular, consider the bichromatic version where the set S is divided into two subsets S_r and S_b and the goal is to find a placement that maximizes the number of points contained from S_r and minimize the number contained from S_b. This is the weighted variant where all points in S_r have weight 1 and all points in S_b have weight -1. Another bichromatic problem, to maximize points covered in S_r while covering *no* point in S_b, can be solved by assigning weight 1 to the points in S_r and the weight $-|S_r|$ to the points in S_b. The solution to the problem is not difficult. For each placement ρ with a negatively weighted point q_i on the *boundary*, we look for a "nearby" placement ρ_ϵ that contains the same points but does not contain q_i. The same idea can be applied in the degenerate case where multiple points lie on the boundary, though if there is more than one negatively weighted point on the boundary then the ϵ translation will not necessarily exist. Thus with minor modifications, the algorithm can be used to solve both the bichromatic and the general weighted variant of the problem with the same running time.

5 Summary

We have provided the asymptotically fastest known solution to the problem of computing a placement (translation and rotation) of a given convex polygon P containing the maximum number of points of a given point set S. We have shown that the algorithm requires output-sensitive $O(n^2km^2\log(nm))$ time and $O(n+m)$ space, where n is the number of points in S, m is the number of vertices of P, and k is the maximum number of points contained ($k \leq n$). Using bucketing, we achieve an even better time bound of $O(c^2nk^2m^2\log(km))$, where c is the ratio of longest to shortest diameter of P.

The algorithm is conceptually simple and self-contained. It uses a line sweep of an arrangement of *containing regions* in a *rotation diagram*. The rotation diagram can also be used to solve motion planning problems in which a convex polygonal robot must stay in contact with a certain point while avoiding or containing other points. The algorithm generalizes at no cost in running time not only to solve the bichromatic variant of the problem, but the more general weighted point set problem. It is asymptotically faster than the best previously known approaches by at least a linear factor, and as much as n^3 depending on k, the number of points covered.

5.1 Extensions and Open Problems

There are obvious generalizations of Problem 1. We may consider containment of arbitrary simple polygons or containment by polyhedra in higher dimensions.

PROBLEM 2. *Given a* simple polygon *P and a planar point set S, find a rigid motion (or even just a translation) ρ that maximizes the number of points contained by $\rho(P)$.*

PROBLEM 3. *Given a convex polyhedron P and a point set S in $\mathbb{R}^3$, find a rigid motion ρ that maximizes the number of points contained by $\rho(P)$ over all possible rigid motions.*

Acknowledgements

This paper benefited from discussions with Scot Drysdale, Amy Briggs and Mark Montague.

References

[1] G. Barequet, M. Dickerson, and P. Pau. Translating a convex polygon to contain a maximum number of points. In *Proc. 7th Canadian Conference on Computational Geometry*, pages 61–66, 1995.

[2] S. Chandran and D. Mount. A parallel algorithm for enclosed and enclosing triangles. *Int. J. Computational Geometry and Applications*, 2(2):191–214, 1992.

[3] B. Chazelle. The polygon placement problem. In F. Preparata, editor, *Advances in Computing Research: Volume 1*, pages 1–34. JAI Press, 1983.

[4] A. Datta, H. Lenhof, C. Schwarz, and M. Smid. Static and dynamic algorithms for k-point clustering problems. In *Proc. 3rd Workshop on Algorithms and Data Structures*, pages 265–276. Lecture Notes in Computer Science 709, Springer Verlag, New York, 1993.

[5] A. Efrat, M. Sharir, and A. Ziv. Computing the smallest k-enclosing circle and related problems. *Computational Geometry: Theory and Applications*, 4:119–136, 1994.

[6] D. Eppstein and J. Erickson. Iterated nearest neighbors and finding minimal polytopes. *Discrete and Computational Geometry*, 11:321–350, 1994.

[7] V. Klee and M. Laskowski. Finding the smallest triangles containing a given convex polygon. *J. Algorithms*, 6:359–375, 1985.

[8] J. O'Rourke, A. Aggarwal, S. Maddila, and M. Baldwin. An optimal algorithm for finding minimal enclosing triangles. *J. Algorithms*, 7:258–269, 1986.

[9] M. Overmars and C. Yap. New upper bounds in Klee's measure problem. *SIAM J. Computing*, 20:1034–1045, 1991.

[10] F. Preparata and M. Shamos. *Computational Geometry.* Springer Verlag, New York, 1985.

[11] G. Toussaint. Solving geometric problems with the rotating calipers. In *Proc. IEEE MELECON '83*, Athens, Greece, 1983.