1-1-1986

Finding Largest Empty Circles with Location Constraints

L Paul Chew
Dartmouth College

Robert L. Scot Drysdale
Dartmouth College

FINDING LARGEST EMPTY CIRCLES WITH LOCATION CONSTRAINTS

L. Paul Chew and Robert L. (Scot) Drysdale III

Technical Report PCS-TR86-130

Finding Largest Empty Circles with Location Constraints

L. Paul Chew(*) and Robert L. (Scot) Drysdale, III
Dept. of Mathematics & Computer Science
Dartmouth College
Hanover, NH 03755

ABSTRACT

Let S be a set of n points in the plane and let CH(S) represent the convex hull of S. The Largest Empty Circle (LEC) problem is the problem of finding the largest circle centered within CH(S) such that no point of S lies within the circle. Shamos and Hoey [4] outlined an algorithm for solving this problem in time O(n log n) by first computing the Voronoi diagram, V(S), in time O(n log n), then using V(S) and CH(S) to compute the largest empty circle in time O(n). Toussaint [7] pointed out some problems with the algorithm as outlined by Shamos and presented an algorithm which, given V(S) and CH(S), solves the LEC problem in time O(n log n). Preparata and Shamos [2] show that the original claim was correct by outlining an algorithm that computes the LEC in O(n) time given V(S) and CH(S). We generalize their method to show that given V(S) and any convex k-gon P, the LEC centered within P can be found in time O(k+n). We also improve on an algorithm given by Toussaint for computing the LEC when the center is constrained to lie within an arbitrary simple polygon. Given a set S of n points and an arbitrary simple k-gon P, the largest empty circle centered within P can be found in time O(kn + n log n). This becomes O(kn) if the Voronoi diagram of S is already given.

(*) Supported in part by NSF grant MCS-8204821.

INTRODUCTION

Let S be a set of n points in the plane, and let CH(S) represent the convex hull of S. The Largest Empty Circle problem (LEC) is to find the largest circle such that no point of S is within the circle and such that the center of the circle is within CH(S).

Shamos and Hoey [4] gave an $O(n \log n)$ algorithm for computing V(S), the Voronoi diagram of S, and pointed out that the LEC problem (and many other problems in computational geometry) can be solved quickly if one starts with the Voronoi diagram. Given n data points in the plane, the Voronoi diagram partitions the plane into n regions, one associated with each point (see Figure 1). The region associated with data point p consists of all points in the plane that lie closer to p than to any of the other $n-1$ data points. The boundaries of these Voronoi regions form a planar graph with $O(n)$ vertices. Each of these Voronoi vertices is the center of an empty circle that intersects three or more points of S.

Note that the LEC must either (1) intersect two of the original points and be centered at a point on the convex hull or (2) intersect three of the original points. Any other circle is incompletely constrained and could thus be enlarged. This is equivalent to saying that the LEC must be located either at a point where V(S) intersects CH(S) or at one of the Voronoi vertices. Thus, the LEC problem can be solved using a three step algorithm: first, compute V(S) in time $O(n \log n)$ [4]; second, check the Voronoi vertices in time $O(n)$ using a simple depth-first-search of the Voronoi diagram graph; third, check the points where V(S) intersects CH(S). Shamos [4, 5, 6] has claimed that the third step can be done in time $O(n)$, but was vague about the details. Toussaint [7] has pointed out problems with the methods outlined by Shamos, and has given an $O(n \log n)$ algorithm for the third step. Preparata and Shamos [2] show that the original claim was correct by outlining an $O(n)$ algorithm for this step. We present a more general algorithm that can be used to find the LEC centered within an arbitrary convex k-gon in time $O(k+n)$ assuming that $V(S)$ is already given.

In the same paper [7], Toussaint gives an $O(n^2 \log n)$ algorithm for solving the LEC problem when the center of the circle is constrained to lie within an arbitrary simple n-gon. We give an $O(n^2)$ algorithm for this problem. More generally, we show that given a set S of n points and a polygon P with k vertices, the largest empty circle centered within P can be found in time $O(kn + n \log n)$ where the term $n \log n$ is needed just for the construction of the Voronoi diagram of S.

We assume that polygons are simple (edges do not cross) and are presented as an ordered list of vertices (clockwise order). All points are presented in the usual Cartesian coordinates. As in [1], to simplify the presentation of the algorithm we form a boundary by preprocessing the points involved so that we need consider only finite edges and regions. This is used strictly to simplify the presentation of the algorithm, but is not otherwise needed. For us then a Voronoi diagram is made up of Voronoi vertices, Voronoi edges, and boundary edges. The Voronoi diagram is given as an <u>edge-ordered representation</u> [1]:

1. if e is an edge joining vertex v to vertex w, then e is represented by the pair of directed edges $\{(v,w),(w,v)\}$;

2. each vertex v has associated with it not only its coordinates, but also a list in counterclockwise order of all directed edges whose source is v;
3. each directed edge (v,w) has associated with it the edge (w,v) as well as the name of the region immediately to the right of (v,w).

The representation of a Voronoi diagram as produced by Shamos' $O(n \log n)$ algorithm can be processed in $O(n)$ time to produce such an edge-ordered representation, or the desired representation can easily be built as the Voronoi diagram algorithm is running.

We make use of the following theorem:

Theorem (Toussaint [7]). Given a set S of n points and a k-gon P, the largest circle C such that the center of C is within P and such that no point of S lies within C must be a circle centered at either: (1) a Voronoi vertex of V(S), (2) a vertex of the k-gon P, or (3) an intersection point of P and V(S). (See Figure 2.)

CONVEX POLYGONS

The following algorithm checks each of the 3 classes of points mentioned in the previous theorem. The vertices of the convex polygon P and the intersection points of P with the Voronoi diagram are checked in the loop in step 3 of the algorithm. The basic idea, like that outlined in Preparata and Shamos [2], is to follow the Voronoi edges around P, staying outside of P and as close to P as possible. Our step is more complex than theirs because we must deal with general convex polygons rather than the special case of the convex hull. The Voronoi vertices are checked in step 4. Note that to check the Voronoi vertices in $O(n)$ time we must be able to determine whether a Voronoi vertex is within P in $O(1)$ time. This is done using an intersection count (a count of intersections with P) for each edge of the Voronoi diagram. This intersection count for each Voronoi edge is used to keep track of whether the the Voronoi vertices are inside or outside of P. For instance, if one endpoint of edge e is outside P and the intersection count for edge e is even (odd) then the other endpoint is outside (inside) P.

Some care must be taken when a vertex of P lies on an edge of the Voronoi diagram or when a Voronoi vertex lies on an edge of P (see, for example, the discussion in [3] on determining whether a point is inside a polygon). For our presentation, we will treat a vertex of P that lies on an edge of the Voronoi diagram as if it is actually located a very small distance to the right or, if the edge extends to the right, a small distance to the right and a bit down. Similarly, a Voronoi vertex that lies on an edge of P will be treated as if it is actually located to the left or to the left and a bit up.

We use s to represent both a point of S and the Voronoi region of the point s. If p is a vertex of the polygon P then we use E(p) to represent the edge of P clockwise from p and with p as an endpoint. We use R(p) to represent the ray with endpoint p in the direction of E(p). E(v,s) represents an edge of region s, starting at vertex v of region s and going clockwise.

Input: a set S of n points, a convex k-gon P.

Output: the largest empty circle centered within P.

Method: (Note that vertices of P that lie on a Voronoi edge and Voronoi vertices that lie on an edge of P can be handled as outlined above.)

0. Compute V(S) the Voronoi diagram for S; work within a rectangle that contains both S and P (this makes all regions and edges finite).
1. Keep an intersection count for each edge of V(S); initially, the count will be zero for each edge.
2. Choose any vertex p of P; find which Voronoi region contains p and let s be the data point for that region; find the edge E(v,s) of region s that R(p) intersects.
3. while more of polygon P do

3a. <u>if</u> R(p) misses E(v,s) <u>then</u>

Mark E(v,s) (used just to help with the time analysis);

Set v = next vertex clockwise around s;

3b. <u>else if</u> E(p) misses E(v,s) <u>then</u>

Set p = next vertex clockwise around P;

Check the circle centered at p through the point s;

3c. <u>else</u>

Increment the intersection count for E(v,s);

Check the circle through s centered at the intersection of E(p) and E(v,s);

Set s = region adjacent to s through E(v,s).

4. Do a depth first search of the Voronoi diagram graph, using the intersection count for each edge to determine which of the Voronoi vertices lie within P (each time you pass an intersection you switch from inside to outside or vice versa); find the LEC among those circles centered at Voronoi vertices within P.
5. Report the largest of the empty circles found in steps 3b, 3c, and 4.

Analysis:

0. O(n log n) [4].
1. O(n) time to initialize since there are O(n) edges.
2. This step takes O(n) time. We find the region containing p by simply checking each data point of S to find the closest. We find the edge e_s that R(p) intersects by simply checking each edge of region s.
3. This loop is done O(k+n) times. To see this, note that each time through this loop one of 3a, 3b, and 3c is executed. We show these instructions are executed at most O(n), O(k), and O(n) times, respectively.

3a. It is clear that the following loop invariant holds: s is a region that P intersects; v is outside of P; either E(v,s) is outside of P or it intersects P. So an edge is marked iff it is outside the polygon P and part of a Voronoi region that P intersects. These marked edges form a simple polygon Q (see Figure 3). (Technically, Q is not quite a simple polygon because some edges are used more than once, but each such edge can be considered as two edges.) As the algorithm runs we march clockwise around Q, marking each edge as we go. Because the edges of Q are also Voronoi edges and there are only O(n) Voronoi edges, step 3a can be executed at most O(n) times.

3b. There are only k vertices in polygon P and the loop ends when we finish the polygon; thus this instruction can be executed at most k times.

3c. Since the polygon P is convex it can intersect a single edge at most twice. There are O(n) edges, so this instruction can be done at most O(n) times.

4. O(n) time to do depth first search on a graph of size O(n).

5. O(1).

This gives us the following theorem:

Theorem. Given a set S of n points, the largest empty circle centered within an arbitrary convex k-gon P can be found in time O(k + n log n). If the Voronoi diagram for S is given then the LEC centered within P can be found in time O(k+n).

As an immediate corollary:

Corollary. Given a set S of n points and V(S) the Voronoi diagram of S, the largest empty circle centered within the convex hull of S can be found in time O(n).

For the LEC problem as stated in the Corollary (that is, find the LEC centered within the convex hull of S) the algorithm above can be simplified. For instance, the vertices of the convex hull are points of S; thus, they do not need to be checked as centers of possible LECs.

With the exception of step 2 in which the region that contains the initial point of P is determined, the method used to find the intersection points of the convex polygon and the Voronoi diagram did not need any of the properties of a Voronoi diagram other than that such a diagram is a convex planar subdivision (a subdivision of the plane in which all regions are convex). Note that the region that contains the initial point of P can be determined in O(n) time by simply comparing the point with the boundaries of each region. This observation immediately leads to the following corollary that holds for convex planar subdivisions or (as in [1]) for any planar subdivisions that can be easily triangulated (for instance, subdivisions with star-shaped regions).

Corollary. The intersection of a convex k-gon and a convex planar subdivision of size n can be found in time O(k+n).

SIMPLE POLYGONS

The algorithm for convex polygons can be used exactly as written to find the LEC centered within an arbitrary simple polygon in time $O(kn+n \log n)$.

Input: a set S of n points, a simple k-gon P.

Output: the largest empty circle centered within P.

Method: Exactly as for the convex case.

Analysis:

0-2. As above.

3. This step takes time $O(kn)$. Because P is no longer convex, we may end up going all the way around a given region in step 3a for each time that we get a new polygon edge in 3b. The best we can say is that for a given vertex p of P steps 3a and 3c will each be executed at most $O(n)$ times. Since there are k vertices of P this gives a time bound of $O(kn)$.

4-5. As above.

The algorithm outlined above clearly checks all the appropriate points. Step 3 has the worst time bound. Note, however, that there can be kn intersections between a polygon of size k and a Voronoi diagram of size n (see Figure 4); so as long as we check all the intersection points this bound cannot be improved. This gives us the following result for simple k-gons.

<u>Theorem</u>. Given a set S of n points and a simple k-gon P, the largest empty circle centered within P can be found in time $O(kn + n \log n)$. If the Voronoi diagram for S is already given then the LEC can be found in time $O(kn)$.

REFERENCES

[1] D. Kirkpatrick, Optimal Search in Planar Subdivisions, SIAM Journal of Computing 12:1(1983), 28-35.

[2] F. P. Preparata and M. I. Shamos, Computational Geometry, Springer-Verlag (1985).

[3] R. Sedgewick, Algorithms, Addison-Wesley (1983).

[4] M. I. Shamos and D. Hoey, Closest-Point Problems, Proc. 16th IEEE Symposium on Foundations of Computer Science (1975), 151-162.

[5] M. I. Shamos, Problems in Computational Geometry, Carnegie-Mellon University (1977).

[6] M. I. Shamos, Computational Geometry, Ph.D. Thesis, Yale University (1978).

[7] G. T. Toussaint, Computing Largest Empty Circles with Location Constraints, International Journal of Computer and Information Sciences 12:5(1983), 347-358.

Figure 1. A Voronoi diagram.

Figure 2. Centers for possible largest empty circles.

Figure 3. The polygon made of Voronoi edges containing the convex polygon P.

Figure 4. kn intersections between a k-size polygon and a Voronoi diagram of size n.

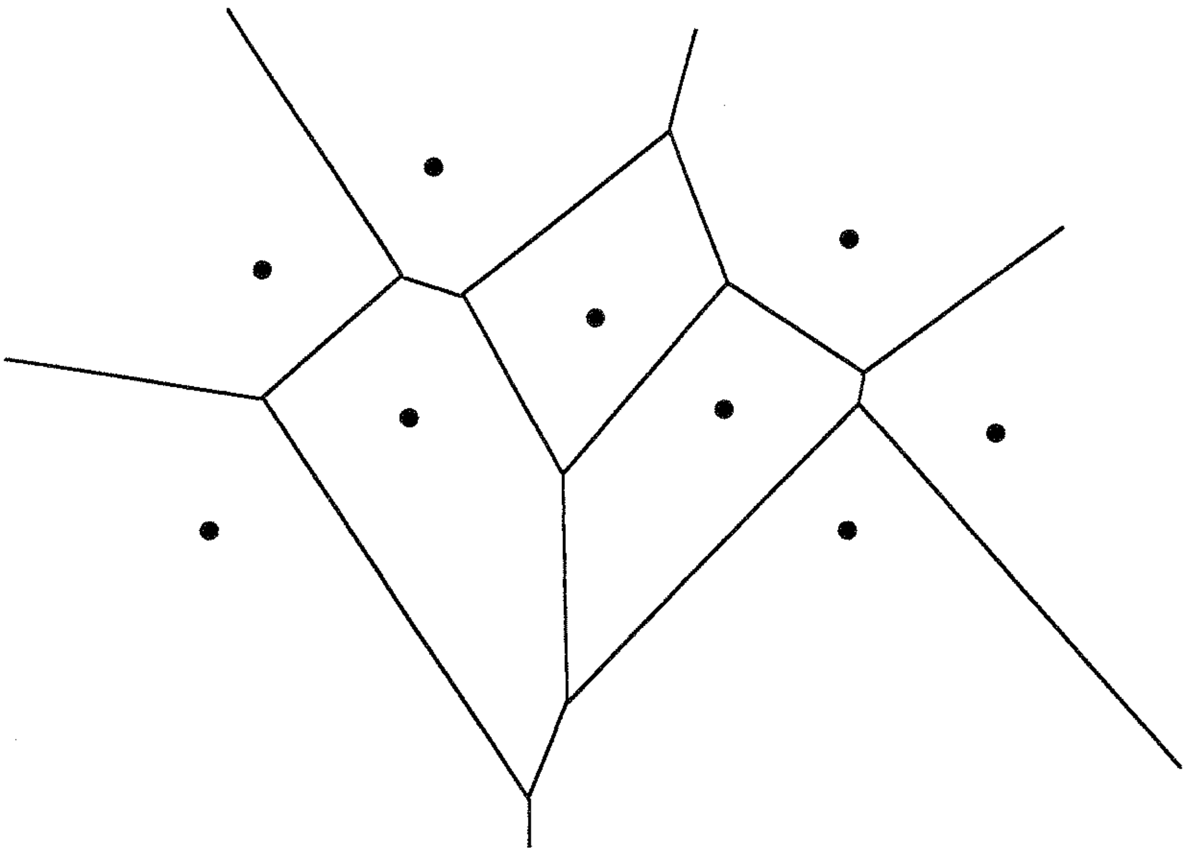

Figure 1

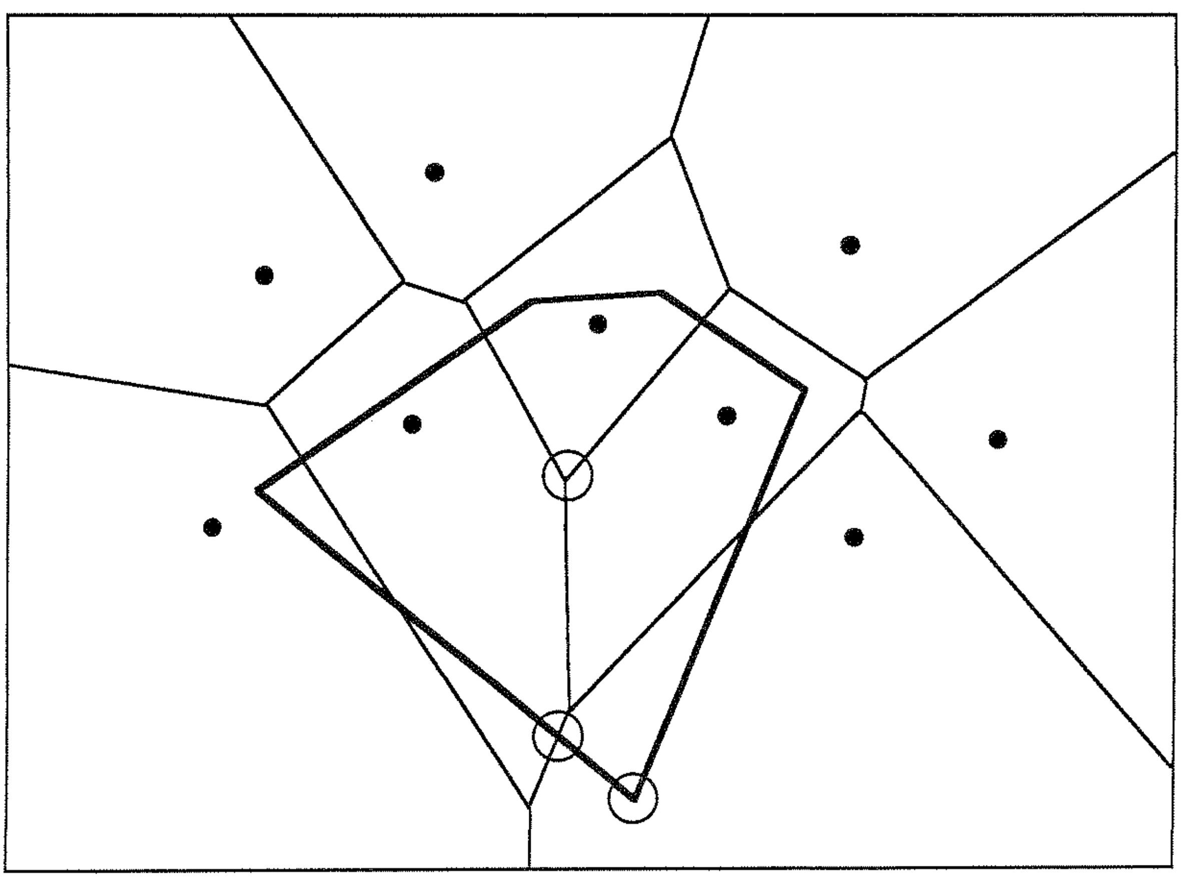

Figure 2

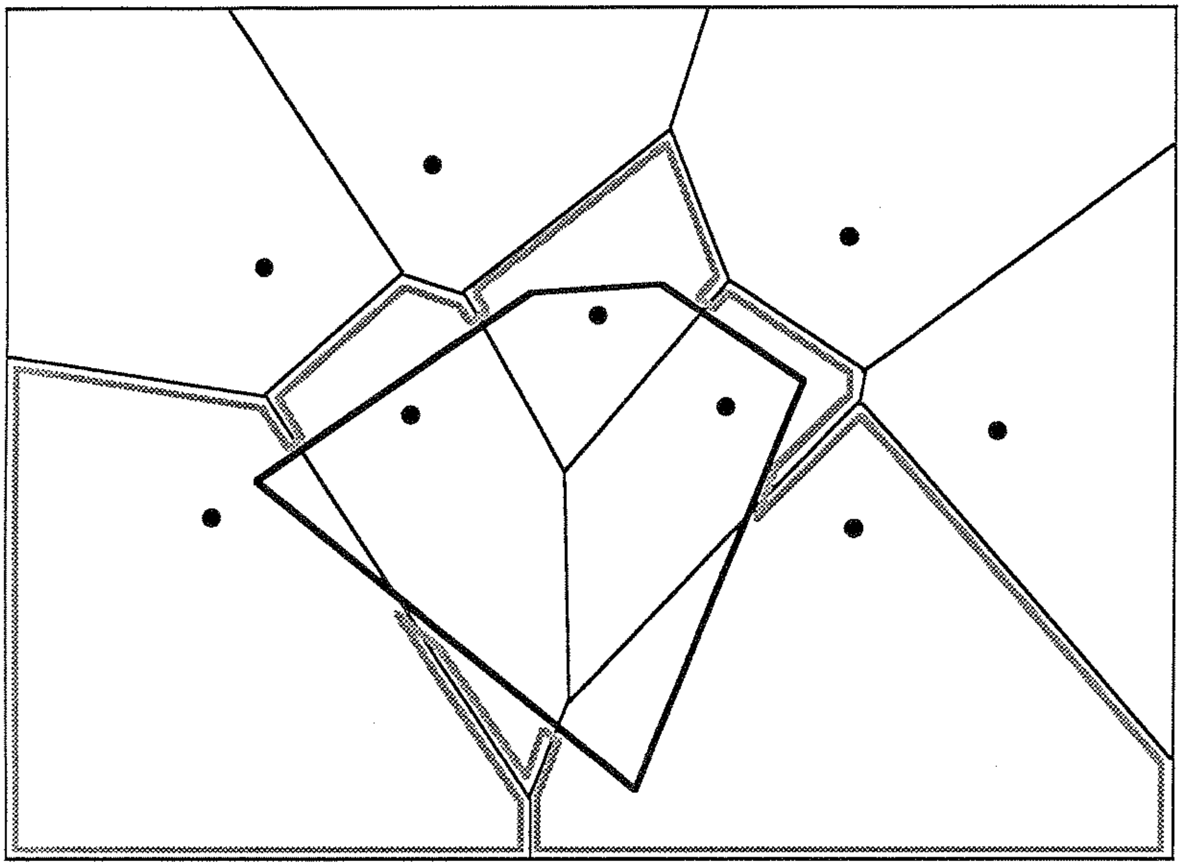

Figure 3

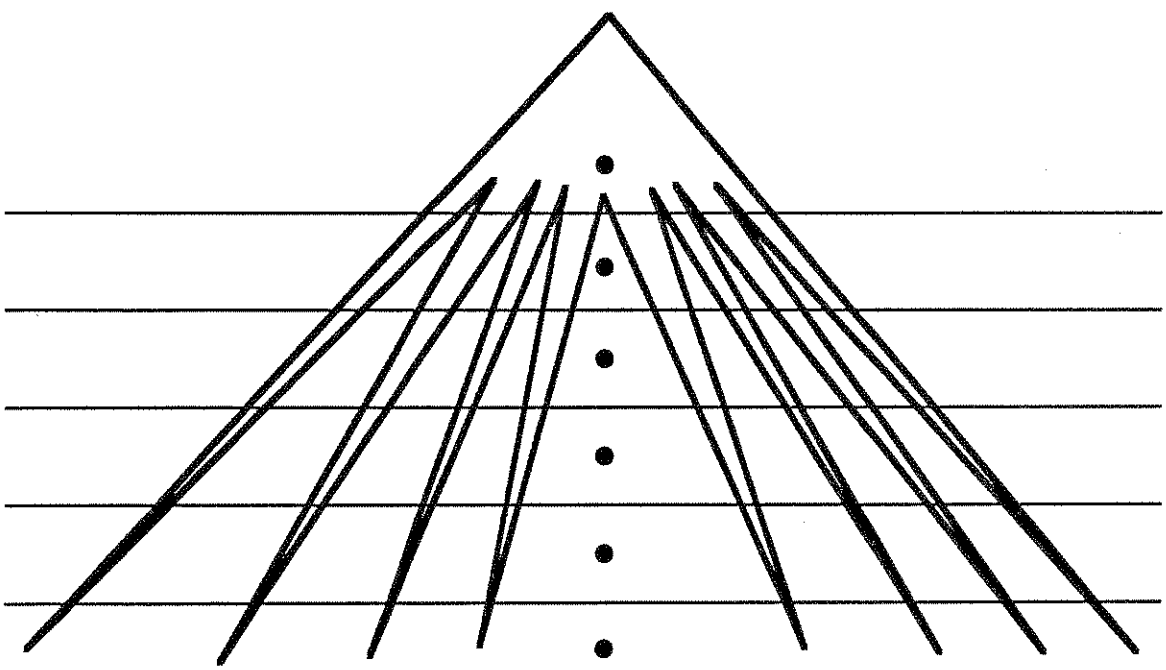

Figure 4