CMSC 754: Lecture 13
Line Arrangements

Reading: Chapter 8 in the 4M's.

Line Arrangements: We have studied a number of the most fundamental structures in computational geometry: convex hulls, Voronoi diagrams and Delaunay triangulations. These are all defined over a finite set of points. As we saw earlier, points and lines in the plane are related to each other through the *dual transformation*. In this lecture, we will studey a fundamental structure defined for a finite set of lines, called a *line arrangement*.

Consider a finite set $L = \{\ell_1, \ldots, \ell_n\}$ of lines in the plane. These lines naturally subdivide the plane into a cell complex, which is called the *arrangement* of L, and is denoted $\mathcal{A}(L)$ (see Fig. 1(a)). The points where two lines intersect form the vertices of the complex, the segments between two consecutive intersection points form its edges, and the polygonal regions between the lines form the faces, also called *cells*. Although an arrangement contains unbounded edges and faces, as we did with Voronoi diagrams (from a purely topological perspective) it is possible to add a vertex at infinity and attach all these edges to this vertex to form a proper planar graph (see Fig. 1(b)). An arrangement can be represented using any standard data structure for cell complexes, a DCEL for example.

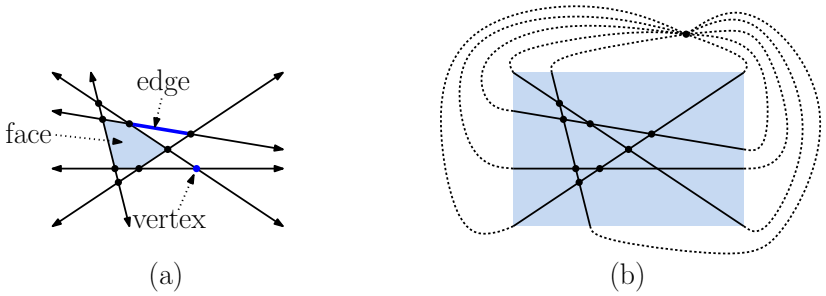


Fig. 1: Arrangement of lines; (a) the basic elements of an arrangement and (b) adding a vertex at infinity to form a proper planar graph.

As we shall see, arrangements have many applications in computational geometry. Through the use of point-line duality, many of these applications involve sets of points. We will begin by discussing the basic geometric and combinatorial properties of arrangements and an algorithm for constructing them. Later we will discuss applications of arrangements to other problems in computational geometry. Although we will not discuss it, line arrangements in $\mathbb{R}^2$ can be generalized to hyperplane arrangements in $\mathbb{R}^d$. In such a case the arrangement is a polyhedral cell complex.

Combinatorial Properties: The *combinatorial complexity* of an arrangement is the total number of vertices, edges, and faces in the arrangement. An arrangement is said to be *simple* if no three lines intersect at a common point. Through our usual general position assumption that no three lines intersect in a single point, it follows that we will be interested only in simple arrangements. We will also assume that no two lines are parallel. The following lemma shows that all of these quantities are $\Theta(n^2)$ for simple planar line arrangements.

Lemma: Let $\mathcal{A}(L)$ be a simple arrangement of n lines L in the plan. Then:

(i) the number of vertices (not counting the vertex at infinity) in $\mathcal{A}(L)$ is $\binom{n}{2} = \frac{1}{2}(n^2-n)$.

(ii) the number of edges in $\mathcal{A}(L)$ is n^2

(iii) the number of faces in $\mathcal{A}(L)$ is $\binom{n}{2} + n + 1 = \frac{1}{2}(n^2+n+2)$.

Proof: The fact that the number of vertices is $\binom{n}{2}$ is clear from the fact that (since no two are parallel) each pair of lines intersects in a single point.

The number of edges follows from the fact that each line contains n lines. This is because each line is cut by each of the other $n-1$ lines (assuming no two parallel lines), which splits the line into n edges.

The number of faces follows from Euler's formula, $v-e+f=2$. To form a cell complex, recall that we added an additional vertex at infinity. Thus, we have $v = 1 + \binom{n}{2}$ and $e = n^2$. Therefore, the number of faces is

$$\begin{aligned} f &= 2 - v + e = 2 - \left(1 + \tbinom{n}{2}\right) + n^2 = 2 - \left(1 + \tfrac{n(n-1)}{2}\right) + n^2 \\ &= 1 + \tfrac{n^2}{2} + \tfrac{n}{2} = 1 + \tfrac{n(n-1)}{2} + n = \tbinom{n}{2} + n + 1, \end{aligned}$$

as desired.

By the way, this generalizes to higher dimensions as well. The combinatorial complexity of an arrangement of n hyperplanes in $\mathbb{R}^d$ is $\Theta(n^d)$. Thus, these structures are only practical in spaces of relatively low dimension when n is not too large.

Incremental Construction: Arrangements are used for solving many problems in computational geometry. But in order to use an arrangement, we first must be able to construct it.[1] We will present a simple incremental algorithm, which builds an arrangement by adding lines one at a time. Unlike the other incremental algorithms we have seen so far, this one is *not randomized*. Its worst-case asymptotic running time, which is $O(n^2)$, holds irrespective of the insertion order. This is asymptotically optimal, since this is the size of the arrangement. The algorithm will also require $O(n^2)$ space, since this is the amount of storage needed to store the final result.

Let $L = \{\ell_1, \ldots, \ell_n\}$ denote the set of lines. We will add lines one by one and update the arrangement after each insertion. We will show that the ith line can be inserted in $O(i)$ time (irrespective of the insertion order). Summing over i, this yields a total running time proportional to $\sum_{i=1}^{n} i = O(n^2)$.

Suppose that the first $i-1$ lines have already been inserted. Consider the insertion of ℓ_i. We start by determining the leftmost (unbounded) face of the arrangement that contains this line. Observe that at $x = \infty$, the lines are sorted from top to bottom in increasing order of their slopes. In time $O(i)$ we can determine where the slope of ℓ_i falls relative to the slopes of the prior $i-1$ lines, and this determines the leftmost face of the arrangement that contains this

[1] This is not quite accurate. For some applications, it suffices to perform a plane-sweep of the arrangement. If we think of each line as an infinitely long line segment, the line segment intersection algorithm that was presented in class leads to an $O(n^2 \log n)$ time and $O(n)$ space solution. There exists a special version of plane sweep for planar line arrangements, called *topological plane sweep*, which runs in $O(n^2)$ time and $O(n)$ space. In spite of its fancy name, topological plane sweep is quite easy to implement.

line. (In fact, we could do this in $O(\log i)$ time by storing the slopes in an ordered dictionary, but this would not improve our overall running time. By our assumption that no two lines are parallel, there are no duplicate slopes.)

The newly inserted line cuts through a sequence of $i-1$ edges and i faces of the existing arrangement. In order to process the insertion, we need to determine which edges are cut by ℓ_i, and then we split each such edge and update the DCEL for the arrangement accordingly.

In order to determine which edges are cut by ℓ_i, we "walk" this line through the current arrangement, from one face to the next. Whenever we enter a face, we need to determine through which edge ℓ_i exits this face. We answer the question by a very simple strategy. We walk along the edges of the face, say in a counterclockwise direction until we find the exit edge, that is, the other edge that ℓ_i intersects. We then jump to the face on the other side of this edge and continue the trace with the neighboring face. This is illustrated in Fig. 2(a). The DCEL data structure supports such local traversals in time linear in the number of edges traversed. (You might wonder why we don't generalize the trapezoidal map algorithm. We could build a trapezoidal map of the arrangement and walk the new segment through a sequence of trapezoids. It turns out that this would be just as efficient.)

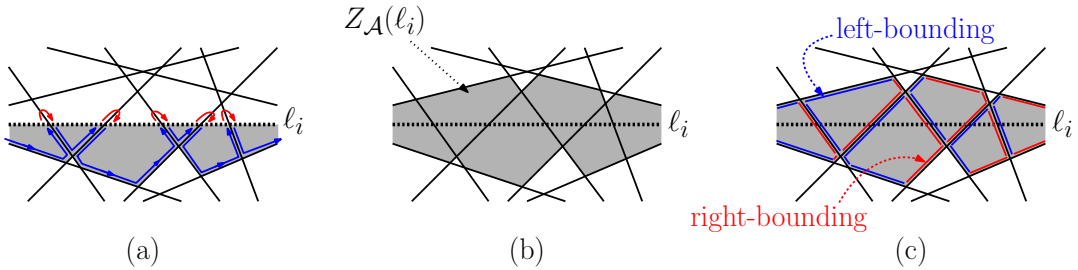


Fig. 2: Adding the line ℓ_i to the arrangement; (a) traversing the arrangement and (b) the zone of a line ℓ_i. (Note that only a portion of the zone is shown in the figure.)

Clearly, the time that it takes to perform the insertion is proportional to the total number of edges that have been traversed in this tracing process. A naive argument says that we encounter $i-1$ lines, and hence pass through i faces (assuming general position). Since each face is bounded by at most i lines, each facial traversal will take $O(i)$ time, and this gives a total $O(i^2)$, which is much higher than the $O(i)$ time that we promised earlier. Why is this wrong? It is based on bound of the total complexity of the faces traversed. To improve this, we need to delve more deeply into a concept of a *zone* of an arrangement.

Zone Theorem: The most important combinatorial property of arrangements (which is critical to their efficient construction) is a rather surprising result called the *zone theorem*. Given an arrangement $\mathcal{A}$ of a set L of n lines, and given a line ℓ that is not in L, the *zone* of ℓ in $\mathcal{A}(L)$, denoted $Z_{\mathcal{A}}(\ell)$, is the set of faces of the arrangement that are intersected by ℓ (shaded in Fig. 2(b)). For the purposes of the above construction, we are only interested in the edges of the zone that lie below ℓ_i, but if we bound the total complexity of the zone, then this will be an upper bound on the number of edges traversed in the above algorithm. The combinatorial complexity of a zone (as argued above) is at most $O(n^2)$. The Zone theorem states that the complexity is actually much smaller, only $O(n)$.

Theorem: (Zone Theorem) Given an arrangement $\mathcal{A}(L)$ of n lines in the plane, and given any line ℓ in the plane, the total number of edges in all the cells of the zone $Z_{\mathcal{A}}(\ell)$ is at most $6n$.

As with many combinatorial proofs, the key is to organize matter so that the counting can be done in an easy way. This is not trivial. We cannot count cell-by-cell, since some cells have high complexity and some low. We also cannot count line-by-line, because some lines contribute many edges to the zone and others just a few. The key in the proof is finding a (clever!) way to add up the edges so that each line appears to induce only a constant number of edges into the zone. (Note that our text counts zone edges a bit differently.)

Proof: The proof is based on a simple inductive argument. For the sake of illustration, let us rotate the plane so that ℓ is horizontal. By general position, we may assume that none of the lines of L are parallel to ℓ. We split the edges of the zone into two groups, those that bound some face from the left side and those that bound some face from the right side. An edge of a face is said to be *left bounding* if the face lies in the right halfplane of the line defining this edge, and a face is *right bounding* if the face lies in the left halfplane of the line defining this edge (see Fig. 2(c)). We will show that there are at most $3n$ left-bounding edges in the zone (highlighted in Fig. 3(a)), and by applying a symmetrical argument to the right-bounding edges, we have a a total of $6n$ edges.

The proof is by induction on n. For the basis case, when $n = 1$, then there is exactly one left-bounding edge in ℓ's zone, and $1 \leq 3 = 3n$. For the induction step, let us assume the induction hypothesis is true for any set of $n-1$ lines, and we will show that it holds for an arrangement of n lines. Consider the rightmost line of the arrangement to intersect ℓ. Call this ℓ_1 (see Fig. 2(c)). Prior to its existence, the induction hypothesis implies that there are at most $3(n-1)$ left-bounding edges in the zone of the remaining $n-1$ lines.

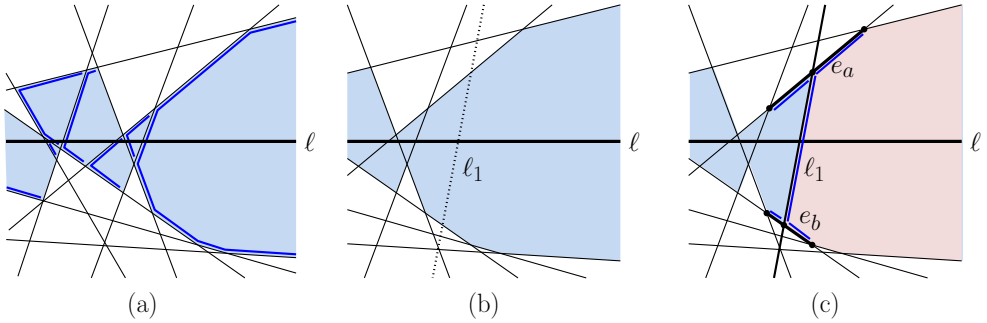


Fig. 3: Proof of the Zone Theorem.

Now, let us add ℓ_1 and see how many more left-bounding edges are generated. Because ℓ_1 is leftmost, it intersects the rightmost face of the zone. Observe that all of the edges of this face are left-bounding edges. By convexity, ℓ_1 intersects the boundary of this face in two edges, denoted e_a and e_b, where e_a is above ℓ, and e_b is below. Its insertion creates a new left-bounding edge running along ℓ_1 between e_a and e_b, and it splits each of the edges e_a and e_b into two new left-bounding edges. Thus, there is a net increase by three edges, for a total of $3(n-1)+3 = 3n$ edges.

We assert that ℓ_1 cannot contribute any other left-bounding edges to the zone. This is because the lines containing e_a and e_b block any possibility of this. Therefore, there are at most $3n$ left bounding edges, as desired.

Applications of Arrangements and Duality: Line arrangements, when combined with the dual transformation, make it possible to solve a number of geometric computational problems. A number of examples are given below. Unless otherwise stated, all these problems can be solved in $O(n^2)$ time and $O(n^2)$ space by constructing a line arrangement. Alternately, they can be solved in $O(n^2 \log n)$ time and $O(n)$ space by applying plane sweep to the arrangement.

General position test: Given a set of n points in the plane, determine whether any three are collinear.

Minimum area triangle: Given a set of n points in the plane, determine the minimum area triangle whose vertices are selected from these points.

Minimum k-corridor: Given a set of n points, and an integer k, determine the narrowest pair of parallel lines that enclose at least k points of the set. The distance between the lines can be defined either as the vertical distance between the lines or the perpendicular distance between the lines (see Fig. 4(a)).

Visibility graph: Given line segments in the plane, we say that two points are *visible* if the interior of the line segment joining them intersects none of the segments. Given a set of n non-intersecting line segments, compute the *visibility graph*, whose vertices are the endpoints of the segments, and whose edges a pairs of visible endpoints (see Fig. 4(b)).

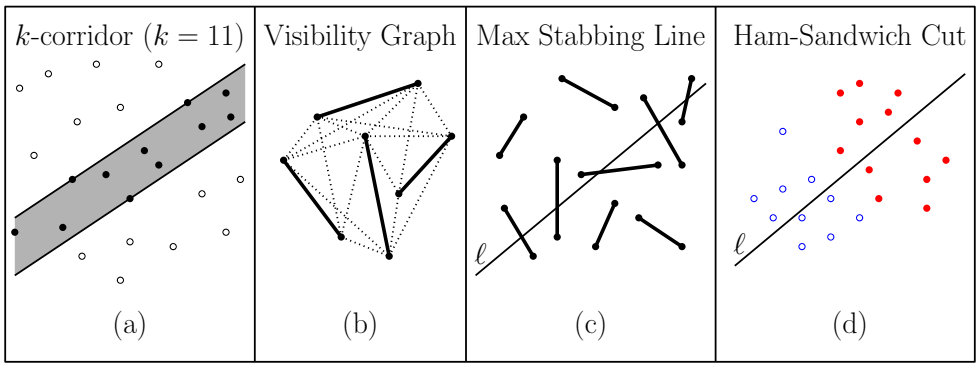


Fig. 4: Applications of arrangements.

Maximum stabbing line: Given a set of n line segments in the plane, compute the line ℓ that stabs (intersects) the maximum number of these line segments (see Fig. 4(c)).

Ham Sandwich Cut: Given n red points and m blue points, find a single line ℓ that simultaneously bisects these point sets. It is a famous fact from mathematics, called the *Ham-Sandwich Theorem*, that such a line always exists. If the two point sets are separable by a line (that is, the red convex hull and the blue convex hull do not intersect), then this can be solved in time $O(n + m)$ (see Fig. 4(d)).

In the remainder of the lecture, we'll see how problems like these can be solved through the use of arrangements.

Sweeping Arrangements: Since an arrangement of n lines is of size $\Theta(n^2)$, we cannot expect to solve problems through the explicit use of arrangements in less than quadratic time. Most applications involve first constructing the arrangement, and then traversing it in some manner. In many instances, the most natural traversal to use is based on a plane-sweep. (This is not the only way however. Since a planar arrangement is a graph, methods such as depth-first and breadth-first search can be used.)

If an arrangement is to be built just so it can be swept, then maybe you don't need to construct the arrangement at all. You can just perform the plane sweep on the lines, exactly as we did for the line segment intersection algorithm. Assuming that we are sweeping from left to right, the initial position of the sweep line is at $x = -\infty$ (which means sorting by slope). The sweep line status maintains the lines in, say, bottom to top order according to their intersection with the sweep line. The events are the vertices of the arrangement.

Note that the sweep-line status always contains exactly n entries. Whenever an intersection event occurs, we can update the sweep-line status by swapping two adjacent entries. Thus, instead of an ordered dictionary, it suffices to store the lines in a simple n-element array, sorted, say, from top to bottom. This means the **sweep-line updates can be performed in $O(1)$ time**, rather than $O(\log n)$(see Fig. 5(a)). We still need to maintain the priority queue, and these operations take $O(\log n)$ time each.

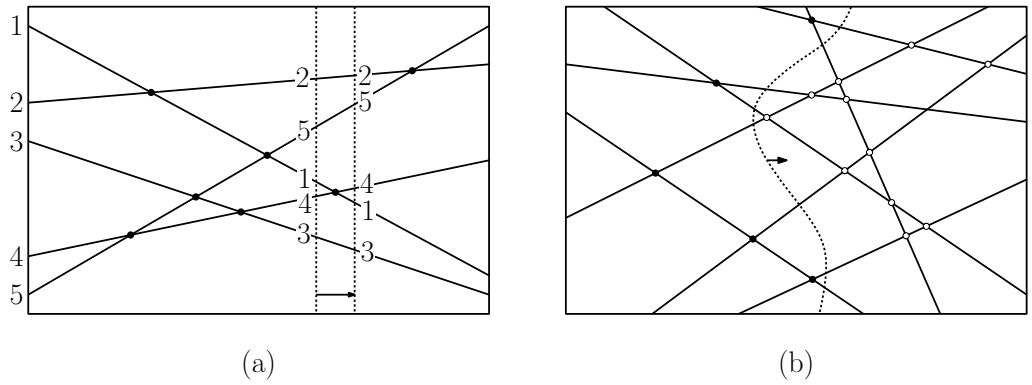


Fig. 5: Sweeping a line arrangement.

Sweeping an arrangement in this manner takes $O(n^2 \log n)$ time, and $O(n)$ space. Because it is more space-efficient, this is often an attractive alternative to constructing the entire subdivision.

Topological Plane Sweep: (Optional) As mentioned above, the priority queue is the slowest part of plane sweeping an arrangement. Remarkably, there is a way to save this $O(\log n)$ factor, but we must abandon hope of sweeping events in purely left-to-right order. There is a somewhat more "relaxed" version of plane sweep, which works for line arrangements in the plane. The method is called *topological plane sweep*. It runs in $O(n^2)$ time (thus, eliminating an $O(\log n)$ factor from the running time) and uses $O(n)$ space.

It achieves efficiency relaxing the requirement that vertices be swept in strict left-to-right order. Rather, it uses a more "local" approach for deciding which vertex of the arrangement

to sweep next.[2] This local approach guarantees that the vertices along each line are swept in their proper order, even though vertices lying on different lines are not necessarily swept in their proper left-to-right order. Intuitively, we can think of the sweep line as a sort of *pseudoline*, that intersects each line of the arrangement exactly once (see Fig. 5(b)). Although we will not present any justification, it method applicable to all the problems we will discuss in today's lecture.

Duality: Many of our applications will involve the dual transformation, which we introduced earlier in the semester. Recall that the dual of a point $p = (a, b)$ is the line $p^* : y = ax - b$, and the dual of a line $\ell : y = ax - b$ is the point $\ell^* = (a, b)$. Also recall the *order-reversing property* that the point p lies above line ℓ (at vertical distance h) if and only if the dual line p^* lies below dual point ℓ^* (also at vertical distance h).

Narrowest k-corridor: We are given a set P of n points in the plane and an integer k, $1 \leq k \leq n$, and we wish to determine the narrowest pair of parallel lines that enclose at least k points of the set. (We call this a *slab* or *corridor*.) We define the *width* of the corridor to be the vertical distance between these. Our objective is to compute the corridor of minimum width that encloses k points (which may lie on the corridor's boundary.) It is straightforward to adapt the algorithm to minimize the perpendicular distance between the lines. We will make the usual general-position assumptions that no three points of P are collinear and no two points have the same x-coordinate.

Consider any corridor defined by parallel lines ℓ_a (above) and ℓ_b (below) (see Fig. 6(a)). Since the lines are parallel, these points share the same a-coordinate, which implies that the line segment $\overline{\ell_a^* \ell_b^*}$ is vertical (see Fig. 6(b)). The vertical distance between the lines (that is, the difference in their y-intercepts) is the same as the vertical distance between the dual points.

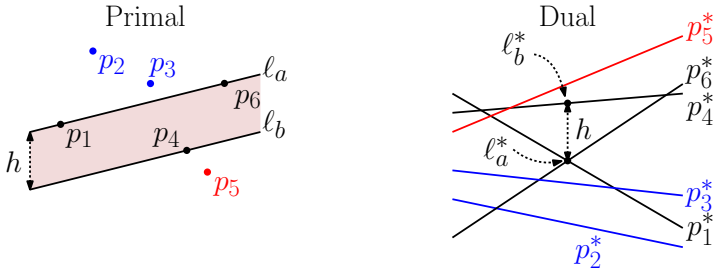


Fig. 6: A 3-corridor in primal and dual forms. (Note that the corridor is not as narrow as possible.)

By the order-reversing property, points that lie above/below/within the corridor (shown in blue, red, and black in Fig. 6(a), respectively) are mapped to dual lines that pass below/above/through this segment (see Fig. 6(b)). Thus, we have the following equivalent dual formulation of this problem:

Shortest vertical k-stabber: Given an arrangement of n lines, determine the shortest vertical segment that stabs (intersects) k lines of the arrangement.

[2] For details, see "Topologically sweeping an arrangement" by H. Edelsbrunner and L. J. Guibas, *J. Comput. Syst. Sci.*, 38 (1989), 165–194.

It is easy to show that the shortest vertical k-stabber may be assumed to have one of its endpoints on a vertex of the arrangement. (If the vertical line has endpoints in the interior of two edges, we can slide it left or right and decrease its length.)

3-stabber: The 3-stabber is the simplest to describe. A perform a simple plane sweep of the arrangement (using a vertical sweep line). Whenever we encounter a vertex of the arrangement, we consider the distance from this vertex to the edge of the arrangement lying immediately above this vertex and the edge lying immediately below. (These are illustrated by the blue broken lines in Fig. 7(a).) We can solve this problem by plane sweep in $O(n^2 \log n)$ time and $O(n)$ space. (By using topological plane sweep, the extra $\log n$ factor in the running time can be removed.)

Note that we can use this to test whether points are in general position. It is easy to prove that a set of n points has three or more collinear points if and only if the dual arrangement's minimum 3-stabber is of length zero.

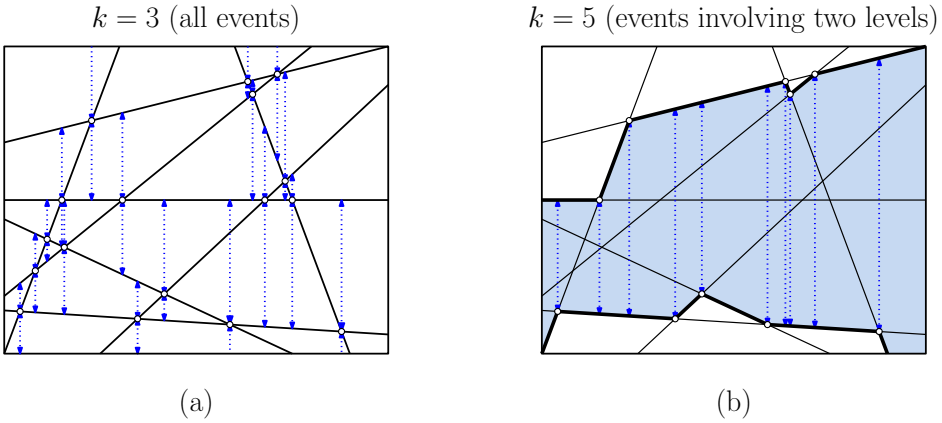


Fig. 7: The critical events in computing the shortest vertical 3-stabber (a) and 5-stabber (b).

k-stabber: Whenever we encounter a vertex in the plane sweep, we determine the distance to the lines of the arrangement that lie $k-2$ above and $k-2$ below (see the blue broken lines of Fig.7(b)). The reason for the "-2" is to account for two lines that pass through the vertex itself. Recalling that the sweep-line status can be stored in a simple n-element array, it is possible to access these entries in $O(1)$ time for any value of k.

Halfplane Discrepancy: Next we consider a problem derived geometric sampling. Suppose that we are given a collection of n points P lying in a unit square $U = [0,1]^2$. We want to use these points for random sampling purposes. In particular, the property that we would like these points to possess is that for any halfplane h, we the fraction of points of P that lie within h should be roughly equal to the area of intersection of h with U. More precisely, define $\mu(h)$ to be the area of $h \cap U$, and $\mu_P(h) = |P \cap h|/|P|$. A sample is good if $\mu(h) \approx \mu_P(h)$, for any choice of h.

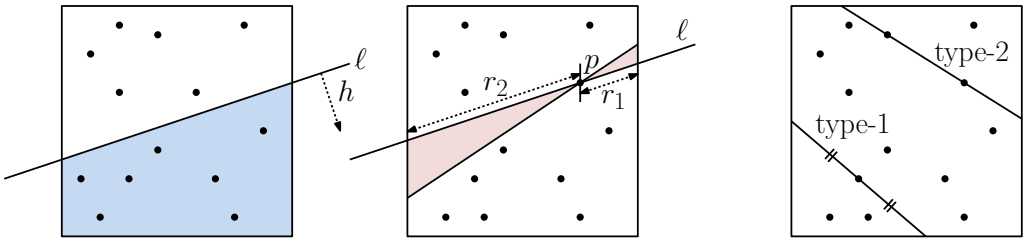
To make this more formal, we define the *discrepancy* (or more accurately, the *halfplane discrepancy*) of a finite point set P with respect to a halfplane h to be

$$\Delta_P(h) = |\mu(h) - \mu_P(h)|.$$

For example, in Fig. 8(a), the area of $h \cap U$ is $\mu(h) = 0.625$, and there are 7 out of 13 points in h, thus $\mu_P(h) = 7/13 = 0.538$. Thus, the discrepancy of h is $|0.625 - 0.538| = 0.087$. Define the *halfplane discrepancy* of P to be the maximum (or more properly the supremum, or least upper bound) of this quantity over all halfplanes:

$$\Delta(P) = \sup_h \Delta_P(h).$$

Let's consider the problem of computing the discrepancy of a given point set P.



(a) (b) (c)

Fig. 8: Discrepancy of a point set.

Since there are an uncountably infinite number of halfplanes that intersect the unit square, we should first derive some sort of *finiteness criterion* on the set of halfplanes that might produce the greatest discrepancy.

Lemma: Let h denote the halfplane that generates the maximum discrepancy with respect to P, and let ℓ denote the line that bounds h. Then either:

(i) ℓ passes through one point of P, and this point is the midpoint of the line segment $\ell \cap U$, or

(ii) ℓ passes through two points of P.

Remark: If a line passes through one or more points of P, then should this point be included in $\mu_P(h)$? For the purposes of computing the maximum discrepancy, the answer is to either include or omit the point, whichever produces the larger discrepancy. The justification is that it is possible to perturb h infinitesimally so that it includes none or all of these points without altering $\mu(h)$.

Proof: We will show that any line can be moved until it satisfies either (i) or (ii) in such a manner that the discrepancy never decreases. First, if ℓ does not pass through any point of P, then (depending on which is larger $\mu(h)$ or $\mu_P(h)$) we can move the line up or down without changing $\mu_P(h)$ and increasing or decreasing $\mu(h)$ to increase their difference, until it does pass through a point of P. Next, if ℓ passes through a point $p \in P$, but is not the midpoint of the line segment $\ell \cap U$, then we claim that we can rotate this line about p and hence increase or decrease $\mu(h)$ without altering $\mu_P(h)$, to increase their difference.

To establish the claim, consider Fig. 8(b). Suppose that the line ℓ passes through point p and let $r_1 < r_2$ denote the two lengths along ℓ from p to the sides of the square. Observe that if we rotate ℓ through a small angle θ, then to a first order approximation, the gain

due to area of the triangle on the right is $r_1^2\theta/2$, since this triangle can be approximated by an angular sector of a circle of radius r_1 and angle θ. The loss due to the area of the triangle on the left is $r_2^2\theta/2$. Thus, since $r_1 < r_2$ this rotation will decrease the area of region lying below h infinitesimally. A rotation in the opposite increases the area infinitesimally. Since the number of points bounded by h does not change as a function of θ, the discrepancy cannot be achieved as long as such a rotation is possible.

We say that a line is *type-1* if it satisfies condition (i) and it is *type-2* if it satisfies condition (2) (see Fig. 8(c)). We will show that the discrepancy for each types of lines can be computed in $O(n^2)$ time.

Type-1: For each point $p \in P$, there are only a constant number of lines ℓ (at most two, I believe) through this point such that p is the midpoint of $\ell \cap U$. It follows that there are at most $O(n)$ type-1 lines. We can compute the discrepancy of each such line in $O(n)$ time, which leads to a total running time of $O(n^2)$.

Type-2: Consider a type-2 line ℓ that passes through two points $p_i, p_j \in P$. This line defines two halfplanes, one above and one below. We'll explain how to compute the discrepancy of the lower halfplane, h^-, and the upper halfplane, h^+, is symmetrical. First, observe that we can compute $\mu(h^-)$ in constant time, so all that remains is computing $\mu_P(h^-)$, that is, the number of points lying on or below ℓ.

If we apply our standard dual transformation, ℓ is mapped in the dual plane to a point ℓ^* at which the dual lines p_i^* and p_j^* intersect. This is just a vertex in the line arrangement $\mathcal{A}(P^*)$. By the order-reversing property of the dual transformation, the points lying on or below ℓ coincide with the dual lines that lie on or above this vertex.

We can compute this quantity in constant time for each vertex of the line arrangement. Recall that the sweep-line status is stored in a simple n-element array, sorted from top to bottom. The vertex in the arrangement corresponds to two consecutive entries in the sweep-line status, say at positions $k-1$ and k. The number of dual lines lying on or above the vertex is therefore just k (assuming that we index the array from 1 to n).

For example, consider the vertex being swept in Fig. 5(a). These intersecting lines are at indices $k-1 = 3$ and $k = 4$ in the sweep-line status, and hence there are 4 lines on or above this vertex, which implies that there are 4 points lying on or below the corresponding type-2 line $\overline{p_1p_4}$ in the primal configuration. Since we can compute the discrepancy for each type-2 line in $O(1)$ time, the overall time to compute the discrepancies of all type-2 lines is $O(n^2)$.

Levels: The analysis that was done above for type-2 lines suggests another useful structure within a line arrangement. We can classify each element of the arrangement according to the number of lines lying above and below it. We say that a point is at *level* k, denoted $\mathcal{L}_k$, in an arrangement if there are at most $k-1$ lines (strictly) above this point and at most $n-k$ lines (strictly) below it. It is easy to see that $\mathcal{L}_k$ is an x-monotone polygonal curve (see Fig. 9(a)). For example, $\mathcal{L}_1$ is the upper envelope of the lines, and $\mathcal{L}_n$ is the lower envelope. Assuming general position, each vertex of the arrangement is on two levels, which meet at this vertex in a "knocked-knee" manner. Given the arrangement $\mathcal{A}(L)$ of a set of n lines, it is an easy

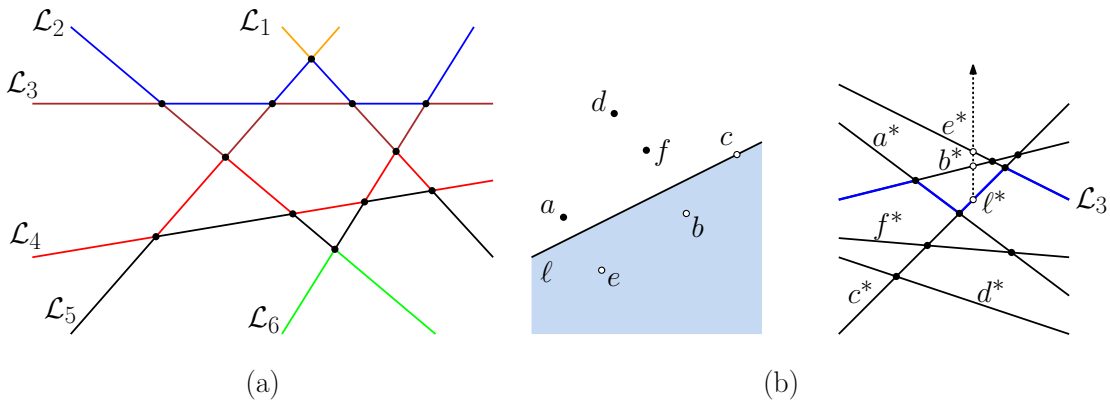


Fig. 9: Levels in an arrangement and k-sets.

matter to label every edge of the arrangement with its level number, by tracking its index in the sweep-line status.

There is a dual equivalence between a level in an arrangement and a concept called k-sets. Given an n-element point set P and integer $1 \leq k \leq n$, a k-element subset of $P' \subseteq P$ is called a *k-set* of P if there exists a halfplane h such that $P' = P \cap h$. For example, if (p_i, p_j) is an edge of the convex hull of P, then $P' = \{p_i, p_j\}$ is a 2-set of P. A classical question in combinatorial geometry is, as a function of n and k, what is the maximum number of possible k-sets that any n-element set can have. (The current best bounds range between $O(n \log k)$ and $O(nk^{1/3})$.)

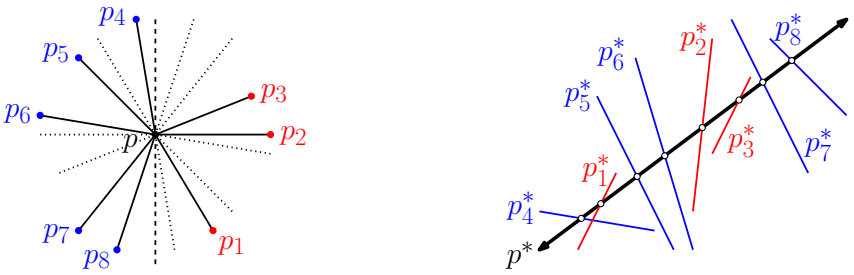
There is a close relationship between the k-sets of P and level k of the dual arrangement $\mathcal{A}(P^*)$. To see this, let us first distinguish between two types of k-sets. We say that a k-set is a *lower k-set* if it lies in the halfplane beneath a line ℓ and otherwise it is an *upper k-set*. Let's just consider lower k-sets, since upper k-sets are symmetrical (by reflecting the points about the x-axis).

Consider any lower k-set defined by some line ℓ. We may assume that ℓ passes through a point of P, and hence there are $k-1$ points strictly below ℓ. The associated dual point ℓ^* lies on an edge of the dual arrangement, and by the order-reversing property of the dual transformation, there are $k-1$ lines of $\mathcal{A}(P^*)$ that pass strictly above ℓ^*. (For example, in Fig. 9(b), the lower 3-set $\{c, b, e\}$ is defined by a line ℓ, which passes through c. In the dual setting, the point ℓ^* lies on the dual line c^* and lies on level $\mathcal{L}_3$ because lines b^* and e^* lie above it.) The upper k-sets can be identified with level $\mathcal{L}_{n-k+1}$, because each point on this level has k lines passing on or below it, and hence $n-k+1$ lines on or above.

Sorting all angular sequences: Earlier, we introduced the problem of computing visibility graphs. We will not explicitly discuss the solution of that problem here, but we will discuss a fundamental subroutine in this algorithm. Consider a set of n points in the plane. For each point p in this set we want to sort the remaining $n-1$ point in cyclic order. Clearly, we can compute the cyclically sorted order about any given point in $O(n \log n)$ time, and this leads to an overall running time of $O(n^2 \log n)$. We will show that we can do this $O(n^2)$ time. (This is rather surprising. Lower bounds on sorting imply that we cannot sort n sets of $n-1$

numbers faster than $\Omega(n^2 \log n)$ time. But here we are exploiting the special structure of the cyclically ordered point sets.)

Here is how we do it. Suppose that p is the point around which we want to sort, and let $\langle p_1, \ldots, p_n \rangle$ be the points in final angular order about p (see Fig. 10(a)). Consider the arrangement defined by the dual lines p_i^*. How does this order manifest itself in the arrangement?



(a) (b)

Fig. 10: Arrangements and angular sequences.

Consider the dual line p^*, and its intersection points with each of the dual lines p_i^*. These form a sequence of vertices in the arrangement along p^*. Consider this sequence ordered from left to right. It would be nice if this order were the desired circular order, but this is not quite correct. It follows from the definition of our dual transformation that the a-coordinate of each of these vertices in the dual arrangement is the slope of some line of the form $\overline{pp_i}$ in the primal plane. Thus, the sequence in which the vertices appear on the line is a *slope ordering* of the points about p_i, which is not quite the same as the *angular ordering*.

However, given this slope ordering, we can simply test which primal points lie to the left of p (shown in blue in Fig. 10(a)), and separate them from the points that lie to the right of p (shown in red in Fig. 10(a)). We partition the vertices into two sorted sequences, and then concatenate these two sequences, with the points on the right side first, and the points on the left side later. For example, in Fig. 10, we partition the slope-sorted sequence $\langle 4, 1, 5, 6, 2, 3, 7, 8 \rangle$ into the left sequence $\langle 4, 5, 6, 7, 8 \rangle$ and the right sequence $\langle 1, 2, 3 \rangle$, and then we concanate them to obtain the final angle-sorted sequence $\langle 1, 2, 3, 4, 5, 6, 7, 8 \rangle$.

Thus, once the arrangement has been constructed, we can reconstruct each of the angular orderings in $O(n)$ time, for a total of $O(n^2)$ time. (Topological plane sweep can also be used, but since the output size is $\Omega(n^2)$, there no real benefit to be achieved by using topological plane sweep.)