1 Delaunay Triangulation, Voronoi Diagram, and Gabriel Graph

Show that p_i and p_j have an edge in $\mathcal{GG}(S)$ if and only if $\overline{p_ip_j}$ appears in the $\mathcal{DT}(S)$ and $\overline{p_ip_j}$ crosses its dual Voronoi edge.

($\Rightarrow$) For two points $p_i, p_j \in S$, $\overline{p_ip_j}$ is the edge that connects them, and C_{ij} is the circle with $\overline{p_ip_j}$ as the diameter. We say that C_{ij} is an empty circle with no points of S in its interior and therefore $\overline{p_ip_j} \in \mathcal{GG}(S)$; we then consider a third point $p_k \in S$, where $p_k \neq p_i, p_j$ and p_k is not interior to C_{ij}. Point p_k is either:

1. p_k is on the circumference of C_{ij}. Therefore, by definition[1], $\Delta p_ip_jp_k$ is a face in the Delaunay triangulation, and so $\overline{p_ip_j} \in \mathcal{DT}(S)$ (as do the other edges formed by the three points). $\Delta p_ip_jp_k$ is a right triangle because they are three points on a circle whose diameter is the hypotenuse. The Voronoi vertex that is the end point of the dividing Voronoi edge between p_i and p_j must be located within the right triangle, equidistance from the three points, and the edge must intersect the circumcenter given by the midpoint of $\overline{p_ip_j}$, because there are no other points intertior or on the circle for which more Voronoi edges would form.

2. p_k is exterior to C_{ij}. Therefore, by definition[2], $\overline{p_ip_j} \in \mathcal{DT}(S)$ and the $\overline{p_ip_j}$ could only cross the dividing Voronoi edge that separates the regions of site p_i and p_j.

($\Leftarrow$) Given $\overline{p_ip_j}$ appears in the $\mathcal{DT}(S)$ and $\overline{p_ip_j}$ crosses its dual Voronoi edge, I show by contradiction that $\overline{p_ip_j}$ is also in the $\mathcal{GG}(S)$[3].

Suppose that $\overline{p_ip_j}$ doesn't cross its dual Voronoi edge. This means that the dividing edge was "pushed" either above or below $\overline{p_ip_j}$, and $\overline{p_ip_j}$ now intersects two other edges of a third Voronoi cell, given by p_k. So there is a Voronoi vertex equidistance from the three points and this vertex and the edge it's on doesn't intersect the midpoint of $\overline{p_ip_j}$. This position means the distance between p_i and p_k, and the distance between p_j and p_k are both shorter than the distance between p_i and p_j. Therefore, taking the circle with the diameter $\overline{p_ip_j}$, p_k must lie interior to the circle. Therefore, $\overline{p_ip_j}$ cannot be in $\mathcal{GG}(S)$. (Note that this edge can still be in $\mathcal{DT}(S)$ if the circle given by $\Delta p_ip_jp_k$ have no interior points; this just simply means that $\overline{p_ip_j}$ doesn't pass through its circumcircle).

Since $\mathcal{GG}(S) \subseteq \mathcal{DT}(S)$, we can find $\mathcal{GG}(S)$ very easily. First, compute the $\mathcal{DT}(S)$, which can be done in $O(n \log n)$ time and $O(n)$ space[4]. Then, We can walk the edges of $\mathcal{DT}(S)$ in $O(n)$ time, and check each if the circle with just that edge, $\overline{p_ip_j}$, as the diameter contains an interior point or not. Since this was an edge in the $\mathcal{DT}$, there is only one potential point that

[1] Theorem 9.6.i [2] pg. 198
[2] Theorem 9.6.ii [2] pg. 198
[3] Jake
[4] Theorem 9.12 [2] pg. 206

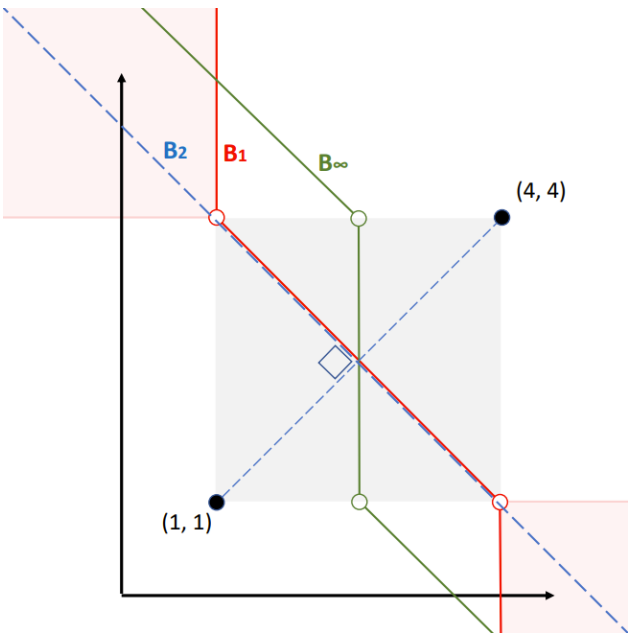
could be interior, the third point in the triangle; so we only need to check if the neighbors of p_i, p_j are interior to C_{ij}; assuming general position, there is a constant number of neighbors to check, 2 for each point. If it has interior points, then discard that edge; otherwise, it is an edge in $\mathcal{GG}(S)$. The total runtime is $O(n \log n)$ due to computing the $\mathcal{DT}(S)$, which is optimal and faster than checking every pair of points with every other point for being interior, which would come out to be $O(n^3)$ time.

2 Voronoi Diagrams in L_1 and L_∞ Metrics

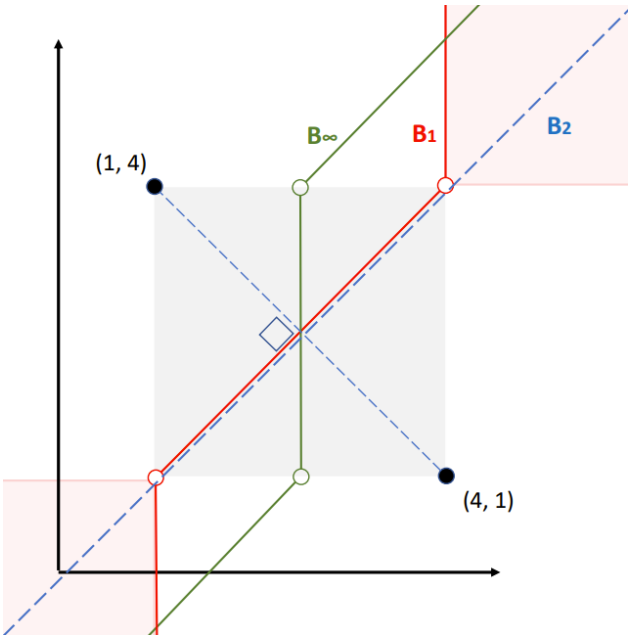
I use the the grid layout in MS powerpoint (grid lines don't export) to draw the below diagrams. The Voronoi diagram in the L_1 metric is given in solid red lines; it is made of the bisectors (B_1's) that separate the regions of two separate points. The Voronoi diagram in the L_∞ metric is given in solid green lines; it is made of the bisectors (B_∞'s) that separate the regions of two separate points. The regular Voronoi diagram (given in the L_2 metric) is given in the thicker dashed blue lines (the segment connecting two points is thin-dashed); it is made of the perpendicular bisectors (B_2's) that separate the regions of two separate points.

I start by characterizing the bisectors between two points $q_i(x_i, y_i)$ and $q_j(x_j, y_j)$ in the three different metrices in different general positions using real values for points; q_i will be left of q_j, i.e. $x_i < x_j$; everything to one side of a bisector B_p in the L_p metric is closer to one point than it is so the other, because the bisectors are equal distance from either points. L_1-distance is $d_1(q_i, q_j) = |x_i - x_j| + |y_i - y_j|$; L_∞-distance is $d_\infty(q_i, q_j) = \max\Big(|x_i - x_j|,\ |y_i - y_j|\Big)$[5]. Points on the bisectors are found by taking half of the distance.

In each of the six cases below, notice that B_2 is bounded by B_1 and B_∞ which have a criss-cross pattern with each other[3]. In general, for $1 \leq p \leq \infty$, we can say that B_p is bounded by B_1 and B_∞, and that all dividing edges in each metrices intersect at a common point in the rectangle given by q_i and q_j. In the case that q_i and q_j form a square region (when $|x_i - x_j| = |y_i - y_j|$), there are two regions in the L_1 metric shaded in red in which any point there could be equal distance from q_i and q_j; for the diagrams below, I take the bisector in the dark, solid red; likewise for the bisector shown for the L_∞ metric.

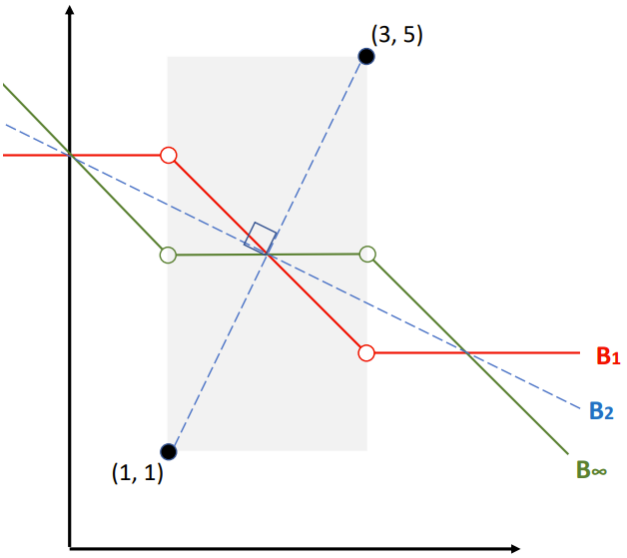


(a) $|x_i - x_j| = |y_i - y_j|$, $y_i < y_j$

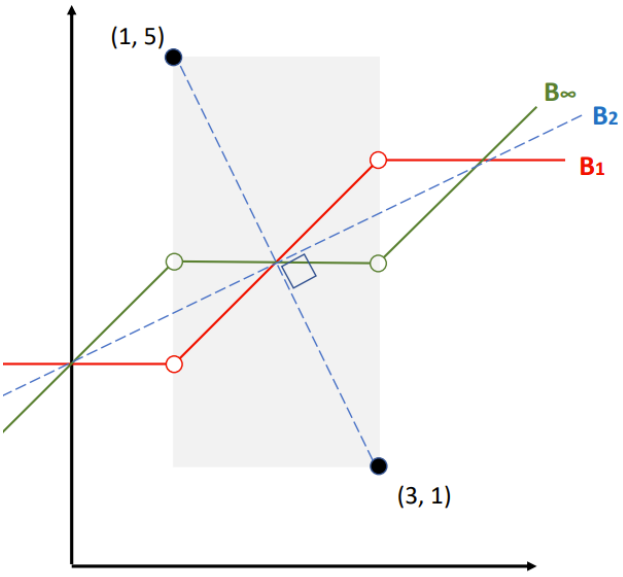


(b) $|x_i - x_j| = |y_i - y_j|$, $y_i > y_j$

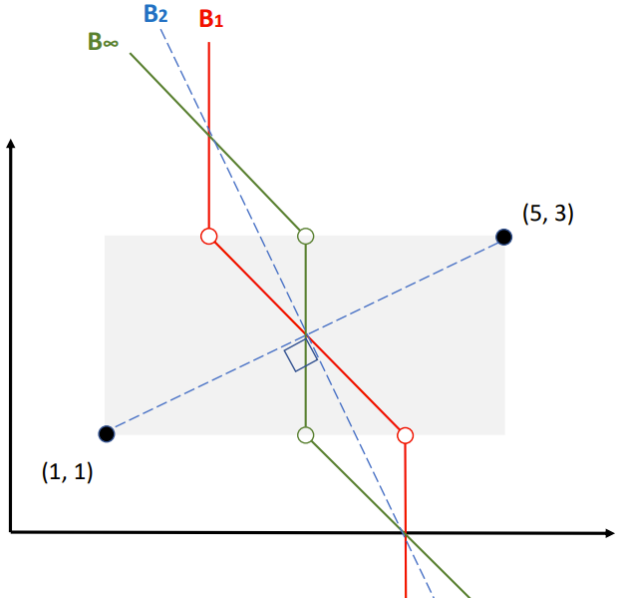
[5] Jake



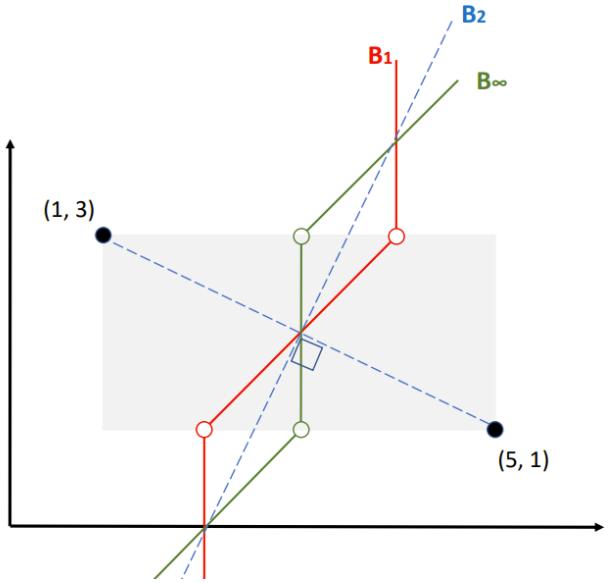
(a) $|y_i - y_j| > |x_i - x_j|$, $y_i < y_j$



(b) $|y_i - y_j| > |x_i - x_j|$, $y_i > y_j$



(c) $|x_i - x_j| > |y_i - y_j|$, $y_i < y_j$



(d) $|x_i - x_j| > |y_i - y_j|$, $y_i > y_j$

(a) **Voronoi Diagram in L_1 Metric**

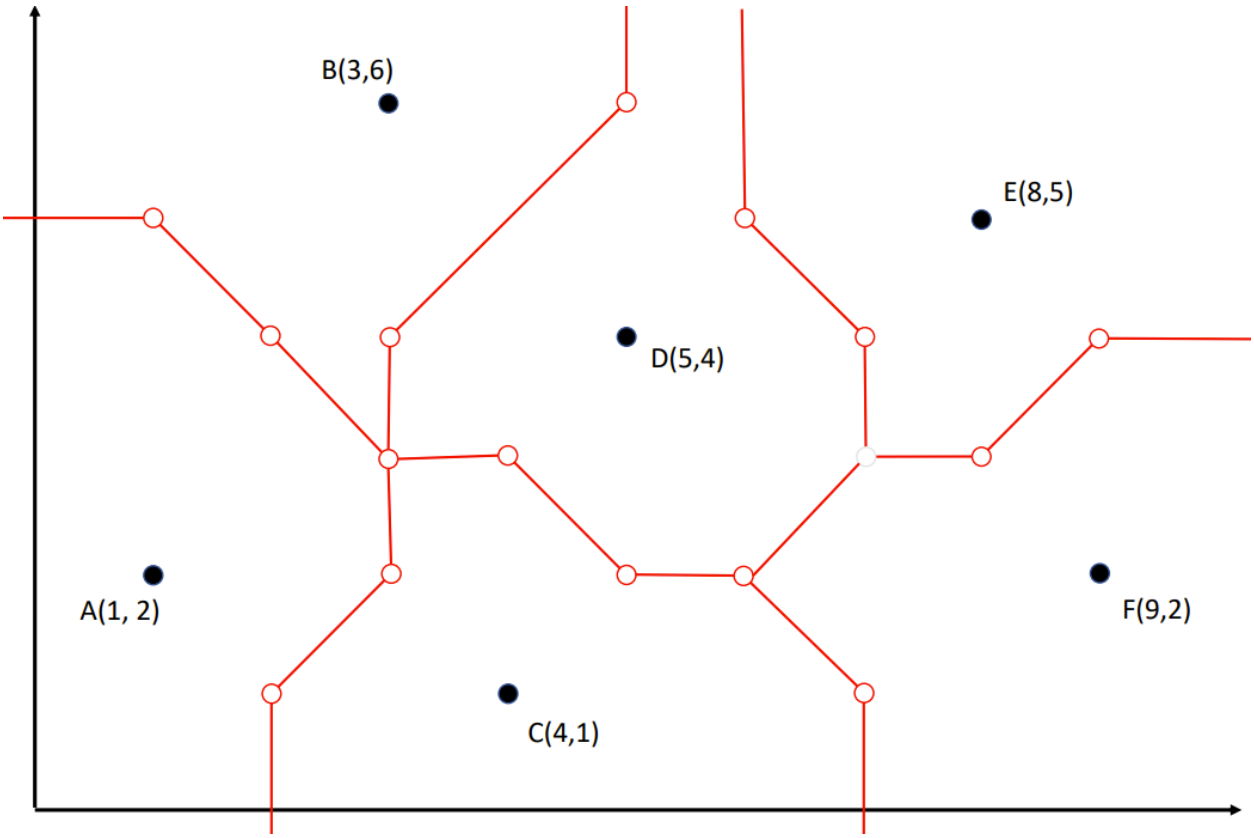


Figure 3: L_1 metric Voronoi diagram for a set of 6 points using the L_1-distance formula above. Fig. 7 is a detailed version that extends the bisectors of pairs of points. See Fig. 4 for the L_∞ Voronoi diagram for these points, and Fig. 5 for the two overlayed.

(b) **Voronoi Diagram in L_∞ Metric**

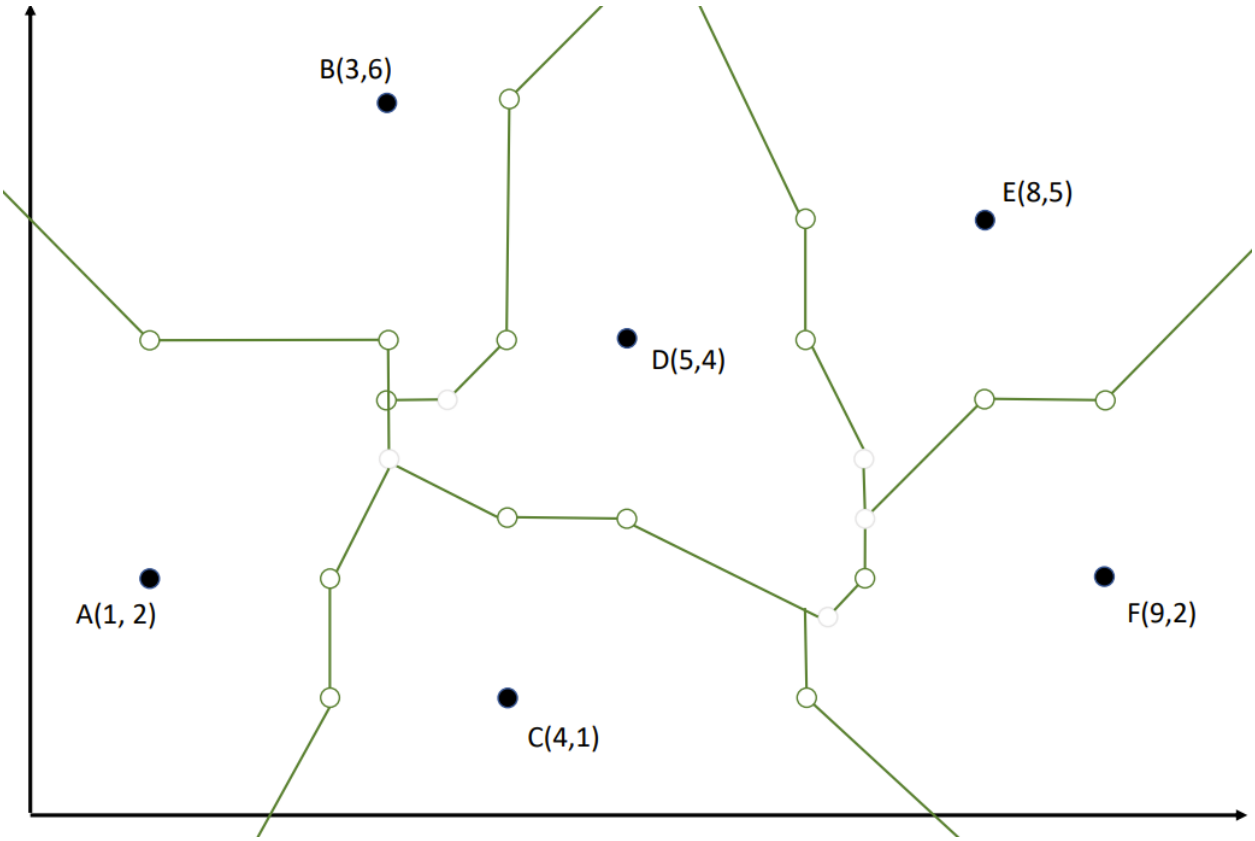


Figure 4: L_∞ metric Voronoi diagram for the same set of 6 points above using the L_∞-distance formula. Fig. 8 is a detailed version that extends the bisectors of pairs of points. See Fig. 3 for the L_1 Voronoi diagram for these points, and Fig. 5 for the two overlayed.

(c) **Voronoi Diagrams of the L_1 and L_∞ Metric**

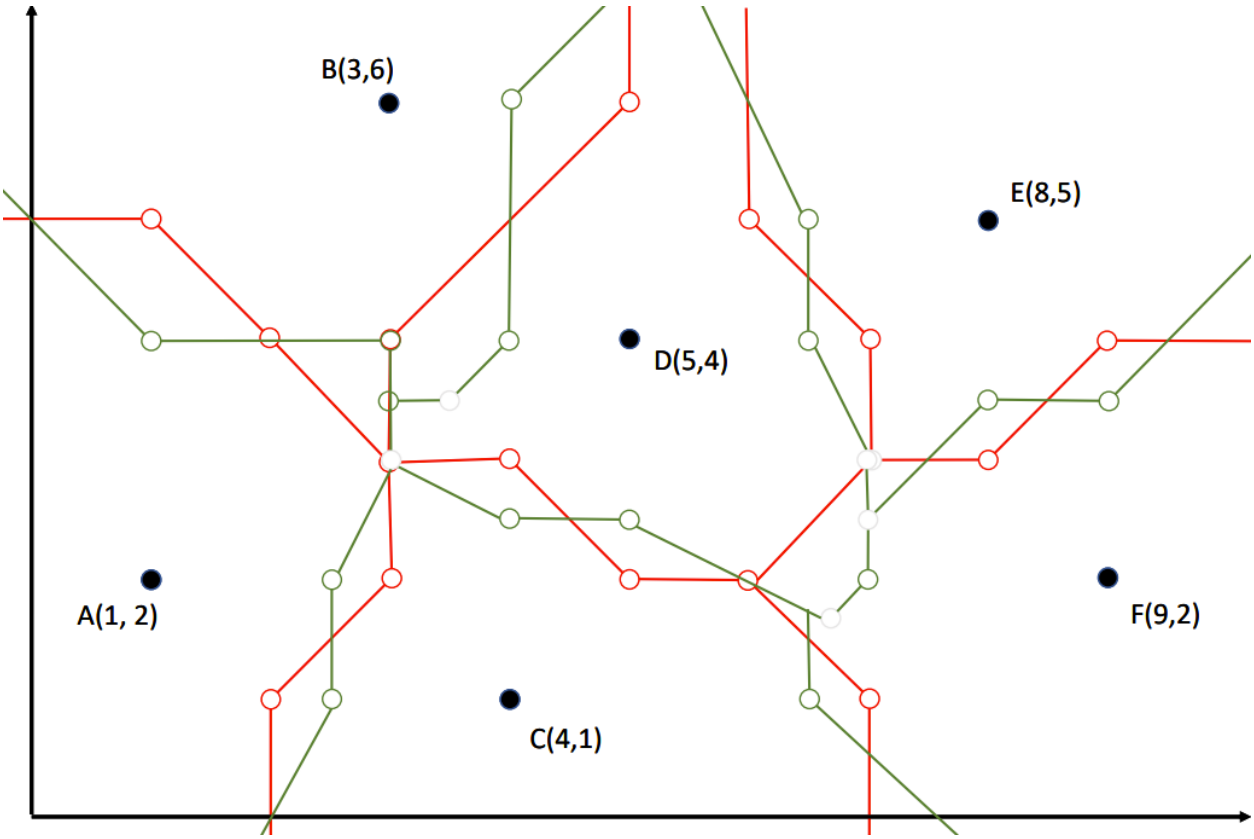


Figure 5: This overlay of the L_1 and L_∞ Voronoi diagrams shows the same criss-cross patterns seen and described above in Fig. 1a - 2d, when we overlay the $B_1(q_i, q_j)$ and $B_\infty(q_i, q_j)$ for two given points in the plane. I expect the corresponding L_2 Voronoi diagram to be contained in the bounds of these two diagrams. See. Fig. 6, where the L_2 Voronoi diagram (Fig. 9) is overlayed.

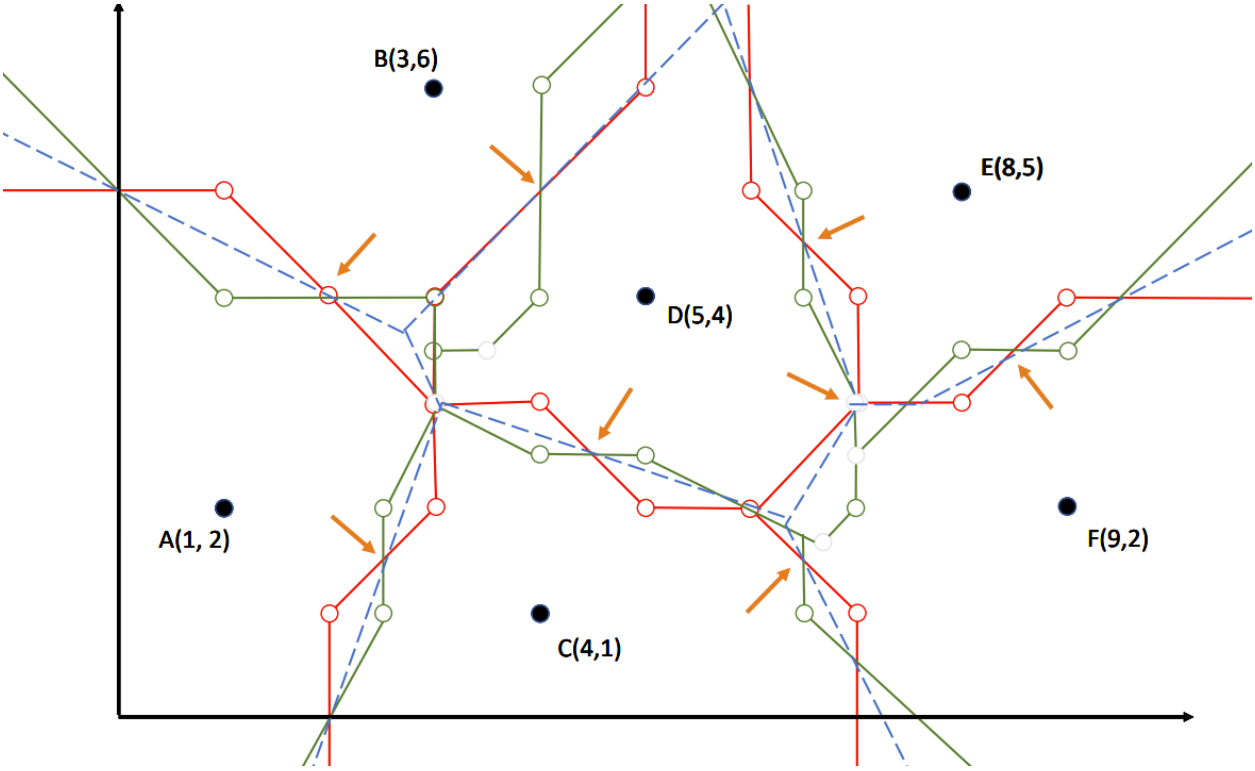


Figure 6: An overlay of the L_1, L_2, L_∞ Voronoi diagrams. We see the blue dashed edges of the L_2 diagram within the bounds of the edges of the L_1 and L_∞ diagrams. The orange arrows point to where three edges of the three diagrams intersect at the mid-point of a segment $\overline{q_i q_j}$. See Fig. 9 for just the L_2 Voronoi diagram.

3 Points in a Diagram

a. We assume general position of S and that we've already computed the Voronoi Diagram for S, $\mathcal{VD}(S)$, which takes $O(n \log n)$ time, and that the edges, vertices, and faces of $\mathcal{VD}(S)$ are stored in a DCEL. We can do point location on q in $O(\log n)$ time to detect which Voronoi cell it is in. Since this is a Voronoi Diagram, whatever face query point q is in, the Voronoi site, point $p \in S$, to which the current face/cell belongs, will be the closest point to q, because q is in a region in which it is closer to $p \in S$ than it is to any other point $r \in S$, $r \neq p$. Therefore, using $q(x_q, y_q)$ and closest point $p(x_p, y_p)$, we can calculate in consant time the largest radius circle C_q can have without including p using the distance formula, $r = \sqrt{|x_q - x_p|^2 + |y_q - y_p|^2 - \epsilon}$; finally, we can report the largest circle centered at q for the problem, $C_q = (x - x_q)^2 + (y - y_q)^2 = r^2$. Space complexity is linear because we are storing the diagram in a DCEL. Overall runtime if we don't have to compute the $\mathcal{VD}(S)$ is $O(\log n)$ for one query point in order to do point location. For m query points, the total runtime is $O(m \log n)$.

b. Again, assume general position and that both $\mathcal{CH}(S)$ and $\mathcal{VD}(S)$ are computed (each takes $O(n \log n)$ time), overlayed, and their edges added to the same DCEL. We remove the partial edges $\mathcal{VD}(S)$ that are outside $\mathcal{CH}(S)$, since the path must be contained within the hull; we can walk the edges of the $\mathcal{CH}(S)$ to find the intersection points with edges of $\mathcal{VD}(S)$ and remove those beyond the hull. This takes $O(n \log n)$ time.

   To find a path from p to q, the "safest" possible path would be some route that is as far from any points in S as possible. This describes the edges of the $\mathcal{VD}(S)$, because points of the edges are halfway between two given sites, and are therefore the furthest possible points from two points in S. This does not necessarily mean all edges are "safe" to walk, i.e. $> k$ distance from a site[6].

   So then, for each point $t \in S$, we create a circle centered at t with radius k. For every partial edge that intersects the circles and are contained in the circle, remove those partial edges from the DCEL, as well. This can be done in $O(n \log n)$ time. We are now left with a DCEL, with possibly disjointed subgraphs, but with edges and vertices that are $> k$ distance away from a site in S.

   From both p and q, each creates a list of the nearest Voronoi edge or vertex to them, which, as stated, will be $> k$ distance from a Voronoi site. Then, using depth-first search, we calculate a path from p to q (or vice versa) (DFS is linear runtime). Report the result of the first successful DFS if a path is found; otherwise, a path that met the conditions did not exist and we report no path.

   Finally, overall space complexity is linear for the DCEL.

[6] Diane discussed with us possible ways to find a path that met the k-tolerance condition, including walking on the Voronoi edges and how to tackle it from the Piano Mover's problem perspective.

4 Project

5 Appendix

The below figures are the detailed versions of previous figures in problem 2. The faded out portion of the diagrams are the bisectors dividing regions of different pairs of points that are not part of the final voronoi diagrams. The faded colored letters indicate which half plane belonged to which region, and the faded grey letters are those half planes of the individual bisectors not in the final diagrams.

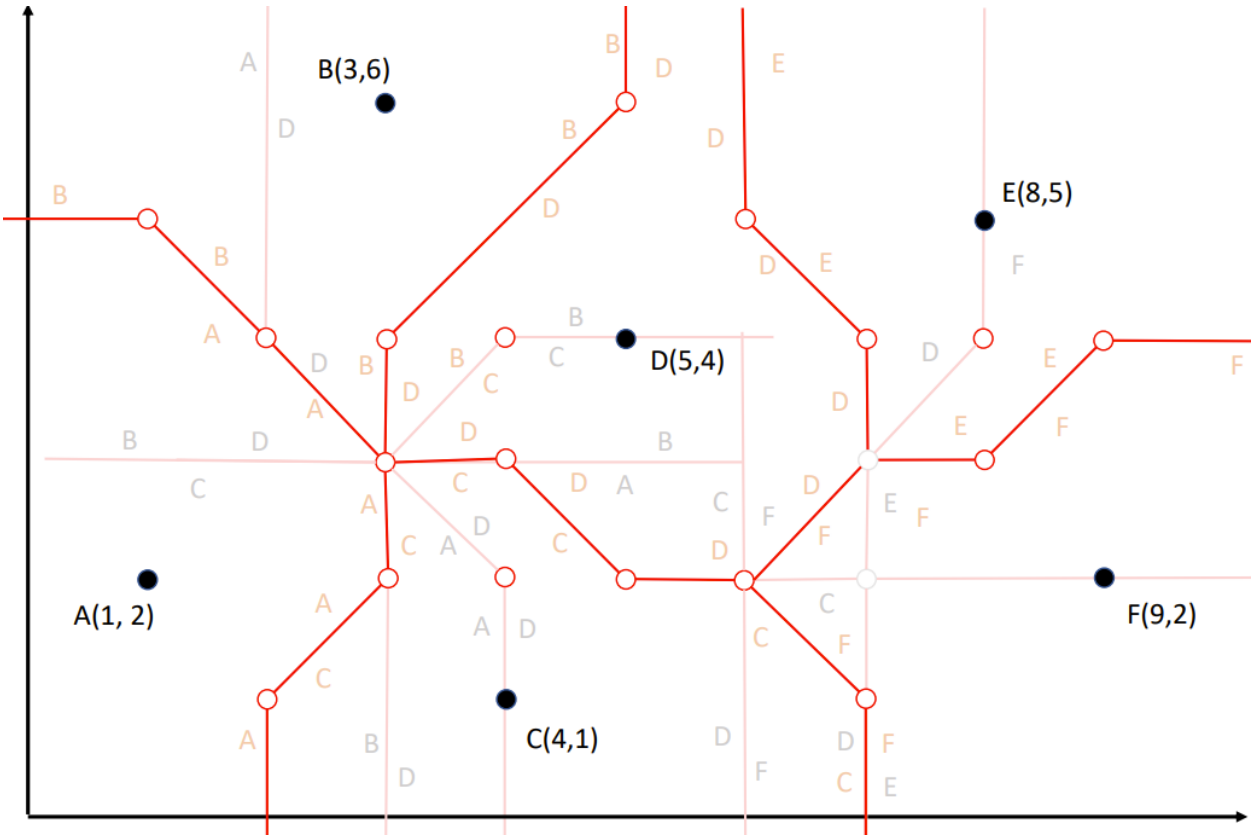


Figure 7: Detailed version of L_1 metric Voronoi Diagram of Fig. 3

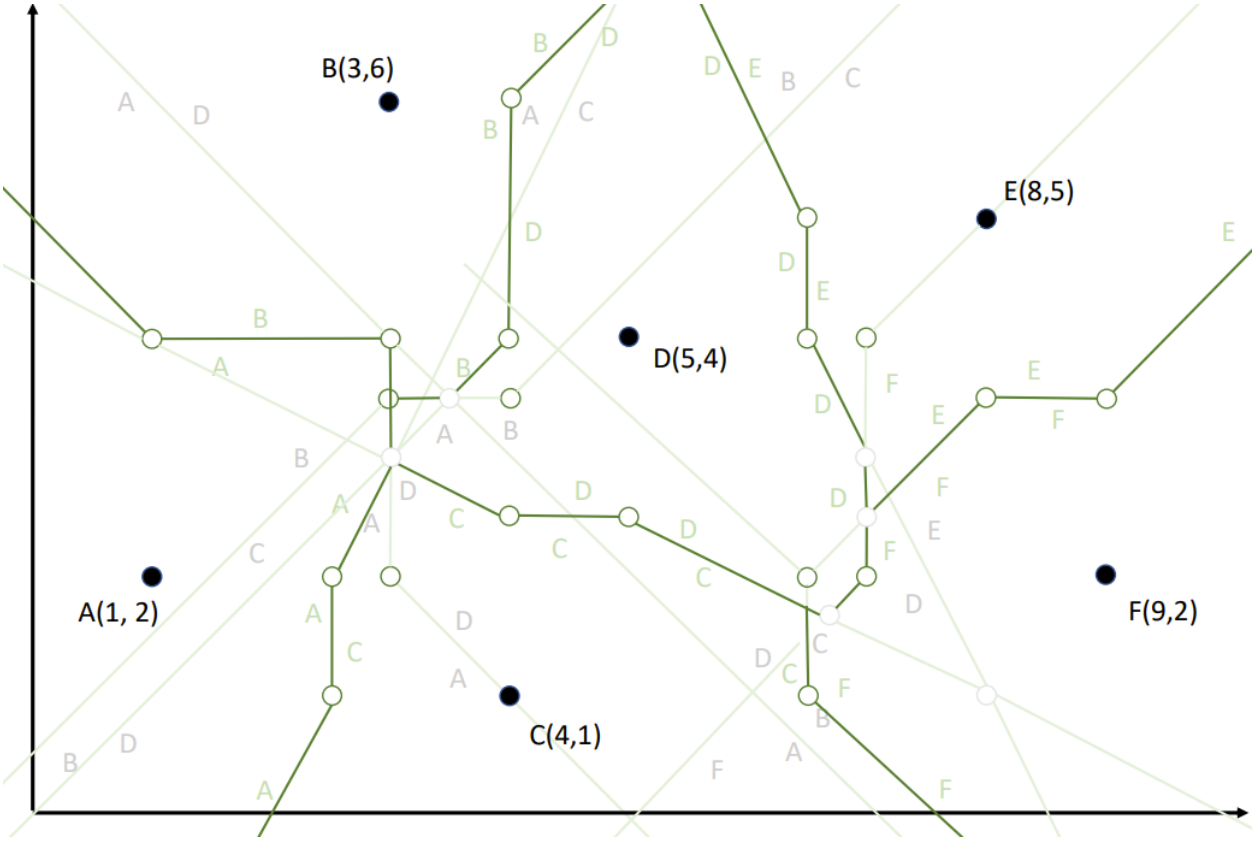


Figure 8: Detailed version of L_∞ metric Voronoi Diagram of Fig. 4

References

[1] Jake and Diane's office hours, classmates: Stephanie, Alex, Anju with homework problem discussions.

[2] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. 2008. Computational Geometry: Algorithms and Applications (3rd ed. ed.). Springer-Verlag TELOS, Santa Clara, CA, USA.

[3] D.T Lee and C.K. Wong, "Voronoi Diagrams in L_1 and (L_∞) Metrics with 2-Dimentional Storage Applications". SIAM J Comput. Vol. 9, No. 1, 200-212. 1980.

[4] Yunzhi Shi, "Sarah's Interactive Voronoi Diagram". 2017.

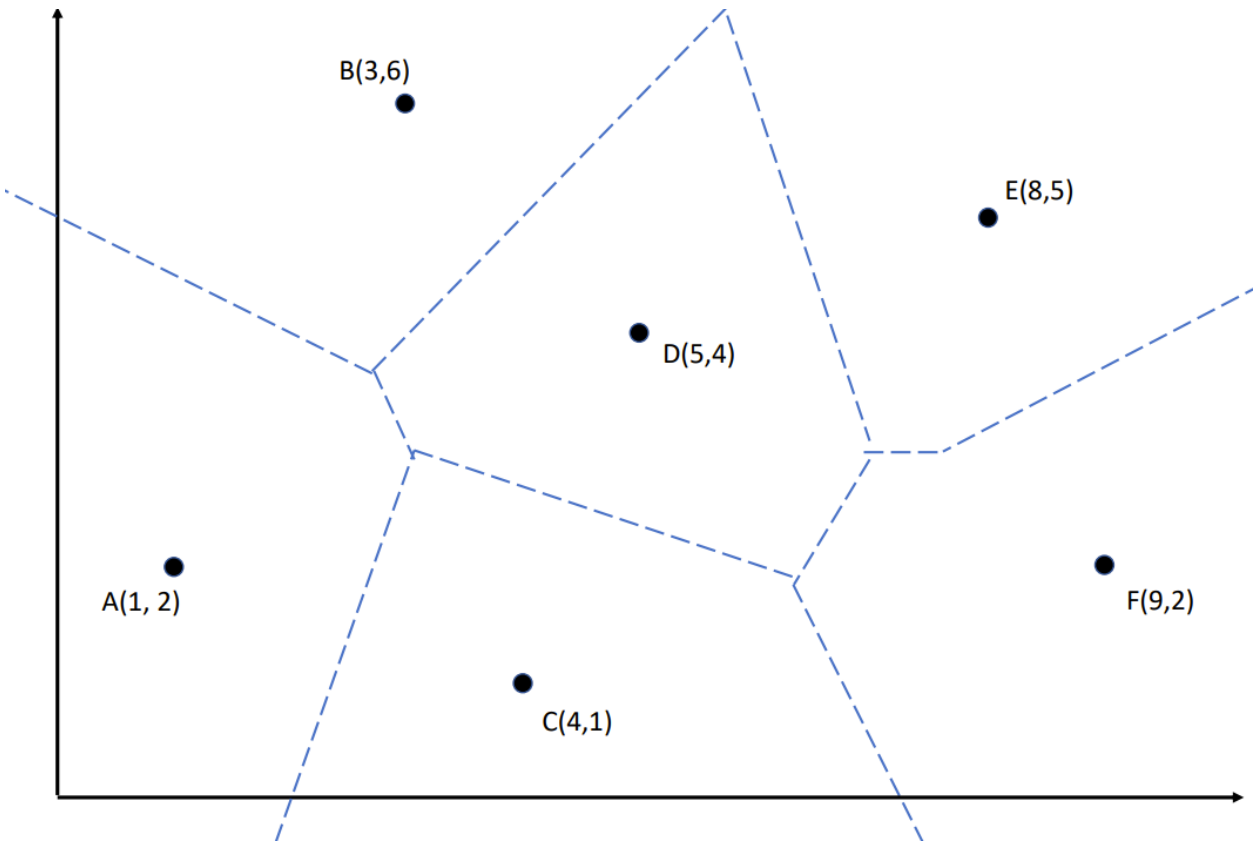


Figure 9: The L_2 Voronoi diagram for the 6 points examined in problem 2.