CSC 565: Graph Theory **North Carolina State University**
Fall 2019 **Computer Science**

Lecture 21: November 4, 2019

Lecturer: Don Sheehy <drsheehy@ncsu.edu> *Scribe: Ellis Ackerman, Roshani Narasimhan*

Play with planarity Game called "Planarity" gives a scrambled planar graph, where you can move around the vertices to find an embedding of the graph (no crossings). http://planarity.net

Tutte's Algorithm

Goal Find a linear embedding of a planar graph. *Prerequisites for Tutte's algorithm* We need the graph to be a 3-connected graph.
Why do we need a 3-connected graph for Tutte's Algorithm? Let G be a 3-connected and planar graph. We need to fix an outer-face and we can do this since the faces are the non-separating induced cycles of G. Thus we know the edges that will bound a face. In the original version of Tutte's Algorithm, there are two steps:

- Fix the outer face - the vertices are placed equally-spaced on the circumference of a circle.
- Each of the other vertices is placed in the interior, at the centroid of its neighbors.

Does Tutte's Algorithm terminate eventually? Say we put a vertex in the interior, at the centroid of its neighbours, then this system will have a new centroid and so on. This is equivalent to letting a spring system enter an equilibrium state, assuming all the springs have the same spring constant and the forces exerted by the neighbors, on the vertex at the centroid cancel out each other. Thus adding new vertices inside would at some point of time bring the system in equilibrium, thus a solution exists and the algorithm converges. The condition stated above imposes two linear constraints for each vertex in the interior, as is evident from the Hooke's law - one in the x direction and one in the y direction.

Solutions obtained from the algorithm We formalize this notion using block-matrices.

$$LP = F0,$$

where L is the Laplacian of G and P are the positions of all vertices. F are the forces on the vertices of the outer face, and 0 corresponds to the fact that the center of the system, which consists of internal vertices, is at equilibrium. This matrix system tells us that,

$$L_1 B B^T L_2 P_{\text{out}} P_{\text{in}} = F0.$$

Here L_1 and L_2 are the parts of the Laplacian which correspond to the Laplacians of the outer and inner faces, respectively. By matrix-block multiplication, we have,

$$L_2 P_{\text{in}} = -B^T P_{\text{out}}.$$

By inverting L_2, we would be able to solve this system for P_{in}, which would result in the locations of inner vertices, based on the positions of the outer vertices.

How many solutions are possible? Maxwell observed that sometimes, there is not just one 'reciprocal diagram' but a family of 'reciprocal diagrams' (which obtained from the solution of the rows/columns of the matrix which are not linearly dependent on each other). If L_2 is not invertible, there could be many solutions to

this system. Recall that by the *Matrix-Tree Theorem* (which states that the determinant of the Laplacian with first row and first column removed is the number of spanning trees of G), L_2 is indeed invertible (has a positive determinant). Thus, we can solve for for the locations of the interior vertices,

$$P_{\text{in}} = -L_2^{-1} B^T P_{\text{out}}.$$

What could go wrong if G was not 3-connected? Let us consider the example of a K_3 subgraph in G. If we want to add the third vertex, then it should be placed at the then the centroid for the two vertices. The centroid would lie on the edge connecting the two vertices, thus adding the third vertex would result in overlapping of edges. The figure below illustrates this notion.

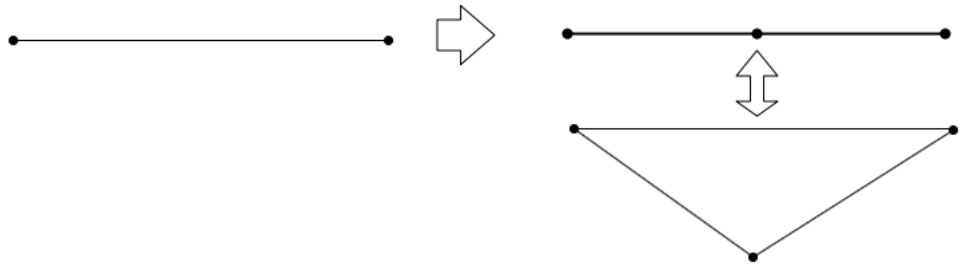

Figure 1: Adding a vertex to the interior of two existing vertices results in a line with overlapping edges instead of forming a triangle.

So how do we prove that this algorithm is correct and is an embedding of the graph, and not just a drawing? If we return to thinking of the edges as springs in equilibrium, then we are essentially thinking of a net as we stretch the outer-face of the graph. If we try to pull away of crossings, we create monotone paths (edges increasing in one direction) in the drawing. We note that faces must be convex. A polygon is not convex if it has a **reflex vertex** or a **winding**. In both these cases, we can find a line in the plane that separates the four vertices into two sets, thereby creating monotone paths between the interleaved vertices. This indeed creates a K_4 topological minor and we know that a graph containing K_4 topological minor is not planar.

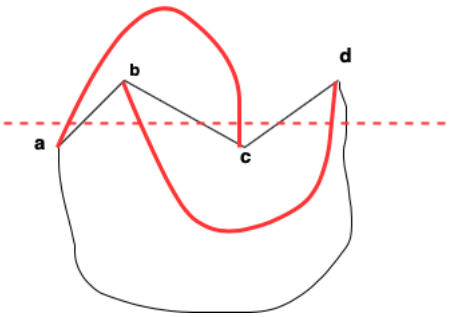

Figure 2: Note the monotone paths marked in red from a to c and b to d, when the graph contains reflex vertices. The graph shown above is not planar.

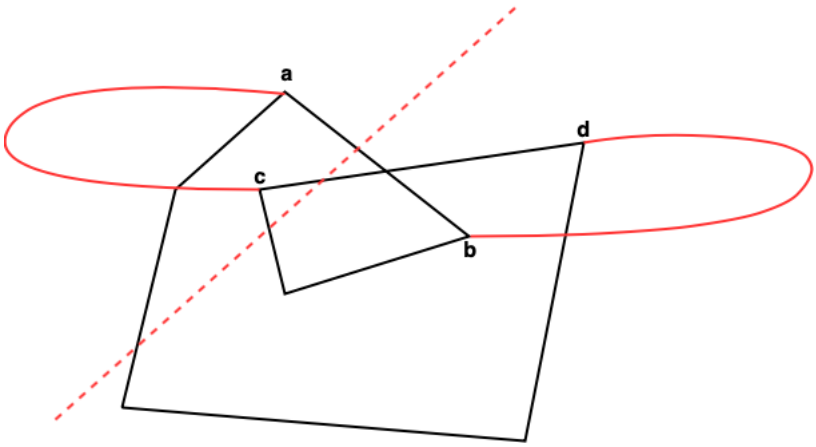

Figure 3: Note the monotone paths marked in red from a to c and b to d, when the graph contains a winding. The graph shown above is not planar.

The goal now is to show that as we glue various faces (which we have asserted to be convex polygons) of G together, they do not overlap. We do this by showing that locally, there are no crossings.

Definition: Ply is the number of polygons at any point. This is a function on the plane. For each point in the plane, it returns the number of polygons "stacked" at that point. Thus, our goal becomes checking at

the alleged embedding by Tutte's Algorithm yields a drawing in which the ply is one inside the drawing and zero out outside of it. This will be good to show that there are no crossings.

We must check the following:

- For every edge in G, **faces are on opposite sides of the shared edge**. If the faces are on the same side of an edge, it means there is an overlap of faces. This implies that the ply is greater than 1. We can think of this as a sheet of paper being folded into two. The crease signifies the shared edge and surfaces of the paper denote the ply (which is equal to 2).

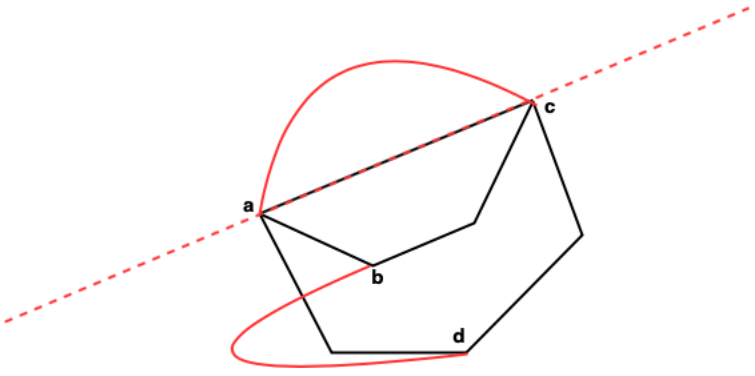

Figure 4: This figure shows that though there are no explicit crossings in the drawing of the graph, there is a shared edge, which makes the ply equal to two in certain portion of the drawing. Note the monotone paths marked in red from a to c and b to d, when the graph contains a winding. Tutte's algorithm would not yield such a drawing.

- We must also check that there is **no winding around a vertex**, when we are gluing the polygons together. We ensure this by checking that the sum of the angles of the edges coming out of the vertex are less than 2π. If there is a winding around a particular vertex, it means that we circle about more than once, which is possible only when there is a crossing among one or more edges. This means that we can find monotone paths between two points on different winding (which will lead to a K_4 topological minor, which has 4 vertices on same face). Thus the existence of monotone paths imply existence of sub-graphs that cannot exist in a planar graph.

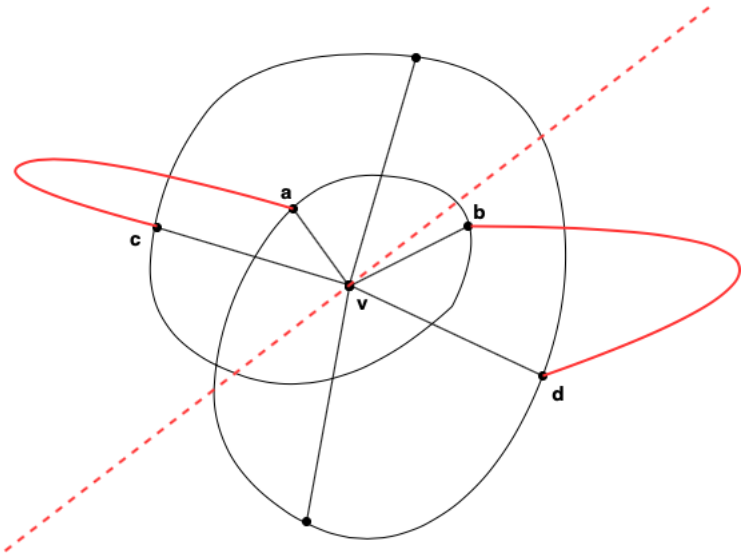

Figure 5: Around the vertex v, due to the winding we can find monotone paths between a and c ; b and d. Ply within the inner winding is 2.

Locally "good" to globally "good" *If G satisfies conditions locally, does it ensure that G has a planar embedding?*

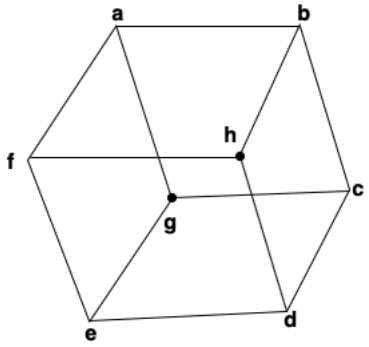

Figure 6: Drawing for a cube which does not comply with Tutte's algorithm. This drawing has the inner vertices g and h placed at the centroid of its neighbouring vertices. However the outer hexagon formed by a,b,c,d,e and f does not form a face.

Let us consider the drawing of the cube provided above. We note that:

- The internal vertices are placed at the centroid of its neighbours. (The forces of exerted by the outer vertices are balanced at the inner vertices).
- At the internal vertices, there is no winding, which means that the angles made by the edges at these vertices are equal to 2π.
- Each edge has faces to the opposite sides.
- Faces look to be convex, however the outer face is not actually a face of the cube! The outer hexagon is not a non-induced separating cycle. Recall that it is important that the outer-face of the embedding of the graph, must be a non-separating induced cycle (implied by the fact that the graph is 3-connected).
- We observe crossings at edges as well.
- Ply is two everywhere inside the embedding.

Thus even though the local conditions for the vertices and the edges are satisfied, we note that the above drawing of a cube is not a planar embedding.

Interesting facts about ply:

- For a planar graph, ply does not change at the edges - as faces are on the opposite sides of the edge.
- Ply will be constant at a vertex, as the number of polygons which meet at the vertex must be constant.
- For a planar graph, the ply in the inner faces of the graph is 1 and the ply outside of the outer face is 0.

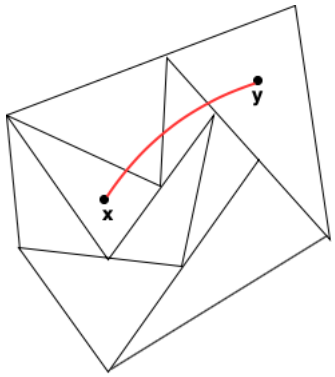

Figure 7: Tracking change in the ply at two points x and y within the graph.

We now consider the ply and the drawing of the graph above. Let us assume that for two points, x and y in the graph, $\text{ply}(x) \neq \text{ply}(y)$. If something is not true globally, then it must be not true somewhere locally. Thus, if we "walk" from x to y, then at some point the ply must change, i.e. a boundary is crossed. But we have shown that locally the embedding is well-behaved. The ply is constant on the interior on edges and vertices. Thus the ply can only change when the outer-face is crossed.

Lingering Issues and Problem Variations

Mean Graphs
Consider the graph $K_3 \times P_{n/3}$. The center section gets smaller and smaller. These sort of situations can be hard for computers to represent and we can run into numerical precision errors.

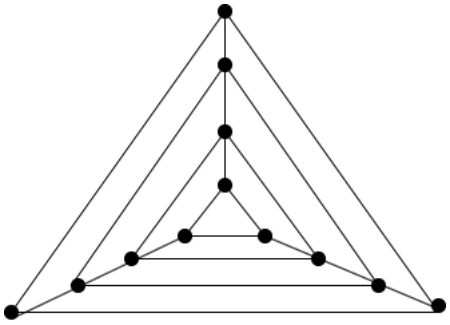

Figure 8: $K_3 \times P_{n/3}$ as a computationally difficult problem.

Low-Precision Embdeddings
Can we embed a graph into the plane such that the vertices are on grid points. The original embedding problem let us place vertices at any point in the real plane, but this variation now constricts us to the integers, changing the allowed precision. Many problems become harder under this restriction, including graph embeddings.

Spectral Graph Embedding
Instead of solving the linear system, we consider the eigenvalues of the Laplacian for each direction,

$$Lx = \lambda x \text{ and } Ly = \lambda y.$$

We are looking for the force vectors directed towards or away from the origin. While this method does not give us force equilibrium as Tutte's Algorithm does, it does give us an organized way to embed the graph. This method leads to the notion of "scaling". Note that L has $n-1$ eigenvectors and it is not always clear which is "best." This method was first discussed in applications of computational chemistry.