Christian Zinck

DMP Planarity Decision

For my project I chose to implement the DMP planarity decision algorithm by path addition, from Demoucron, Malgrange, and Pertuiset.[1] I based my implementation off pseudocode found online.[2] The first step was to eliminate some easy cases, as the source suggests. Self and multiple edges are removed because their presence does not affect the planarity of the graph. Only the largest biconnected component is considered because only a biconnected component, (a collection of nodes that have at least two edges), has the potential to be nonplanar. Additionally, all graphs with less than 5 vertices are planar, all graphs with $E > 3V - 6$ are nonplanar, and all acylic graphs are planar.

The algorithm starts by finding a single cycle, which is a graph with two faces. This cycle will be chordless because there are no vertices connected by an edge that is not part of the cycle. Paths will be added iteratively to the cycle, thereby incrementing the number of faces and chordless cycles by one. This is repeated until all faces have been successfully added, and the graph is deemed planar, or a path is found to have no planar connection, and the graph is deemed nonplanar. The difficulty in the implementation comes from managing the regions of the graph, path selection, and deciding how many possible embeddings a path can have.

To manage the regions, I chose to maintain a list of cycles known as a basis. A basis of any graph is a complete collection of chordless cycles in the graph, which for a biconnected graph includes all vertices and edges. Every time a path is added to the graph, it will split a region in two, creating two new regions, both boundaries of which contain the path edge list. The original region is removed from the basis as it is no longer chordless.

To select a path, I chose to brute force test combinations of endpoints within the graph until one was found. Each node in the graph can be checked against all other nodes to see if a path exists between the two, that doesn't share a vertex or edge with the current graph. The alternative would have been to search for path possibilities within the basis. Since the basis size grows faster than the number of nodes involved and the maximum size of the basis is much larger than the maximum number of nodes, the larger the problem becomes the less efficient it is to search for paths using cycles from the basis.

To determine the number of possible embeddings for a path, I used the idea that for each region the path's endpoints are members of, a path can be embedded in that region. This means if two nodes share a path that has not already been embedded, that path can be embedded in any one of their common regions. If a path has 0 possible embeddings, the graph is nonplanar because there exists a path that must be embedded, but there is no way to do so in the plane. If a path has one possible embedding, it must be embedded in that region, because there is no other region it can be embedded in. If a path has more than 1 possible embedding, it cannot be embedded until it is certain that no path with one possible embedding exists. This is so a path with multiple possible embeddings doesn't block a path with a single embedding that the algorithm hasn't checked yet.

I have tested the implementation on the graphs of known planarity in the 'tests' directory and it decides correctly. It has a worst case time complexity of $O(FV^2)$ when all nodes have a path with more than 1 possible embedding, however this is the case in very sparse graphs with few faces. The and a best case time complexity of O(F) when all paths have exactly 1 possible embedding. This occurs in a triangulated planar graph as no new path can be formed inside a triangle, thereby eliminating all existing possible embeddings except for the outer face. The algorithm has a slower average time complexity than other competing algorithm categories, such as vertex and edge addition, which both have linear time algorithms from Even and Tarjan and Boyer and Myrvold respectively.[3, 4] That said, for triangulated graphs, it can run faster than its competitors due to the fact that the number of faces in a graph is always less than the number of edges or number of vertices.

References

1. Graphes planaires: reconnaissance et construction des représentations planaires topologiques - DEMOUCRON, MALGRANGE, et al. - 1965
2. A Planarity Algorithm. (n.d.). Retrieved August 21, 2018, from http://www.mathcs.emory.edu/~rg/book/chap6.pdf
3. Network Flow and Testing Graph Connectivity - EVEN and TARJAN - 1975
4. On the Cutting Edge: Simplified O(n) Planarity by Edge Addition - BOYER and MYRVOLD - 2004