**Q2**

The algorithm running sequentially works as follows: it looks through all edges and finds direct common pairs of points between the two points on such edges (essentially a path of size 2). It checks if there is exactly 1 edge that an edge created by such pairs would intersect, and that edge being the edge in question. The angle between the two current points and the edges that connect them to the edge in question must also have an added angle greater than PI radians. If so, this edge is “flippeable” and will be replaced by the edge connected by the pair of points. When this occurs, the algorithm must relook at all edges of the graph that now includes the new edge and excludes the old one. If no flip is to be done, then the algorithm checks out a new edge.

The parallel implementation is quite similar, however threads are delegated different edges based on when they attempt to look at them. If an edge is locked, it moves onto the next open edge and examines it. Because multiple threads are interacting with similar edges (as they must consider attached edges as well that may be looked at as a main edge in another thread), some sort of synchronization is required. The flipping portion is the most critical part of the algorithm, thus locking this actions seems necessary. Whenever an edge is deemed flappable, it first checks if an edge associated to it had previously been removed while being locked out. If it had been removed, it moves on to look at the next edge. If all associated edges still exist, it removes the edge and is set to relook at the entire graph.

This graph displays the speedup from 1 thread to up to 4 threads. It seems as though