Origami algorithm

This is the conceptual algorithm for my origami project, I have no idea how to code some parts of this.

A tree is a list of leaf and river nodes, the connections between them and any spaces(between to parallel nodes) where cuts are made.

*Let N=1*

*Place leaf i at point Pi such that*

*To find the long leg- draw a path along the gridlines, from I to j, which does not cross any edges. add up the lengths of the horizontal and vertical displacements separately. All displacements are considered positive, even if they backtrack. The longer of these two numbers is the long leg.*

*To find the tree distance from I to j, start at I, and find the shortest path along the tree to j. this distance is the tree distance. For this calculation, pretend that there is no distance between nodes which have a cut between them, unless the cut is between I and j.*

*If there is a cut between leaf I and leaf j,*

*place leaf i at point Pi as well.*

*Make a cut either horizontally or vertically (whichever is best), from point Pi – tree distance(in the appropriate direction) to Pi +tree distance.*

*Long leg (i , j) ≥ N\* tree distance(i,j)*

*For all I,J*

*If this is impossible, N++*

The long leg calculation ensures that the allocated boxes don’t overlap, but allows node to be closer near cuts. The tree distance is similar to tree maker’s algorithm, and I have used N ( the number of squares the paper is divided into) instead of the scaling factor lambda.

I’m not sure how to implement this as code, but I think there should be a function that takes a paper object, and returns an optimized version. That new one could be displayed on screen, or be later used to generate crease patterns.