

# Growing leaves

Graham Jones, [www.indriid.com](http://www.indriid.com), August 2022

## 1 Introduction

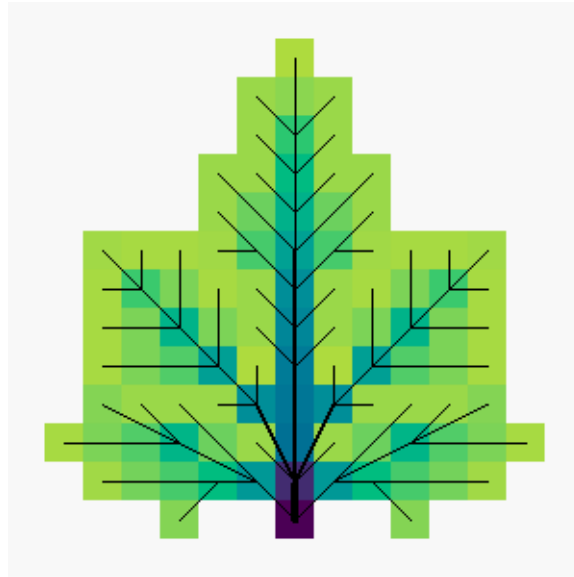


Figure 1: A mathematical leaf.

A leaf with a larger area collects more light, but requires a bigger network of veins to deliver water and nutrients to every part. A ‘good’ leaf maximizes area while minimizing transport costs. This is a simplified account of a paper by Qinglan Xia which describes a model in which these conflicting aims are balanced against each other:

*The Formation of a Tree Leaf*

<https://pdodds.w3.uvm.edu/research/papers/others/2007/xia2007a.pdf>

The leaf is made of square cells on a grid, together with a network of veins which supplies each cell from the base of the leaf.

The network of veins is a sort of mini-tree with its own ‘root’ at the base of the leaf. In graph theory terms, it is rooted tree (a tree with one distinguished vertex or node). There is a unique directed path from the root to every other node. We can talk about ‘the’ edge of a node, since only one edge leads to a node. Each node sits at the centre of a cell. I’ll use the words ‘cell’ and ‘node’ interchangeably.

## 2 The algorithm

Here's the basic algorithm.

1. Start with a tree,  $G$ , containing one node, the root.  
REPEAT:
  2. Find the squares just outside  $G$  that can be added with the smallest possible increment in  $F(G)$ . If the increment in  $F(G)$  is less than a threshold, add them as new cells, otherwise STOP.
  3. Optimize  $G$ , by repeatedly visiting every node  $n$ , and trying to find a better parent for it.

The tree  $G$  is a list of nodes.  $F(G)$  is the total transport cost of  $G$ .

In step 2, 'just outside' means being an 8-neighbor of a node in  $G$ . The increment in  $F(G)$  can be found by making a new tree  $G_2$  with the candidate added, and calculating  $F(G_2) - F(G)$ . The threshold is called  $\epsilon$  and it determines how big the leaf grows.

In step 3, the candidate new parents for  $n$  are those which are either an 8-neighbor of  $n$  or an 8-neighbor of  $n$ 's current parent, but are not  $n$  itself, nor a descendant of  $n$ . For each candidate, make a new tree  $G_2$  with the change in parent. If  $F(G_2) < F(G)$ , replace  $G$  with  $G_2$ . Since changing  $G$  in one place may make improvements elsewhere possible, the process is repeated until no node can be improved.

This is a simple but inefficient way to implement the algorithm. There is pseudo-code in section 6.

Along with  $\epsilon$ , there are two parameters  $\alpha$  and  $\beta$  which determine the shape and 'style' of the leaf.

## 3 The total transport cost $F(G)$

### The nodes

Each node  $n$  has the following information which together define the topology (branching pattern) and geometry of the tree.

- The parent node  $\text{parent}(n)$ . The root does not have a parent.
- A list  $\text{children}(n)$  of child nodes, which may be empty.

- Integers  $x(n)$  and  $y(n)$  giving the 2D coordinates of  $n$ . The root is at  $(0,0)$  and is assumed to be supplied from the plant by an invisible edge coming from  $(0,-1)$ .

Each node  $n$  also has values which are used to calculate the transport cost.

- The length  $d(n)$  of the edge of  $n$ .
- Weight  $w(n)$ , which is the total amount of fluid required at  $n$ , for  $n$  itself and all its descendants.
- The local ‘bending cost’  $b(n)$ .
- The total bending cost  $m(n)$  of a node  $n$  is the product of  $b()$  values along the path leading from the root to  $n$ .
- The local transport cost  $f(n)$  which is equal to  $m(n)^\beta w(n)^\alpha d(n)$ .

The total transport cost  $F(G)$  is the sum of all the  $f(n)$ .

### The weights

Each cell needs the same amount of fluid, which is assumed to be 1. This means  $w(n)$  is equal to the number of cells that are descendants of  $n$ , plus 1. It is proportional to the cross-sectional area of a vein. See Fig 2.

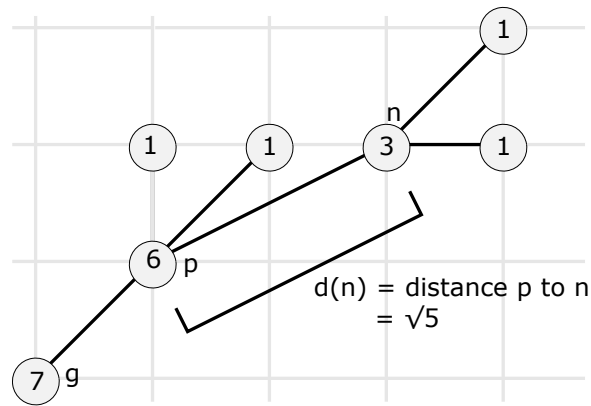


Figure 2: A part of a tree with weights and distances. Node  $n$  has parent  $p$  and grandparent  $g$ . The weight of  $n$  and other nodes are in the circles. The distance of  $n$  to  $p$  is  $d(n) = \sqrt{2^2 + 1^2} = \sqrt{5}$ .

## The bending costs

For most nodes, the local bending cost  $b(n)$  is derived from the angle between the edge of  $\text{parent}(n)$  and the edge of  $n$ . This requires  $n$  to have a parent and a grandparent  $\text{parent}(\text{parent}(n))$ , so it doesn't work near the root. If  $\text{parent}(n)$  is the root, the invisible edge coming from  $(0,-1)$  mentioned above is used. For the root itself, we set  $b(\text{root}) = 1$ .

If the angle  $\theta$  of the bend is less than a right angle,

$$b(n) = 1 / \cos \theta.$$

Otherwise  $b(n) = \infty$ , which means that 'backwards' bends never appear in the tree. See Fig 3.

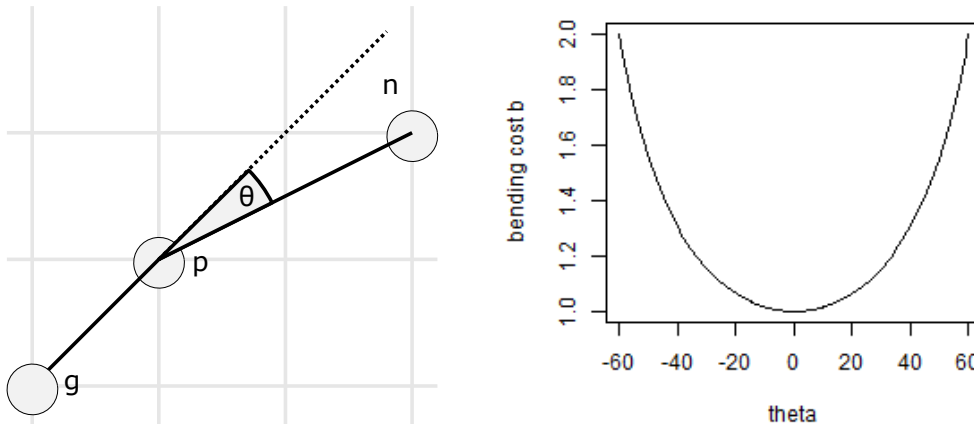


Figure 3: on the left, node  $n$  and the bend angle  $\theta$ . Here,  $\cos \theta = (2 \times 1 + 1 \times 1) / (\sqrt{5}\sqrt{2}) = 3/\sqrt{10}$  and  $b(n) = \sqrt{10}/3 = 1.054$ . On the right, a graph of  $b(n)$ .

Starting with  $m(\text{root}) = 1$ , other nodes can be given  $m()$  values by repeatedly applying  $m(n) = b(n)m(\text{parent}(n))$ .

### 3.1 The parameters $\alpha$ and $\beta$

Assume the veins have the same shaped cross-section, for example circular. If the cost of a vein was proportional to the circumference, it would be proportional to  $w(n)^{1/2}$ . On the other hand, if there was no saving in using one big vein instead of several little ones, the cost would be  $w(n)$ . The formula  $w(n)^\alpha$  as  $\alpha$  ranges from 0.5 to 1 interpolates between these possibilities. Values of  $\alpha$  outside this range seem harder to justify physically.

Bends presumably incur some cost, requiring extra material for rigidity as well as containing the fluid, but I don't know any explanation for Xia's particular formula for  $b(n)$ .

### 3.2 Does it stop growing?

When some cells are added, the veins are optimized, and this may mean that the next cells can be added at a smaller cost than the previous ones. It's not obvious that the cost will ever reach  $\epsilon$  and the leaf might keep growing forever. Xia proved a theorem that ensures that it does stop when  $\alpha > 1/2$ .

My proof is based on Xia's, but I hope simpler. For convenience I assume that  $\alpha \leq 1$  as well. We start with a lemma and a definition.

**Lemma 1** *Suppose  $x \geq 1$ . The function  $f(x) = (x + 1)^\eta - x^\eta$ , is decreasing if  $\eta \leq 1$  and increases without limit if  $\eta > 1$ .*

**Proof** Exercise!

Let  $\text{path}(n)$  be the set of nodes in the path from root to  $n$ , including  $n$  and the root. The **potential function**  $P(n, x)$  provides the cost of adding an extra weight  $x$  along the path from root to a node  $n$ . This is

$$P(n, x) = \sum_{u \in \text{path}(n)} m(u)^\beta [(w(u) + x)^\alpha - w(u)^\alpha] d(u). \quad (1)$$

**Theorem 1** *The algorithm for growing a leaf stops if  $1/2 < \alpha \leq 1$*

**Proof** Let  $R(G)$  be the Euclidean distance from the root to a node in  $G$  which is furthest away from the root. When a node is added,  $R(G)$  may or may not increase, but there will be an infinite number of new nodes which do increase  $R(G)$  as  $G$  grows. Let  $r$  be one of them, so it is a square just outside the  $G$  at least as far away as  $R(G)$ . Assume that  $b$  is the chosen parent of  $r$  inside  $G$ . The cost of adding  $r$  is at least

$$\Delta(r) = d(r)m(u)^\beta + P(b, 1)$$

Here  $d(r)$  is the distance from  $b$  to  $r$ . Since  $m(u)^\beta$  is always at least 1,

$$\Delta(r) \geq d(r) + \sum_{u \in \text{path}(n)} [(w(u) + 1)^\alpha - w(u)^\alpha] d(u).$$

We're assuming  $\alpha \leq 1$ , so by the Lemma, each of the terms  $[(w(u) + x)^\alpha - w(u)^\alpha]$  gets smaller as  $w(u)$  gets bigger. The biggest any  $w(u)$  could be is the number of nodes  $|G|$  in  $G$ , which is equal to the area covered by  $G$ . So

$$\Delta(r) \geq d(r) + \sum_{u \in \text{path}(n)} [(|G| + x)^\alpha - |G|^\alpha] d(u).$$

Using  $(|G| + 1)^\alpha - |G|^\alpha \leq 1$ , we get

$$\Delta(r) \geq [(|G| + 1)^\alpha - |G|^\alpha] \left( d(r) + \sum_{u \in \text{path}(n)} d(u) \right).$$

The term in big brackets is the length of the path from  $r$  to the root, so it must be at least  $R(G)$ .

$$\Delta(r) \geq [(|G| + 1)^\alpha - |G|^\alpha] R(G)$$

Now  $|G|$  cannot be any bigger than the circle centered at the root and with radius  $R(G) + 1 \leq 2R(G)$ . (We add 1 because parts of squares might stick out of the circle.) So  $|G| \leq 4\pi R(G)^2 \leq (4R(G))^2$ , so  $R(G) \geq (1/4)|G|^{1/2}$ , and

$$\begin{aligned} \Delta(r) &\geq (1/4)[(|G| + 1)^\alpha - |G|^\alpha] |G|^{1/2} \\ &\geq (1/4)[(|G| + 1)^{\alpha+1/2} - |G|^{\alpha+1/2}] \end{aligned}$$

Now we can use the Lemma with  $\eta = \alpha + 1/2$ . Since  $\alpha > 1/2$ ,  $\eta > 1$ , which shows  $\Delta(r)$  will eventually get bigger than  $\epsilon$  as  $|G|$  gets bigger.

## 4 Examples

In the pictures, the veins are shown with width proportional to the square root of  $w(n)$ , or the diameter of the vein. The colors are based on the transport cost for a cell and all its descendants, with darker colors for bigger costs.

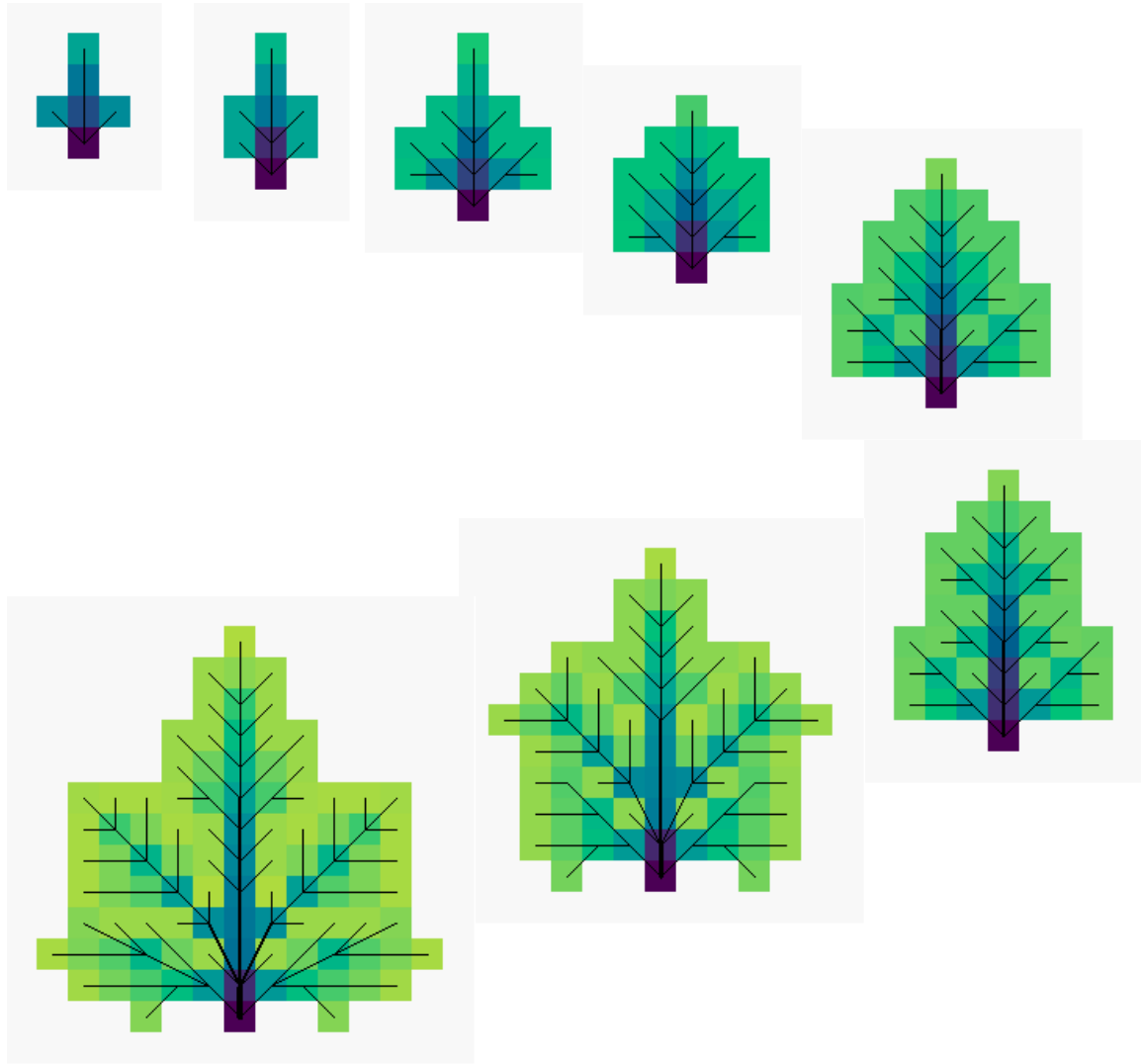


Figure 4:  $\alpha = 0.70, \beta = 0.70, \epsilon = 5.00$ . This is the same example at the start, showing some steps during its growth.

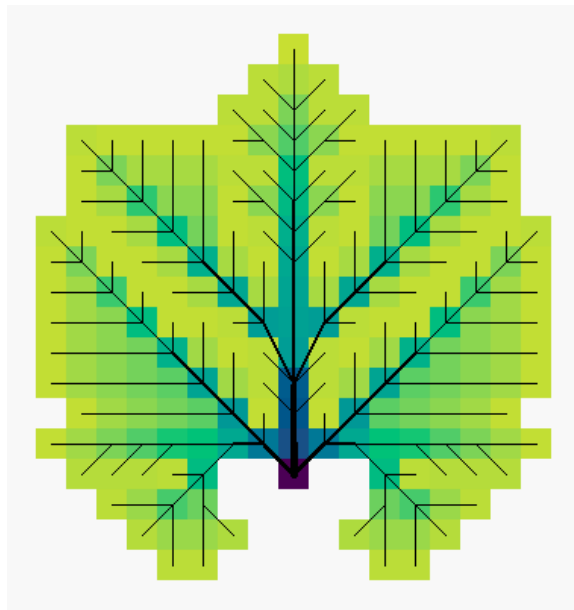


Figure 5:  $\alpha = 0.68, \beta = 0.38, \epsilon = 5.00$ . This one uses the parameters from Xia's paper to make a leaf like a maple. Mine is not the same, though it is similar. See section 5.2

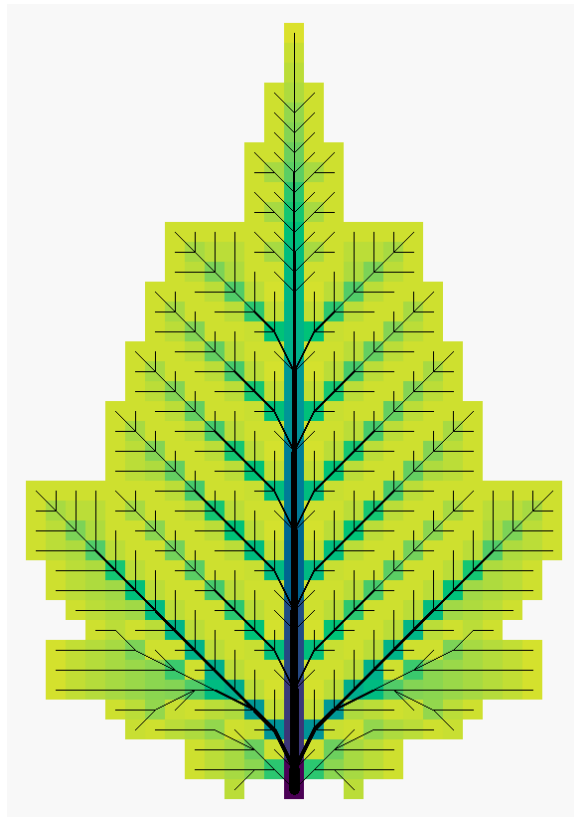


Figure 6:  $\alpha = 0.55, \beta = 1.00, \epsilon = 5.00$ . Small alpha, large beta.



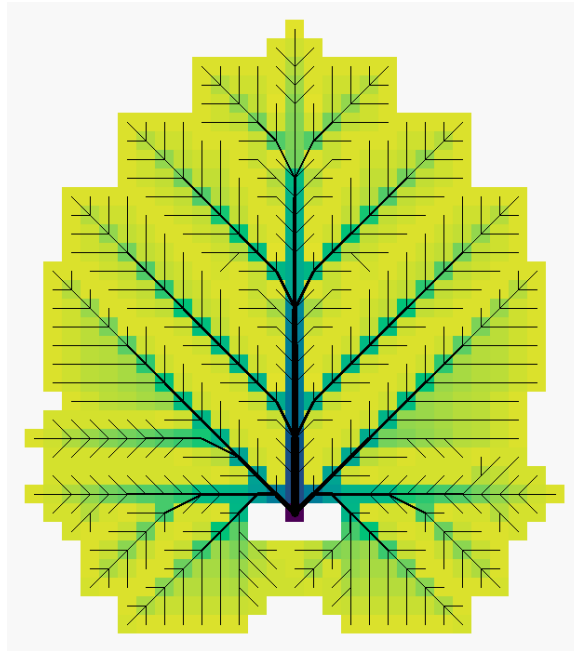


Figure 7:  $\alpha = 0.70, \beta = 0.40, \epsilon = 7.00$ . Medium alpha, small beta. Symmetry is broken when the leaf meets itself below the root.

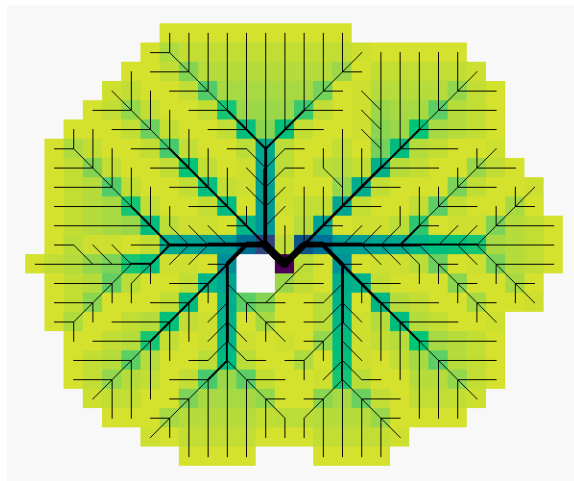


Figure 8:  $\alpha = 0.70, \beta = 0.01, \epsilon = 5.00$ . Medium alpha, tiny beta. Maybe like a lily pad?

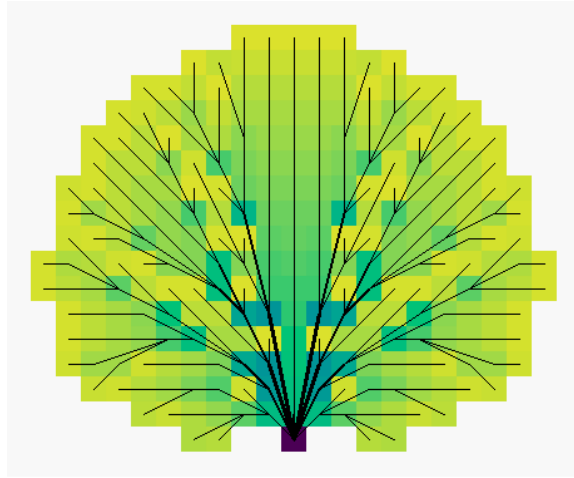


Figure 9:  $\alpha = 0.95, \beta = 1.50, \epsilon = 15.00$ . Large alpha and large beta. Maybe like a ginkgo?



Figure 10: Ginkgo leaf in autumn (detail). Credit: Joe Schneid, Louisville, Kentucky - Own work, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=9488972>

## 5 Some thoughts

### 5.1 The nature of the model

The model is like a caricature: it captures some important features of how real leaves grow, while ignoring or distorting other aspects. There are quite a few models of this sort in mathematical biology, because biology is far too complicated to model in full detail.

Lots of leaves have features not captured in the model, and some, like grass and conifer needles, seem to use a different algorithm altogether.

I find it thought-provoking that large alpha and large beta (Fig 9) make a fan-shaped leaf a bit like a ginkgo leaf. In evolutionary terms, the ginkgo is very lonely: it split from its nearest relative about 300 million years ago. Perhaps the ginkgo struggled to compete with plants that had more efficient veins? At any rate, it's not a hypothesis that is likely to occur to you if you were playing with an algorithm that mimicked leaves using fractals or some such.

### 5.2 Some wrinkles in the algorithm

Xia uses a sequence of increasing epsilon values  $\epsilon_1, \epsilon_2, \dots, \epsilon_k$  satisfying

$$0 < \epsilon_1 < \epsilon_2 < \dots < \epsilon_k = \epsilon$$

for each leaf. Cells are added until the next value is reached, then the veins are optimized. However, the exact sequences are not given. I chose to collect just the cells that can be added at minimum cost, add those, then optimize the veins.

When veins are optimized, there can be ties between different possible parents for a cell. Somehow, these ties must be broken. This is noticeable in Figs 7 and 8.

Real leaves do not grow on a square grid. Some features, like the hole in Fig 8 are artifacts of the grid. More generally, the available angles for bends is very limited.

## 6 Pseudo-code

This is aimed at people who want to write their own code. I recommend starting with the simple version.

### 6.1 Pseudo-code for simple version

```
transport_cost(G) {
  fill in w(n) for every node, from tips to root (postorder)
  fill in b(n) for every node
  fill in m(n) for every node, from root to tips (preorder)
  find d(n) and fill in f(n) for every node
  return the sum of the f(n)'s
}
```

For  $w(n)$ , we visit every child of a node before the node itself. At a tip  $x$ ,  $w(x) = 1$ . Otherwise, we add:

$$w(n) = 1 + \sum_{c \in \text{children}(n)} w(c).$$

For  $m(n)$  we visit parents before children. At the root,  $m(n) = 1$ . Otherwise, we multiply:

$$m(n) = b(n)m(\text{parent}(n)).$$

```
add_cells(G, epsilon) {
  A = empty list
  for (every cell c outside the tree) {
    for (every 8-nbr p of c which is inside the tree) {
      make a new tree G2 with c as child to p
      calculate cost(c,p) = transport_cost(G2)-transport_cost(G)
      Add triple (c,p,cost(c,p)) to A
    }
  }
  find the minimum cost minc of all the cost(c,p)'s in A
  if (minc < epsilon) {
    add all (c,p) with minimum cost equal to minc to G
  }
  return G
}
```

This is not quite the same as Xia's algorithm - see section 5.2.

```

optimize_veins(G) {
  for (every nonroot node n) {
    p = parent(p)
    Q = list of every 8-nbr of n and every 8-nbr of p
    for (every q in Q which is not n nor a descendant of n) {
      make a new tree G2 with n having parent q not p
      if (transport_cost(G2) < transport_cost(G)) {
        G = G2
      }
    }
  }
  return G
}

```

The order in which nodes are visited make a slight difference to the way the leaf develops, and can break symmetry - see section 5.2.

Finally, the main loop:

```

Set alpha, beta, epsilon
Make a one-node tree G for the root.
repeat:
  G2 = add.cells(G, epsilon)
  if (G2 no bigger than G) {
    break
  }
  F = transport_cost(G)
  repeat:
    G2 = optimize_veins(G)
    F2 = transport_cost(G2)
  until (F2 >= F)
until false

```