Visualising Populations of Rooted Labeled Trees on a Lattice

Steven Gustafson and Natalio Krasnogor
University of Nottingham
smg@cs.nott.ac.uk nxk@cs.nott.ac.uk

January, 2003

Abstract

This paper describes an algorithm for visualising rooted labeled trees on a lattice, described by Daida [1]. The method for generating points on the lattice is described and previous work is extended with a technique for visualising populations of trees in three dimensions and an accompanying measure for comparing populations based on their three dimensional structure, which encapsulates many key features of a population's structural properties.

1 Rooted Labeled Trees

We define a rooted labeled tree T as a set of nodes N, where $N = F \cup T$. T is a set of symbols that label leaf nodes in the tree and F is the set of symbols to label internal nodes, where $F \cap T = \{\}$. The sets F, T represent functions and terminals, respectively, in a parse tree representation of computer programs (discussed in Section 5). Let us assume that the degree of a node labeled by F is always 3 (except the root which has degree 2), and nodes from T always have degree 1.

We further label the nodes in a tree T by the standard array representation of binary trees, where a node n has the integer label k that serves as an array index. A node at position k can then have edges leading to children at positions $k * 2$ and $k * 2 + 1$. The root node in the tree has label $k = 1$.

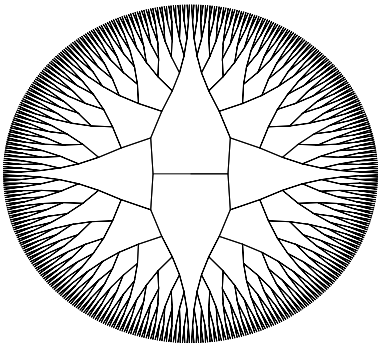

Figure 1: A lattice for a full binary tree of depth 10. The root nodes is denoted by the centermost point, at the mid-point of the center most horizontal line.

2 Lattice Points Generation

Daida [1] notes that an absolute labeling and ordering of nodes allows the construction of a lattice where a tree can be concisely and compactly visualised. The lattice consists of points on the circumference of concentric circles, where each circle represents nodes at a particular depth (the greater the depth the greater the radius of the circle). Figure 1 shows the lattice for a full binary tree of depth 10 where each lattice point is connected to its children points.

To create this visualisation, an algorithm is used to find points on the circumference of concentric circles, where a circle representing depth d has 2^d points (one for each node in the tree at depth d). The points on each circle are labeled in increasing degree from degree 0 (where the origin is the center of the circle, and degree 0 is due North). Points are labeled counterclockwise from degree 0, which places the points labeled $k * 2$ and $k * 2 + 1$ closest to point k on the circle with the next smalled radius. Note in Figure 1, the root of the tree is at the center of the circle and lies in the middle of the centermost line.

Figure 2 shows how the initial points are chosen on the set of concentric circles. In the left figures, the circle with radius 0 is divided into 2 parts and a point is place there (the origin) with the vertical line. The next circle with radius 1 is divided into four parts with the horizontal line and points are placed at the intersection of the circle and this line. The right figure

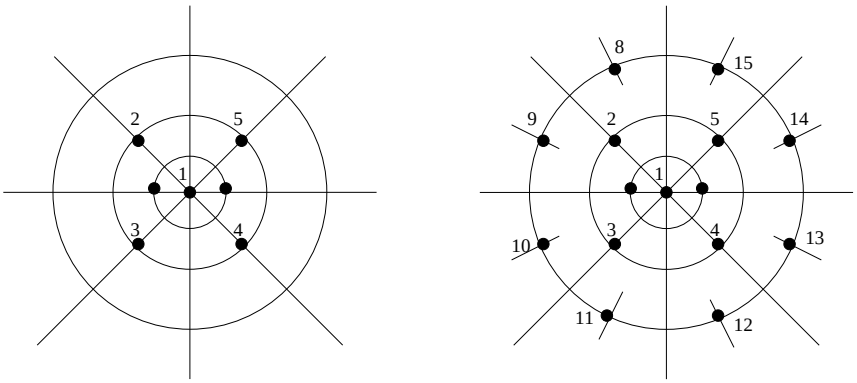

Figure 2: Divide concentric circles into even parts and place points on the intersection of the circle and division lines, avoiding points which would lie at the same angle as previously placed points.

shows the placing of 8 points on the outermost circle.

The following algorithm divides a circle into p parts, where $p = 2^d$ is the number of points on a circle with radius d. We find the coordinates of the 2^d points on each circle and use the coordinates for every other point, skipping the points which would lie on a line already used to divide smaller radius circles (as described above) and would be at the same angle from previous points.

```
function lattic_points (maxdepth)
   for(depth=0, depth <= maxdepth+1, depth++)
      points = exp(2,depth)
      angle = (2 * PI)/points
      for(p=1 , p <= points, p=p+2)
         if( !(depth==1 &&  p==1) )
            x = depth * sin(p * angle)
            y = depth * cos(p * angle)
            output x,y
```

We generate the points which represent, in order, the locations for nodes in an array representing binary trees. The final step is to then list our full binary tree T as an array, $k = 0, \ldots, 2^d$ and pair each point with its corresponding node.

To generate the proper edges, shown in Figure 1, we only need to do a depth first search on this array of coordinate and node pairs to generate the

Figure 3: Left: A binary tree with 7 nodes. Right: The same binary tree projected onto a lattice where the root (highest node in the left figure) is the centermost node in the figure.

beginning and end coordinate of each edge. Thus, a full binary tree is shown in the left figure in Figure 3, where each point is marked with an circle.

3 Visualising Populations of Trees

While the above visualisation of a binary tree on a lattice is compact, we also wish to understand the structure of a population of trees. To do this, we generate a global binary tree, T_G, represented as an array (as described above). Then, a depth first search is performed through *every* tree in the population. For every node in every tree that is encountered at absolute position k, increment $T_G[k]$ by 1 ($T_G[k]$ is initially 0). Plotting the same tree in three dimensions with the height of each node representing the number of trees in the population with a node in that position gives us a view of the structure of the entire population.

Figure 4 shows the same tree from Figure 3 plotted in three dimensions. The left figure of Figure 4 shows the points on the lattice in two-dimensions. This tree represents a population of trees, where the height of a node corresponds to the number of times the population contains a tree with this node.

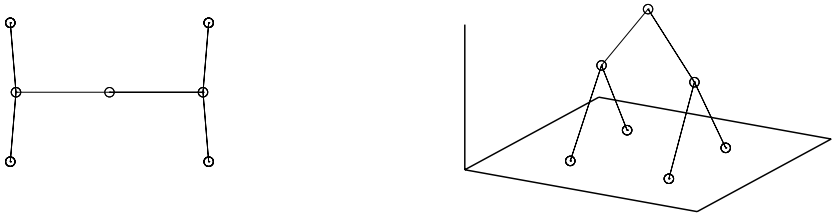

Figure 4: Left: A binary tree with 7 nodes on a lattice (from Figure 3. Right: The same binary tree in three dimensions, which now represents a forest of trees. This figure could represent a forest of three trees, one with only a root, one with a root and two children, and one with a root and two children and those children each with two children.

4 Comparing 3D Population Structures

It would be useful to compare different population structures without referring to the actual visualisation. We wish to capture the essence of the visualisation in a measure.

To create a measure for comparing structures based on the visualisation we identify the three key characteristics of the visualisation and incrementally build a measure to capture those characteristics. Each point in our visualisation is represented by a triplet of x, y, z, where x, y are the coordinates on the XY plane and z is the height of the point, which represents the number of times the population samples this node.

4.1 Total Nodes and Average Nodes in a Tree

The first characteristic is the total number of nodes in a population. Obviously, this rough measure allows smaller and bigger populations to be distinguished. The total nodes in the population is easily found by summing all the z_k values: $\sum_k z_k$, where z_i is the number of times the population samples the node at position i in the absolute labeling of our trees.

4.2 Area Approximation

Two populations may have equal number of nodes but differ dramatically in the structure they take. The next characteristic is the overall area that the structure occupies in three-dimensions. This value will be affected by fuller subtrees which appear closer to the root. The population structure can be defined as edges connecting each parent to its two children. In our array representation, when a parent at absolute position k has a child at $k*2$ then there is an edge from (k,$k*2$). To assign a value to the structure that a population creates in three-dimensions, we approximate the area under that structure by finding the area under the edges, a plane perpendicular to the XY-plane. The areas of these planes give more weight to nodes nearer to the root in the current visualisation method.

The integral of the line/edge in three dimensions represented by the coordinates of each parent and child pair, (x_1, y_1, z_1) and (x_2, y_2, z_2), points can be described as:

$$\int_0^A \frac{z_2 - z_1}{A} x + z_1 \; dx, \tag{1}$$

where

$$A = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2}$$

The sum the above integrals for each (parent,child) pair in our graph defines an area approximation. Note that this value will be highest when every tree in the population represents a full binary tree, and lowest when every tree in the population is a single node. We will refer to this measure on a population as AM.

Figure 5 represents a sample of populations, their corresponding structures and measures. Note that if two populations p_1, p_2 have the same number of nodes n, then the population which has more nodes closer to the root will have a higher AM value. This is because a population with more nodes nearer to the root will have points with higher position in the structure connected to children with longer edges. The same ordering of structures could be found by a weighted summation of z_k values, where the weight decreases as the depth increases.

4.3 A Volume Approximation of 3D Structures

The area appriximation measure captures parent child relationships in the population but fails to consider the relationships between nodes at the same

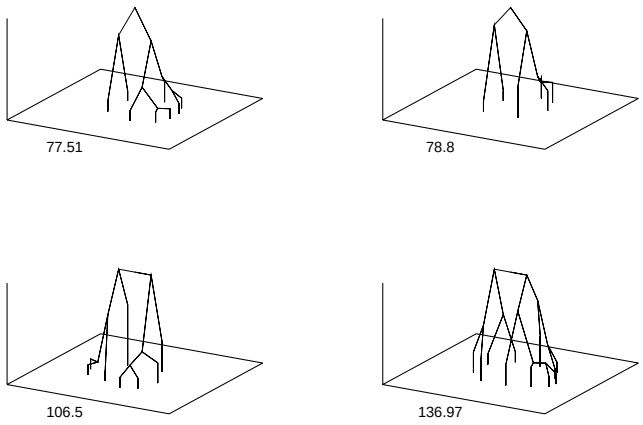

Figure 5: Four structures and the corresponding area approximation measure, denoting the summation of areas under the edges.

depth, i.e. how balanced and full are the trees in the population. To incorporate this information into another measure of our three dimensional structure, we now consider the volume that the structure occupies, a volume approximation, VM.

To approximate the volume, we define two types of polygons on our original lattice, a triangle which has points of a parent and its two children, and a quadrangle which has points of two neighboring parents at the same depth, and two children, one child from each parent. The area of the polygon is found by listing the coordinates of each polygon corner counter-clockwise and summing the determinant of each adjacent coordinate. Thus, for points defining a three sided polygon $(x_0, y_0), (x_1, y_1), (x_2, y_2)$, the area follows:

$$\frac{1}{2}((x_0y_1 - x_1y_0) + (x_1y_2 - x_2y_1) + (x_2y_0 - x_0y_2)) \tag{3}$$

Figure 6 shows our original lattice with polygons drawn in.

To find the volume of these polygon towers, where each polygon corner has the height equal to that point/node's occurrence in the population, we simply approximate the height by summing the heights of each corner and taking the average. An example of the three dimensional structure with polygons drawn in Figure 7.

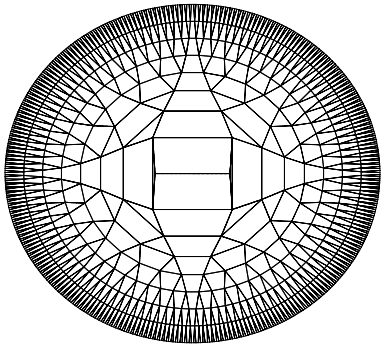

Figure 6: The lattice with all points and edges drawn and the additional edges defining the polygons used in the volume approximation measure. Note that the triangles measure the area between children, while the quadrangle captures the area between neighbouring cousins (children from different parents at the same depth).

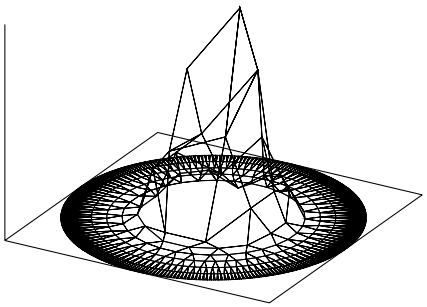

Figure 7: A structure with polygons drawn in. The measure of volume used in this paper approximates the height of the polygons to the be the average of the height of the corners.

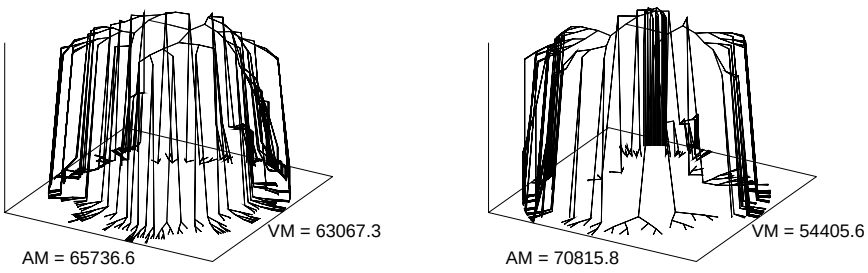

Figure 8: Two structures and their corresponding area (AM) and volume (VM) measures. The right structure contains many 'gaps' caused by the presence of many sparse trees in the population. The distinction is uncovered by the VM measure and gives more value to the left structure which is fuller and samples the population at depths near the root more.

This volume approximation, VM, gives more information and a different ordering among structures than our previous measure. The main benefit is that its emphasises the importance of the entire structure being evenly balanced, rather than just particular subtrees, as the space between parents and children and similar depths is considered. An example if found in Figure 8, where the two populations of trees which are quite full. The value of each measure AM and VM is shown.

Note that the left population has a smaller AM value but *higher* VM value. This population samples nodes closer to the root much more evenly than the right population. In fact, the right population has a characteristic that is quite deceptive: This population samples nodes more at deeper depths, but does not sample evenly from all possible subtrees at higher depths (the importance of this is discussed in the Section 5). Obviously, there are other possibilities to find an approximate volume for these structures.

The fact that we can easily change the distances and scaling in the lattice to give more importance to different depths makes this proposed measure ideal as it represents the visualisation. The key elements of the visualisation are: representing the structure of a population of trees compactly, representing the relationship between parents and children and the relationship

between nodes at the same depths. The measure based on volume approximation captures these elements by finding the heights of the polygons which represent the parent-child relationship and the relationship between neighboring nodes at the same depth. This provides an accurate measure with respect to the three dimensional structure represented in our visualisations. Lastly, the volume measure could be improved by a more exact measure of the volume of the polygon towers.

5 Remarks

The motivation of this paper comes from the machine learning field of evolutionary algorithms. Here, we are interested in *evolving* a population of possible solutions to a specified problem. The subfield of evolutionary algorithms in which we are focusing on is genetic programming, which uses a parse tree for representing its possible solutions to problems. Parse trees are easily represented as rooted labeled trees, were internal nodes represent functions in a programming language and leaf nodes represent inputs to those functions, all defined over a specific problem.

In genetic programming, the algorithm must search over the space of all possible structures, rooted labeled trees, and contents, which functions and terminals to use at node positions. Thus, the ability of populations of trees to represent different types of structure is critical, particularly in more advanced topics within the research field.

It was our interest to develop a method to represent populations of these structure compactly and a measure to compare them. Thus, the publication [1] which describes the original lattice visualisation fit perfectly into our ongoing work. This representation does give much more importance in the visualisation to the root portions of trees, and the bias is useful for developing measures to compare structures as many operators and problems in genetic programming are affected by depth and sampling differently at different depths.

References

[1] J.M. Daida. Limits to expression in genetic programming: Lattice-aggregate modeling. In D.B. Fogel et al., editors, *Proceedings of the 2002 Congress on Evolutionary Computation*, pages 273–278. IEEE Press, 2002.