Visualizing Tree Structures in Genetic Programming

Jason M. Daida, Adam M. Hilss, David J. Ward, and Stephen L. Long

Center for the Study of Complex Systems and the Space Physics Research Laboratory
The University of Michigan, 2455 Hayward Avenue, Ann Arbor, Michigan 48109-2143

Abstract. This paper presents methods to visualize the structure of trees that occur in genetic programming. These methods allow for the inspection of structure of entire trees of arbitrary size. The methods also scale to allow for the inspection of structure for an entire population. Examples are given from a typical problem. The examples indicate further studies that might be enabled by visualizing structure at these scales.

1 Introduction

"Structure always affects function." ([1], p. 268)

In saying this, the author was commenting on the nature of complex networks. To many who study the anatomy of such networks, the topology of interconnections can have a pronounced effect on function. True, the author was thinking along the lines of complex networks like an electrical power grid or the Internet. However, he could have just as well been referring to the kinds of solution outcomes that happen in GP.

In particular, one might say that networks are prevalent in GP. Each individual can be framed as a topology of interconnections between functions and terminals. Each population can be framed as a topology of interconnections not only between individuals at time t_i, but also of the times preceding t_i.

One could also suspect that such networks in GP are complex. After all, the metaphor from which GP draws from is generally considered to be complex. One might even speculate that the mathematical properties of complex networks (e.g., scale-free networks) might also apply to GP. To say, however, that GP *produces* complex networks is a matter of conjecture. No one in the field has proven this. Nevertheless, there has been evidence that suggests that the topology of interconnections has a significant effect on the outcomes for GP.

For example, some of the early theoretical work in GP has alluded to the consequences of structure in determining solution outcomes. Rosca and Ballard [2, 3] hypothesized that solution outcomes are determined, in part, by rooted-tree schema. They argued that during a run, GP identifies these structures first and subsequently builds upon these structures to form solutions.

Other researchers have independently supported Rosca and Ballard's theory, which highlighted the importance of root structure. This support has come largely in the form of empirical studies that featured selected benchmark problems. These would include [4, 5]. Of these two, McPhee and Frietag's work made the strongest empirical

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 1652–1664, 2003.

case that root structures play a key role in determining solution outcomes for GP in general.

Structure, of course, has played a significant role in other studies in GP as well. For example, structure was a key determinant of fitness in one of the earliest adaptations of the Royal Road problem [6] (in genetic algorithms) to GP [7]. Work by Goldberg and O'Reilly [8, 9] resulted in the development of the ORDER and MAJORITY fitness problems in which the structures of solution outcomes were driven by the problem at hand. Structure has played a key role in the analysis by Soule, Foster et al. [10-13], who has looked at the evolution of shape in GP outcomes. Soule and Foster's work on the evolution of tree shapes has since been carried forward and extended by Langdon [14, 15].

Structure plays a key role, too, in the prevailing theory on GP. Poli's work on a GP schema theory (and later Langdon and Poli) [15-18] presumes schemata that are structurally based (i.e., has a certain size and depth). Bloating has also been associated with structure; it has been Langdon and Poli's contention that bloating occurs partly as a result of a random walk on a landscape of tree-shape distributions (e.g., [19, 20]).

However, in spite of this longstanding interest in the role of structure, the direct visualization of such structures has *not* progressed far. Nearly all visualizations of structure have been along the lines of "for illustrative purposes only," as was first done by Koza [21]. There have not been a viable means of inspecting structures for typical trees that have been discussed in GP, let alone a viable means of inspecting structures for an entire population. Consequently, our investigations have focused on the visualization of tree structures and other topologies of interconnections between nodes. The purpose of this paper, then, is to describe our methods for visualizing whole tree structures, which would be about two orders of magnitude larger in tree size than what has typically been published. The purpose of this paper is to also describe our methods for visualizing structures for entire populations, which would be another two orders of magnitude beyond that of an individual tree.

This paper is organized as follows. Section 2 provides the mathematical context and summarizes the current conventions in visualizing trees. Section 3 describes our method for visualizing the structure for an entire tree and provides examples of these visualizations. Section 4 extends the methods described for individual trees and scales them to encompass populations. Section 5 concludes this work.

2 Background

2.1 Mathematical Preliminaries

The conventions and terms that are used in this paper are those of Knuth [22] and Stanley [23, 24].

A *tree T* is formally defined as a finite set of nodes such that:

a) There is one node that is designated as the *root* of the tree, and
b) The remaining nodes (excluding the root) are partitioned into $m \geq 0$ disjoint, non-empty sets $T_1,\ldots,T_m$, each of which is a tree. The trees $T_1,\ldots,T_m$ are called *subtrees* of the root.

The number of subtrees associated with a node is called the *degree* of that node. A node of degree zero is called a *terminal node* or *leaf*. Each root is considered as the

parent of the roots of its subtrees; the roots of these subtrees are said to be the *children* of their parent and *siblings* to each other.

A *plane tree* or *ordered tree* is defined similarly to a *tree*, except that (b) becomes

b´) The remaining nodes (excluding the root) are placed into an ordered partition $(T_1,\ldots,T_m)$ of $m \geq 0$ disjoint, non-empty sets $T_1,\ldots,T_m$, each of which is a plane tree.

If only the relative orientation of nodes is considered and not their order, a tree is said to be *oriented*.

Let $m \geq 2$ in the definition of a tree. An m-ary tree is defined similarly to a *tree*, except that (a) and (b) become

a´´) Either T is empty, or else there is one node that is designated as the *root* of T, and

b´´) The remaining nodes (excluding the root) are put into an ordered partition of exactly m disjoint (possibly empty) sets $T_1,\ldots,T_m$, each of which is an m-ary tree.

An m-ary tree is said to be *complete* if every node that is not a leaf has m children. In deference to GP (and not Knuth), we use term *full* to refer to a complete m-ary tree that has m^d terminal nodes, where d is the depth of that tree and the root node is considered at depth 0. A *binary* tree is a 2-ary tree.

Note that an m-ary tree is *not* a special case of the formal definition of a tree. Consequently, a binary tree is not assumed to be an instance of a *tree T*. By definition, however, a complete binary tree is such an instance.

The treatment of trees in this paper differs from the current treatment of trees in GP. GP uses conventions that tend to focus on node content. (For example, internal nodes in GP are assumed to be functions of a specified arity. In turn, arity determines the number of children for that node.) In contrast, the treatment of trees in this paper uses the conventions that tend to focus on structure.

2.2 Current Conventions

In GP, the convention for drawing trees starts with the root node at the top and grows downward with the terminal nodes at the bottom. This is a prevalent convention, starting with [21] through current works like [25]. *Right*, *left*, *top*, and *bottom* are subsequently denoted. *Depth*, as opposed to *height*, is also indicated. This convention follows the standard practice in computer science at large ([22], p. 311). Stanley explicitly states this convention ([23], p. 295): that for m-ary trees (which computer scientists typically use), lines (edges) joining a node (vertex) to the roots of its subtrees are drawn at equal angles that are symmetric with respect to a vertical axis. This convention works if both m and the number of nodes remain small (i.e., small m-ary trees). For example, Knuth depicts two small m-ary trees that still consume a page of illustration (for one, m is 2 and number of nodes is 63; for the other, m is 11 and the number of nodes is 61).[1]

For many problems, drawing a large m-ary tree is not needed, since a small m-ary tree would suffice as a representative sample. For GP, however, we claim that it is instructive to visualize m-ary trees of a few thousand nodes. Our reasoning is that such visualizations would show broader patterns of solution structure that are not readily seen with smaller samples or with structural metrics (like *number of nodes* or

[1] Even so, Knuth adjusted convention slightly by laying each trees on its side.

depth level). Unfortunately for these larger trees, there is little by way of a drawing convention. Consequently, the next section introduces a method for drawing large, *m*-ary trees. To simplify matters, we restrict our attention to plane binary trees.

3 Visualizing Structure for a Tree

3.1 Visualization Method

Our method for visualizing trees involves mapping *m*-ary (binary trees for this paper) to a circular grid. For binary trees, this grid can be derived by starting with a full binary tree, which we designate as *C*. The nodes of *C* are labeled by a positive integer *k* ($k \in \Im^+$) in the following way:

1. The root node is designated as $k = 1$ and at depth level 0.
2. The leftmost node for each depth level *n* is designated as $k = 2^n$.
3. Nodes at depth level 1 are labeled from left to right as $k = 2$ and $k = 3$, respectively.
4. Nodes at each subsequent depth level *n* are labeled from left to right using the following sequence: $\{2^n, 2^n+1, 2^n+2, \ldots 2^{n+1}-1\}$.

The nodes of this labeled full binary tree can be mapped to a circular grid in polar coordinates (r, θ) by first assigning depth levels to a set of concentric rings. The exact mapping of depth levels to ring radii is arbitrary, so long as depth level 0 remains at the center and increasing depths are assigned to rings of increasing radius. For convenience, we let depth level *n* be mapped to a ring of radius *n*. Figure 1a gives an example of a full binary tree of depth 3 that has been labeled in this manner.

Ring radius is subsequently determined by using the following mapping $\rho(k)$: $k \rightarrow r$, such that $k \in \Im^+$, $r \in \Im^+$, and

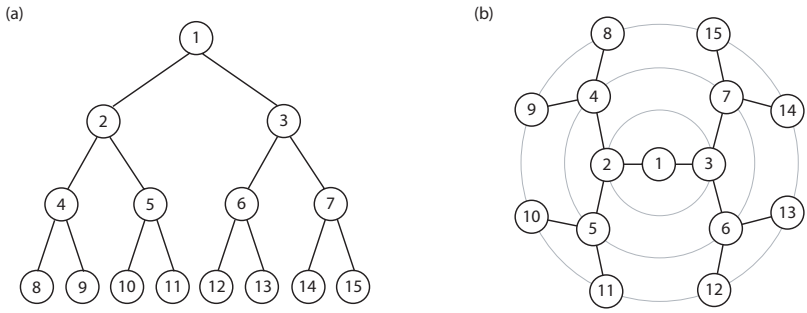

Fig. 1. Mapping a Full binary tree to a circular grid. (a) Full binary tree of depth 3. (b) Corresponding circular grid

$$\rho(k)=\begin{cases} 0, & k=1. \\ \lfloor \log_2 k \rfloor & k>1. \end{cases} \tag{1}$$

Nodes on a ring are positioned by using the following mapping $\phi(k)$: $k \rightarrow \theta$, such that $k \in \Im^+$, $r \in \Re$, and

$$\phi(k)=\begin{cases} 0, & k=1. \\ \pi\left(\frac{1}{2}+\frac{1}{2^{\rho(k)}}+\frac{k \bmod 2^{\rho(k)}}{2^{\rho(k)-1}}\right) & k>1. \end{cases} \tag{2}$$

For the moment, we designate set L_C of plane binary trees to be defined by the collection of labeled nodes (vertices) of C in polar coordinate space. Figure 1b gives an example of a full binary tree of depth level 3 that has been mapped to set L_C.

We can show that an arbitrary plane binary tree A (or for that matter, an arbitrary binary tree) can be mapped to L_C by showing that a set of labels corresponding to A is a subset of L_C. To do so, we traverse A in preorder[2] and label each node visited in the following manner:

1. The root node of tree A is designated as $k = 1$ and at depth level 0.
2. The root node of the left subtree is labeled $2l$, where l is the label of that node's parent.
3. The root node of the right subtree is labeled $2l+1$, where l is the label of that node's parent.

We designate this set of labels for A as L_A. Figure 2 depicts an example of an arbitrary plane binary tree A that is mapped to L_C. If A is a full plane binary tree, it can be readily shown that this preorder labeling does result in an identical labeling that corresponds to C.

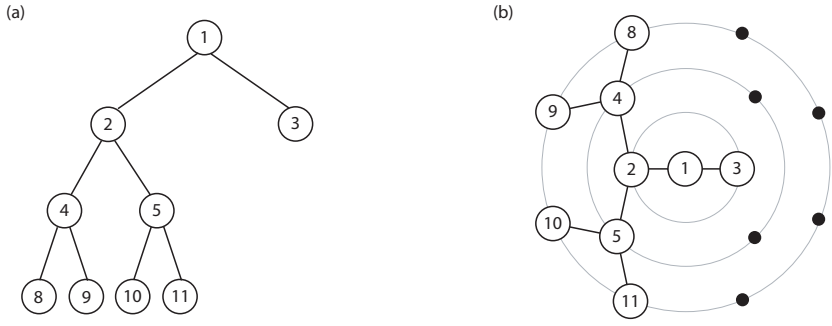

Fig. 2. Mapping an arbitrary plane binary tree to a circular grid. (a) Arbitrary plane binary tree of depth 3. (b) Corresponding circular grid

[2] Preorder traversal is defined recursively and is described in [22], p. 319. There are three steps that are taken: visit the root; traverse the left subtree; traverse the right subtree.

If one forgoes the information content in each node and treats each node as a vertex, one can use the described visualization method to examine the gross structure of (plane binary) trees. The compact representation allows for the depiction of structures consisting of one to several thousands of vertices.

3.2 Visualization Examples

Figure 3 shows an example of a full binary tree of depth level 10 (2047 nodes) and an example of plane binary tree of depth level 26 (461 nodes) that result from GP. From the standpoint of typical metrics that quantify aspects of tree structure (i.e., *number of nodes* and *depth level*), the GP binary tree is unremarkable. The GP tree was generated under circumstances that were also unremarkable and is a solution from a symbolic regression problem that has been documented elsewhere in the literature [26-29].[3]

In comparison to a full binary tree, the GP plane tree that is shown in Fig. 3 is sparse and is characterized by a few radial spikes in which much of the nodes are found. There are significant areas of the lattice that are not populated. It is asymmet-

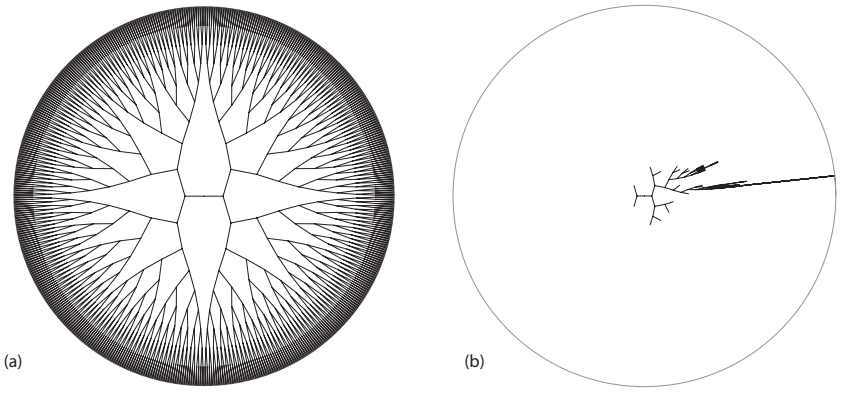

Fig. 3. Two examples of plane binary trees. (a) Full binary tree, depth 10. (b) Structure of a GP-generated solution (using arity-2 functions), depth 26. The gray circle around (b) represents depth 26 and is shown for reference

ric. As it turned out, though, this gross pattern is representative of the population from which it was taken.

---

[3] The problem is an instance taken from symbolic regression and involves solving for the function $f(x)= 1 + 3x + 3x^2 + x^3$. Fitness cases are 50 equidistant points generated from $f(x)$ over the interval [-1, 0). The function set was {+, –, ×, ÷}, which correspond to arithmetic operators of addition, subtraction, multiplication, and protected division. A terminal set was {X, **R**}, where X is the symbolic variable and **R** is the set of ephemeral random constants that are distributed uniformly over the interval [-1, 1]. Most of the GP parameters were similar to those mentioned in Chapter 7 in [21]. Unless otherwise noted: population size = 500; crossover rate = 0.9; replication rate = 0.1; population initialization with ramped half-and-half; initialization depth of 2–6 levels; and tournament selection ($n_{tournment}$ = 7). Other parameter values were maximum generations = 200 and maximum tree depth = 26.

Figure 4 shows that individual tree structure in the context of 47 other individuals. All of these trees have been taken from the same population of 500. Figure 5 shows another 48 individuals out of a population of 500, except that these individuals were generated with a slightly different set of GP parameters (i.e., fitness-proportionate, rather than tournament selection). In both figures, although many of the plane binary trees structures look similar (even identical) they are, in fact, distinct individuals. Close inspection shows minor structural differences between even the most similar-looking individuals.

Figures 4 and 5 pose a number of questions. What drives the shape of these structures? Are these figures representative of GP as a whole? Are there limits to the kinds of structures that can be generated in GP? How do structures evolve over time? Why is there a lack of structural diversity? Does the type of selection influence the kinds of structures that are generated? How does content correlate with structure? Which metrics best characterize these structural features? How do structures vary between trials? Clearly, there is a high degree of structural correlation in this small sample. Unfortunately, the questions that can be raised by a suggestion of structural pattern are more than what can be adequately answered by this paper.

4 Visualizing Structure in a Population

4.1 Visualization Method

Although only 9.6% of the their respective populations have been depicted in Figs. 4 and 5, the high degree of structural correlation depicted in those figures suggests that the rest of their populations are also structurally correlated. We can test this possibility by starting with a cumulative distribution on L_C for an entire population, i.e.,

$$\mathrm{L}_P = \sum_{\forall A \in P} \mathrm{L}_A, \tag{3}$$

where A is a tree in a population P, $L_P \supset L_C$ and $\mathbf{L}_i$ is a vector corresponding to L_i such that

$$\mathrm{L}_i \equiv \sum_{\forall a \in L_i} \mathrm{i}_a . \tag{4}$$

Note that $\mathbf{i}_a$ specifies a unit component vector, and that a is a label in L_i.

In other words, the (un-normalized) cumulative distribution of a population P can be treated as a sum of the vectors corresponding to the trees in that population. The vector that corresponds to each tree A is defined as a sum of unit vectors, where each unit vector corresponds to an occupied grid-point in L_C. Consequently, the tree that spans the population is described by L_P.

For example, suppose a population P consists of four binary trees that have the labels {1, 2, 3, 6, 7, 14, 15}, {1, 2, 3}, {1, 2, 3, 4, 5}, and {1}. The corresponding un-normalized cumulative distribution for this population would be $\mathbf{L}_P = 4\mathbf{i}_1 + 3\mathbf{i}_2 + 3\mathbf{i}_3 + \mathbf{i}_4 + \mathbf{i}_5 + \mathbf{i}_6 + \mathbf{i}_7 + \mathbf{i}_{14} + \mathbf{i}_{15}$, with $L_P = \{1, 2, 3, 4, 5, 6, 7, 14, 15\}$.

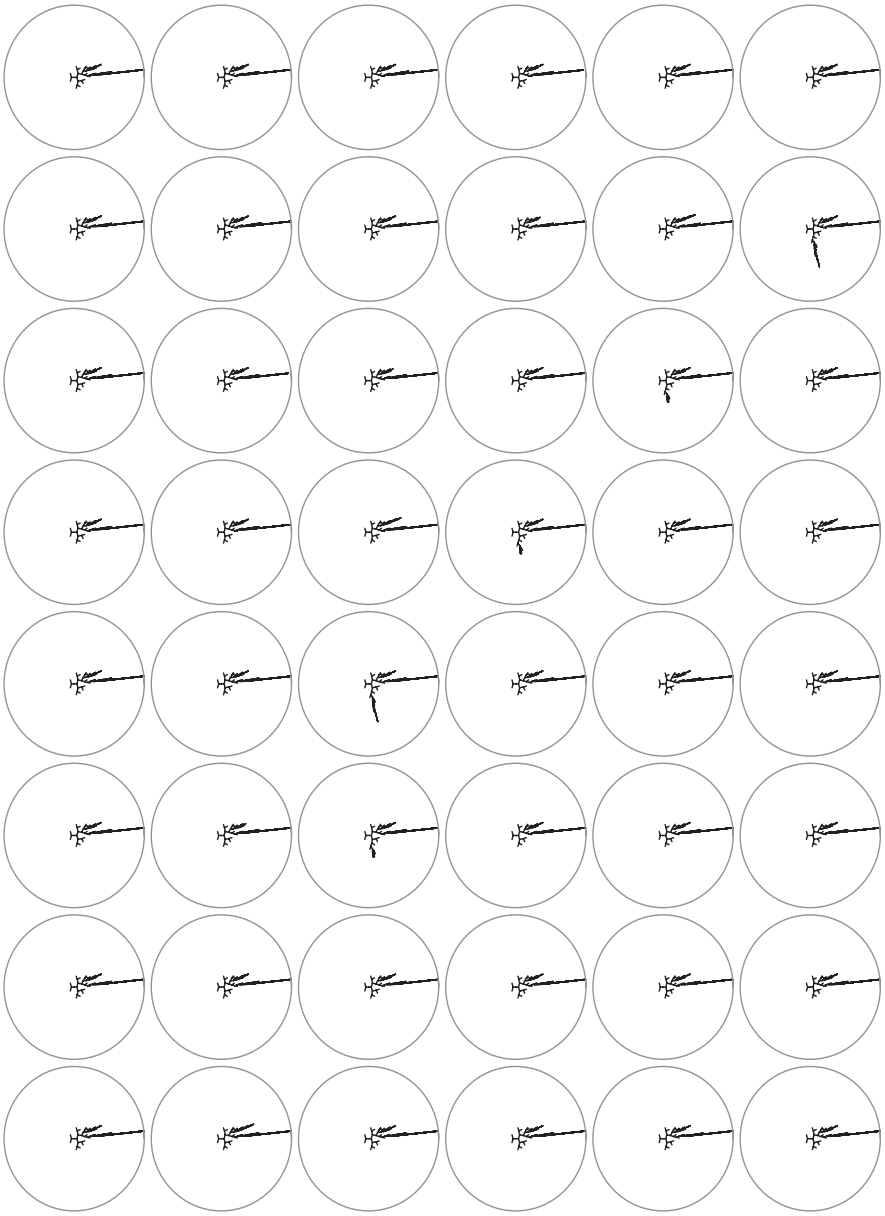

Fig. 4. Sample of a population. These are 48 representatives out of a population of 500. The individual that is shown in Figure 3b is the second from the top-left corner. These results are for a symbolic regression problem (binomial-3) using tournament selection ($n_{tournment} = 7$). As reference, a gray circle around each individual represents depth level 26

Fig. 5. Another population sample. These are 48 representatives out of a population of 500. These results are for the same symbolic regression problem and GP parameters that were used to generate the results of Fig. 5. The exception was that fitness-proportionate instead of tournament selection was used to generate these results. As reference, a gray circle around each individual represents depth level 26

Visualization of a population can subsequently be done in three steps:

1. Compute $\mathbf{L}_P$.
2. Normalize each component of $\mathbf{L}_P$ by the number of individuals in population P.
3. Construct the tree that corresponds to L_P, except that the grayscale value of each line from parent to child is proportionate to the magnitude of the component vector corresponding to that child. (Assume that higher magnitudes map to darker grays).

The resulting visualization is tantamount to overlaying all of the trees in a population. Structures that are used most often show up darker; the least used structures, lighter.

4.2 Visualization Examples

The left-side graph of Fig. 6 corresponds to the population of which Fig. 4 is a subset. The right-side graph of Fig. 6 corresponds to the population of which Fig. 5 is a subset. Each graph summarizes the structure of 500 trees, which represents an entire population.

While individual trees provide an insight to the degree of structural similarity within a population, they are but a hint. This limitation exists even though our method of visualization of single trees does enables one to see more structure than has been possible by orders of magnitude. It is still not enough for understanding structural trends at a population level.

For those reasons, we would argue that the visualizations of Fig. 6 do help in seeing some of these larger trends. In particular, we can observe the following:

- The darkest regions of each graph occur in the center and radiate outward. (Darker regions represent locations of higher structural correlation.) This indicates that structures near the root are highly correlated within a population. This observation is consistent with theory and observations about the rapid convergence of root structure within GP (e.g., [2, 3, 5]).
- The lightest regions tend to occur near the edges of each graph. This means that

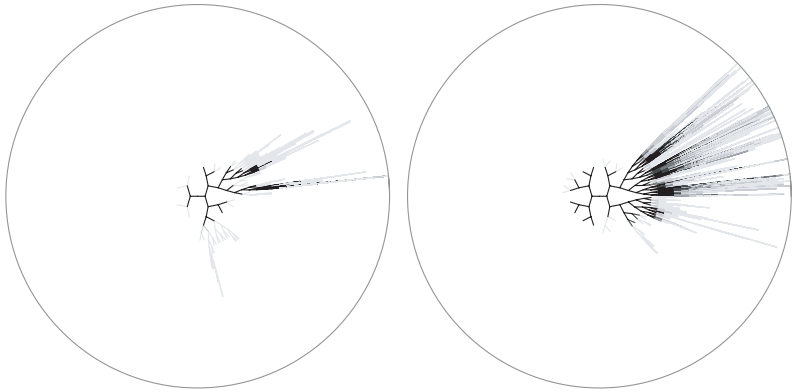

Fig. 6. Visualization of populations. The left graph corresponds to the population for Fig. 4; the right graph, the population for Fig. 5

the structures proximal to the terminals are the least likely to be found from individual to individual within a population. That they tend to be the lightest parts of each graph in Fig. 6 is also consistent with theory and observations (e.g., [4]).

- The graph on the left seems to exhibit less structural diversity than the graph on the right. Both graphs in Fig. 6 correspond to the same problem and to mostly the same GP parameters. The exception is that the graph on the left corresponds to a trial using tournament selection, while the one on the right corresponds to a trial using fitness-proportionate selection.

Of these three observations, the last one requires more effort to interpret. It is somewhat arguable to judge by visual inspection alone whether a population is more or less structurally diverse than another. Fortunately, the visualization method for populations requires the computation of a cumulative distribution. We can use the information from Equations 3 and 4 to plot cumulative distribution by rank (i.e., from grid points that are the most used to grid points that are the least used). The result of this is shown in Fig. 7.

The top graph of Fig. 7 corresponds to the cumulative distribution under tournament-selection, while the bottom graph corresponds to the cumulative distribution under fitness-proportionate selection. The plots of cumulative distributions show distinctly different curves. The plot for tournament selection shows that grid points are either occupied by most of the population or by nearly none at all—which is what one would expect for a structurally similar population. In comparison, the fitness proportionate case is heavy-tailed and indicates that many more structures are being preserved. This finding is consistent with theoretical work in tournament selection, which argues that diversity is lost under tournament selection [30].

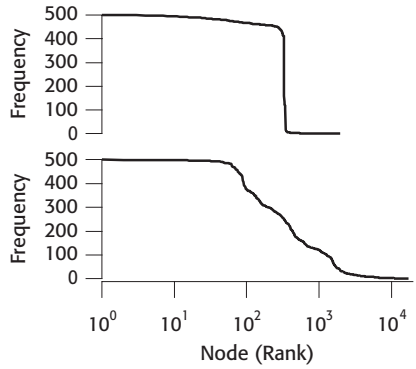

Fig. 7. Distribution curves

5 Conclusions

"Structure always affects function."

Regardless of how strongly stated was this sentiment, work done in the research community does suggest that structure plays a significant role in understanding how GP works. This paper has reviewed a number of investigations that involve the structure of solution outcomes under GP. In spite of these efforts in understanding the role of structure in GP, there has been a deficit in visualization methods that can depict these structures at the scales that would be of interest to the GP community. Consequently, this paper has described two methods to visualize the structure of trees that occur in GP. The first method allows for the inspection of structure of entire trees and can be scaled for trees consisting of several thousand nodes. The second method allows for the visual inspection of structure at the level of an entire population. Both methods are amenable for further quantitative analysis.

Examples were given from a typical problem in GP. We used these examples as a case study that demonstrates how these visualization methods can be used. We further indicated how such visualizations could either raise new questions or complement existing investigations.

References

[1] S.H. Strogatz, "Exploring Complex Networks," *Nature*, vol. 410, pp. 268–276, 2001.

[2] J.P. Rosca, "Analysis of Complexity Drift in Genetic Programming," in *GP 1997 Proceedings*, J. R. Koza, et al., Eds. San Francisco: Morgan Kaufmann Publishers, 1997, pp. 286–94.

[3] J.P. Rosca and D.H. Ballard, "Rooted-Tree Schemata in Genetic Programming," in *Advances in Genetic Programming 3*, L. Spector, et al. Eds. Cambridge: The MIT Press, 1999, pp. 243–271.

[4] C. Gathercole and P. Ross, "An Adverse Interaction Between Crossover and Restricted Tree Depth in Genetic Programming," in *GP 1996 Proceedings*, J. R. Koza, et al., Eds. Cambridge: The MIT Press, 1996, pp. 291–96.

[5] N.F. McPhee and N.J. Hopper, "Analysis of Genetic Diversity through Population History," in *GECCO '99 Proceedings*, vol. 2, W. Banzhaf, et al., Eds. San Francisco: Morgan Kaufmann Publishers, 1999, pp. 1112–1120.

[6] M. Mitchell, S. Forrest, and J.H. Holland, "The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance," in *Proceedings of the First European Conference on Artificial Life.*, F.J. Varela and P. Bourgine, Eds. Cambridge: The MIT Press, 1992, pp. 245–254.

[7] W. Punch, et al., "The Royal Tree Problem, A Benchmark for Single and Multiple Population GP," in *Advances in GP*, vol. 2, P. J. Angeline and J.K.E. Kinnear, Eds. Cambridge: The MIT Press, 1996, pp. 299–316.

[8] D.E. Goldberg and U.-M. O'Reilly, "Where Does the Good Stuff Go, and Why?," in *Proceedings of the First European Conference on Genetic Programming*, W. Banzhaf, et al., Eds. Berlin: Springer-Verlag, 1998.

[9] U.-M. O'Reilly and D.E. Goldberg, "How Fitness Structure Affects Subsolution Acquisition in GP," in *GP 1998 Proceedings*, J. R. Koza, et al. Eds. San Francisco: Morgan Kaufmann Publishers, 1998, pp. 269–77.

[10] T. Soule, J.A. Foster, and J. Dickinson, "Code Growth in Genetic Programming," in *GP 1996 Proceedings*, J.R. Koza, et al., Eds. Cambridge: The MIT Press, 1996, pp. 215–223.

[11] T. Soule and J.A. Foster, "Code Size and Depth Flows in Genetic Programming," in *GP 1997 Proceedings*, J.R. Koza, et al., Eds. San Francisco: Morgan Kaufmann Publishers, 1997, pp. 313–320.

[12] T. Soule and J.A. Foster, "Removal Bias: A New Cause of Code Growth in Tree Based Evolutionary Programming," in *ICEC 1998 Proceedings*, vol. 1. Piscataway: IEEE Press, 1998, pp. 781–786.

[13] W.B. Langdon, et al., "The Evolution of Size and Shape," in *Advances in Genetic Programming 3*, L. Spector, et al., Eds. Cambridge: The MIT Press, 1999, pp. 163–190.

[14] W.B. Langdon, "Size Fair and Homologous Tree Crossovers for Tree Genetic Programming," *Genetic Programming and Evolvable Machines*, vol. 1, pp. 95–119, 2000.

[15] W.B. Langdon and R. Poli, *Foundations of Genetic Programming*. Berlin: Springer-Verlag, 2002.

[16] R. Poli and W.B. Langdon, "A New Schema Theory for GP with One-Point Crossover and Point Mutation," in *GP 1997 Proceedings*, J.R. Koza, et al., Eds. San Francisco: Morgan Kaufmann Publishers, 1997, pp. 279–285.

[17] R. Poli and W. B. Langdon, "Schema Theory for Genetic Programming with One-Point Crossover and Point Mutation," *Evolutionary Computation*, vol. 6, pp. 231–252, 1998.

[18] R. Poli, "Exact Schema Theorem and Effective Fitness for GP with One-Point Crossover," in *GECCO 2000 Proceedings*, L. D. Whitley, et al., Eds. San Francisco: Morgan Kaufmann Publishers, 2000, pp. 469–476.
[19] W.B. Langdon and R. Poli, "Fitness Causes Bloat," in *Soft Computing in Engineering Design and Manufacturing*, P.K. Chawdhry, R. Roy, and R.K. Pant, Eds. London: Springer-Verlag, 1997, pp. 23–27.
[20] W.B. Langdon, "Quadratic Bloat in Genetic Programming," in *GECCO 2000 Proceedings*, L.D. Whitley, et al., Eds. San Francisco: Morgan Kaufmann Publishers, 2000, pp. 451–458.
[21] J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge: The MIT Press, 1992.
[22] D.E. Knuth, *The Art of Computer Programming: Volume 1: Fundamental Algorithms*, vol. 1, Third ed. Reading: Addison–Wesley, 1997.
[23] R.P. Stanley, *Enumerative Combinatorics I*, vol. 1. Cambridge: Cambridge University Press, 1997.
[24] R.P. Stanley, *Enumerative Combinatorics II*, vol. 2. Cambridge: Cambridge University Press, 1999.
[25] C. Jacob, *Illustrating Evolutionary Computation with Mathematica*. San Francisco: Morgan Kaufmann, 2001.
[26] J.M. Daida, et al., "Analysis of Single-Node (Building) Blocks in GP," in *Advances in Genetic Programming 3*, L. Spector, W.B. Langdon, U.-M. O'Reilly, and P. J. Angeline, Eds. Cambridge: The MIT Press, 1999, pp. 217–241.
[27] J.M. Daida, et al., "What Makes a Problem GP-Hard? Analysis of a Tunably Difficult Problem in Genetic Programming," in *GECCO '99 Proceedings*, vol. 2, W. Banzhaf, et al., Eds. San Francisco: Morgan Kaufmann Publishers, 1999, pp. 982–989.
[28] O.A. Chaudhri, et al. "Characterizing a Tunably Difficult Problem in Genetic Programming," in *GECCO 2000 Proceedings*, L.D. Whitley, et al., Eds. San Francisco: Morgan Kaufmann Publishers, 2000, pp. 395–402.
[29] J.M. Daida, et al., "What Makes a Problem GP-Hard? Analysis of a Tunably Difficult Problem in Genetic Programming," *Journal of Genetic Programming and Evolvable Hardware*, vol. 2, pp. 165–191, 2001.
[30] T. Bickle and L. Thiele, "A Mathematical Analysis of Tournament Selection," in *ICGA95 Proceedings*, L.J. Eshelman, Ed. San Francisco: Morgan Kaufmann Publishers, 1995, pp. 9–16.