What Makes a Problem GP-Hard? Validating a Hypothesis of Structural Causes

Jason M. Daida, Hsiaolei Li, Ricky Tang, and Adam M. Hilss

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

Abstract. This paper provides an empirical test of a hypothesis, which describes the effects of structural mechanisms in genetic programming. In doing so, the paper offers a test problem anticipated by this hypothesis. The problem is tunably difficult, but has this property because tuning is accomplished through changes in structure. Content is *not* involved in tuning. The results support a prediction of the hypothesis – that GP search space is significantly constrained as an outcome of structural mechanisms.

1 Introduction

What makes a problem GP-hard? A common hypothesis of why a problem is hard lies with a metaphor of a rugged landscape. Deception, ruggedness, epistasis – all these terms have been evoked at one time or another to describe attributes of a fitness landscape that might be encountered in genetic programming (GP). These concepts have been developed in the genetic algorithms and evolutionary computation communities over the past several decades. There is evidence to suggest that these concepts have legitimacy for GP, as well.

However, there may be more to problem difficulty in GP than what may have been thought. In [1, 2], we presented a hypothesis that argues for the existence of structural mechanisms. The existence of these mechanisms is predicted to have significant consequences. One significant consequence is that GP is greatly limited to the kinds of structures it can create.

In particular, some structures would be easy for GP to produce. Other structures would be difficult. With our theory, it has been possible to create a map of where either difficult or easy structures might exist. The map indicates that only a small portion of the total possible structures would be considered easy. The rest are difficult.

Such a map raises two questions. What happens if GP needs to produce a solution that would best be served by a "difficult" structure? Can there be a purely structural basis to problem difficulty?

We have been investigating ways to validate the predictions that have been set forth by our hypothesis. This paper represents one of those efforts and describes a link between structure and problem difficulty. To start, we have narrowed the scope of the first question. The question becomes this: What happens if GP needs to produce a solution that can *only* be served by a particular set of structures?

If, indeed, there is a purely structural basis for problem difficulty in GP, it should be possible to craft a problem that tests for this. Admittedly, such a problem would be

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 1665–1677, 2003.

unusual. Content in functions and terminals would be rendered irrelevant. Fitness would be based on measurements of structure (i.e., like *number of nodes* and *depth of tree*). While GP was not designed for a problem like this, the hypothesis suggests its existence. In essence, GP would be asked to evolve *nothing*, as far as substance is concerned. However, the hypothesis predicts that some *nothing*s would be harder to achieve than others. It further predicts that most of *nothing* is difficult for GP. The problem that we present in this paper is called the *Lid*, a tunably difficult problem that is based on structure.

In this paper, we use the *Lid* as a probe that empirically tests for locations in the map that identifies where "hard" or "easy" structures exist. This paper presents the first of these results.

Section 2 summarizes both current theories and our hypothesis of how tree structure might determine problem difficulty. Section 3 describes our proposed tunably difficult problem. Section 4 describes our experimental procedure while Section 5 shows the results. Section 6 discusses the results in the context of the theory. Section 7 concludes.

2 Theory

2.1 Previous Work

That structure has any influence on solution outcomes was alluded to in some of the earliest references to "bloat" (i.e., [3, 4]). For example, Altenberg (in [3]) suggested "[P]rogram bloating with junk code may simply be driven by the recombination operator, a neutral diffusion process with upward drift." He further stated, "[P]rograms might be expected to bloat because of the asymmetry between addition and deletion of code at the lower boundary of program size." Although he did not directly implicate tree structures, Altenberg was probably the first to articulate the idea that there might be a neutral, random diffusive process that occurs regardless of whatever selection pressure there might be on content.

Altenberg's conjecture was independently picked up two years later by Soule and Foster. In a series of papers [5-8], they described bloating as a pervasive GP phenomenon that happens in a number of problems. While they did not go so far as to say the phenomenon is caused by a neutral process, they did propose a "fix" that was largely structural (i.e., invoke a penalty on solution size). Soule and Foster's work helped to initiate compelling theories that explain phenomena that occur across a broad variety of problems encountered in GP (e.g., [9, 10]).

Soule and Foster eventually teamed with Langdon and Poli in [11]. Together they furnished the linkage between the generation of random trees and bloating. In later work [10, 12], Langdon further elaborated on this linkage and hypothesized that the distribution of solution shapes follows that of the theoretically derived distribution of tree shapes. He posited that GP solutions are a result of a random walk on the "surface" of the distribution of tree shapes in the search space defined by tree size and depth.

Whether the distribution of tree shapes has sufficient explanatory power to account for problem difficulty is discussed later in this paper.

There are two other noteworthy investigations.

Goldberg and O'Reilly [13, 14] have isolated and examined how program dependencies may affect GP tree structures. Their work helped to establish a link between the form of a solution and the function that this solution is to embody.

Punch et al. published a paper describing a tunably difficult problem that they deemed the *Royal Tree* [15]. The *Royal Tree* problem was one of the earliest instances of borrowing the *Royal Road* function, which had been used in the GA community for some time. It is also significant in that it was the first instance of a tunably difficult problem that was structurally based. Its program dependencies were arguably straightforward and easily quantifiable. The solutions to the problem were easy to visualize. What was surprising was that it did not take much to turn the *Royal Tree* problem from a "hard" problem into an "impossible" one, in spite of the significant computational resources that were used. For the most part, however, the community has apparently dismissed the problem as anomalous – "real" coding solutions do not appear like *Royal Tree* programs; it's "obvious" that the *Royal Tree* problem is hard because the solutions need to be full trees; and so on.

2.2 Theoretical Context of This Paper

With the exception of the *Royal Tree*, nearly all of the previous work presumes a role of content in contributing toward the dynamics observed in GP. Content is presumed to be a necessary condition. In contrast, the theoretical context of this paper does not presume this. Rather, our hypothesis examines dynamics that can be brought about by considering just structural causes. If there is a presumption, it would be this: that much of the dynamics driven by structure would occur at a lower level than dynamics driven by content.

Although we describe our hypothesis in other papers [1, 2], it is worthwhile to summarize their findings here.

- The iterative growth of trees in standard GP is analogous to a physical process of diffusion-limited aggregation (also known as ballistic accretion).
- Objects that are created by diffusion-limited aggregation have a characteristic density (i.e., because they are a kind of fractal).
- If the iterative growth of trees in standard GP is analogous to diffusion-limited aggregation, it is possible that trees also have a characteristic "density" or range of "densities" (which can be measured by tree sizes and depths).
- We can create a model of iterative tree growth to map out which structures are likely.
- The model predicts for four regions in the space of number of nodes and tree depth: I (easy), II (transitional), III (hard), IV (out-of-bounds). A map of these regions is shown in Fig. 1. This particular map presumes GP with an arity-2 function set.

One can think of these regions as a kind of "bottle." Region I would denote the interior of this "bottle." Most GP trees would be found in here. Region II would denote the walls of this "bottle." Fewer trees would be found here. Region III would denote the outside of this "bottle." Even fewer trees would be found here. Region IV represents the out-of-bounds, since these regions are impossible for an arity-2 tree to exist.

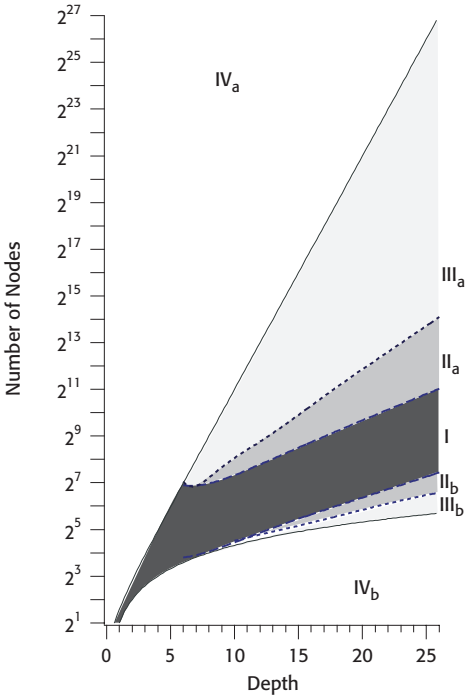

Fig. 1. Predicted regions of difficulty (from [2])

Note that the model itself focuses only on the role of tree structure on determining problem difficulty. We do *not* imply that structure alone accounts for all the phenomena observed in determining problem difficulty. Rather, the model was designed with the intent of isolating the effect of structure away from content. We view Regions I–IV, then, as representing limiting conditions within GP search space. Evidence presented in [1,2] supports the notion that content-driven mechanisms (which also determine problem difficulty) occur within these limiting conditions.

Note, too, that this type of iterative random growth does result in a substantially different "surface" than would be suggested by the distribution of binary tree shapes, as suggested in [10, 16].

Finally, we note that although this surface is different, our reasoning has drawn heavily from those works mentioned in Sect. 2.1. In a sense, we show what happens when one extends Langdon's model by including consequences that stem from iterative random growth.

2.3 Motivation for a New Test Problem

A partial verification of our hypothesis would include a comparison with GP results on problems with an arity-2 function set. (A limited comparison was given in [1, 2].) However, known problems – like *multiplexer* or *sextic* – do not account just for structural effects. Known problems inherently involve content or syntax. Even the structure-based *Royal Tree* problem (see [15]) involves some knowledge of node syntax, if only to properly compute a fitness score.

There is no known problem in GP that isolates information structure away from information content. Furthermore, there is no known problem that tests for variations in problem difficulty due only to variations in information structure.

Given that the hypothesis has predicted for significant variations in problem difficulty that would arise just from information structure, it has been incumbent upon us to show that such a problem exists. The remainder of this paper describes such a problem.

3 The *Lid* Problem

The objective of the *Lid* problem is to derive a tree of a given size and depth. If our hypothesis is correct, the *Lid* problem should be tunably difficult simply by specifying particular sizes and depths. This would occur because trees of certain sizes and depths would be more likely than others.

Given this fairly broad objective, there are a number of variants of the *Lid* problem that can be described. However, for the purposes of this paper, we describe one variant that should naturally "fit" standard versions of GP. The described variant is a depth-first version, which means that GP would need to solve for the depth of a target tree first before solving for the number of nodes of that target tree. This ordering suits standard GP, since meeting a depth criterion has been a relatively easy task.

We leave for future work descriptions and results from other variants. As far as we have been able to determine, however, all variants have displayed tunable difficulty.

3.1 Problem Specification

As in the *ORDER* and *MAJORITY* problems [13, 14], the function set for the *Lid* problem is the single primitive function JOIN {*J*}, which is of arity-2. As in the *Royal Tree* problem [15], the terminal set for the *Lid* problem is the single primitive {*X*}.

The *Lid* problem requires the specification of a target tree depth (i.e., d_{target}) and a target number of tree terminals (i.e., t_{target}). We assume that *Lid* trees are rooted, and that the convention for measuring depth presumes that the node at the root lies at null (0) depth.

How an arbitrary *Lid* tree measures against its target specifications is given by the following metrics for depth and terminals[1] – Eqs. (1) and (2), respectively.

$$metric_{depth} = W_{depth}\left(1 - \left(\frac{\left|d_{target} - d_{actual}\right|}{d_{target}}\right)\right) \tag{1}$$

$$metric_{terminals} = \begin{cases} W_{terminals}\left(1 - \left(\frac{\left|t_{target} - t_{actual}\right|}{t_{target}}\right)\right) & \text{if } metric_{depth} = W_{depth} \\ 0, & otherwise \end{cases} \tag{2}$$

where d_{actual} and t_{actual} correspond to the depth and number of terminals for the measured *Lid* tree, respectively. Weights W_{depth} and $W_{terminals}$, are chosen arbitrarily such that

$$W \equiv W_{depth} + W_{terminals} = 100. \tag{3}$$

[1] There are several variants of these metrics. For example, one could have GP solve for terminals first, instead of depth. Considerations of these variants are deferred to future work.

Raw fitness is defined as

$$fitness_{raw} = metric_{depth} + metric_{terminals}. \quad (4)$$

3.2 Tunability Considerations

There are two parameters that are suggested by our hypothesis that can be used for tuning problem difficulty: d_{target} and t_{target}.

Varying t_{target} while fixing d_{target} results in solutions that vary in size, but not in depth. Note that it can be shown that t_{target} is directly related to the size of a target *Lid* tree, i.e.,

$$Size_{target} = 2t_{target} - 1. \quad (5)$$

In this case, the initial combinatorial search space in J and X for each target tree remains invariant across t_{target}, although the size of each specified tree grows exponentially.

Varying d_{target} while fixing t_{target} results in solutions that should vary in depth but not in size. In this case, the combinatorial search space in J and X for each target tree remains invariant across d_{target}. We note, however, that the numbers of trees that satisfy a particular 2-tuple of (d_{target}, t_{target}) will vary as d_{target} varies. For example, there are 64 instances that satisfy (d_{target}, t_{target}) = (7, 8); however, just one instance satisfies (d_{target}, t_{target}) = (3, 8).

4 Experimental Procedure

This paper considers an empirical test of information structure driving problem difficulty. In particular, if the shape of a tree is a major factor in determining problem difficulty, we should be able to detect this by changing the target shape of a tree. We would subsequently note how difficult it is for GP to attain that shape. For this paper, we used a simple measure of problem difficulty. Out of N trials, we noted how many of these trials were ones in which GP successfully derived the target shape.

We subsequently considered the case of taking two one-dimensional slices along the x- and y-axes of the map of theoretically derived regions shown in Fig. 1. In essence, we used the *Lid* problem as a kind of "probe" that "sounds" for the predicted regions in GP search space. We basically set the tunable *Lid* parameters and measured the difficulty at that point. We sampled for changes in difficulty along two slices across these regions. A horizontal slice was determined by varying d_{target} while fixing t_{target}. A vertical slice was determined by varying t_{target} while fixing d_{target}.

We used a modified version of lilgp [17] as was used in [18]. Most of the modifications were for bug fixes and for the replacement of the random number generator with the Mersenne Twister [19]. We configured lilgp to run as a single thread.

Most of the GP parameters were identical to those mentioned in Chapter 7 [20]. 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 fitness-proportionate selection. Other parameter values were maximum generations = 200 and the maximum depth = 512 (assuming a root node to be at depth

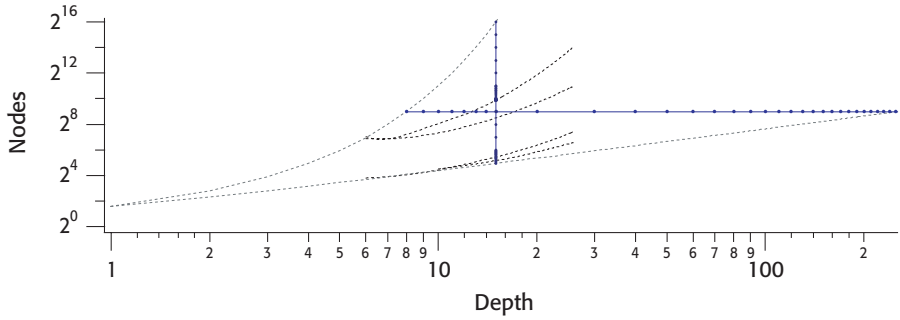

Fig. 2. Sampling of search space using *Lid*

= 0). Note that as in stock lilgp, crossover for this experiment was biased towards the internal nodes of an individual (i.e., 90% internal, 10% terminals).

The parameters of the *Lid* problem were as follows: W_{depth} = 30 and $W_{terminals}$ = 70. Although sensitivity studies of varying W_{depth} and $W_{terminals}$ have been conducted, the results of these studies have been set aside for a future paper. Those studies have suggested that while varying the *Lid* parameters can influence results, those results are still consistent with the findings of this paper.

Figure 2 shows both slices superimposed over the map of theoretically derived regions. Lines indicate the locations of the slices. Filled circles indicate the locations of the sample taken with *Lid*. For the horizontal cut, the target number of terminals t_{target} was fixed at 256 (which corresponds to a tree that is 511 nodes large), while target depth d_{target} was varied. For the vertical cut, the target depth d_{target} was fixed at 15, while the target number of terminals t_{target} was varied. Sampling along either cut was higher where the rate of change in measured difficulty was higher.

Note that for clarity in Figure 2, the *x*-axis has been changed from a linear scale to a logarithmic one. Note also that the map of regions stops at depth 26. For depths greater than 26, model results have been left pending for a future paper.

For each 2-tuple of (d_{target}, t_{target}), a total of 1,000 trials were run. Consequently, there were 32 data sets collected of 1,000 trials apiece for the horizontal cut. There were 60 data sets of the same number of trials for the vertical cut. A total of 92,000 trials were completed. Trials were run on Linux and Sun UNIX workstations.

5 Results

Horizontal Cut

Figure 3 shows the results from the horizontal cut. This figure shows two graphs that share target depth d_{target} as a common abscissa. Each curve represents the ensemble performance of a total of 16 billion individuals that were distributed among 1,000 trials in 32 data sets. The sampling of these individuals was such that only the best individual per trial was depicted.

The domain of these graphs is the span of allowable depths for a binary tree (which are also *Lid* individuals) for a specified number of terminals. For the target terminal specification t_{target}= 256, the minimum allowable depth is 8 (a full tree); the maximum allowable depth, 255 (a sparse tree).

The top graph represents the success rate of identifying a perfect individual in a trial per 1,000 trials as a function of target depth d_{target}, which serves as the horizontal cut's tuning parameter. This plot is analogous to a curve of tunable difficulty as was shown in [21, 22]. The harder a problem is, the lower the success rate.

The bottom graph depicts a scatter plot of the size (i.e., number of nodes) of an individual as a function of target depth d_{target}. As with the previous plots, there are 32,000 data points shown, with 1,000 points per target depth d_{target}. For the horizontal cut discussed in this paper, a size of 511 nodes represents a successful trial. A line delineates this size for this graph. When all 1,000 points for a data set have the same number of nodes, those points are delineated with an 'x.' The 'x' markers appear for d_{target} = 12, 13, 14, 15, 20, 30, 40, 50, which means that *all* of the trials at those depths solved for the correct solution. This scatter plot subsequently provides a measure of how close individuals were to their corresponding target. (As it turned out, no trial in either horizontal or vertical cuts failed to reach its corresponding target depth. This means, then, that target depth and actual depth can be treated as the same.) The harder a problem is, the further away the sizes are from the target size of 511 nodes.

Vertical Cut

Figure 4 shows the results from the vertical cut. This figure shows two graphs that share target size (nodes) as a common abscissa. Note that target size (Eq. 5) was used instead of target terminals t_{target} so that the results can be compared to Fig. 1 (which has an ordinate in nodes, as opposed to terminals). Each curve represents the ensemble performance of a total of 30 billion individuals that were distributed among 1,000 trials in 60 data sets. The sampling of these individuals was such that only the best individual per trial is depicted.

The domain of these graphs is the span of allowable sizes for a binary tree (which are also *Lid* individuals) for a specified depth. For the target depth specification d_{target} = 15, the minimum allowable size is 31 (a sparse tree); the maximum allowable size, 65,535 (a full tree).

As in the horizontal cut, the top graph represents the success rate of identifying a perfect individual in a trial per 1,000 trials as a function of target size, which serves as the vertical cut's tuning parameter.

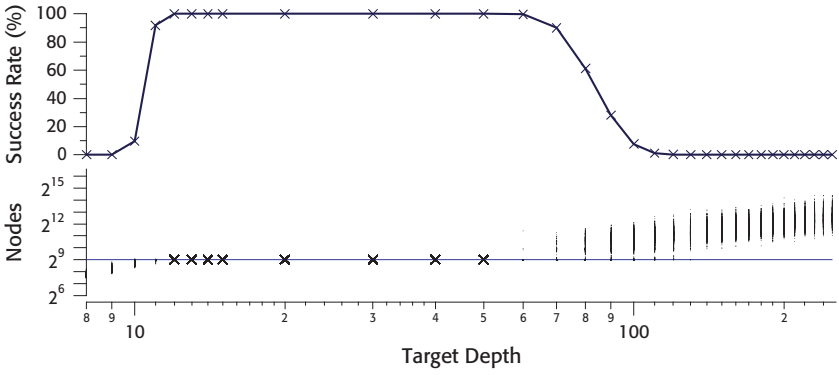

Fig. 3. Horizontal cut results

The bottom graph depicts a scatter plot of the size (i.e., number of nodes) of an individual as a function of target size. There are 60,000 data points shown, with 1,000 points per target size. For the vertical cut discussed in this paper, a successful trial is represented when the target size equals the measured size. A 45° line delineates this condition for this graph. When all 1,000 points for a data set have resulted in success, those points are delineated with a '+.' The '+' markers appear for target size = 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 127, 255, 511, 915, which means that *all* of the trials at those sizes solved for the correct solution. This scatter plot subsequently provides a measure of how close individuals were to their corresponding target. The harder a problem is, the further away the measured sizes are from the 45° line.

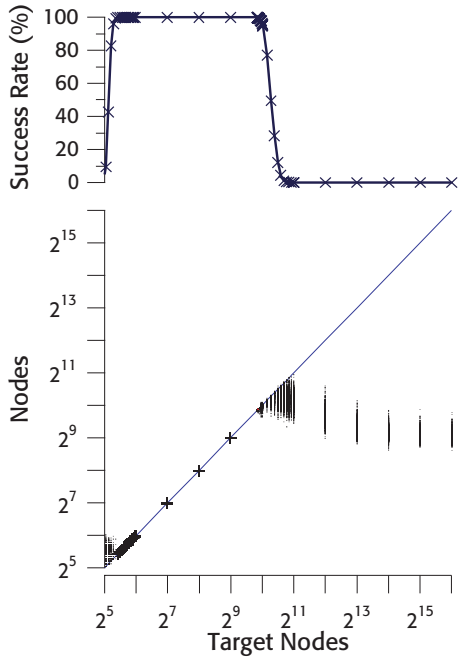

Fig. 4. Vertical cut results

6 Discussion

6.1 Regions

Observation: The empirical results support our predicted outcome of four regions of problem difficulty.

The curves for success rate for Figs. 3 and 4 indicate the existence of three regions of problem difficulty (with the fourth – "out-of-bounds" – as being impossible to achieve with binary trees). Presented in order of increasing difficulty, these regions are:

Region I. ("Easy") This region would be characterized by a plateau at or near 100% in the success-rate curve. For the horizontal cut, it measured to a plateau that spans from about depth 12 to 60. For the vertical cut, it measured to a plateau that spans from about 41 to 969 nodes. (Note: as in [1,2], this region was measured by using the upper one-percentile isopleth relative to the absolute maximum of the region, which in this case was 100%.)

Regions II. ("Transitional") These regions (i.e., IIa and IIb) would be characterized by a transition from Region I to Regions III (i.e., IIIa and IIIb). For the horizontal cut, the upper-bound transition was measured from about depth 60 to 110, while the lower-bound transition was measured from about depth 10 to 11. For the vertical cut, the upper-bound transition was measured from about 969 to 1,639 nodes, while the lower-bound transition was from about 31 to 41 nodes.

Regions III. ("Hard") These regions (i.e., IIIa and IIIb) would be practically categorized as those regions that might not be accessible to GP, even though these regions still represent legitimate tree structures. For the horizontal cut, the upper-bound region measured from about depth 110 to the end of the allowable domain (i.e., 255), while the lower-bound region measured from depths 8 to 10. For the vertical cut, the upper-bound transition was measured from about 1,639 nodes to the end of the allowable range (i.e., 65,535 nodes) while the lower-bound transition was not measurable.

Region IV. ("Out-of-Bounds") Region IV consists of those depths that are not attainable by means of a binary tree.

6.2 Contrast with Current Theory

Observation: The frequency distribution of shapes corresponding to binary trees does not have sufficient explanatory power to account for the results.

The current theory (*cf.* [10, 16]) posits that solution size and shape are the results of a random walk on the node-depth surface of the frequency distribution of tree shapes for a given size and depth. The current theory also posits that tree shapes tend to be biased towards growth, particularly in the direction of the most likely shape for a tree of a given size and depth. Finally, the current theory predicts that this random walk would most commonly occur along the "ridgeline" of this surface, with most individuals being found within the upper- and lower-5% boundaries for a given number of terminals.

If this were true, a problem like the *Lid* should show a positive correlation for problem difficulty with this surface. To determine this, we independently derived and recomputed the surface as described in [10] so that those results can be compared to our results (particularly the success-rate curves in Figs. 3 and 4).

The top graph in Fig. 5 shows a view of this surface as depicted in [10]. It shows the "out-of-bounds" boundary for binary trees, the "ridgeline," and the upper- and lower-five percentile boundaries. The top graph represents the common visualization [10, 16] of a binary tree distribution.

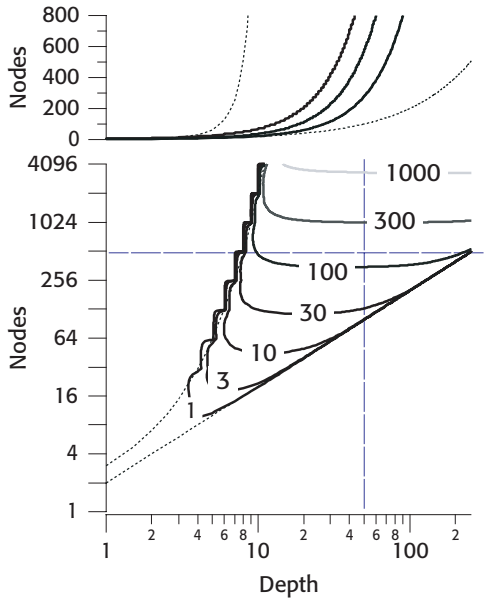

Fig. 5. Two views of the distribution of binary trees

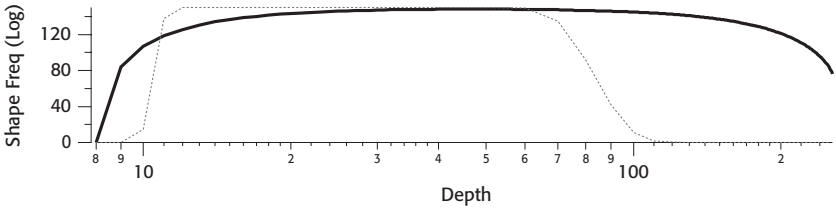

Fig. 6. Distribution of binary trees for 256 terminals

One might guess that the upper- and lower-5% boundaries are analogous to contour lines. They are not. The bottom graph in Fig. 5 shows the same surface, except visualized with the contour lines of constant frequency distribution. The contours are given in $\log_{10}$ quantities. If anything, the surface is dominated by the peak. Just in terms of what is shown in the bottom figure, the peak (located near the upper right corner) is greater than $10^{1,000}$. This value is 700 orders of magnitude greater than most of the surface that has been shown.

Also for Fig. 5, lines have been superimposed on the bottom graph to indicate the location of the horizontal and vertical cuts of this paper.

Figure 6 shows the frequency distribution of trees for the horizontal cut. The corresponding success-rate curves from the *Lid* problem are superimposed in light gray. While Fig. 6 loosely has the shape as shown in the horizontal-cut results (Fig. 3), the locations of the transition zones differ considerably from the empirical data.

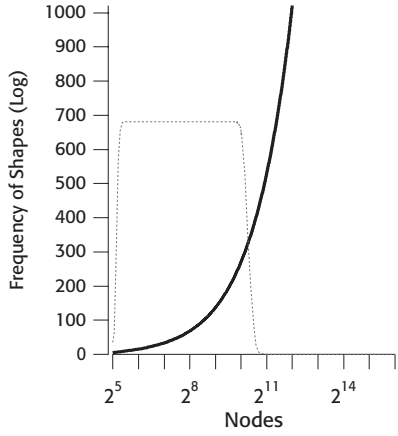

Fig. 7. Distribution of binary trees for depth 15

Figure 7 shows the frequency distribution of trees for the vertical cut. The corresponding success-rate curves from the *Lid* problem are superimposed in light gray. If the current hypothesis is sufficient in explaining the *Lid* results, the frequency distribution curves should track these results. Figure 7 shows that the frequency distribution curve has a significantly different shape than the vertical-cut results (Fig. 4). For the range shown, the frequency of binary tree shapes is exponentially increasing. The rate of that increase is dramatic: we computed the distributions for only a sixteenth of the total span of the cut, but the frequency distribution of trees had surpassed $10^{1,200}$. In other words, the frequency distribution of trees should predict (in this range) that the *Lid* problem would become easier for the vertical cut when the number of target nodes is greater. Our results have shown otherwise.

7 Conclusions

As this paper has shown, tree structure can have a substantial impact in determining problem difficulty. It has provided empirical support for our hypothesis that the itera-

tive random growth of information structures alone can be a significant and limiting factor that determines problem difficulty. As we have described in [1,2], iterative random growth is a result of structural mechanisms that "piggyback" on the operators that generate variation in GP. This paper lends further support for the existence and significance of these structural mechanisms.

In working towards validating our hypothesis, we developed a tunably difficult problem called the *Lid*. It was designed, in part, to serve as a kind of "probe" that "sounds" for the predicted regions in GP search space. Its tunable difficulty stems from a predicted outcome of our theory: that some structures are easier for GP to attain than others. As the results indicate, most structures are difficult for GP.

We have shown that the current hypothesis (*cf.* [16]) does not have sufficient explanatory power to account for the results.

Finally, we note that this paper does *not* say that program dependency, fitness landscapes, epistasis, etc. are invalid. Rather, that context-free mechanisms may also need to be considered when determining what makes a problem GP-hard.

References

[1] J.M. Daida, "Limits to Expression in Genetic Programming: Lattice-Aggregate Modeling," in *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 1. Piscataway: IEEE, 2002, pp. 273–278.

[2] J.M. Daida and A. Hilss, "Identifying Structural Mechanisms in Standard GP," in *GECCO 2003*.

[3] L. Altenberg, "Emergent Phenomena in Genetic Programming," in *Evolutionary Programming – Proceedings*, A.V. Sebald and L.J. Fogal, Eds. Singapore: World Scientific Publishing, 1994, pp. 233–241.

[4] W.A. Tackett, "Recombination, Selection and the Genetic Construction of Computer Programs," Ph.D. Dissertation, *Electrical Engineering*. Los Angeles: University of Southern California, 1994.

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

[6] T. Soule, et al., "Using Genetic Programming to Approximate Maximum Clique," in *GP 1996 Proceedings*, J.R. Koza, et al. Eds. Cambridge: The MIT Press, 1996, pp. 400–405.

[7] 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.

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

[9] W.B. Langdon, "Scaling of Program Fitness Spaces," *Evolutionary Computation*, vol. 7, pp. 399–428, 1999.

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

[11] 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.

[12] W.B. Langdon, "Size Fair and Homologous Tree Genetic Programming Crossover," in *GECCO 1999 Proceeding*, vol. 2, W. Banzhaf, et al., Eds. San Francisco: Morgan Kaufmann Publishers, 1999, pp. 1092–1097.

[13] U.-M. O'Reilly and D.E. Goldberg, "How Fitness Structure Affects Subsolution Acquisition in Genetic Programming," in *GP 1998 Proceedings,* J. R. Koza, et al., Eds. San Francisco: Morgan Kaufmann Publishers, 1998, pp. 269–277.

[14] D. E. Goldberg and U.-M. O'Reilly, "Where Does the Good Stuff Go, and Why?," in *Proceedings of the First European Conference on GP*, W. Banzhaf, et al., Eds. Berlin: Springer-Verlag, 1998.
[15] W. Punch, D. Zongker, and E. Goodman, "The Royal Tree Problem, A Benchmark for Single and Multiple Population Genetic Programming," in *Advances in Genetic Programming*, vol. 2, P. J. Angeline and J. K.E. Kinnear, Eds. Cambridge: The MIT Press, 1996, pp. 299–316.
[16] W. B. Langdon and R. Poli, *Foundations of Genetic Programming*. Berlin: Springer-Verlag, 2002.
[17] D. Zongker and W. Punch, "lilgp," v. 1.02 ed. Lansing: Michigan State University Genetic Algorithms Research and Applications Group, 1995.
[18] J. M. Daida, et al., "Analysis of Single-Node (Building) Blocks in Genetic Programming," in *Advances in Genetic Programming 3*, L. Spector, et al., Eds. Cambridge: The MIT Press, 1999, pp. 217–241.
[19] M. Matsumoto and T. Nishimura, "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudorandom Number Generator," *ACM Transactions on Modeling and Computer Simulation*, vol. 8, pp. 3–30, 1998.
[20] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge: The MIT Press, 1992.
[21] 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.
[22] J. M. Daida, et al., "What Makes a Problem GP-Hard? Analysis of a Tunably Difficult Problem in Genetic Programming," *Journal of GP and Evolvable Hardware*, vol. 2, pp. 165–191, 2001.