Probing for Limits to Building Block Mixing with a Tunably-Difficult Problem for Genetic Programming

Jason M. Daida
The University of Michigan
2455 Hayward Avenue
Ann Arbor, Michigan 48109-2143
1 (734) 647-4581
daida@umich.edu

Michael E. Samples
The University of Michigan
2455 Hayward Avenue
Ann Arbor, Michigan 48109-2143
msamples@umich.edu

Matthew J. Byom
The University of Michigan
2455 Hayward Avenue
Ann Arbor, Michigan 48109-2143
mjbyom@umich.edu

ABSTRACT
This paper describes a tunably-difficult problem for genetic programming (GP) that probes for limits to building block mixing and assembly. The existence of such a problem can be used to garner insight into the dynamics of what happens during the course of a GP run. The results indicate that the amount of mixing is fairly low in comparison to the amount of content that could be present in an initial population.

Categories and Subject Descriptors
I.2.2 [**Artificial Intelligence**]: Automatic Programming – *program synthesis.*

General Terms
Algorithms, Performance, Experimentation, Theory.

Keywords
Tunably-difficult problems, *Highlander* problem, building blocks, initial populations.

1. INTRODUCTION
Are there limits to the degree to which building blocks can be mixed and assembled by genetic programming (GP)? If so, what are they, and what is the nature of those limits?

Intuitively, one would think that such limits do exist. GP cannot arbitrarily pick and choose what it needs in its derivation of a solution. Rather, the selection of content material is constrained by that material's placement in a tree structure. Material that is closer to the terminals of a tree is more likely to be picked than material that is at or closer to a root node. It is unlikely, then, that GP would derive a solution that has been formed from all of the root nodes of individuals that are in a typical population.

GECCO'05, June 25-29, 2005, Washington, DC, USA.

This intuitive understanding has been supported by a number of rigorous studies that have investigated the processes by which GP derives a solution. For example, researchers have shown that GP generally selects and locks in a root node rapidly and before the remaining content of a solution has been determined (e.g., [12, 14, 16, 22]). These findings have been corroborated by a few model problems that exploit the difficulty of dislodging a root node during the course of a GP run (e.g., [10, 13]).

Root nodes and rooted tree schema are not the only constraints in assembling building blocks that researchers have identified. Root nodes and rooted tree schema are parts of tree structures. The fact that content is arranged in trees does have its own implications, which researchers have subsequently investigated. This would include work such as [6, 11, 19]. Subtrees, of course, are the basis of formal definitions of building blocks (e.g., [14, 20]). The nature of putting building blocks together has also been investigated with the development of several other model problems including ORDER and MAJORITY [11, 18, 19], Royal Tree [21], and binomial-3 [5, 9].

Although an intuition of building block mixing limits seems reasonable given these constraints, a test of this intuition is not so easy. On one hand, building blocks are difficult to separate from a problem's domain. There is reason to suspect that classes of problems in one domain (like data modeling) might have intrinsically different building block properties than a class of problems in another domain (like Boolean) (see [1]). It would be reasonable, then, to suspect that limits to building block mixing and assembly would be particular to a domain. On the other hand, there is work that has totally abstracted content away from GP and has focused exclusively on structure [6, 7]. There is reason to suspect that there are indeed constraints to building block mixing and assembly, but it could be argued that those constraints tell us little about the actual mixing and assembly. The link between constraints that are purely the result of structure and those that would come from a typical GP problem are left to be resolved.

This paper articulates a linkage between constraints that could arise from purely structural concerns and those that would arise as a result of a problem's domain. As work by Sastry et al. [23] implies, the interaction between content and structure is highly nonlinear. The purpose of this paper, then, is to describe a model problem that can be used to probe for these interactions.

This paper is organized as follows. Section 2 describes a conjecture about how building blocks and tree structures could

interact. If the conjecture is true, it should be possible to construct a tunably-difficult model that exploits this interaction. Section 3 gives a specification for the proposed model problem, which we have called *Highlander*. Section 4 describes an experiment that tests this problem. Section 5 describes the results of that experiment. Section 6 discusses the results, and Section 7 concludes.

2. CONJECTURE

One rigorous way to answer the questions posed at the outset of this paper would be to work through the schema theorems, which account for the consequences of GP selecting and amplifying ensembles of nodes to build up solutions (e.g., [14, 20]). However, it is possible to simplify matters if one considers a slightly different framing than using these theorems. Instead of framing the questions as to whether GP can select and assemble ensembles of nodes, we frame the questions as to whether GP can transport nodes that it would need to a solution. We understand that having the needed nodes in a tree is not equivalent to having those same nodes arranged as schema. Nevertheless, the issue of whether those nodes exist in a tree is a limiting condition to whether schema with those nodes would exist.

This framing allows us to re-express the question of whether there are limits to the degree to which building blocks can be mixed and assembled in the following manner: Given an initial population P_0, let every node in this population be uniquely identified to form a set V_0. What is the maximal fraction that can be instantiated by a single GP individual (i.e., the maximum ratio of the number of elements from V_0 in a tree relative to the total number of elements in V_0)?

At first glance, the answer could approach "1," because GP uses a variable-length structure. Given enough time, there should be enough recombination and trees should be large enough such that an individual may consist of every single node in V_0. Figure 1 suggests otherwise.

Figure 1 suggests that there are several competing processes that would affect the transport of initial population nodes to individuals in subsequent generations. These processes are:

Rate of Mixing (recombination). There is a finite rate at which elements of V_0 can be mixed, since elements of V_0 are grouped as individuals in P_0 and individuals are recombined two at a time.

Rate of Individual Loss (selection). It is known that certain methods such as tournament selection can result in a significant fraction of individuals in one generation failing to recombine and to contribute to the formation of individuals in the next generation (e.g., [17]). As a result, elements in V_0 can be permanently lost.

Rate of Aggregation (structure). There has been work that indicates that tree structures grow in a particular manner and are subsequently constrained to certain ranges in shape [6]. A side consequence is that not all nodes are accessible for recombination even if they were still present in a population. As a result, elements in V_0 can be practically lost.

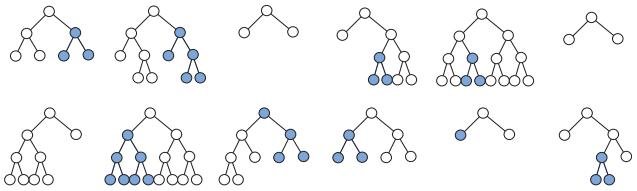

Figure 1. Example of nodes used and others lost in an initial population. Dark (blue) nodes are those that would be used in the creation of individuals for the next generation.

If these and perhaps other competing processes exist in a GP run, it should be possible to exploit differences in these rates to derive a tunably-difficult problem. The next section describes how this could be done.

3. THE HIGHLANDER PROBLEM

The overall objective of our proposed test problem is to identify the maximal fraction of V_0 that can be instantiated by a single GP individual. We can do so by having GP solve for a *specified* fraction β and then by using this specification as a tuning parameter to determine at which fraction GP fails to solve. Section 3.1 provides a qualitative description of how this would work, while Section 3.2 describes the problem's specifications. Section 3.3 details the tunability of the proposed problem.

3.1 Problem Description

Figure 2 illustrates how a problem like *Highlander* would work. For the purposes of this illustration, the initial population is very small. If all of the nodes were counted in this population, there would be just 10 nodes. Each node would be labeled with a unique number. To test whether GP would be able to transport nodes in an initial population to an individual, we would need to specify a percentage of an initial population to appear in a solution. If building an individual with the requested material can be done before material is lost either to structure or selection, GP should be able to build an individual with the requested percentage of initial population nodes. Individuals would subsequently be scored on the basis of how many nodes with unique labels would exist in a tree. The individual tree shown in Figure 2 is consequently an example of a solution for a specified percentage of 30%.

3.2 Problem Specification

As in *ORDER* and *MAJORITY* [11, 19], the function set for *Highlander* is the single primitive function JOIN { J }, which is of arity-2. As in *Royal Tree* [21], the terminal set is the single primitive { X }. As in *Lid* [7], the semantics of { J } and { X } are not a factor and serve only as placeholders, since the corresponding program to each tree is not executed for evaluation. For this paper, we presume only crossover.

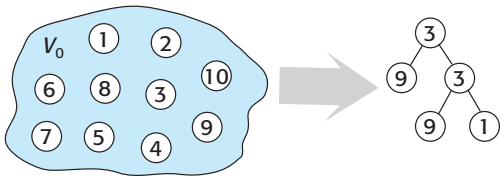

Figure 2. Example of how *Highlander* would work.

The *Highlander* problem requires the specification of a target β:

$$\beta \equiv \frac{\mathrm{N}_{desired}(T)}{\mathrm{N}(V_0)}, \qquad (1)$$

where N(A) denotes the number of *uniquely* labeled elements in a set A; $\mathrm{N}_{desired}(T)$ is a specified number of uniquely labeled nodes that belong to a set T; set T corresponds to the ordered set of nodes that belong to an individual tree; set V_0 consists of all the nodes that make up the initial population, whereby each element in V_0 is presumed to be labeled with a unique ID.

The ID labeling scheme is similar to that employed by McPhee and Hopper [16]. Their scheme called for tagging each node in the initial population with integer label pairs (`ID:memID`). The `ID` part of their label is assigned just once at population initialization and consists of an integer that is unique to a node relative to the set of nodes that make up the initial population (i.e., V_0). One can think of `ID` as a serial number that can be used to track individual nodes. The `memID` part of their label can change from generation to generation and is used for providing an audit trail for subtree memberships. In our case, we use just the `ID` portion to compute N(T).

Raw fitness is defined as

$$fitness_{raw}(T) = \left| \beta - \frac{\mathrm{N}(T)}{\mathrm{N}(V_0)} \right|. \qquad (2)$$

For example, if β=1, it means that for an individual tree to score perfectly (i.e., raw fitness = 0), a tree would need to minimally consist of all nodes in V_0. The size (i.e., number of nodes) of that tree can exceed that of N(V_0), since there can be repeated instances of an element in V_0 that appear in that tree.

3.3 Tunability Considerations

In theory, different settings for β should result in *Highlander* being easier or harder to solve for a given set of GP parameters. One can use a fairly simple metric to characterize problem difficulty: the percentage of successful runs that produce at least one "perfect" individual (i.e., has a raw fitness score of zero). Easier problems would have higher percentages of successful runs.

Although GP parameters are not directly involved in the tuning, parameters that correspond to population initialization can be significant factors. For example, the size of a population (i.e., M) determines in part the size and membership of V_0. Likewise, the initial population parameters (e.g., method of population initialization, range of depths considered in population initialization, etc.) can also determine the size and membership of V_0.

Other GP parameters can also influence results. For example, method of selection and selection parameters can affect the rate of individual loss. Parameters affecting internal node bias and recombination can directly influence recombination rates and indirectly influence aggregation rates.

Table 1 . Parameter settings

Parameter	Setting
Run Objective	Evolve an individual that contains a specified fraction β of V_0
Terminal Set	X
Function Set	J
Selection	Tournament q=7 or Proportionate
Population Size M	[50, 100, 500, 1 000, 1500]
Initialization Method	Ramped Half-and-Half
Initialization Depths	2Š6 Levels
Max Generations G	50
Maximum Depth	2048
Internal Node Bias	90% internal, 10% terminals
Termination Criteria	Run reaches G
Highlander β (in %)	See text
Success Predicate	Fitness is within 5% of specified β

4. Experimental Procedure

We investigated the tunability of the proposed problem using several common parameter configurations for GP. Table 1 lists the parameter settings considered in this experiment.

We used a modified version of lilgp [25] similar to that used in [8]. Most of the modifications were for bug fixes and for the replacement of the random number generator with the Mersenne Twister [15]. Other significant modifications included augmenting the data structure associated with each node to include an integer `ID` that serves as that node's serial number. Each `ID` is unique to a node and is generated once during population initialization. We configured lilgp to run as a single thread.

Table 1 describes the ten different experimental configurations that we used, since we considered two different selection methods and five different population sizes.

Difficulty was measured as the percentage of successful runs that produced at least one "perfect" individual and to a precision of three decimal places. This translated to 1000 trials per β value per configuration. Each configuration was adaptively sampled over the range of β (0%, 100%] (i.e., more samples were taken in rapidly varying regions than in slowly varying ones). A total of 238 data points were taken, which corresponds to 238,000 trials that were conducted for all ten configurations.

Trials were run on a Linux grid, which consists of 67 nodes of dual processor Athlon MP CPUs. The results correspond to approximately two CPU-years of computation.

5. Results

Figure 3 shows the difficulty curves corresponding to the ten different configurations of this experiment. There are five sets of plots shown in Figure 2, with each plot corresponding to a particular population size M. Each plot includes two difficulty curves, which correspond to the configurations for tournament selection and fitness proportionate selection at that particular population size. Problem difficulty was measured as the percentage of the number of trials that resulted in a "perfect" individual. (*Each* data point in a curve represents the computed percentage of successful trials out of 1000 trials.) Note that higher values of β generally correspond to more difficult variations of *Highlander* for GP to solve.

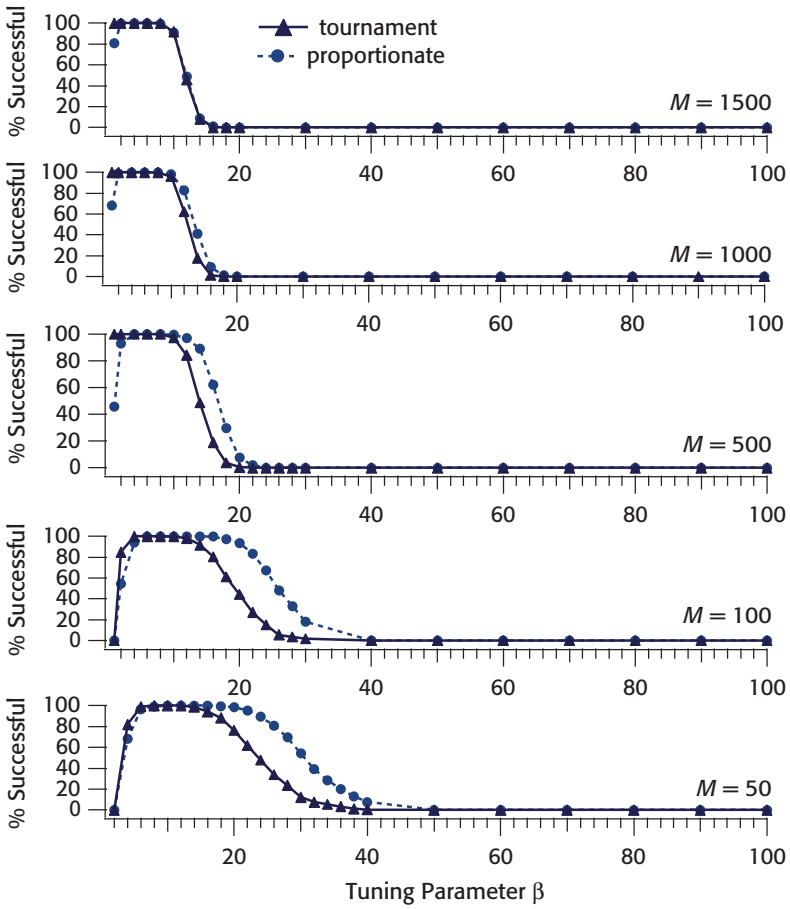

Figure 3. Difficulty curves for *Highlander*. Each plot corresponds to a population size and depicts the percentage of successful trials as a function of tuning parameter β. Parameter β values that correspond to easier settings have higher percentages of successful trials.

Since tree structure has been implicated as a contributing factor to the amount of material that a GP solution can or cannot have, Figures 4 and 5 show a subset of the results shown in Figure 3 as structural plots, which were introduced in [24]. One scatter plot summarizes the size versus depth results that correspond to one data point on a difficulty curve for $M = 500$. Although there are 23 data points depicted per curve for $M = 500$, only 10 data points are shown per curve as scatter plots in Figures 4 and 5. (One can presume that the depicted scatter plots are representative of what happens in size and structure as β is varied). Each point in a scatter plot corresponds to one trial—the measurement at that point specifically corresponds to the best-of-trial solution for that particular trial.

For comparison, we also included two sets of boundaries to accompany each scatterplot. The light (gray) lines indicate the maximum extent for size and depth that are achievable using complete binary trees (i.e., using a function set that consists exclusively of functions of arity-2). The dark (blue) lines indicate the theoretical boundaries of the size and depth ranges for typical trees (see [6]). Those boundaries were not computed beyond depth 26, which is the reason why those boundaries seem truncated in both Figures 4 and 5.

It is significant that in both Figures 4 and 5, that there is *not* much difference between each plot once β has been increased beyond a certain value. The implications of this are discussed in the next section.

6. Discussion

There are four observations that can be made about the results given in Section 5. These include the following:

- The Highlander Problem is tunably difficult.
- There is evidence for the existence of an upper bound to the amount of initial population material that can be expressed in a single individual.
- Structural processes played a significant limiting role in the Highlander solutions that GP identified.
- The existence of an attractor in size and depth space suggests that the material that GP uses is not dispersed throughout an initial population, but is instead concentrated in a subset of individuals in that population

These observations are discussed in Sections 6.1 to 6.4, respectively.

6.1 Tunability

Observation: *The Highlander Problem is tunably difficult.*

The results clearly demonstrate that the *Highlander* problem is tunably difficult as a function of parameter β. In particular, there were three general kinds of regions: an "easy" region where the likelihood of being able to solve for the problem was high; a

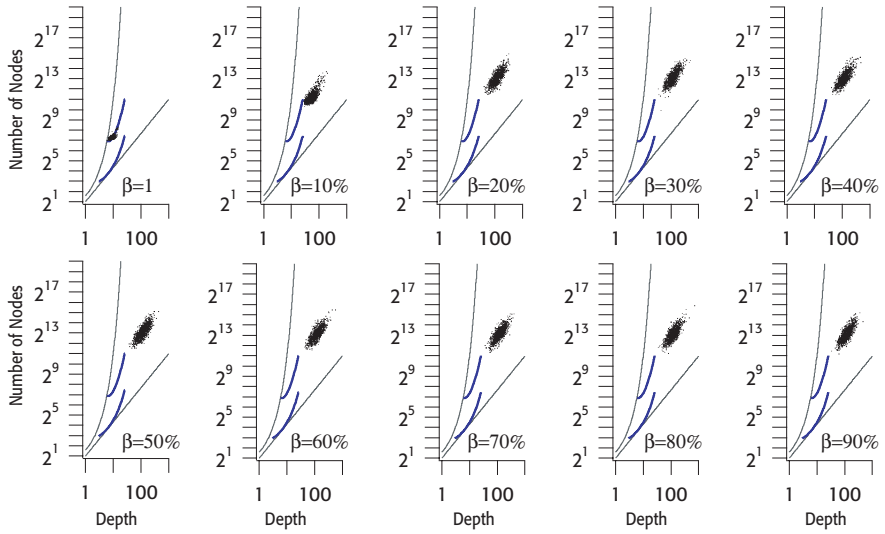

Figure 4. Size versus Depth scatter plots for Tournament Selection, Population Size M = 500. For clarity, only scatter plots for selected values of β are shown.

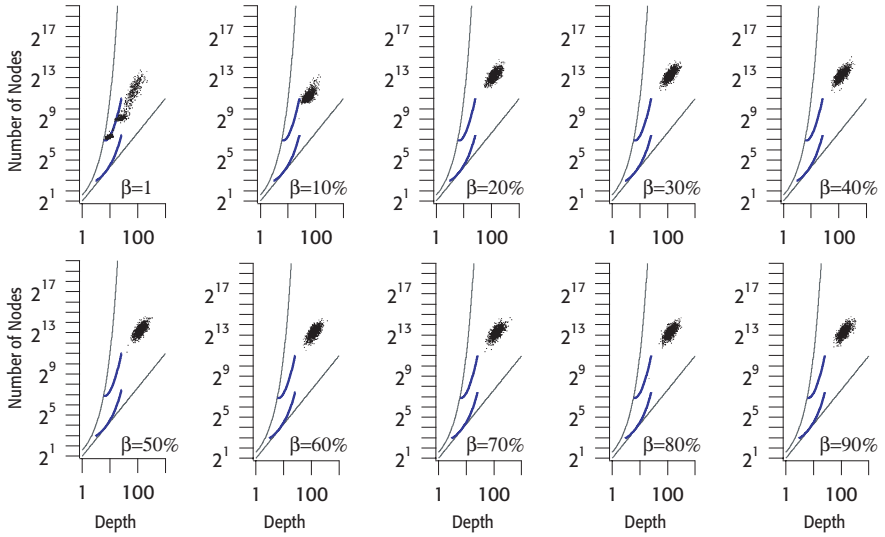

Figure 5. Size versus Depth scatter plots for Fitness Proportionate Selection, Population Size M = 500. For clarity, only scatter plots for selected values of β are shown.

"hard" region where the likelihood of being able to solve for the problem was low; and transition regions that were either monotonically increasing or decreasing.

The results also indicate that the dynamic range for problem difficulty is significant. The measured dynamic range between "easy" and "hard" regions is at least three orders of magnitude: for "easy" regions, 1000 trials out of 1000 trials resulted in a "perfect" solution; for "hard" regions, 0 trials out of 1000 trials resulted in a "perfect" solution. This range could be greater than three orders of magnitude: the exact range is unknown since difficulty was measured to only three digits of precision.

What were surprising was that the "easy" region corresponded to a fairly small range of values for β and that this range of values corresponded to low fractions of the total number of possible initial population nodes that can be found in a single individual. For example, for population size M = 500, the range that defines the "easy" region is approximately [2%, 12%] (under proportionate selection. That range is even smaller for population size M = 1500 (i.e., approximately [2%, 8%]).

The implication is that GP was able to assemble just a small portion of the total number of possible initial population nodes into a single individual. For example, for a population size of M = 1000, only about 12% of all nodes that were created during ramped half-and-half (depths 2–6) initialization can "easily" be transported and assembled into a single individual. Under these same run parameters, it would be very difficult to assemble individuals if those individuals required more than 20% of all of the initial population nodes.

We understand that for many GP problems, these limited ranges may be sufficient. There is a significant amount of redundancy implicit in the expression of function and terminal sets in an intial population. Nevertheless, this range of values does represent a

limiting condition to the amount of mixing that can be done concerning building blocks. If critical ensembles of nodes are needed but exist outside this range, it will likely not be possible for GP to derive a solution.

We note that *Highlander*'s property of tunable difficulty has markedly different behaviors in comparison with two other problems, *binomial-3* [5, 9] and *Lid* [7]. For these problems, generally became much easier to solve when using tournament selection $q = 7$ than proportionate selection. For *Highlander*, the problem remained just as, if not more, difficult to solve under tournament selection than under proportionate selection.

6.2 Upper Bound to Transport

Observation: *There is evidence for the existence of an upper bound to the amount of initial population material that can be expressed in a single individual.*

The results suggest both an upper and a lower bound to the amount of initial population material (i.e., V_0) that can appear in any one GP individual. Of these two bounds, the lower bound is the least surprising, since Equation 1 approaches zero when the amount of material from the initial population (e.g., a large initial population) approaches infinity.

However, the existence of an upper bound has not been anticipated by current theory [e.g., [14]]. Although one could suspect that it would be next to impossible to have just one individual contain all of the nodes that were present in the initial population, it is not intuitive that an upper bound would manifest just after a β of 10% for either selection method for a population size as modest as 500.

The existence of an upper bound does support our conjecture of competing rates. As mentioned in Section 2, there may be at least three rates involved: rate of mixing (recombination), rate of aggregation (structure) and rate of individual loss (selection). The first two rates would contribute positively towards assembling an individual, while the third would contribute negatively. For example, if a solution required that a certain percentage V_0 be used, GP would generally have to mix and aggregate it from individuals in a population before elements of V_0 are lost because of selection.

6.3 Structural Reasons for Tunability

Observation: *Structural processes played a significant limiting role in the Highlander solutions that GP identified.*

Although theoretical results have not been computed at the depths to which many of the *Highlander* solutions were measured, the location of all *Highlander* solutions in size and depth space is consistent with observations that were made in [4] concerning structural mechanisms and constraints.

A summary of our hypothesis is as follows:

- The iterative growth of tree structures 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, trees may 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.
- Our model predicts four regions in the space of number of nodes and tree depth: I (easy), II (transitional), III (hard), and IV (out-of-bounds). A map of these regions is shown in Figure 6. This particular map presumes GP with an arity-2 function set.

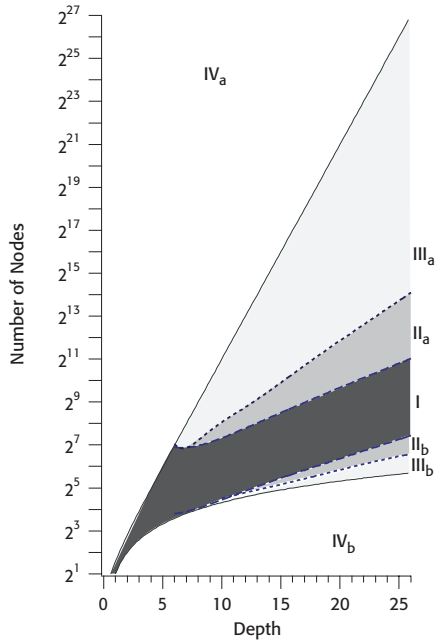

Figure 6. Predicted Regions of Difficulty. Because tree growth is constrained by structure, most typical binary trees that are created by GP fall into Region I.

The model results have been compared to several different problems of known difficulty [2, 4, 6], including *quintic*, *sextic*, *6-input multiplexer*, *11-input multiplexer*, and *binomial-3*. In spite of the varying levels of difficulty, nearly all (better than 90%) of the results fall within Region I. The results are significant, in part because Region I accounts only for about 0.005% of the entire allowable search space in size and number of nodes between depths 0–26. (Region I's area looks larger in Figure 1 because the *y*-axis is scaled as log 2.)

In that work, it was noted that in spite of problem difficulty, the trajectory of search was constrained to the most likely shapes that aggregation yields. Such a trend is consistent with what is shown in Figures 4 and 5. It is notable that regardless of the wide latitude given for the kinds of solutions that were allowable for the Highlander, the size and depth characteristics for the derived solutions fell within a relatively narrow range.

6.4 Location of Potential Building Blocks in an Initial Population

Observation: *The existence of an attractor in size and depth space suggests that the material that GP uses is not dispersed*

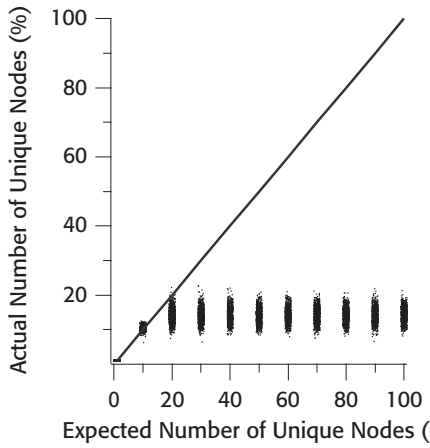

Figure 7. Numbers of distinct elements of V_0 in a GP solution as a function of β. Population size M = 500, tournament selection.

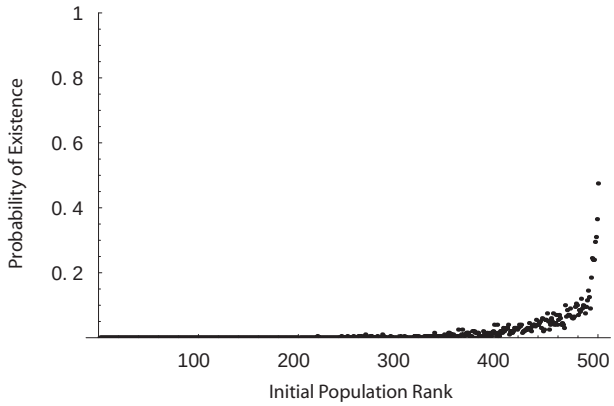

Figure 8. Computed likelihood of a particular initial population individual appearing in the generation 200. Higher ranks correspond to higher fitness. Results are for the *Binomial-3* problem and reported in [3].

throughout an initial population, but is instead concentrated in a subset of individuals in that population.

The existence of two competing rates—rate of mixing and rate of individual loss—could conceivably be enough to account for an upper bound to diversity in the difficulty curves. In other words, the speed at which V_0 can be distributed and mixed between individuals is in tension with the speed at which individuals are lost (i.e., failing to recombine or to replicate) from a population. An equilibrium point would be where the remainder of V_0 is evenly distributed among every individual in population. At that point, the loss of an individual would not result in a loss from V_0. An upper bound to the amount of initial population material would be the result of specifying β such that the number of elements of V_0 remaining in a population would exceed that which is possible at equilibrium.

The problem with this line of thinking is that it does not account for what appears to be an attractor that exists in the space defined by size and depth—both of which are structural metrics. This attractor appears in Figures 4 and 5 at values of β, which correspond to 0.0% success rate. The position of this attractor did not move once a particular rate in β was reached. There are similar attractors that were found at other population sizes.

If the elements of V_0 that were used by GP solutions were dispersed widely throughout an initial population, one would expect solution size and depth to vary as β varies. After all, the minimum sized solutions to the *Highlander* problem increases as β increases. Instead, there seems to be an attractor that remains invariant in size and depth space regardless of β.

Insight into the nature of this attractor might be gleaned by examining the numbers of distinct elements of V_0 in a GP solution as a function of β. (The tuning parameter β can be thought of as the *desired* number of distinct elements of V_0 in a GP solution that is normalized by the total number of distinct elements in V_0.) Figure 7 shows this plot for the tournament selection case (for the data shown in Figure 4). When *Highlander* builds a successful solution, points of desired and measured numbers should fall on the 45° diagonal. However, the average number of distinct elements of V_0 remained constant after β = 20%—about 15%.

As it turns out, this value of 15% is consistent with findings in [3]. That paper focused on tracking the number of initial population individuals that were represented at the end of 200 generations for a completely different problem, but using similar GP parameters as those used to generate the results given in Figure 7. As that paper turned out, less than 20% of an initial population's individuals were represented at a time when most of a population should consist of elements of a GP solution. (e.g., see Figure 8.) Although the number of unique nodes in a *Highlander* solution indicates nothing of the number of initial population individuals and although the number of represented initial population individuals indicates only roughly the number of unique initial population nodes, the results suggest that similar phenomena may be involved.

7. Conclusions

At the outset of this paper, two questions were posed: Are there limits to the degree to which building blocks can be mixed and assembled by GP? If so, what are they, and what is the nature of those limits?

To the first question, the answer is "yes." In answering this question, we devised a model problem—called *Highlander*—that serves as a probe to explore for these limits. It was specifically designed to be tunably difficult, which would serve to indicate the boundaries of when GP could and could not mix and assemble building blocks. It was specifically designed to exploit various processes in GP that are most likely to produce this phenomenon of tunable difficulty. As a result, this tunable problem has a heritage that is apart from those found in genetic algorithms, and for that matter, in other branches of genetic and evolution programming.

We tested this model's property of tunable difficulty in one of the larger computational experiments in the field. A total of 238,000 trials were evaluated on a Linux grid to measure the behavior of this problem as a function of population size, selection method, and its tuning parameter. The results indicated, with fairly common choices for population size and selection method, that the degree to which building blocks can be assembled is fairly low. In particular, the results showed that only 2% to 18% of all nodes that were created during population initialization would ever make it into a solution.

We acknowledge that this range may be sufficient for many problems in GP because there is a significant amount of

redundancy that is implicit with the expression of function and terminal sets. Nevertheless, this range of values does represent a limiting condition to the amount of mixing that can be done concerning building blocks.

To the second question, the answer is "the limits exist because there are competing processes within a GP process." We have speculated that there are three main processes. The first process allows for the mixing and assembly of building blocks (recombination). The second process potentially removes materials that could be used in the assembly of these blocks (selection). The third constrains the way these building blocks are assembled, regardless of the selection pressure that is applied (aggregation).

The results from our computational experiment have yielded evidence for the existence of all three of these processes. This evidence is in addition to the most obvious one—the problem is clearly tunable—but further experimentation is needed to identify all of the processes that may or may not be limiting factors in the assembly of building blocks. This paper looked at just three variables; there are others (such as those associated with initial population creation) that are likely to figure prominently. Such work, however, was beyond the scope of this paper.

We are optimistic that the use of a model problem like *Highlander* could be used as an experimental complement in the development of GP theory. Although three main processes were conjectured that afford for this problem to be tunably difficult, it remains to be seen whether a theoretical model can be developed that can explain some of the phenomena discussed in this paper.

8. ACKNOWLEDGMENTS

We thank the following individuals for their assistance: A. Hilss, D. Ward, S. Long, and P.Chuisano for theory and software support; R. O'Grady and C. Kurecka for grid computer protocol development; R. Middleton and A. Mackenzie for metaphor research; R. Tang for *Lid* research; M. Hodges, M. Pizzimenti, M.Byom, and J. Kriesel for their discussion. The first and second authors thank I. Kristo.

9. REFERENCES

[1] Banzhaf, W., et al. *GP: An Introduction*. Morgan Kaufmann, San Francisco, 1998.

[2] Daida, J.M. Limits to Expression in Genetic Programming: Lattice-Aggregate Modeling. in *CEC 2002*, IEEE, Piscataway, 2002, 273–278.

[3] Daida, J.M. Towards Identifying Populations that Increase the Likelihood of Success in Genetic Programming. in *GECCO 2005*, 2005.

[4] Daida, J.M. What Makes a Problem GP-Hard? A Look at How Structure Affects Content. in Riolo, R.L. and Worzel, W. eds. *GP Theory and Practice*, Kluwer Academic Publishers, Dordrecht, 2003, 99–118.

[5] Daida, J.M., Bertram, R.B., Polito 2, J.A. and Stanhope, S.A. Analysis of Single-Node (Building) Blocks in GP. in Spector, L., et al. eds. *Advances in GP 3*, MIT Press, Cambridge, 1999, 217–241.

[6] Daida, J.M. and Hilss, A.M. Identifying Structural Mechanisms in Standard GP. in Cantú-Paz, et al. eds. *GECCO 2003*, Springer-Verlag, Berlin, 2003, 1639–1651.

[7] Daida, J.M., et al. What Makes a Problem GP-Hard? Validating a Hypothesis of Structural Causes. in Cantú-Paz, et al. eds. *GECCO 2003*, Springer-Verlag, Berlin, 2003, 1665–1677.

[8] Daida, J.M., et al. What Makes a Problem GP-Hard? Analysis of a Tunably Difficult Problem in GP. in Banzhaf, W., et al. eds. *GECCO '99,* Morgan Kaufmann, San Francisco, 1999, 982 – 989.

[9] Daida, J.M., et al. What Makes a Problem GP-Hard? Analysis of a Tunably Difficult Problem in GP. *GPEM*, *2* (2). 165–191.

[10] Gathercole, C. and Ross, P. An Adverse Interaction Between Crossover and Restricted Tree Depth in GP. in Koza, J.R., et al. eds. *GP 1996*, MIT Press, Cambridge, 1996, 291–296.

[11] Goldberg, D.E. and O'Reilly, U.-M. Where Does the Good Stuff Go, and Why? in Banzhaf, W., et al. eds. *EuroGP,* Springer-Verlag, Berlin, 1998, 16–36.

[12] Hall, J.M. and Soule, T. Does GP Inherently Adopt Structured Design Techniques? in O'Reilly, U.-M., et al. eds. *GP Theory and Practice II*, Kluwer Academic Publishers, Boston, 2004.

[13] Langdon, W.B. and Poli, R. An Analysis of the MAX Problem in Genetic Programming. in Koza, J.R., et al. eds. *GP 1997*, Morgan Kaufmann, San Francisco, 1997, 222–230.

[14] Langdon, W.B. and Poli, R. *Foundations of GP*. Springer-Verlag, Berlin, 2002.

[15] Matsumoto, M. and Nishimura, T. Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudorandom Number Generator. *ACM Trans Mod and Comp Sim*, *8* (1). 3–30.

[16] McPhee, N.F. and Hopper, N.J. Analysis of Genetic Diversity through Population History. in Banzhaf, W., et al. eds. *GECCO '99*, Morgan Kaufmann, San Francisco, 1999, 1112 – 1120.

[17] Motoki, T. Calculating the Expected Loss of Diversity of Selection Schemes. *EC*, *10* (4). 397–422.

[18] O'Reilly, U.-M. The Impact of External Dependency in GP Primitives. in *CEC 1999*, IEEE Press, Piscataway, 1998, 306–311.

[19] O'Reilly, U.-M. and Goldberg, D.E. How Fitness Structure Affects Subsolution Acquisition in GP. in Koza, J.R., et al. eds. *GP 1998*, Morgan Kaufmann, San Francisco, 1998, 269–277.

[20] Poli, R. General Schema Theory for GP with Subtree-Swapping Crossover. in Miller, J.F., et al. eds. *EuroGP 2001*, Springer-Verlag, Berlin, 2001, 143–159.

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

[22] Rosca, J.P. Analysis of Complexity Drift in GP. in Koza, J.R., et al. eds. *GP 1997*, Morgan Kaufmann, San Francisco, 1997, 286–294.

[23] Sastry, K., et al. Population Sizing for GP Based on Decision Making. in O'Reilly, U.-M., et al. eds. *GP Theory and Practice II*, Kluwer Academic, Boston, 2004, 49–65.

[24] Soule, T., et al. Code Growth in Genetic Programming. in Koza, J.R., et al. eds. *GP 1996*, MIT Press, Cambridge, 1996, 215 – 223.

[25] Zongker, D. and Punch, W. lilgp, Michigan State University Genetic Algorithms Research and Applications Group, Lansing, 1995.