How Different are Genetic Programs? Entropy Methods for Studying Diversity and Complexity in Genetic Programming

Mori, Naoki, McKay, R.I. (Bob), Nguyen, Xuan Hoai and Essam, Daryl

***Abstract*— We propose a new approach to the study of Genetic Programming dynamics, using genotypic entropy metrics. The aim of the approach is to provide a single framework for simultaneously studying diversity and individual complexity, both of the raw trees and their non-redundant sub-components. The entropy metrics are based on the entropy of small subtrees, and we show that the analysis is largely independent of the exact choice of subtree shape.**

To facilitate the study of redundancy, we apply powerful new methods for detecting and removing redundancy; while these methods are tailored to the particular function set used here, they are readily extensible to other function sets.

We demonstrate the effectiveness of the approach by applying it to a well-known symbolic regression problem. Using the new methods, we measure quantitative relationships between solution complexity and diversity, explore relationships between population structure and problem success or failure, and analyse the relationships between the different solutions found by GP.

I. Introduction

It is well known that Genetic Programming (GP - [1]–[4]) populations start off with low individual complexity and high diversity, and tend toward increasing individual complexity, and reduced diversity, with an increasing proportion of the complexity, and probably the diversity, residing in the redundant components of the individuals. But to what extent can these assertions be quantified? To be sure, it is possible to compare the initial and final diversities using such metrics as edit distance [5], or the initial and final complexities using metrics such as tree size or depth [6]. But if we ask, what are the relative changes, how can we answer? If we ask, how much does the population complexity transfer from the diversity of individuals to their internal complexity, how can we quantify the result? We aim, in this paper, to provide answers to these questions. We are convinced that (unlike the case with Genetic Algorithms (GA)), a full understanding of diversity in GP requires an understanding of its interaction with individual complexity, and that a prerequisite for such an understanding is an ability to quantify them in common terms. We hope that this paper provides a stepping stone in this direction.

In this paper, we first consider previous work on GP diversity and complexity in section II. Section III introduces the subtree entropy metrics which are the primary focus of this paper, while section IV briefly introduces our simplification approach. The experimental settings for the study are described in section V, and examples of the sorts of analyses which may be conducted are presented in section VI, while section VII provides the results of these analyses. Finally, in section VIII, we discuss how these methods provide new information about the dynamics of GP systems, and how we hope to extend this work in the future.

II. Background

Genetic Programming takes its inspiration from the field of Genetic Algorithms [7], hence it is hardly surprising that the interest in diversity in the former has extended to the latter. However the evolutionary dynamics of GP systems are more complex than those of GA. Since the complexity of individuals is free to change, GP's evolutionary dynamics also involve changes in complexity. To date, these two areas have largely been considered separately, as will be apparent from the subsequent discussion. However it is clear that there are close interactions. If the individuals in a population become too simple, they also rapidly lose diversity (as those who experiment with parsimony pressures often discover); conversely, bloating of individuals leads to a high level of genotypic diversity, but not necessarily of phenotypic diversity. Hence we will first discuss some of the previous research in these areas.

A. Diversity and Diversity Measures in GP

Diversity studies in GP fall largely into two classes, phenotypic diversity studies and genotypic diversity studies. We consider them separately.

1) Phenotypic Diversity: Phenotypic diversity measures the variety of different fitness values (or at least, function values) generated by a GP algorithm. There have been two main strands of such studies, corresponding to two phenotype diversity measures:

- The number of different fitness values [8]
- Entropy of fitness [9]

Fitness-based (and other phenotypic) measures have a number of advantages, chief among them being their ease of computation - since phenotype and fitness values must be calculated anyway in order to run the algorithm, the additional overhead of computing the metric is low. Phenotypic diversity is particularly useful when we wish to understand the behaviour of the selection operator, and understand the degeneracy of the fitness space [10], [11].

However there is a loss of information in mapping from genotype to phenotype space. The mapping may be many-to-one (many genotypes sharing the same phenotype value), resulting from (among other causes)

- degeneracy in the semantics of the solution - e.g. periodicity in the solution function
- degeneracy in the fitness assignment - e.g. rank-based fitness values such as those used in Pareto ranking [7]

This degeneracy means that phenotypic diversity measures give us limited insight into other aspects of evolutionary dynamics, notably the effects of the variation operators (mutation and crossover), and of bloat and other changes in individual complexity.

2) Genotypic Diversity: A wide range of genotype diversity measures have been defined in previous research, including at least:

1) The number of different genotypes [12]
2) The total number of nodes (tree size) [10]
3) Tree depth [10]
4) Edit/Levenshtein distance [5]
5) Cell-content distance [13], [14]

Measures 1∼3 are computationally tractable, and provide a rough understanding of GP evolutionary dynamics. However they do not reflect the internal tree structure, which is critical to understanding the overall behaviour of GP systems.

The fourth and fifth metrics, based on edit and content distance, do incorporate information about the internal structure of trees. However simple edit and content distances depend on the absolute position of each node. For example, although $f(x) * g(x) = g(x) * f(x)$, the edit and cell-content distances between $f(x) * g(x)$ and $g(x) * f(x)$ may be very large (depending on the complexities of f and g). As the evolutionary operators we are interested in – crossover and mutation – are applied in GP without regard to absolute position, this loss of information is crucial.

Variants of the edit/Levenshtein distance metric (5) can avoid the dependence on absolute position, but they still ignore the non-positional contribution of subtrees.

A second problem with edit distance metrics arises from computational cost considerations. Primarily to avoid quadratic computational cost, edit distance metrics usually compare each individual only with the elite. Thus two almost identical populations, differing only in their elite, may be credited with vastly different diversities.

Finally, all the measures 1∼3 above suffer another important problem: they give no guidance as to how to estimate the complexity (or internal diversity) of an individual in measures that are commensurate with the diversity metric.

B. Complexity Metrics

Individual complexity has been a major focus of GP research [15], [16], with a particular emphasis on the twin phenomena of bloat and of redundant code [6], [17]–[22]. To date, the emphasis has primarily been on the causes of bloat, and of mechanisms to avoid or ameliorate it, rather than on metrics. To a large extent, the research has focused on tree size or depth as complexity metrics, though Langdon and Banzhaf have examined the repetition of large subtrees [23], [24].

Conversely to the situation with diversity metrics, the current methods for studying individual complexity, and the metrics used, are incommensurate with those used for diversity. This renders it difficult to study both within a common framework. We believe that studying the interaction of individual complexity and population diversity is important for fully understanding the evolution of both in GP systems.

C. Theoretical Research on Diversity

Burke et al [10] studied the relationships between diversity measures – such as the number of genotypes, the number of phenotypes, edit and content distances, the entropy of phenotypes, and the set of ⟨terminal, non-terminal, depth⟩ values – and the fitness. They reported that there is only a weak correlation between diversity measures and fitness in symbolic regression problems.

Gustafson et al [11] studied the relationship between the difficulty of problems and bloat using the depth, size, edit distance and entropy of phenotype as measures. They explained the difficulty of problems primarily by the entropy of phenotype. However they did not find any effective relationship between fitness and tree size.

D. Redundancy

In analysing GP dynamics, redundancy is a key issue. GP has redundancy in the genotype-to-phenotype mapping – that is, several individuals with different genotypes may nevertheless have the same phenotype. These different genotypes may have different complexities, and a GP algorithm is not constrained to find the simplest. As a result, GP can – and generally does – suffer from the phenomenon of bloat, in which both before and after the population has converged phenotypically, the complexity of the individuals increases rapidly.

If we wish to understand the behavior of GP systems – their genotypic diversity, complexity, building block evolution etc., it is crucial to understand the behavior not only of the overall genotype, but also of its effective core – that part which actually affects the semantics, and hence determines the selective pressure on the system.

In [25], we introduced a new method of simplification known as Equivalent Decision Simplification (EDS), and showed that – at least for real arithmetic domains such as those considered here – it is far more effective in simplifying expressions (though at higher computational cost) than the better-known algebraic simplification methods, and thus better suited to our purposes. We use EDS in the work reported here.

III. Subtree Entropy

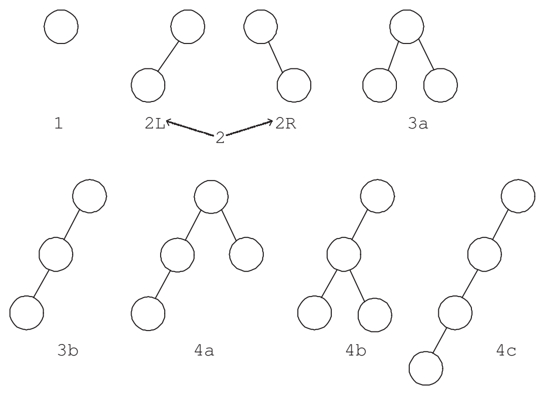


Fig. 1. Subtree Templates

In this research, we are proposing a new diversity measure called “subtree entropy”. This measure is based on the entropy of fixed-shape subtrees (templates) of the individual or population under consideration. A template is decided on before analysis and fixed. Figure 1 shows examples of subtree templates of various sizes.

Our method extracts instances of the template subtrees from the target set of trees, and calculates the entropy based on those subtrees. We call this entropy “subtree entropy”.

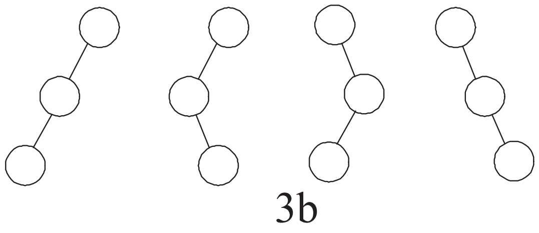


Fig. 2. Four different Structures for Template 3b (having the same connection relationships, but differing relative node positions).

For binary operators, if the number of internal nodes in a template is α, the number of leaves is β, the number of non-terminal nodes in the language is M_{op}, and the number of terminal nodes in the language is M_{t}, then the total number of different subtrees which belong to this template, N_{all} is[1]:

$$N_{\mathrm{all}} \leq M_{\mathrm{op}}^{\alpha}(M_{\mathrm{op}} + M_{\mathrm{t}})^{\beta} \tag{1}$$

GA diversity research using entropy focuses only on population entropy. However the structural complexity may vary from individual to individual in GP, so the internal entropy of the individual is also important. Therefore, in this study, we propose two kinds of subtree entropy: "population-based subtree entropy" H_{pop} and "individual-based subtree entropy" H_{indiv}.

A. Subtree Templates

We present detailed results from only one template, namely "3a" from figure 1. However we have carried out extensive studies of entropy based on all the subtree templates shown in the figure. These are presented in detail in the experimental sections, in subsection VII-A. In general, we found that the choice of template had little effect.

We chose template "3a" for detailed study because

- It appears very typical, at least of small subtree templates
- It is relatively easy to compute
- In contrast with template "3b", there is no symmetry issue (figure 2 shows the different subtrees of type "3b")
- Apart from $\sin$, the sample problem used in this study has only binary operators

B. Population-based Subtree Entropy H_{pop}

Population-based Subtree Entropy may be computed as follows:

1) Let S_{pop} be the set of instances belonging to the given template, from all individuals in population P.
2) Let the number of occurrences of subtree s in S_{pop} be N_s. We treat the ratio ($N_s/|S_{\mathrm{pop}}|$) as an estimate of the probability p_s of s. H_{pop} is defined as:

$$H_{\mathrm{pop}} = -\sum_{s \in S_{\mathrm{pop}}} p_s \log_e p_s \tag{2}$$

[1]Equality may not be reached because some nonterminals may have too low arity to match some templates

C. Individual-based Subtree Entropy H_{indiv}

Individual-based Subtree Entropy may be computed as follows:

1) Let S_i be the set of subtrees belonging to the given template in each individual i in population P.
2) Let the number of occurrences of subtree s' in S_i be $N_{s'}$. We treat the ratio ($N_{s'}/|S_i|$) as an estimate of the probability $p_{s'}$ of s'. Individual i's subtree entropy H_i is defined as:

$$H_i = -\sum_{s' \in S_i} p_{s'} \log_e p_{s'} \tag{3}$$

3) Set H_{indiv} equal to the mean value of H_i in the population, i.e. $\langle H_i \rangle$.

IV. Simplification of GP Individuals

We call the operation of converting a tree structure into an equivalent but smaller structure "simplification". In this study, we used Equivalent Decision Simplification (EDS). Briefly, EDS simplifies trees by evaluating subtrees over the set of instances used to compute the fitness function, and testing whether these values are the same (within some accuracy bound) as those yielded by a set of canonical small trees; if so, the subtree is replaced by the canonical small tree. In [25], we demonstrated that EDS finds substantially more simplifications than syntactically-based algebraic methods (though at greater computational cost), and so is well suited to our purpose.

V. Experimental Setting

In this study, we use standard GP and a typical symbolic regression problem [2]–[4], [25], [26]. The underlying experimental setting is the same as for [25].

A. Problem Domain

The chosen problem is, given the 20 random X and Y values shown in Figure (3) over the range $[-\pi, \pi]$, to find an expression for the target function $\cos 2X$ (in the figure, the 20 points are represented by + symbols). The 20 points are generated by dividing the range into 20 even intervals, and sampling uniformly randomly across each interval. The sample is generated once for each run, then held constant throughout the run.

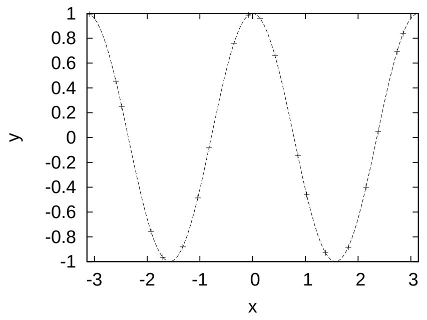


Fig. 3. 20 sample points

Problem

Objective function : $\cos 2X \quad [-\pi, \pi]$
Operators : $\{+, -, *, \%, \sin\}$

Operands : $\{X, 1\}$

where "%" is protected division, satisfying $X\%0 \rightarrow 1$. (note that the target function, cos, is not included in the operator set).

B. GP Settings

The GP settings used in our experiments are as follows:

- Number of runs: 1000
- Generations per run: 200
- Population size: 500
- Crossover rate: 0.9, using subtree crossover
- Mutation: subtree mutation, rate 0.1
- Selection: tournament selection, tournament size 3
- Initialization: ramped half-and-half initialization
- Tree Depth limit: initial limit = 6; subsequent limit = 15
- Repair mechanism: crossover and mutation were attempted up to 100 times

The raw fitness (RF) is calculated from the sum of the absolute errors at the 20 data points. Given 20 fitness points $S_{\mathrm{t}} = \{X_i, i = 1, 2, \ldots, 20\}$, the fitness f of the individual which represents function $g(X)$ is given by:

$$E = \sum_{i=1}^{20} \mid \cos 2X_i - g(X_i) \mid \tag{4}$$

$$f = \frac{1}{1+E} \tag{5}$$

An individual is regarded as a solution when all 20 errors are less than 0.01, as follows:

$$g(X) = \begin{cases} \text{solution}, & \forall X_i \in S_{\mathrm{t}}, \\ & \mid \cos 2X_i - g(X_i) \mid \leq 0.01 \\ \text{non}-\text{solution}, & \text{otherwise} \end{cases} \tag{6}$$

C. Numerical Computation Issues

This work leads to some complex issues in numerical computation, which we consider in more detail [25]. Briefly, we need to set bounds on some approximations used:

- When a value should be treated as zero, in both GP runs, and in analysis (we use a bound of 10^{-9})
- When two approximately equal solutions should be treated as phenotypically identical, especially for computing the phenotypic entropy (we use a bound of 0.00286 on differences in the instance values, chosen so that in particular, individuals accepted as solutions will always be treated as phenotypically identical)
- Analogously, how to compute the genotypic entropy

VI. Experimental Analyses

1000 independent runs of the GP system were performed, using 1000 different random seeds, the entire populations being saved for the subsequent analyses. We performed two different sets of analyses.

A. Analysis 1

This preliminary analysis investigates the effect of the different templates on the results. Extensive experimental studies were conducted on a range of different metrics, but for reasons of economy, we present here only the results for the population entropy metric (defined in subsection III-B). They are typical. We display and compare results produced by a range of different subtree templates. Firstly, we compare results of templates 2L and 2R (see figure 1), to see whether orientation has any noticeable effect. Then, we compare templates of different topologies, and finally, templates of different sizes.

B. Analysis 2

This section aims to show the additional insight into GP systems which may be gained through subtree entropy. We first present an analysis of the runs using a typical set of tools, namely tree size (as complexity metric) and phenotype and genotype entropies (as diversity metrics), though using our new EDS as a simplification tool [25]. We then introduce the subtree entropy metric, to understand the further insights which become available through its use.

VII. Results and Discussion

In this section, we show a number of graphs and tables describing the results of our experiments. In all plots, the abscissa (x axis) shows the generation, while the ordinate (y axis) shows the value of the particular parameter under investigation.

In our analyses, we wished to see whether there was any difference between fast-solving, slow-solving and failing runs, so we selected three separate sets of 100 runs from the 1000-run sample by stratified random sampling:

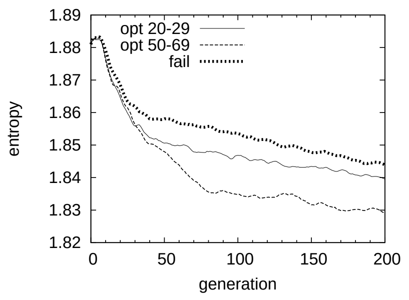
- C_{20}, consisting of 100 runs which found a solution between generations $20 \sim 29$.
- C_{50}, consisting of 100 runs which found a solution between generations $50 \sim 69$.
- C_{fail}, consisting of 100 runs which did not find a solution within the 200-generation limit of the runs.

Most plots show three graphs, corresponding to C_{20}, C_{50} and C_{fail} (respectively opt 20-29, opt 50-69 and fail in the legends), each being the average of 100 trials.

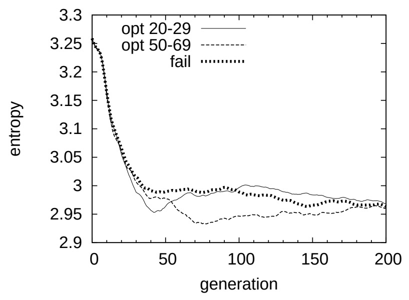
A. Comparing Different Subtree Templates

Here, we consider the effects of different subtree templates on the resulting entropy metrics. In particular, we look at the effect on the population entropy. Figure 4 shows a detailed comparison, for classes C_{20}, C_{50} and C_{fail}, of the population entropy metric as calculated using each of the different subtree templates from figure 1. The important point to note here is that, while the results are specific to this particular metric and scenario, the general conclusions carried across to all the subtree entropy metrics studied. For this particular metric, we found that:

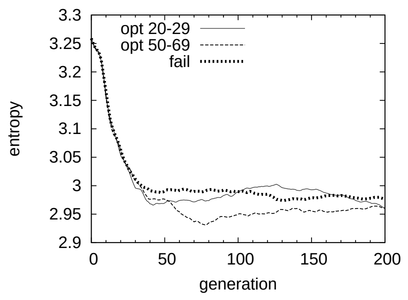
- The behaviour of the metric for template "1" (figure 4a) differs substantially from those for all other templates (hardly surprising, since template "1" ignores all structure information, depending only on the content of nodes)
- The behaviours of templates "2L" and "2R" (figures 4b, c) are very similar (figure 4d), so we have ignored orientation of subtrees in subsequent work, and only consider topology
- The behaviour of templates "3a" and "3b" (figures 4e, f) are very similar, as are the behaviours of templates "4a", "4b" and "4c" (figures 4g, h, i)
- The behaviours of size 2 templates are quite similar to those of size 3, and the differences between size 3 and size 4 templates are small



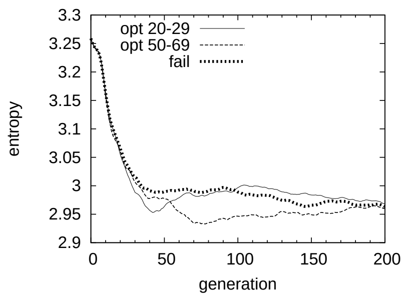
(a) Template 1



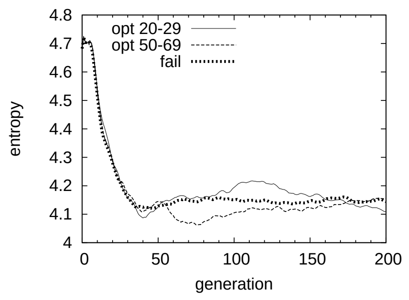
(b) Template 2L



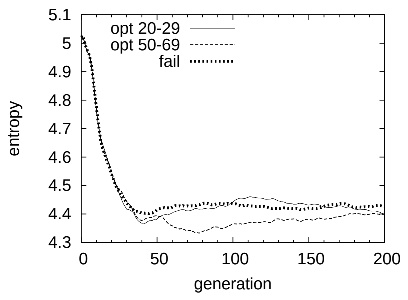
(c) Template 2R



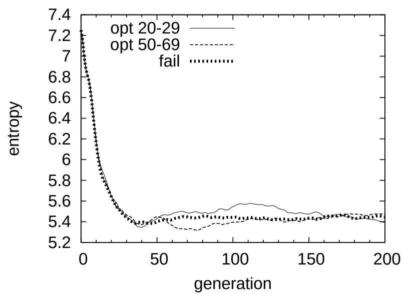
(d) Templates 2L & 2R



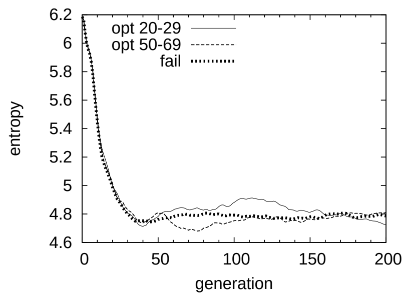
(e) Template 3a



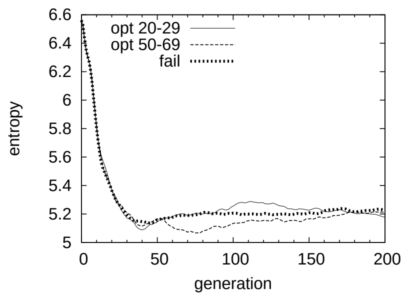
(f) Template 3b



(g) Template 4a

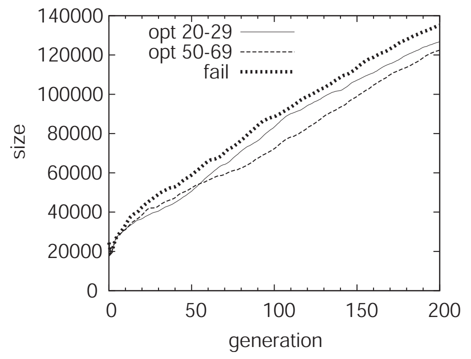


(h) Template 4b

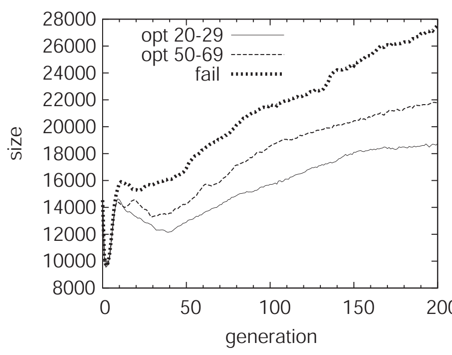


(i) Template 4c

Fig. 4. Population Entropy for Different Templates



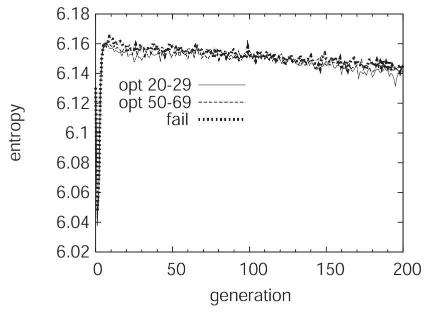
(a) Size (before simplification)



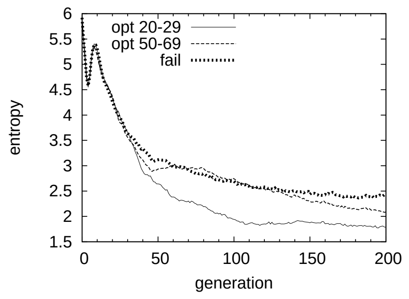
(b) Size (after simplification)

Fig. 5. Variation in the Total Number of Nodes (Size)

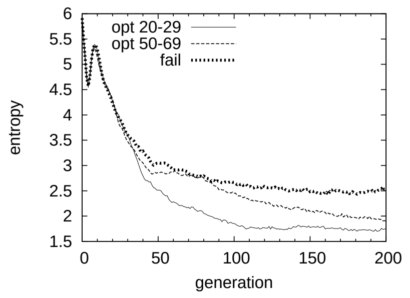
There are practical issues in extending the work beyond size 4 templates. Of course, it is obvious that there are computational cost issues - the number of subtrees grows exponentially with the subtree size, rendering large template entropies computationally infeasible. But equally important, since the number of bins also increases exponentially, but with a larger exponent, the number of instances in each bin decreases with increasing subtree size. Thus exponentially increasing numbers of individuals and runs are required, to avoid the entropy calculation being swamped by stochastic noise in the near-empty bins. We have not found it feasible to repeat these experiments for subtrees of size beyond 4. Pragmatically, as described in section III, we focused in the rest of our study on metrics calculated from template 3a, as both computationally efficient, and typical of the results from other templates. All subsequent discussion is based on that template.



(a) Genotype Entropy (before simplification)



(b) Genotype Entropy (after simplification)



(c) Phenotype Entropy

Fig. 6. Variation in Entropy of Genotype and Phenotype

B. *Evolutionary Dynamics of Genetic Programming*

In this subsection, we consider the evolutionary dynamics of Genetic Programming; for economy of exposition, we use the plots of C_{20}, C_{50} and C_{fail} rather than the overall population plots, but pay attention mainly to the overall shapes of the graphs, rather than the differences between them.

In [25], we presented a typical classical analysis of such runs (though using EDS rather than rule-based methods for simplification). Briefly, we were able to conclude from figures 5 and 6 (from [25]) that:

- The genotype size gives us very little information beyond the expected, that it increases over time
- The genotype entropy increases rapidly to a maximum, then declines gradually (so there is some small level of convergence in genotypes – perhaps due to pressure exerted by the depth boundary)
- The phenotypic entropy behaves in slightly unexpected ways, starting at a value just below 6, then falling rapidly to a local minimum around 4.5 (generation 4), rising rapidly to around 5.4 (generation 10), then falling monotonically toward an asymptote. We noted in [25] that EDS could provide some explanation for this.
- The effective-code genotype entropy closely mimics the behaviour of the phenotype entropy, strongly suggesting that EDS is finding close to the minimal effective core of individuals
- Unlike the total node size, the effective code size is highly informative, yielding good explanation for the behaviour of phenotype entropy: initially, many large but unfit individuals are lost from the population; then there is an increase in size as somewhat fit individuals are built up. At that stage (around generation 10, some relatively fit, small expressions, such as $\sin X$, start to dominate, and are then used as building blocks in larger, even fitter expressions (see [25] for more detailed analysis).

When we introduce subtree entropy to the analysis, we immediately see considerable similarity between the individual subtree entropy H_{indiv} plots in figure 7 and the size plots in figure 5 (they would be even more similar if we had drawn the size plots on a logarithmic scale to reflect the trees' mean node depths). This supports our interpretation of subtree entropy as a complexity metric.

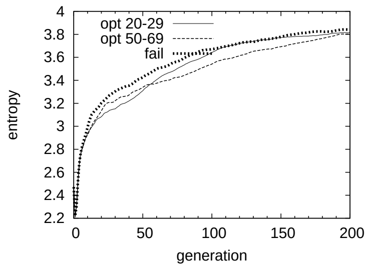
The overall shape of the reduced-code population subtree entropy plot (figure 8(b)) is not too dissimilar to that of the reduced-code genotype entropy (figure 6(b)), with the exception of the missing inflexion between generations 3 and 10. In fact, the lack of this inflexion somewhat bears out our explanation of it in the previous case, since that size-based explanation would be less likely to affect a content-based metric such as subtree entropy.

When it comes to the original code plots, however, we see an important difference. While the genotypic entropy plot (figure 6(a)) provided little useful information, the subtree entropy plot (figure 8(b)) shows useful structure similar, but not identical, to the reduced code plot.

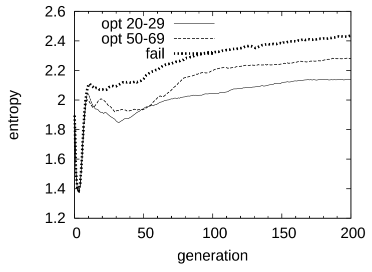
TABLE I
GENOTYPIC ENTROPY (MEAN OF C_{20}, C_{50}, C_{fail})

Generation:	0	1	2	50	100	150	200
Before Simplification							
H_{indiv}	2.47	2.22	2.30	3.36	3.62	3.75	3.82
H_{pop}	4.68	4.72	4.71	4.13	4.15	4.14	4.14
After Simplification							
H_{indiv}	1.90	1.50	1.39	2.01	2.19	2.25	2.29
H_{pop}	4.45	4.44	4.40	2.85	2.87	2.91	2.94

However the most significant insight comes when we compare figures 7 and 8 – something we can sensibly do, because they share a common ordinate scale, while figures 5 and 6

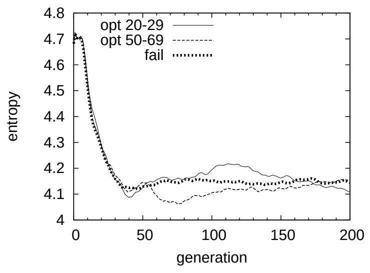


(a) $H_{\rm indiv}$ (before simplification)

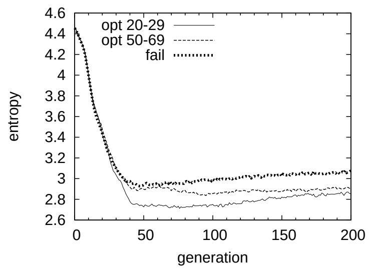


(b) $H_{\rm indiv}$ (after simplification)

Fig. 7. Variation in $H_{\rm indiv}$ (3a)



(a) $H_{\rm pop}$ (before simplification)



(b) $H_{\rm pop}$ (after simplification)

Fig. 8. Variation in $H_{\rm pop}$ (3a)

do not. To facilitate this analysis, table I shows the numeric values (averaged over C_{20}, C_{50} and C_{fail}) for generations 0, 1, 2, 50, 100, 150 and 200.

Firstly, we note that the generation 1 individual subtree entropy is less than half the population subtree entropy (for both the total and the effective code); by generation 50, the difference has narrowed to around $3/4$ (both). By generation 200, it has decreased further, to around 0.92 (original), 0.78 (simplified). To put this another way, in the initial population, the subtree entropy is approximately equally distributed between the internal complexity of individuals and the diversity between them. By the end of the run (reading from the right hand edges of the figures), over 90% of the subtree entropy is concentrated in the internal complexity of un-pruned individuals, less than 10% is used for diversity. The situation is rather different with effective code, with 22% of the subtree entropy still residing in diversity of the effective code, and only 78% being consumed by the individual complexity. Finally, we note that by generation 200, 71% (2.94 out of 4.14) of the total subtree entropy is concentrated in the effective code. This is also true to a lesser extent of the individual subtree entropy (60%, or 2.29 out of 3.82).

We wish to emphasise that these comparisons could not sensibly be made using previous diversity metrics; they would be “comparing apples with oranges”. It is only because we are measuring both diversity and complexity with a common tool – subtree entropy – that these comparisons make sense. Of course, these analyses need to be repeated on other problems, to determine to what extent these behaviours are repeated in different domains. In the long term, our aim is to see if there are any systematic relationships between individual and population subtree entropies as evolution progresses, and if so, to look for mathematical quantisations of those relationships.

VIII. Conclusions

We have proposed novel GP analysis methods, using subtree entropy and and equivalent decision simplification. Applying these methods, we were able to obtain the following:

- An analysis of the quantitative dynamics of problem complexity (bloat).
- A quantitative comparison of problem complexity and diversity, respectively reflected in individual-based and population-based subtree entropy.

We plan to extend this work in the following ways:

- We intend to apply the methods to other problems, both to other symbolic regression problems, and to a wider range of problems.
- While subtree entropy gives us a way to integrate the study of GP complexity and diversity, we are particularly interested to extend this work into machine learning problems, permitting the integration of information-based error, diversity and complexity metrics. It is our long-term belief that such an integration could lead to a better understanding of how to trade off error, complexity and diversity in GP systems.
- The subtree entropy studies can be viewed as providing a lower bound for the population and individual information complexity; complementarily, compression metrics can give an upper bound. We are currently undertaking experimental work using tree compression, in the hope that it can lead to a better understanding of the behaviour of GP systems.

In addition, this work has a broader long-term aim. Entropy has largely been studied in the context of fixed-complexity systems, and provides tremendous insight into the behaviours of those systems. While less is known about the entropic behaviour of variable complexity systems such as GP (or the evolution of DNA or proteins, for that matter), we are convinced that entropy studies offer a key to understanding these systems once the appropriate theoretical frameworks have been developed. We view this experimental study as a first step in inspiring such a development.

ACKNOWLEDGMENT

The authors would like to acknowledge helpful discussions with Prof. Hajime Kita of Kyoto University, Japan, and assistance with preliminary experimental work from Mr Tuan Hao Hoang, of the University of New South Wales, Australia.

We especially wish to acknowledge the unfailing support and encouragement of Prof. Keinosuke Matsumoto of Osaka Prefecture University, Japan.

This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientists (B), 18700227, 2006-2007, and by the support program for new faculty of Seoul National University.

REFERENCES

[1] N. L. Cramer, "A representation for the adaptive generation of simple sequential programs," in *Proceedings of an International Conference on Genetic Algorithms and the Applications*, J. J. Grefenstette, Ed., Carnegie-Mellon University, Pittsburgh, PA, USA, 24-26 July 1985, pp. 183–187. [Online]. Available: http://www.sover.net/ nichael/nlc-publications/icga85/index.html

[2] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA, USA: MIT Press, 1992.

[3] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann, dpunkt.verlag, Jan. 1998.

[4] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza, *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, 2003. [Online]. Available: http://www.genetic-programming.org/gpbook4toc.html

[5] U.-M. O'Reilly, "Using a distance metric on genetic programs to understand genetic operators," in *IEEE International Conference on Systems, Man, and Cybernetics, Computational Cybernetics and Simulation*, vol. 5, Orlando, Florida, USA, 12-15 Oct. 1997, pp. 4092–4097. [Online]. Available: http://ieeexplore.ieee.org/iel4/4942/13793/00637337.pdf

[6] W. B. Langdon, "Quadratic bloat in genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, Eds. Las Vegas, Nevada, USA: Morgan Kaufmann, 10-12 July 2000, pp. 451–458.

[7] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[8] J. P. Rosca, "Genetic programming exploratory power and the discovery of functions," in *Evolutionary Programming IV Proceedings of the Fourth Annual Conference on Evolutionary Programming*, J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, Eds. San Diego, CA, USA: MIT Press, 1-3 Mar. 1995, pp. 719–736. [Online]. Available: ftp://ftp.cs.rochester.edu/pub/u/rosca/gp/.ep.exploratory.ps.gz

[9] ——, "Entropy-driven adaptive representation," in *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, J. P. Rosca, Ed., Tahoe City, California, USA, 9 July 1995, pp. 23–32. [Online]. Available: ftp://ftp.cs.rochester.edu/pub/u/rosca/gp/95.ml.gpw.ps.gz

[10] E. Burke, S. Gustafson, and G. Kendall, "A survey and analysis of diversity measures in genetic programming," in *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, Eds. New York: Morgan Kaufmann Publishers, 9-13 July 2002, pp. 716–723.

[11] S. Gustafson, A. Ekart, E. Burke, and G. Kendall, "Problem difficulty and code growth in genetic programming," *Genetic Programming and Evolvable Machines*, vol. 5, no. 3, pp. 271–290, Sept. 2004. [Online]. Available: http://www.cs.nott.ac.uk/ smg/

[12] W. B. Langdon, "Data structures and genetic programming," University College London, Gower Street, London WC1E 6BT, UK, Research Note RN/95/70, Sept. 1995. [Online]. Available: http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/papers/WBL.aigp2.ch20.ps

[13] A. Ekart and S. Z. Nemeth, "A metric for genetic programs and fitness sharing," in *Genetic Programming, Proceedings of EuroGP'2000*, ser. LNCS, R. Poli, W. Banzhaf, W. B. Langdon, J. F. Miller, P. Nordin, and T. C. Fogarty, Eds., vol. 1802. Edinburgh: Springer-Verlag, 15-16 Apr. 2000, pp. 259–270.

[14] E. D. de Jong, R. A. Watson, and J. B. Pollack, "Reducing bloat and promoting diversity using multi-objective methods," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, Eds. San Francisco, California, USA: Morgan Kaufmann, 7-11 July 2001, pp. 11–18.

[15] K. E. Kinnear, Jr., "Generality and difficulty in genetic programming: Evolving a sort," in *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, S. Forrest, Ed. University of Illinois at Urbana-Champaign: Morgan Kaufmann, 17-21 July 1993, pp. 287–294.

[16] B.-T. Zhang and H. Mühlenbein, "Balancing accuracy and parsimony in genetic programming," *Evolutionary Computation*, vol. 3, no. 1, pp. 17–38, 1995.

[17] N. F. McPhee and J. D. Miller, "Accurate replication in genetic programming," in *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, L. Eshelman, Ed. Pittsburgh, PA, USA: Morgan Kaufmann, 15-19 July 1995, pp. 303–309.

[18] T. Soule and J. A. Foster, "Support for multiple causes of code growth in GP," 20 July 1997, position paper at the Workshop on Evolutionary Computation with Variable Size Representation at ICGA-97.

[19] W. B. Langdon, "Fitness causes bloat in variable size representations," University of Birmingham, School of Computer Science, Tech. Rep. CSRP-97-14, 14 May 1997, position paper at the Workshop on Evolutionary Computation with Variable Size Representation at ICGA-97. [Online]. Available: ftp://ftp.cs.bham.ac.uk/pub/tech-reports/1997/CSRP-97-14.ps.gz

[20] P. J. Angeline, "Subtree crossover causes bloat," in *Genetic Programming 1998: Proceedings of the Third Annual Conference*, J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, Eds. University of Wisconsin, Madison, Wisconsin, USA: Morgan Kaufmann, 22-25 July 1998, pp. 745–752.

[21] W. Banzhaf and W. B. Langdon, "Some considerations on the reason for bloat," *Genetic Programming and Evolvable Machines*, vol. 3, no. 1, pp. 81–91, Mar. 2002.

[22] S. Luke, "Issues in scaling genetic programming: Breeding strategies, tree generation, and code bloat," Ph.D. dissertation, Department of Computer Science, University of Maryland, A. V. Williams Building, University of Maryland, College Park, MD 20742 USA, 2000.

[23] W. B. Langdon and W. Banzhaf, "Repeated sequences in linear GP genomes," in *Late Breaking Papers at the 2004 Genetic and Evolutionary Computation Conference*, M. Keijzer, Ed. Seattle, Washington, USA: AAAI, 26 July 2004.

[24] ——, "Repeated patterns in tree genetic programming," in *Proceedings of the 8th European Conference on Genetic Programming*, ser. Lecture Notes in Computer Science, M. Keijzer, A. Tettamanzi, P. Collet, J. I. van Hemert, and M. Tomassini, Eds., vol. 3447. Lausanne, Switzerland: Springer, 30 Mar. - 1 Apr. 2005, pp. 190–202.

[25] N. Mori, B. McKay, X. H. Nguyen, and D. Essam, "Equivalent decision simplification: A new method for simplifying algebraic expressions in genetic programming," in *Proceedings of the 2007 Asia-Pacific Symposium on Intelligent and Evolutionary Systems*, Yokosuka, Japan, 2007.

[26] N. X. Hoai, "Solving trignometric identities with tree adjunct grammar guided genetic programming," in *2001 International Workshop on Hybrid Intelligent Systems*, ser. LNCS, A. Abraham and M. Koppen, Eds. Adelaide, Australia: Springer-Verlag, 11-12 Dec. 2001, pp. 339–352.