An Efficient Structural Diversity Technique for Genetic Programming

Armand R. Burks
BEACON Center for the Study of Evolution in Action
Michigan State University, U.S.A.
burksarm@msu.edu

William F. Punch
BEACON Center for the Study of Evolution in Action
Michigan State University, U.S.A.
punch@msu.edu

ABSTRACT

Genetic diversity plays an important role in avoiding premature convergence, which is a phenomenon that stifles the search effectiveness of evolutionary algorithms. However, approaches that avoid premature convergence by maintaining genetic diversity can do so at the cost of efficiency, requiring more fitness evaluations to find high quality solutions. We introduce a simple and efficient genetic diversity technique that is capable of avoiding premature convergence while maintaining a high level of search quality in tree-based genetic programming. Our method finds solutions to a set of benchmark problems in significantly fewer fitness evaluations than the algorithms that we compared against.

Categories and Subject Descriptors

I.2.8 [**ARTIFICIAL INTELLIGENCE**]: Problem Solving, Control Methods, and Search

Keywords

genetic programming; diversity; premature convergence

1. INTRODUCTION

Premature convergence is one of the biggest problems that traditional evolutionary algorithms face. Premature convergence can be thought of as evolutionary stagnation, wherein the population reaches a point at which it is very difficult, or even impossible, to improve in fitness. As the population becomes more genetically similar, the traditional genetic operators such as crossover and mutation become less likely to escape local optima.

Evolutionary algorithms (EAs) have widely been applied to solve problems with enormous search spaces that usually contain an abundance of local optima. Therefore, the ability to effectively explore such spaces is critical to finding solutions. In many EAs, as the population converges genetically, the population also converges around a local peak in the fitness landscape. This has two notable impacts on the quality of the search process: (1) as individuals in the population become more genetically similar, the traditional crossover and mutation operators only produce offspring that highly resemble their parents and (2) the EA is reduced to a local search, incapable of effectively moving the population away from the local fitness peak in order to find global optima.

GECCO '15, July 11–15, 2015, Madrid, Spain

DOI: http://dx.doi.org/10.1145/2739480.2754649

1.1 Genetic Programming

We consider the role of genetic diversity in avoiding premature convergence in genetic programming (GP). GP represents a class of heuristic techniques that extend from the traditional genetic algorithm. In classical tree-based GP, individuals are encoded with a tree structure, similar to an abstract syntax tree. As one of the main goals in GP is to evolve computer programs, this tree structure provides a generic way of representing such programs, and it allows the search through the space of possible programs with genetic operators such as crossover and mutation.

1.2 Premature Convergence in GP

The problem of premature convergence is perhaps even more puzzling for traditional tree-based GP because of the variable-size tree representation and the nature of the traditional genetic operators. Although it has been shown that GP populations do not converge the same way as in the classical binary-encoded genetic algorithm [10], premature convergence is still a problem for GP. Earlier analysis work showed that in classical tree-based GP using only subtree crossover, very few individuals from the initial population contribute *any* genetic material to the final population [12]. That work also showed that the entire population often inherits the rooted portion of their trees from a single ancestor, referred to as *Eve*, within just a few generations from the onset of the run. Additionally, over 70% of the population became completely identical to the *Eve* individual in the top four levels of their trees [12].

Other studies into the nature and effectiveness of the typical subtree crossover operator in GP have revealed some interesting properties that contribute to the major loss of genetic diversity [3, 13]. In the traditional subtree crossover, a node is uniformly randomly selected as the crossover point in each parent. The resulting subtrees rooted at each crossover point are exchanged between the parents to create two offspring. This directly contributes to the problem that trees tend to converge to the same structure near the top. Gathercole and Ross showed that nodes deeper in the tree are frequently selected for crossover, leaving upper portions of the trees untouched [3].

Due to the branching factors of the trees, there is a disproportionately large number of nodes in lower levels compared to those higher up in the trees. This means that uniformly randomly choosing a node is more likely to select a node at a lower level. Coupled with a depth limit, which is commonly used to avoid unrestricted tree growth, lower crossover points will result in failed crossover attempts, as they produce offspring that violate the depth limit. This has partly been addressed by simply adding a bias to probabilistically select function (internal) nodes more often than terminal (leaf) nodes. However, Poli and Langdon also showed that subtree crossover tends to exchange little genetic material, on average, producing offspring that are highly identical to one of the two parents [13].

2. EXISTING DIVERSITY TECHNIQUES

2.1 Genetic Diversity Metrics

Several metrics for measuring genetic diversity have been proposed [1]. Many of these metrics attempt to measure the degree to which two genotypes differ by taking into account some aspect of their tree structures. For example, one approach to calculating the the distance between trees considers the overlapping nodes when the two trees are overlaid [2]. After overlaying the trees, the distance is then calculated by taking the total number of differing nodes divided by the size of the smaller tree.

As Burke et al. discuss [1], the main intuition behind promoting diversity is that increased diversity gives the evolutionary process more opportunities to discover solutions. However, it is not always clear which type of diversity metric to employ, or how much diversity is sufficient. For many problem domains, although two genotypes are considered different by a genotypic distance metric, the two genotypes can encode the same (or very similar) behavior or phenotype, and ultimately yield the same fitness value.

With this issue in mind, other approaches have been proposed to explicitly maintain a diverse set of behaviors or phenotypes in the population [8, 11]. The fitness uniform selection scheme [8] attempts to maintain a spread of fitness values over the range of possible fitness values. On the other hand, the novelty search algorithm [11] completely ignores fitness and instead searches for unique behaviors during evolution to avoid the deceptive nature of fitness functions for certain problems. In its original implementation, the novelty search algorithm also benefits from a genetic diversity maintenance technique.

The Structure Fitness Sharing (SFS) algorithm [7] is another method that attempts to promote diversity based on tree structure. Motivated by the fitness sharing concept [4], SFS uses labels on tree structures to decrease the fitness of structures that are over-represented in the population. Instead of directly calculating and comparing genotypic distances, SFS labels each unique structure in the population and keeps track of the total individuals and best fitness for each structure.

2.2 Structured Populations

Instead of trying to enforce genetic diversity through an explicit diversity measure, alternative approaches impose a structure on the population to achieve the goal of avoiding premature convergence [5, 6]. One such approach is the Age-Layered Population Structure (ALPS) [5]. ALPS uses the concept of "genotypic age" to segregate the population into a hierarchy of several age layers, where genotypic age attempts to measure how long an individual's genetic material has been evolving.

The main idea behind ALPS is to localize competition by genotypic age so that younger, less-fit, individuals do not immediately become dominated by older more-fit individuals. This principle allows new randomly generated individuals to be regularly inserted into the population without immediately being outcompeted. This is a key feature of ALPS because the new random individuals could be located within a different region of the search space and contain valuable genetic material that can help the population escape local optima. Without the localized competition of the hierarchical structure, these new random individuals would likely be dominated by older individuals before having a chance to propagate useful genetic material throughout the population.

2.3 Multi-objective Optimization

Multi-objective approaches have also been proposed to optimize fitness while maintaining diversity. The FOCUS (find only and complete undominated sets) method transforms single objective problems into a three-dimensional search to simultaneously optimize fitness, diversity, and tree size [2]. The FOCUS method maintains genetic diversity by using the overlapping tree distance metric described in Section 2.1 to give preference to individuals with the least average squared distance to the entire population.

The Age-Fitness Pareto Optimization algorithm [14] is another multi-objective approach that attempts to maintain genetic diversity while optimizing fitness. Based on the genotypic age concept of ALPS, genotypic age is minimized in order to allow newer, younger solutions in different regions of the search space to direct search effort toward promising areas. This method allows a new random individual to be added to the population each generation without it being immediately dominated by older, more-fit individuals. This method avoids premature convergence without the need for an $O(n^2)$ distance calculation or a the extra overhead involved in maintaining a complicated hierarchical population structure. This method was demonstrated to outperform ALPS on a set of Symbolic Regression problems [14], and it is the basis upon which our approach is implemented.

3. METHOD

Inspired by the findings that GP populations tend to prematurely converge to an identical structure near the top of the trees [12], our approach uses *genetic markers* to actively avoid convergence to a particular rooted tree structure. This is achieved by maintaining a number of unique genetic markers in the population.

3.1 Genetic Markers

In order to place pressure on the population to explore different rooted tree structures, we developed *genetic markers* to gauge the genetic diversity among the top portions of the trees in the population. Figure 1 illustrates the process of generating a genetic marker by inspecting a partial tree. We perform a depth-first traversal, up to some depth, d, and the resulting Lisp-style partial expression becomes the genetic marker. This process is relatively inexpensive, and it can be done as the tree is parsed during the evaluation of

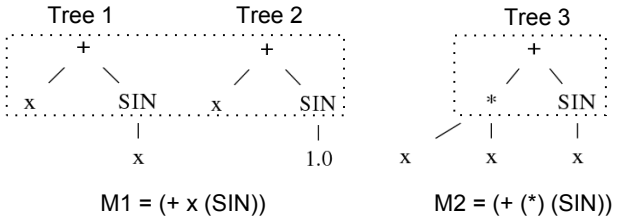

Figure 1: Performing a depth-first traversal from the root to depth 1 on three trees to create the two unique genetic markers $M1$ = "(+ x (SIN))" and $M2$ = "(+ (*) (SIN))".

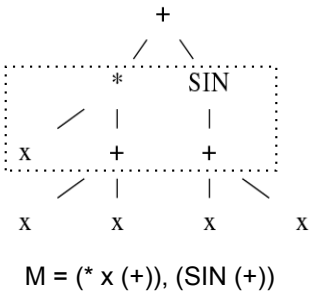

Figure 2: Performing a depth-first traversal starting at level 1 and stopping at level 2 to create the genetic marker M = "(* x (+)), (SIN (+))".

an individual to save extra computational expense. It is important to note that the genetic marker need not contain a complete subtree. Also, the genetic marker can begin at an arbitrary level in the tree, in which case, the genetic marker will represent an ordered list of fragments as in Figure 2. Requiring that the list of fragments remain ordered (left to right) ensures that we capture the structure of the tree at that level, even though we do not need to preserve the root node.

3.2 Genetic Marker Density

Using genetic markers as described in Section 3.1, individuals can be grouped as in Figure 1 in order to determine the density of each unique genetic marker. This density measure allows us to determine how widely spread a particular genetic marker is in the population. The density, ρ_m, of a genetic marker, m, is the fraction of individuals in the population that contain the genetic marker, and it can be calculated as follows:

$$\rho_m = \frac{|P_m|}{|P|}, P_m = \{\forall p \in P \mid M(p) = m\} \quad (1)$$

where, P is the current population, P_m is the subset of individuals in the population that contain the genetic marker m, and $M(p)$ is a function that returns the genetic marker of individual p.

This density measure only requires $O(n)$ time to assign density values to the individuals in the population. Although it is well-known that the fitness evaluation is the most expensive part of the evolutionary process for large problems, the density calculation still introduces less overhead than the tree distance method that we discussed in Section 2.1. The tree distance method requires $O(n^2)$ comparisons because each individual must be compared to every other individual. Additionally, unlike the distance method which needs to examine the entire tree, our density measure only inspects partial trees. Even though we only consider a small portion of each tree, our density measure is able to capture useful dynamics of how genetic material propagates throughout the population, which we will discuss further in Section 5.1.

In addition to gauging how genetic markers propagate throughout the population, this measure of density can also be used to prevent a genetic marker from becoming too dense. This is the central idea of our approach to avoiding premature convergence. We now discuss how we use genetic marker density to promote and maintain genetic diversity.

3.3 Optimizing Density and Fitness

Our approach, which is an extension of the Age-Fitness Pareto Optimization algorithm [14], attempts to prevent the rooted portions of the trees in the population from becoming too structurally similar by maintaining a diverse set of genetic markers. Instead of using genotypic age, we focus on the density of genetic markers to promote genetic diversity. The main intuition behind this is that by maintaining more genetic markers, we can avoid converging to any one structure and therefore maintain a higher level of genetic diversity in the population. We calculate the density of each unique genetic marker in the population, and then minimize density while optimizing fitness in our multi-objective approach.

Each generation, the algorithm proceeds by probabilistically performing crossover, reproduction, and mutation on randomly selected parents to breed the next generation. The parent population as well as the new population are temporarily retained, doubling the population size from n to $2n$. We then use the Pareto tournament selection method [14] to remove dominated individuals in order to shrink the population down from $2n$ back to the target population size, n. This works by randomly selecting a tournament of individuals and then removing all dominated individuals from the population until the target size is reached or all remaining individuals in the population are non-dominated.

In order to determine domination, we calculate the density of each unique genetic marker in the population, and assign each individual a density value corresponding to that of its genetic marker. An individual A is said to dominate individual B only if A is at least equal to B on all objective values and A is strictly better than B in at least one objective value. In other words, A and B can be equal in either density or fitness, but A must either be more fit than B or A's density value must be smaller. In the case that individuals are equal on both objectives, we attempt to break ties by tree size, preferring the individual that contains fewer nodes. If individuals are equal on both objectives, and they are the same size, we simply choose one in order to prevent the population from becoming too large due to ties. As Schmidt and Lipson note [14], it is theoretically possible for the current set of non-dominated individuals to be larger than the target population size after performing Pareto tournament selection, but we did not observe this in any of our experiments.

Since selection still favors fitter individuals, there is still an incentive to improve fitness, which means that fitter individuals will still receive more breeding opportunities. However,

since the density objective favors individuals that have less-represented genetic markers, highly-fit individuals are less likely to quickly sweep the population. This allows newer, perhaps less-fit, genotypes more of a chance to propagate valuable genetic material throughout the population. This also allows us to add a new randomly generated individual to the population at the end of each generation without it being immediately dominated in fitness alone.

One of the known reasons for premature convergence is that strong selection pressure causes an imbalanced sampling of genetic material because more fit individuals are constantly favored and selected [6]. The density objective actively avoids this issue by favoring less-represented genetic markers while still exploiting the fitter solutions. This is because individuals that contain denser genetic markers are required to be more fit in order to remain non-dominated.

4. EXPERIMENTAL SETUP

In order to determine the effectiveness of our approach, we compared its performance to the original Age-Fitness Pareto Optimization algorithm as well as the Age-Layered Population Structure (ALPS). For the comparison experiments, we used standard GP as a control, where we consider standard GP as a generational tree-based GP system using fitness-based tournament selection and subtree crossover with 90% internal node bias. We implemented the ALPS and Age-Fitness Pareto Optimization algorithms, as described in [5] and [14] respectively, in our GP system.[1] We compared the algorithms on the following problems described below.

Boolean 11-Multiplexer (11-MUX) as in [10]: This problem requires a boolean function to correctly decode $k = 3$ address bits as a binary number in order to select as output one of the $2^k = 8$ data bits. The fitness cases are all combinations of 11-bit binary strings, and fitness is the proportion of correctly output instances. We used the terminal set $T = \{A_0 - A_2, D_0 - D_7\}$, where each A_i is an address bit, and each D_i is a data bit. The function set contained the standard boolean operators $F = \{AND, OR, NOT, IF\}$.

Even *n*-Parity [10]: This problem requires a boolean function to determine for each binary number of length n whether or not the number has an even number of bits set. Therefore, the fitness cases are all 2^n combinations of n-bit binary strings, and fitness is the proportion of correctly classified fitness cases. We used the following terminal set: $T = \{D_0, D_1, ..., D_{n-2}, D_{n-1}\}$, where each D_i represents the ith bit in the n-bit binary string. The function set $F = \{AND, OR, NAND, NOR\}$ contained standard boolean operators for this problem, with the exclusion of the XOR operator. It has previously been noted [9, 10] that this problem is more difficult without the XOR operator due to the complexity of the required structure of the solution, so this provides more of a challenge for the algorithms. We compared the algorithms using $n = 5$.

Modified Quartic Symbolic Regression as in [9]: GP tries to evolve the function $4x^4 + 3x^3 + 2x^2 + x$ instead of the more common $x^4 + x^3 + x^2 + x$. The addition of coefficients to the terms makes the required tree structure of the solution more complex because we do not include constants in the terminal set, so the coefficients must be manufactured arithmetically. We used the terminal set $T = \{x\}$ and the function set $F = \{+, -, *, \%, SIN, COS, EXP, RLOG\}$ where % is the protected division operator, and $RLOG$ is the protected natural logarithm operator as described in [10]. We used 20 randomly selected points between $[-1.0, 1.0]$, and fitness is based on the sum of absolute errors between an individual's output value and the target function value, if the difference is greater than 0.01. In the results reported in Section 5, fitness is normalized between $[0, 1.0]$ by using the following formula:

$$f = \frac{1}{(1 + e)} \tag{2}$$

where e is the cumulative absolute error.

Parameter	Value
Population size	500
Tournament size	2
Crossover probability	0.90
Reproduction probability	0.10
Max tree depth	17
Max tree size	300
Age layers (ALPS)	10
Age gap (ALPS)	20
Elites (ALPS)	2
Elites (standard GP)	50
Random initialization	ramped half & half
Maximum evaluations	10 million

Table 1: GP settings for all problems.

Table 1 lists the GP settings that we used for all algorithms for all problems. We did not use point mutation, as initial experiments showed that it had minimal impact on the results. In order to provide an accurate analysis of the performance of each algorithm, we conducted 100 independent trials for each problem. Each trial lasted until the optimal fitness (1.0 in each problem) was reached or 10 million fitness evaluations were used. All results represent the mean across the 100 trials unless stated otherwise.

For the density/fitness algorithm, genetic markers were generated using the first two levels in the tree, starting at the root for all problems, with the exception of the Symbolic Regression problem for which we used the first three levels. Initial experiments suggested that using the first three levels yields better performance for the Symbolic Regression problem. This is likely due to the larger function set with more differences in arity among the functions.

5. RESULTS

We first validate the genetic marker definition with an analysis of genetic marker propagation in a standard GP setup contrasted with that of our approach. Next, since we are interested in avoiding premature convergence while finding the solution as fast as possible, we compare the algorithms in terms of the mean number of fitness evaluations used until the optimal solution was found. From the set of all 100 independent replicate trials, we recorded the total number of fitness evaluations at the time that the optimal fitness was reached, for the trials in which the optimal fitness was reached. We then compare the algorithms in terms of search quality (fitness over time). In order to provide a fair comparison, we present the best fitness over time in fitness

[1]All source code and configuration files necessary for reproducing our results are available at https://github.com/burks-pub/gecco2015

evaluations, averaged over the 100 trials, because the algorithms consume a different number of fitness evaluations per generation. Finally, we test the scalability of our approach on the Even n-Parity problem. For space considerations, all tables and figures refer to standard GP as SGP, Age-Fitness Pareto Optimization as A/F, and Density/Fitness as D/F.

5.1 Genetic Marker Analysis

We analyzed the density of all genetic markers among the top 6 levels of the trees in the population for 1000 generations, otherwise using the same experimental settings detailed in Section 4. We use generations here because we are interested in how genetic markers change after each round of breeding. We simultaneously tracked the genetic markers from all levels in a top-down fashion, using two levels at a time. For each pair of levels, we recorded the genetic marker with the maximum density, using the density measure described in Section 3.2. For each level, i, in each tree in the population, genetic markers were generated as we described in Section 3.1, starting at level i and continuing through level j. For example, the genetic marker at levels $L_{0:1}$ will contain all the nodes of a tree from the root (level 0) through level 1. Except where noted, the results of this analysis were similar for all the benchmark problems, so we will only show results for the Symbolic Regression problem due to space considerations. The results shown represent the mean across 100 independent trials.

Figure 3 shows the mean maximum density of all genetic markers across the top 6 levels of the trees in the population in a standard GP setup. The population very quickly becomes nearly identical among the topmost portions of the trees, as a large percentage of the population shares the same genetic marker in the first few levels. By generation 200, nearly 60 percent of the population is identical across the top 6 levels in the trees. This confirms earlier findings that a large percentage of the population quickly converges to the same structure [12]. More importantly, this shows that our definition of genetic markers and the corresponding density measure captures useful dynamics of GP populations.

Figure 4 shows the effect that our approach has on genetic diversity, which highly contrasts with that of a standard GP setup as in Figure 3. This shows that our algorithm maintains a very diverse set of tree structures in the population while preventing a single structure from becoming too dense.

Interestingly, for the Symbolic Regression problem, the maximum density among all genetic markers in the topmost portion of the trees ($L_{0:1}$) still increases over time using our approach. However, this is not nearly as rapid as in the standard GP setup shown in Figure 3. As we discussed in Section 3.3, individuals with more dense genetic markers must have relatively high fitness in order to dominate individuals that have more unique genetic markers. This dynamic places a selective pressure on the population to improve in fitness while still exploring newer rooted tree structures. Figure 4 also shows that our approach maintains a very diverse set of genetic markers across the lower levels in the trees, even though we only explicitly minimize density at the topmost genetic marker level. This demonstrates that diversity among the upper portions of the trees has a major impact on the diversity at deeper levels in the trees.

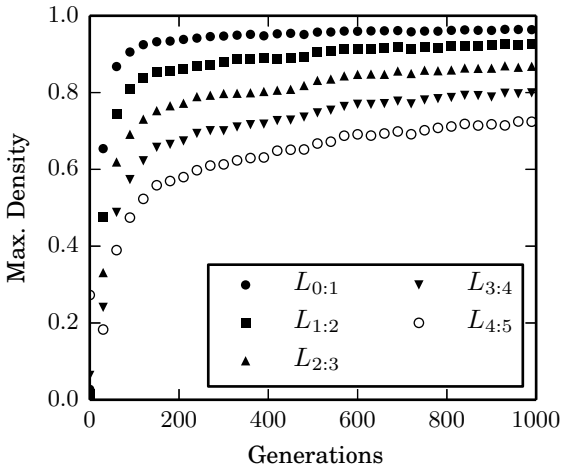

Figure 3: Mean maximum genetic marker density in the top 6 levels of the trees in the population using a standard GP setup on the Quartic Symbolic Regression benchmark problem.

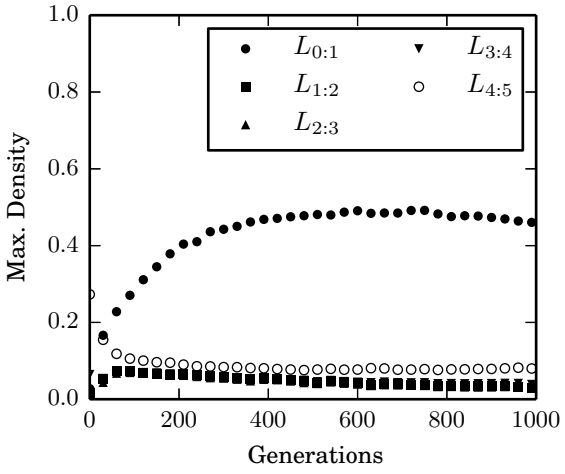

Figure 4: Mean maximum genetic marker density in the top 6 levels of the trees in the population using the density/fitness algorithm on the Quartic Symbolic Regression benchmark problem. Note that values for levels $L_{1:2}$, $L_{2:3}$, and $L_{3:4}$ are overlapping.

5.2 Performance

Figures 5, 6, and 7 show that our algorithm reaches the optimal fitness value in significantly fewer evaluations than the other algorithms in each benchmark problem (Mann-Whitney U test, $p < 0.001$). Table 2 shows that across all problems, D/F reaches optimal fitness at least twice as fast as the second fastest algorithm. Notably, D/F is 5 times faster than A/F on the Even 5-Parity problem, and over 15 times faster than ALPS on the 11 Multiplexer problem. These results show that although ALPS and A/F are successful at avoiding premature convergence, they can do so at the cost of requiring more fitness evaluations to find the optimal solution. However, our approach demonstrates that it

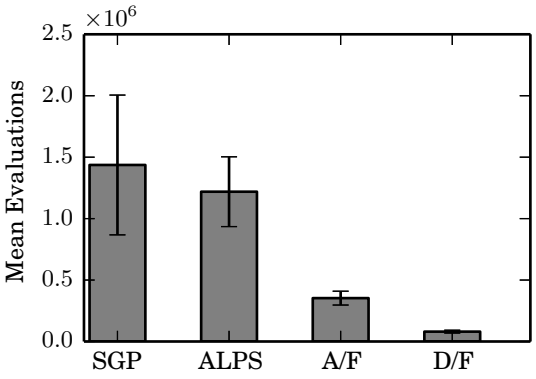

Figure 5: Mean number of evaluations (millions) until optimal fitness was reached on the Boolean 11-Multiplexer problem. Error bars represent the 95% confidence interval around the mean.

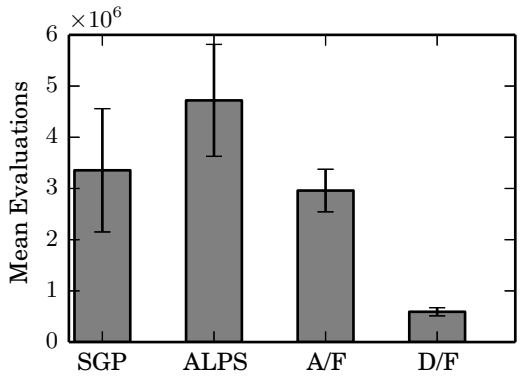

Figure 6: Mean number of evaluations (millions) until optimal fitness was reached on the Even 5-Parity problem. Error bars represent the 95% confidence interval around the mean.

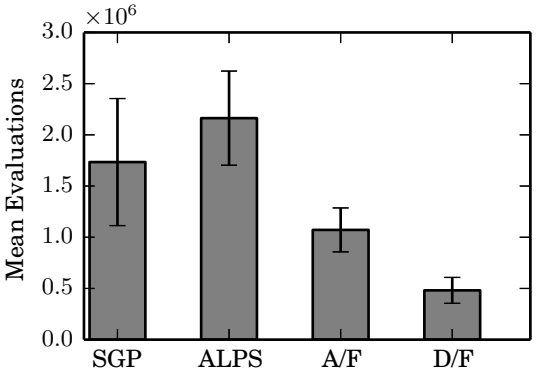

Figure 7: Mean number of evaluations (millions) until optimal fitness was reached on the modified Quartic Symbolic Regression problem. Error bars represent the 95% confidence interval around the mean.

	11-MUX	5-Parity	Regression
SGP	17.89	5.67	3.60
ALPS	15.18	7.98	4.49
A/F	4.40	5.00	2.23

Table 2: Performance factors compared to D/F mean evaluations required to find the solution for 11-MUX (80,304), 5-Parity (591,788) and Regression (481,249).

	11-MUX	5-Parity	Regression
SGP	57	23	56
ALPS	98	25	93
A/F	100	97	100
D/F	100	100	98

Table 3: Number of runs in which the optimal fitness was reached for each problem.

is possible to avoid premature convergence while still finding the solution in a relatively small number of fitness evaluations.

Table 3 shows that D/F found a solution in all of the trials for the 11 Multiplexer and Even 5-Parity problems, and in 98% of the runs for the Symbolic Regression problem. We conducted an additional 100 trials on the Symbolic Regression problem using genetic markers composed of only the first two levels in the trees as in the other problems. With this configuration, D/F found a solution in all of the runs. However, the mean number of evaluations required to find a solution was 1.9 times slower than when genetic markers were composed of the first three levels in the trees. This suggests that for some problems, there may be a trade-off between robustness and efficiency for our algorithm, depending on the composition of the genetic markers. This also suggests that the problem domain may influence the genetic marker dynamics.

5.3 Search Quality

Our second goal is to improve the general quality of search, so next we compare the algorithms in terms of the mean best fitness over time (fitness evaluations). Figures 8, 9, and 10 plot the mean best fitness value over time for each algorithm on each problem. Near the onset of the run, SGP and ALPS reach higher fitness values than both Pareto optimization approaches. This is because A/F and D/F are not only optimizing fitness, but also age and density, respectively. However, A/F and D/F surpass SGP and ALPS, reaching the higher fitness values much earlier in the run, on average. Figures 8 - 10 also show that D/F reaches the higher fitness values much earlier than A/F. This gives rise to the dramatic differences in the required number of evaluations to reach the optimal fitness, as reported in Figures 5 - 7.

5.4 Scalability

We performed an additional set of experiments in order to determine how well our algorithm scales on the n-Parity problem. We do not include results for SGP or ALPS for $n > 5$ because SGP only found a solution in one trial and ALPS only found a solution in two trials for $n = 6$ under these settings. Similarly, we do not report results for A/F beyond $n = 6$, because it only found one solution for $n = 7$. We

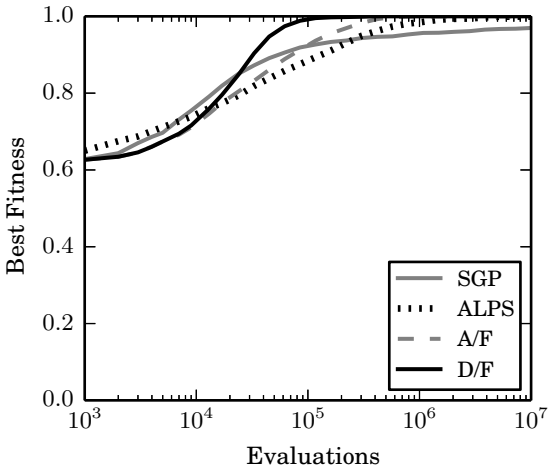

Figure 8: Mean best fitness over time (fitness evaluations) on the Boolean 11-Multiplexer problem. Error bars are omitted for clarity.

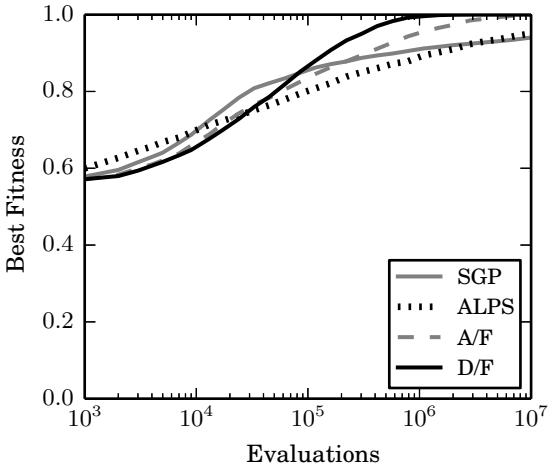

Figure 9: Mean best fitness over time (fitness evaluations) on the Even 5-Parity problem. Error bars are omitted for clarity.

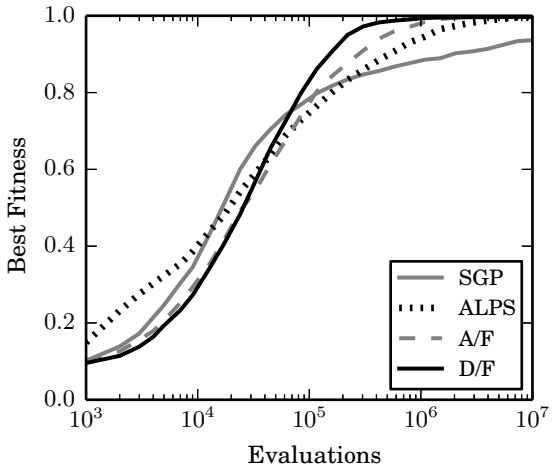

Figure 10: Mean best fitness over time (fitness evaluations) on the modified Quartic Symbolic Regression problem. Error bars are omitted for clarity.

	n = 5	**n = 6**	**n = 7**	**n = 8**
SGP	3.36 (23)	–	–	–
ALPS	4.72 (25)	–	–	–
A/F	2.96 (97)	6.56 (6)	–	–
D/F	0.59 (100)	3.08 (88)	4.48 (37)	5.73 (6)

Table 4: Mean number (in millions) of fitness evaluations required to reach the optimal fitness value for the Even n-Parity problem. Numbers in parentheses indicate the number of trials, out of 100 trials, in which a solution was found.

used the same function set as described in Section 4, so this problem becomes even more difficult as n increases because the required structure for the solution becomes increasingly complex without the XOR operator.

Table 4 shows the mean number of fitness evaluations, in millions, required for $n = 5, 6, 7, 8$, as well as the percentage of trials in which a solution was found. D/F is able to find a solution at a significantly higher rate than A/F in the Even 6-Parity problem. For $n = 6$, our approach finds a solution more than twice as fast as A/F. Additionally, our approach requires less evaluations for both $n = 7$ and $n = 8$ than A/F for $n = 6$. Of course, as n increases, in the absence of the XOR operator, the success rate of our algorithm decreases, and the required number of evaluations to find a solution increases. However, our approach still found a solution in 6 trials with $n = 8$ without the XOR operator, while the other algorithms did not find a solution in the majority of the trials with $n = 6$.

6. CONCLUSIONS

We have presented a new, efficient technique for avoiding premature convergence and improving search quality in tree-based genetic programming. By using tree fragments as genetic markers, the Density/Fitness algorithm prevents the population from converging to a single structure by incorporating genetic marker density into the optimization process. It has already been demonstrated that standard GP

often experiences a rapid loss of genetic diversity, wherein the topmost portions of the trees in the population quickly converge to the same structure [12]. Our results show that by actively combating this phenomenon, we can significantly improve the performance, search quality, and scalability of GP, transforming the convergent nature of traditional GP into a more sustainable search algorithm.

In this study, we restricted the genetic markers to only contain the topmost portion of each tree in the population. This loose genotypic diversity measure, which does not consider the rest of the genotype (lower levels in the tree), significantly improved performance and search quality over standard GP as well as the other algorithms that we compared against. However, some questions that arise are how to effectively and efficiently incorporate more of the genotype into the algorithm, and whether or not this would yield further improvements.

One simple approach to this would be to increase the number of levels used in each genetic marker over time in order to focus on different levels in the trees in the population. However, the computational overhead increases with depth, as more of the tree needs to be traversed in order to create the genetic marker. An alternative approach could simply change the level at which we begin creating the genetic marker, as in Figure 2. Initial experiments with the latter approach did not show an improvement over the results we presented here, although further investigation into the rate at which the level changes, or different methods for changing the genetic markers may prove beneficial.

7. ACKNOWLEDGMENTS

This material is based in part upon work supported by the National Science Foundation under Cooperative Agreement No. DBI-0939454. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

This work was supported in part by Michigan State University through computational resources provided by the Institute for Cyber-Enabled Research.

8. REFERENCES

[1] E. Burke, S. Gustafson, and G. Kendall. Diversity in genetic programming: An analysis of measures and correlation with fitness. *Evolutionary Computation, IEEE Transactions on*, 8(1):47–62, 2004.

[2] E. de Jong, R. Watson, and J. Pollack. Reducing bloat and promoting diversity using multi-objective methods. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO 2001, pages 11–18. Morgan Kaufmann, 2001.

[3] C. Gathercole and P. Ross. An adverse interaction between crossover and restricted tree depth in genetic programming. In *Proceedings of the 1st Annual Conference on Genetic Programming*, pages 291–296, Cambridge, MA, USA, 1996. MIT Press.

[4] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[5] G. S. Hornby. Alps: The age-layered population structure for reducing the problem of premature convergence. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 815–822, New York, NY, USA, 2006. ACM.

[6] J. Hu, E. Goodman, K. Seo, Z. Fan, and R. Rosenberg. The hierarchical fair competition (hfc) framework for sustainable evolutionary algorithms. *Evol. Comput.*, 13(2):241–277, June 2005.

[7] J. Hu, K. Seo, S. Li, Z. Fan, R. C. Rosenberg, and E. D. Goodman. Structure fitness sharing (sfs) for evolutionary design by genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 780–787. Morgan Kaufmann Publishers, 2002.

[8] M. Hutter and S. Legg. Fitness uniform optimization. *Evolutionary Computation, IEEE Transactions on*, 10(5):568–589, 2006.

[9] D. Jackson. Mutation as a diversity enhancing mechanism in genetic programming. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, pages 1371–1378, New York, NY, USA, 2011. ACM.

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

[11] J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evol. Comput.*, 19(2):189–223, June 2011.

[12] N. F. McPhee and N. J. Hopper. Analysis of genetic diversity through population history. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1112–1120. Morgan Kaufmann, 1999.

[13] R. Poli and W. Langdon. On the search properties of different crossover operators in genetic programming. In *Genetic Programming*, pages 293–301. Morgan Kaufmann, 1998.

[14] M. Schmidt and H. Lipson. Age-fitness pareto optimization. *Genetic Programming Theory and Practice VIII*, 8:129–146, 2011.