A New Methodology for the GP Theory Toolbox

Jeffrey K. Bassett
George Mason University
Fairfax, USA
jbassett@cs.gmu.edu

Uday Kamath
George Mason University
Fairfax, USA
ukamath@gmu.edu

Kenneth A. De Jong
George Mason University
Fairfax, USA
kdejong@gmu.edu

ABSTRACT

Recently Quantitative Genetics has been successfully employed to understand and improve operators in some Evolutionary Algorithms (EAs) implementations. This theory offers a phenotypic view of an algorithm's behavior at a population level, and suggests new ways of quantifying and measuring concepts such as exploration and exploitation. In this paper, we extend the quantitative genetics approach for use with Genetic Programming (GP), adding it to the set of GP analysis techniques. We use it in combination with some existing diversity and bloat measurement tools to measure, analyze and predict the evolutionary behavior of several GP algorithms. GP specific benchmark problems, such as ant trail and symbolic regression, are used to provide new insight into how various evolutionary forces work in combination to affect the search process. Finally, using the tools, a multivariate phenotypic crossover operator is designed to both improve performance and control bloat on the difficult ant trail problem.

Track: Genetic Programming

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: Artificial Intelligence—*Learning* [Knowledge acquisition]; F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Theory, Algorithms, Performance

Keywords

Genetic Programming, Quantitative Genetics, Evolutionary Algorithms, Bloat

1. INTRODUCTION

Recently, Quantitative Genetics has been successfully applied to Evolutionary Algorithms for the purpose of creating customized reproductive operators for different representations [3]. Unlike most EA theories, it can be applied to new representations with relative ease, because it only operates at the phenotype level, and ignores what happens to the genetic structure. This is achieved by defining a set of phenotypic traits, and then characterizing populations as probability distributions in this trait space. Various equations, like Price's Theorem, then model how evolutionary forces affect the trajectory of populations through the phenotype space. These models are well suited for doing post-run analysis, making them ideal for building analysis tools for studying existing EAs, and explaining their behaviors. Certain terms in these equations can be interpreted as measuring exploration and exploitation, allowing one to get a sense for how well an algorithm is performing these tasks, and if it is creating the proper balance between them.

Since these tools have proven useful for analyzing EAs, in this paper we show how they can be effectively used to analyze GP algorithms. Specifically we answer the following questions:

1. How much phenotypic variation are the reproductive operators generating?
2. What impact does bloat have on the problem and what role does it play in changing the search dynamics?
3. How do our choices of a selection mechanism affect the balance between exploration and exploitation?

To do this, we will first need to review how the theory and tools work, and what to look for in order to interpret the results. We introduce quantitative genetics by examining some simple EAs and problems. This provides a demonstration of the exploration and exploitation measurements in an environment where it can be easily grasped, and sets the stage for various behavioral analogies with genetic programming later. We then take a multi-modal, deceptive GP problem known as the ant trail problem, and perform various experiments involving different breeding and selection mechanisms in order to examine the GP behavior and show the explanatory power of the new methodology. Certain GP behaviors highlighted through the experiments not only confirm research done previously but also add some novel insights about GP. The general nature of our analysis is validated by showing similar behaviors in the symbolic regression problem. Employing the lessons learned, we then build a multivariate phenotypic crossover operator and show its success in finding better fit individuals while controlling the bloat on the ant trail problem.

GECCO'12, July 7-11, 2012, Philadelphia, Pennsylvania, USA.

2. RELATED WORK

Some of the earliest Evolutionary Computation (EC) work related to quantitative genetics was the recognition that operators with high parent-offspring fitness correlation tend to produce better results [14]. This is closely related to the notion of heritability. A more formal introduction of quantitative genetics theory for use with EAs was done by Altenberg [2] in which he used Price's Theorem [18] as a method for analyzing EA operators to improve their evolvability.

Mühlenbein and Schlierkamp-Voosen [16] developed the Breeder Genetic Algorithm (BGA) based on the equation for the response to selection. Further work with their gene-pool crossover operator was based on an analysis of that same equation. This work ultimately led to the development of Estimation of Distribution Algorithms (EDAs) [15].

Langdon [8] developed tools based on quantitative genetics for analyzing EA performance. He used both Price's Theorem and Fisher's Fundamental Theorem [17] to model GP gene frequencies, and how they change in the population over time.

Bassett and De Jong [3] adapted techniques from multivariate quantitative genetics [7] for use with EAs. This allowed them to define a set of traits that describe a phenotypic version of the search space, and observe an algorithm's behavior from this perspective.

Although not directly related to quantitative genetics, Radcliffe [19] developed a theoretical framework that has certain similarities to the methods used in quantitative genetics. He extended the schema theorem to a more general approach that can be applicable at a phenotypic level.

3. QUANTITATIVE GENETICS BASICS

Quantitative genetics [6, 20] is a theory that is capable of explaining evolutionary dynamics without any reference to the underlying genetics structure of the individuals. This is achieved by dealing with individuals in terms of a set of quantitative traits. A quantitative trait is any aspect of an individual that can be measured as a number. These create an abstraction that allows us to observe the behaviors of an algorithm at the phenotypic level. If we define a set of traits that describe the search space at a phenotypic level, we can observe how the population behaves as it traverses that space.

The theory characterizes populations as joint probability distributions over the set of traits. Equations are then used to describe how these distributions change from one generation to the next as a result of evolutionary forces like selection, reproduction-with-variation, and heritability. Thus the equations suggest methods for measuring these forces and observing them throughout an EA run, giving us more information for diagnosing problems with our algorithms.

3.1 Quantitative Genetics Equations

The quantitative genetics models are built on the idea of a non-overlapping generational evolutionary process. This makes them easily applicable to generational EAs like Genetic Algorithms (GAs) and GP. Figure 1 provides a model of two successive generations during an EA run. A subset of parents (left) are selected and produce offspring, either through crossover, mutation or cloning. Directed edges are draw from each selected parent to all the offspring (right) that it produces.

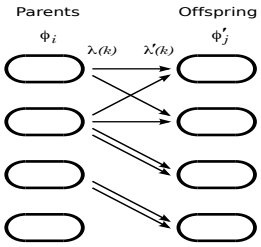

Figure 1: Sample of two successive generations of an EA, for explaining the equations. The parents (left) generate offspring (right). The arrows connect selected parents with the offspring they produce.

When constructing the model, it is important that each directed edge represent the same amount of "influence" that a parent has on it's offspring. In the figure, each edge represents an influence of 1/2. This means that two edges must be drawn between parent and offspring in instances where only cloning or mutation are performed. This has the effect of giving the same denominator to certain fractional values within the equations so that they can be added properly.

A vector of quantitative traits ϕ_i is associated with each parent i. Similarly a vector of traits ϕ'_j is associated with each offspring j. The relationship between parents and offspring is defined by the set of edges, forming a directed graph. The two functions $\lambda(k)$ and $\lambda'(k)$ are defined to get the index of corresponding parent and offspring, for a given edge k.

To abbreviate, we refer to the traits of all parents as ϕ and all offspring as ϕ'. Similarly ϕ_λ refers to all traits of the *selected* parents, and $\phi'_{\lambda'}$ again refers to all the traits of the offspring, although in the case of figure 1 there are two copies of each child.

We define several covariance matrices to describe the population distributions and the forces that cause them to change. $\mathbf{P} = \mathrm{Var}(\phi_\lambda)$ and $\mathbf{O} = \mathrm{Var}(\phi')$ are covariance matrices that describe the trait distributions of actual populations. $\mathbf{P}$ describes the distributions of the *selected* parents, and $\mathbf{O}$ describes the distribution of the offspring. $\mathbf{D} = \mathrm{Var}(\phi'_{\lambda'} - \phi_\lambda)$ describes the amount of trait variation that is introduced by the reproductive operators, and $\mathbf{G}' = \mathrm{Cov}(\phi'_{\lambda'}, \phi_\lambda)$ is a cross-covariance matrix that can be thought of as quantifying the amount of variation from $\mathbf{P}$ that is retained in $\mathbf{O}$.

Given these matrices, we can now describe how the population trait distributions change from one generation to the next using the following equation:

$$\mathbf{O} = 2\mathbf{G}' + \mathbf{D} - \mathbf{P}, \tag{1}$$

which can be rewritten as

$$\mathbf{O} = \mathbf{P}[2\mathbf{G}'\mathbf{P}^{-1} + \mathbf{D}\mathbf{P}^{-1} - \mathbf{I}], \tag{2}$$

where $\mathbf{I}$ is the identity matrix. In this case, we can view everything within the brackets as defining a transformation matrix that transforms the distribution of the traits of the selected parents ($\mathbf{P}$) into the distribution of the offspring population traits ($\mathbf{O}$).

The factor $\mathbf{G}'\mathbf{P}^{-1}$ is similar, but not the same as, the quantitative genetics notion of narrow-sense heritability (commonly just called heritability). Ours describes the average similarity between an offspring and *one* of it's parents. Cal-

culating heritability in this way exposes the term $\mathbf{DP}^{-1}$, which we refer to as perturbation. It describes the amount of new phenotypic variation that the operators are introducing into the population relative to what already exists. Perturbation can be thought of as measuring an operator's capacity for exploration. Heritability, on the other hand, measures how well traits are retained during reproduction. If heritability is low, that indicates that there an unexpected bias in the search. Thus, operators with high heritability can be thought of as better exploiting the information that is in the population.

Another relationship that can be drawn from equation 2 is $\mathbf{OP}^{-1}$. This does not have a corresponding concept in biology, although it is similar in some ways to broad-sense heritability and repeatability. This term describes the similarity of the parent and offspring *populations*, and so we refer to it as population heritability. This is another measure of exploitation, in addition to narrow-sense heritability. We think it is the better choice because it is measuring the larger scale behavior of the EA.

3.2 Describing Matrices as Scalar Values

The information we are interested in observing (heritability: $\mathbf{G}'\mathbf{P}^{-1}$, perturbation: $\mathbf{DP}^{-1}$ and population heritability: $\mathbf{OP}^{-1}$) are actually matrices. As such they contain a lot of information and can be difficult to analyze. It would be preferable to represent this information as scalar values. Certainly a lot of information would be lost, but if we can retain what is most important, it will be easier to interpret the results.

We have chosen to do this by using the trace (tr) function. It is simple to calculate (the sum of the diagonal elements in the matrix) and it describes the total variance within all the traits. It is also equal to the sum of the eigenvalues of the matrix.

To avoid issues that can occur when taking the inverse of a matrix, we have also decided to instead perform division on the trace values of the two component matrices. For example, for the heritability matrix $\mathbf{G}'\mathbf{P}^{-1}$, we calculate the scalar value as $\mathrm{tr}(\mathbf{G}')/\,\mathrm{tr}(\mathbf{P})$. Similarly, $\mathrm{tr}(\mathbf{D})/\,\mathrm{tr}(\mathbf{P})$ would represent perturbation, and $\mathrm{tr}(\mathbf{O})/\,\mathrm{tr}(\mathbf{P})$ gives us a scalar measure of population heritability.

4. EXAMPLE EA ANALYSES

In order to illustrate the capabilities of these tools, we present some examples of their use on some simple EAs. The idea here is to demonstrate the behavior of some canonical algorithms and operators that are known to perform well on a simple problem. This should provide some understanding of how these tools work before we start examining the GP operators. In the two examples we provide, the EA will use a linear genome of 10 real-valued genes, binary tournament selection, and a population size of 100. Thirty runs of the EA are performed in each case.

4.1 Problem

The sample problem we use is often referred to as the sphere function, and is defined as

$$f(X) = \sum_{i=0}^{|X|} x_i^2, \tag{3}$$

where X is a vector of parameters, x_i is the value of the i^{th} parameter, and the total number of parameters is 10. The task is to find the minimum value, which is at the origin.

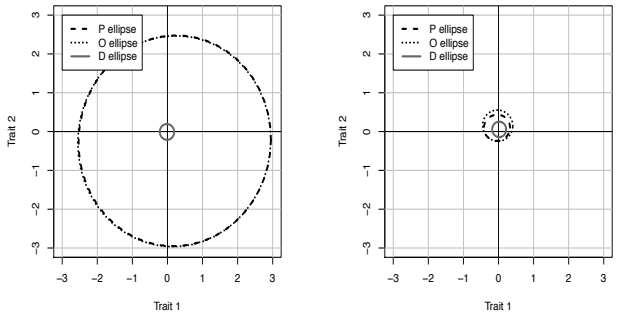

Figure 2: Covariance matrix ellipsoids for P, O and D for the Gaussian mutation EA. Ellipsoids are projected onto the trait1-trait2 plane at generations 1 (left) and 199 (right).

4.2 Quantitative Traits

Our approach is to define a set of traits that describe the phenotypic search space of our problem. When working with function optimization problems, one tends to think of the problem as a landscape defined over a set of parameters – the same parameters defined by X in equation 3. That makes these parameters an ideal set of traits.

4.3 Analysis

The first EA we examine uses the Gaussian mutation operator with a fixed standard deviation of 0.2. Figure 2 plots cross-sections of the ellipsoids of the **P**, **O** and **D** matrices, as projected on the plane defined by trait1 and trait2. Each ellipse represents one standard deviation of the joint distributions defined by the covariance matrices. The left plot is for generation 1, and we can see that the selected parent (**P**) and offspring (**O**) ellipses occupy almost exactly the same space. The ellipse for the mutation (**D**) is much smaller, by comparison, showing that the effects of mutation are small relative to the population.

In the plot on the right, which is from generation 199, things have changed a bit. Here we see that while **D** is the same as in generation 1, it is much larger relative to **P** and **O**. As a result, the offspring population is pushed out to cover a larger area than the parent population does. At this point, selection and mutation have come into equilibrium. Each generation oscillates back and forth, with selection taking **O** and reducing it down to the smaller **P**, and mutation inflating it again, creating a new **O** that is roughly the same as in the previous generation.

We can view these data another way by plotting the total variation in each matrix using the trace function. The leftmost plot in figure 3 plots the trace values for the matrices **P**, **O** and **D** and $\mathbf{G}'$. We see that the curves for **P** and **O** rapidly decrease as the population converges on the solution in phenotype space, but they never reach zero since the population never converges completely. The curves level off at the point when the selection pressure and mutation rate are in equilibrium. Similarly, $\mathbf{G}'$ follows along exactly with the **P** curve. The **D** curve remains constant at 0.4 throughout

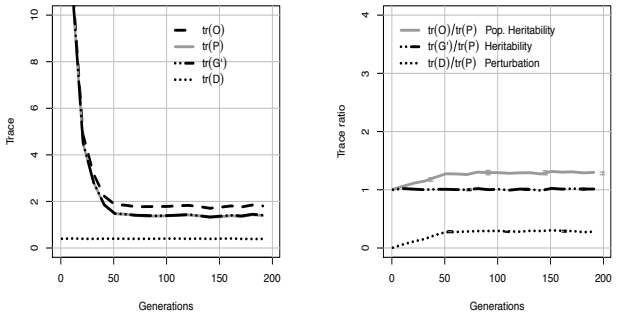

Figure 3: Real valued EA on sphere function with fixed Gaussian mutation.

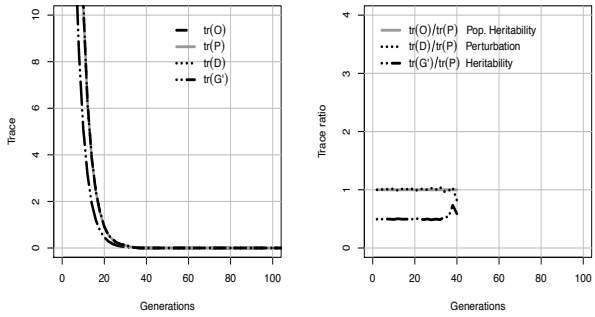

Figure 4: Real valued EA on sphere function with uniform crossover.

the run, which is the value we would expect given a standard deviation of 0.2 and 10 dimensions.

The right-most plot in figure 3 shows the ratios for narrow-sense heritability ($\mathrm{tr}(\mathbf{G}')/\mathrm{tr}(\mathbf{P})$), population heritability ($\mathrm{tr}(\mathbf{O})/\mathrm{tr}(\mathbf{P})$), and perturbation ($\mathrm{tr}(\mathbf{D})/\mathrm{tr}(\mathbf{P})$). The ideal value for both forms of heritability is 1.0, and we see that narrow-sense heritability stays at this value throughout the run. The amount of perturbation starts out low, but as the population converges, the amount of variation that mutation is adding increases relative to the amount of variation in the population, until it reaches the equilibrium point. This same trend is also reflected in the population heritability.

To interpret this plot, what we see is exploration increasing toward the end of the run until we reach a point where selection can no longer push against it. The fact that population heritability has moved so far away from 1.0 means that the algorithm no longer has the ability to exploit new information that it gains via selection.

In a second experiment, we analyze the behavior of an EA that uses only uniform crossover with a 0.5 probability of exchanging any two genes between individuals. We see some interesting differences between the behavior with this operator and the previous one. For example, we see in the left-hand plot of figure 4 that the $\mathbf{D}$ curve is no longer constant, but it adapts. As the phenotypic variance in the population converges, so does the the amount of phenotypic variance added by the operator. This adaptive quality allows the population to converge down to a single point in phenotype space.

When we examine the ratios (figure 4, right), the first thing we notice is that the lines all end at about generation 40. This is approximately the generation when the populations completely converge. Once a population has converged, it is no longer included in the calculations in order to avoid errors caused by dividing by zero. Another thing we notice here is that perturbation is much higher than before, staying at roughly 1.0 throughout the run, indicating a much greater ability to explore than before. At the same time, there is a corresponding reduction in heritability.

What we see with population heritability is probably more interesting, though. It maintains its optimal value of 1.0 throughout much of the run. This means that it is able to exploit new information up until the point when the population converges. This does not necessarily mean that it converged on the global optimum. All it means is that the operator has exhausted it's ability to generate any more phenotypic variation. At this point, both exploration and exploitation stop.

We can observe something very interesting here. Population heritability remains at the ideal level of 1 throughout the run, and at the same time perturbation also remains quite high. If we accept the idea that these really represent exploitation and exploration, then the relationship between these two concepts may not be a simple trade-off, as is often thought. With care, it may be possible to design operators that are good at achieving both at the same time.

We have observed mutation and crossover working independently, but in many EAs the two are used together. When this is done, the corresponding plots of the matrices are essentially a hybrid of those above. The resulting curve for $\mathbf{D}$ will adapt while crossover is dominant in the early stages of the run, but will then reach a constant value toward the end when selection pressure and mutation rate come into equilibrium. The curve for narrow-sense heritability is the same as with crossover-only, remaining constant at 0.5. Population heritability is like the crossover-only in the early generations, remaining constant at 1, and then becomes like mutation in the later generations, rising up to the same equilibrium point. Perturbation is more like the effects of the two operators added together, and in fact looks identical to the population heritability curve.

It is also interesting to consider what happens in certain extreme cases. For example, when the only operator used is cloning, perturbation ($\mathrm{tr}(\mathbf{D})/\mathrm{tr}(\mathbf{P})$) is zero and population heritability ($\mathrm{tr}(\mathbf{O})/\mathrm{tr}(\mathbf{P})$) is one. This means that exploration is low, but exploitation is high. At the other end of the spectrum, an algorithm with a very high mutation rate is little more than a random search algorithm. It will have high perturbation, but it will also have population heritability values greater than one. Therefore it will be capable of exploration, but not able to exploit the information it gains about the search space.

5. EXAMPLE GP ANALYSES

We now illustrate how these ideas can be used to gain insight into GP algorithms. While many of our observations are already well known within the GP community, we consider this a benefit as it only serves to validate what these new tools are capable of. To assist in our analyses, we will

also be using some analysis tools that have been designed specifically for GP.

5.1 GP-Specific Analysis Tools

5.1.1 *Genetic Diversity using Lineage*

Various genotypic diversity measurements have been explored in GP, like tree-edit distances, genetic lineages, entropy, etc., for understanding the genotypic behavior and correlating it with phenotypic behaviors [4]. Genetic Lineage has shown promise on problems where other approaches have failed to show significant correlation to fitness [5]. In the context of GP, with individuals as trees, when an operator like crossover breeds and produces an offspring, the offspring that has the root node of parent has the lineage of that parent. This gives a way to measure the distribution of lineage over generations and also change of unique lineages or genetic convergence based on selection and the breeding operator for the population. Unique ancestors or genetic lineage will be used for measuring the diversity or genotypic convergence of a population in the experiments.

5.1.2 *Bloat Measure*

Since bloat plays an important role in analyzing the GP behavior, bloat has to be measured quantitatively and the metric we use is based on recent research on quantitative definition of bloat [21]

$$bloat(g) = \frac{(\overline{\delta}(g) - \overline{\delta}(0))/\overline{\delta}(0)}{\overline{f}(0) - \overline{f}(g))/\overline{f}(0)}, \quad (4)$$

where $\overline{\delta}(g)$ is the average number of nodes per individual in the population at generation g, and $\overline{f}(g)$ is the average fitness of individuals in the population at generation g.

5.2 Problem Studied

To study the effects of Quantitative Genetic metrics we chose the Santa-Fe artificial ant problem. This is representative of many single agent search problems and also is considered to be highly deceptive for genetic programming as there is no guiding force to encourage the ant to travel any particular path. It was also shown that the best solution was obtained using GP as compared to other heuristics like simulated annealing, hill climbing and GA [11].

5.2.1 *Quantitative Traits for Agent Ant*

Here various search properties are devised to measure the behavior of an agent and use this information as phenotypic traits in equations 1 and 2.

Sum of Distances from Last Trail: This is the Manhattan distance computed for all the moves from the ant's location to its last position on the trail. Measures the "moving away effect" of the agent to the trail.

Sum of Distances to Closest Point on Trail: This is the Manhattan distance computed for all the moves from the ant's location to the closest point on the trail. This trait measures the "closeness" of the agent to the trail.

Sum of Distances from Last Point: This is the Manhattan distance computed for all the moves from the ant's current location to its previous location. This trait measures the "geometric displacement effect" irrespective of trail for the agent.

Count of Null Movements: This is the count of zero movements, i.e. no change in displacement for the agent, summed over all its moves. This trait measures the effect of changing code not altering the behavior of the agent.

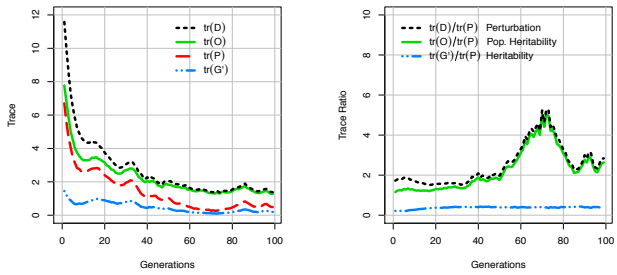

Figure 5: Standard subtree crossover, tournament size 2, depth limit 17

Because these traits represent discrete totals instead of continuous values, they are considered to be semiquantitative traits [12]. While such traits are often used when there are enough distinct values [6, Chapter6], care should be taken that none of the matrices have negative eigenvalues. This does actually happen on occasion in our experiments when calculating the $\mathbf{G}'$ matrix, and should be addressed in future work.

Most of these traits show exponential distribution and hence they are transformed to the new set of derived traits that have a normal distribution, as suggested in [6, Chapter 17]. This is done by taking the *log* of the original values.

We will now try to answer some of the questions posed in the introduction about the GP search process using our quantitative genetics tools. In particular, we will examine premature population convergence, selection and operator interaction and the effects of bloat on search. Experiments were conducted with ECJ [13] using various standard default parameters like population size of 1024, a depth limit of 17, and the ramped half and half method (min/max of 2 and 6) for creating individuals in the initial population.

5.3 Variation from the Operators

We wanted to get an initial sense of how much phenotypic variation the operators were producing. Since subtree crossover is probably the most commonly used GP operator, we chose to make it the focus of this paper. We ran an experiment using subtree crossover with 1.0 probability with a depth limit if 17 and tournament selection with a tournament size of 2. The results are plotted in figure 5. The plot on the left shows the trace values for $\mathbf{P}$, $\mathbf{O}$, $\mathbf{D}$, and $\mathbf{G}'$, and the other shows the ratios of $\mathrm{tr}(\mathbf{D})/\mathrm{tr}(\mathbf{P})$, $\mathrm{tr}(\mathbf{G}')/\mathrm{tr}(\mathbf{P})$, and $\mathrm{tr}(\mathbf{O})/\mathrm{tr}(\mathbf{P})$.

The plot of $\mathbf{D}$ gives us the most direct information about the operator's phenotypic effects. We can see that its behavior is similar to that of an EA using both crossover and mutation. There is a rapid drop in variance early in the run that eases over time and eventually levels off before reaching zero, indicating that the operators are always adding some amount of variance to the population.

The variance reduction behavior that we see early in the run is an interesting effect, and we were curious what could be causing it. When we compared the $\mathbf{D}$ curve with the genetic diversity plot using lineage (figure 7 left), we noticed that the $\mathbf{D}$ curve becomes roughly constant at about

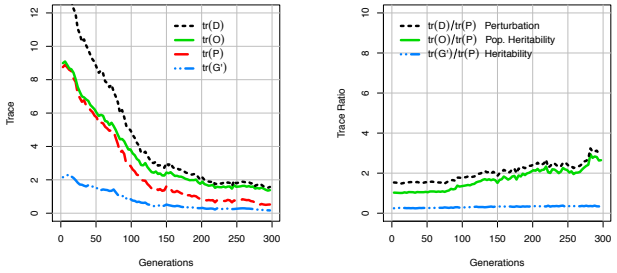

Figure 6: Subtree Crossover, Fitness Proportionate Selection, depth limit 17

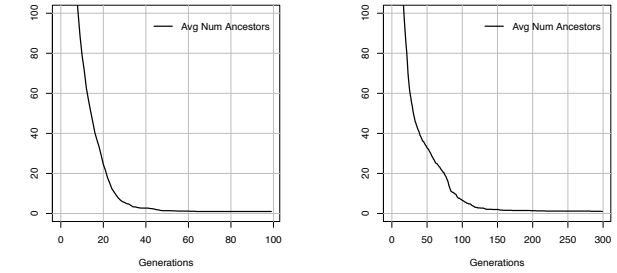

Figure 7: Lineage plot for subtree crossover using tournament selection size 2 (left) and fitness proportional selection (right).

the same time that the lineage diversity reaches zero, and so we hypothesized that the two were related. We tested to see if this relationship is consistent by running an experiment using fitness proportional selection (FPS) instead of tournament selection. This operator has a weaker selection pressure, and thus preserve population diversity for longer periods of time. As we can see in figure 6, **D** again becomes constant at about the same time that genetic diversity is exhausted (figure 7 right). We found this relationship to be consistent at various different selection pressures, as well as with other changes to the GP, such as when elitism is used.

There is one important difference between GP subtree crossover and other EAs using a combination of crossover and mutation. With other EAs, one can adjust the ultimate level at which the curve flattens by changing the mutation rate. In addition, when using Gaussian mutation one has ability to adjust the standard deviation of the Gaussian distribution as well. In GP we have much less control over how much variation subtree crossover adds towards the end of the run, and we may need to rely on other means to strike the right balance between exploration and exploitation.

5.4 The Effects of Bloat on Variation

When comparing the GP subtree crossover results in figure 5 with the EA Gaussian mutation results in figure 3, we notices something curious. In the GP algorithm, the perturbation curve increases, but then decreases again towards the end of the run.

One theory about bloat states that it is a reaction to the destructive effects of the reproductive operators, and as a

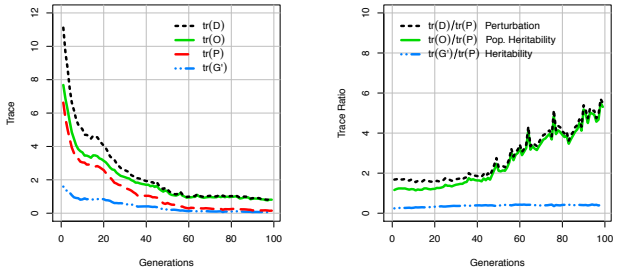

Figure 8: Homologous crossover, tournament size 2, depth limit 17

defense mechanism, bloat servers to reduce the amount of variation that those operators produce [1]. We hypothesized that this is exactly what we was causing the perturbation decrease in figure 5.

To test this, we ran an experiment using a modified version of subtree crossover called homologous crossover. It was designed and successfully employed to control bloat in many GP problems [9]. As the results in figure 8 show, with bloat now largely under control there is no longer a reduction in perturbation, suggesting our hypothesis was correct. ,

5.5 Appropriate Selection Pressure

One key concern within the GP community is avoiding premature convergence to sub-optimal peaks. This means that we need to encourage exploration during the early stages of the run. We have seen that high selection pressure will cause the populations to converge more quickly, reducing exploration. Similarly, it is well know that higher selection pressures lead to more bloat [10], which also tends to reduce exploration. So the solution seems obvious: just use weaker selection pressure.

Unfortunately, the situation is not that simple. If we compare the plots for phenotypic population diversity between figures 5 and 6, we can see that in the later stages of the run, weaker selection pressure will not allow the population to converge on the best solutions in the search space. This leaves us with a dilemma as to what selection pressure would be best. We will propose a solution to this problem in the next section.

5.6 Applying What We Have Learned

We have observed a number of important issues during our analysis so far, two of which we felt could be addressed. First, we noted that subtree crossover always produces some amount of variation, and this may be more variation than is ideal for solving our problem. And second, we observed the need for low selection pressure early in a run, but higher pressure as the population converges. This lead us to make the two modifications described below.

5.6.1 Phenotypic Crossover

We created a version of subtree crossover that we refer to as phenotypic crossover. Described simply, it calculates the **P** covariance matrix, and then creates offspring using subtree crossover, and rejects any that do not fall within the distribution defined by **P**. The result is an operator that produces much higher population heritability.

More specifically, the **P** matrix is used to calculate the probability that any point in phenotype space belongs to the distribution. Two parents are selected, and the operator is guaranteed to produce two offspring that have the same root nodes as the parents. In other words, a son and a daughter are both created, where father and son have the same root node, as do mother and daughter.

A series of candidate sons and daughters are generated and the probability of each belonging to the distribution is calculated. Each is then compared to a number drawn from a uniform random distribution. If this number falls below the probability, the individual is accepted, otherwise it is rejected. Once a son or daughter is accepted, candidates for those positions will no longer be considered. A maximum of 1000 candidates is considered for each position. If either position is left unfilled at that point, the candidate that had the lowest probability is then used to fill the slot.

As the run progresses, we will eventually reach a generation where it is no longer possible to use the **P** matrix for calculating probability, often because one or more traits have converged. The **P** from the previous generation is then used, and will continue to be used for the rest of the run.

The important thing to remember about this crossover operator is that it will produce higher population heritability values. There are probably better ways of achieving this than the approach we have used, but this acts as a proof of concept. These tools suggest some targets to aim for when making modifications. An analysis of the trait values then offers information that may be useful for tracking the problems back to the representation and operators.

5.6.2 *Adaptive Selection*

The second modification we made was to the selection operator. Most standard selection operators maintain a consistent selection pressure throughout the run. Having observed the need for weaker pressure at the beginning of a run and stronger pressure at the end, we decided to create an adaptive tournament selection operator.

Our concern in this paper is to highlight the abilities of these tools rather than to create robust solutions to these issues, therefore the implementation is quite simple. In the first generation, the tournament size is set at 2. Every 10 generations this is increased by 1, up to a maximum tournament size of 7. This allows for a gradually increasing selection pressure as the run proceeds.

5.6.3 *Experiment*

Tests using either one of these modifications separately produced results that were not significantly different from subtree crossover. Together though, the differences were significant. The results are plotted in figure 9.

Best-so-far curves for these different algorithms are plotted in figure 10. We compared fitnesses from generation 100 of the Subtree crossover and the same generation of our modified algorithm. Using the Wilcoxon rank sum test with continuity correction, we verified that our algorithm produced significantly better results with a confidence of p=0.05. The curve for fitness proportional selection does eventually reach a similar fitness to our algorithm, although it does not do so until roughly generation 300.

The plots for average population fitness are similar, where we see that the new algorithm maintains a population that is more fit than the other experiments. The metrics for bloat

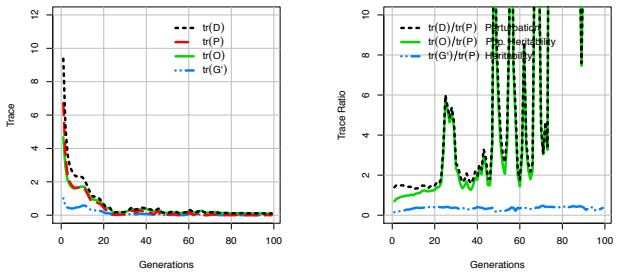

Figure 9: Phenotypic crossover, Adaptive tournament size, depth limit 17

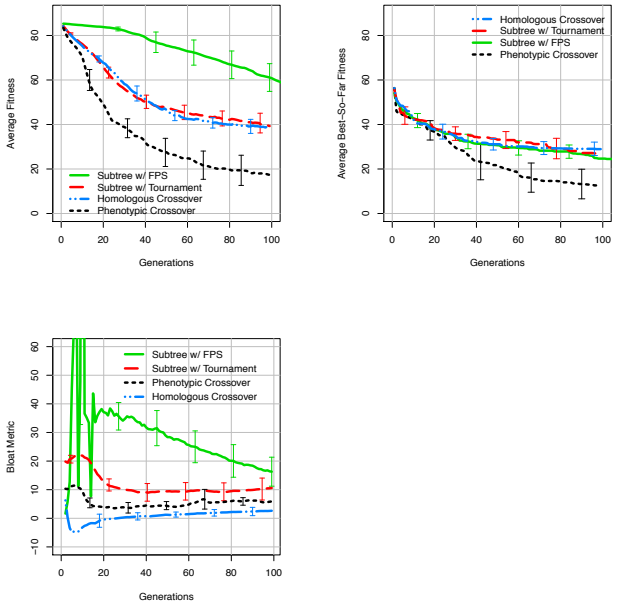

Figure 10: Plots of average population fitness, best-so-far fitness, and the bloat metric for the Ant problem. Whiskers indicate 95% confidence intervals.

show that our approach may have some advantages here as well, especially early in the run, but by generation 100 the improvements are not significant. Ultimately, the insights provided by the quantitative genetics tools allowed us to successfully reduce the number of generations necessary to find a solution.

5.7 Generality of Analysis

Many of these trends are not specific to the ant problem either. We performed an experiment with a different GP problem, the symbolic regression problem using the quintic equation as the target function. The GP used homologous crossover with a depth limit of 17, and tournament selection with a tournament size of 2.

Each trait is defined by sampling the value of the function at some specific x-value input. Ten sample points were used, and they were spaced evenly across the space of legal x-value inputs to the function. The traits were also transformed

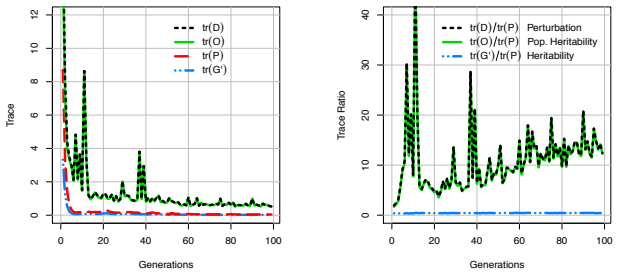

Figure 11: Symbolic Regression, homologous crossover, tournament size 2, depth limit 17

to make the distributions closer to normal using a sigmoid function to push outliers into a range that is more in line with a normal distribution.

The results of the experiment can be seen in figure 11. When compared with the results on the ant problem in figure 8, we can see several trends that we saw before. For example, the curve for **D** follows the same type of path, converging until a fixed level of variation is reached, and then staying there. Also, the perturbation curve and the population heritability curve show the same trend of continual increase over time. This suggests that many of our observations are general features of GP systems. We have not yet tested our improvements on the regression problem, but plan to do so in future work.

6. CONCLUSION AND FUTURE WORK

Quantitative Genetics theory offers a new set of tools for analyzing GP algorithms and how they perform their searches in phenotype space. They allow quantification of abstract concepts like exploration and exploitation, and the ability to observe how well an algorithm is achieving both goals. In this paper, we have shown how these tools provide insights that are not available by studying fitness values or genetic structure alone.

Through our analysis of a standard GP algorithm on the Ant benchmark problem, we were able to identify key points during the runs where exploration and exploitation were out of balance. We identified several causes, including aspects of subtree crossover, a dilemma involving the most effective amount of selection pressure, and bloat itself. These insights allowed us to demonstrate how one might modify designing to address these types of issues. With our proof of concept we were able to improve the fitnesses of the solutions, and reduced the number of generations needed to find them.

7. REFERENCES

[1] L. Altenberg. The evolution of evolvability in genetic programming. In K. E. Kinnear, editor, *Advances in Genetic Programming*, pages 47–74. MIT Press, Cambridge, MA, 1994.

[2] L. Altenberg. The schema theorem and Price's theorem. In L. D. Whitley and M. D. Vose, editors, *Foundations of Genetic Algorithms III*, pages 23–49, San Francisco, CA, 1995. Morgan Kaufmann.

[3] J. K. Bassett and K. A. D. Jong. Using multivariate quantitative genetics theory to assist in ea customization. In *Foundations of Genetic Algorithms 7*. Morgan Kaufmann, San Francisco, 2011.

[4] E. Burke, S. Gustafson, and G. Kendall. A survey and analysis of diversity measures in genetic programming. pages 716–723, 2002.

[5] E. K. Burke, S. Gustafson, G. Kendall, and N. Krasnogor. Is increased diversity in genetic programming beneficial? an analysis of lineage selection. In *Congress on Evolutionary Computation*, pages 1398–1405. IEEE Press, 2003.

[6] D. S. Falconer and T. F. C. Mackay. *Introduction to Quantitative Genetics*. Pearson, fourth edition, 1996.

[7] R. Lande. Quantitative genetic analysis of multivariate evolution, applied to brain: Body size allometry. *Evolution*, 33(1):402–416, Mar. 1979.

[8] W. B. Langdon. *Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming!* The Kluwer international series in engineering and computer science. Kluwer Academic Publishers, Boston, 1998.

[9] W. B. Langdon. Size fair and homologous tree genetic programming crossovers. genetic programming and evolvable machines, 1998.

[10] W. B. Langdon and R. Poli. Fitness causes bloat. In *Soft Computing in Engineering Design and Manufacturing*, pages 23–27. Springer-Verlag, 1997.

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

[12] P. Legendre and L. Legendre. *Numerical Ecology*. Number v. 20 in Developments in Environmental Modelling. Elsevier Science, 1998.

[13] S. Luke et al. ECJ: A java-based evolutionary computation research, 2011.

[14] B. Manderick, M. de Weger, and P. Spiessens. The genetic algorithm and the structure of the fitness landscape. In R. K. Belew et al., editors, *Proc. of the Fourth Int. Conf. on Genetic Algorithms*, pages 143–150, San Mateo, CA, 1991. Morgan Kaufmann.

[15] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions: I. Binary parameters. In H.-M. Voigt et al., editors, *Parallel Problem Solving from Nature – PPSN IV*, pages 178–187, Berlin, 1996. Springer.

[16] H. Mühlenbein and D. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm: I. continuous parameter optimization. *Evolutionary Computation*, 1(1):25–49, 1993.

[17] G. Price. Fisher's 'fundamental theorem' made clear. *Annals of Human Genetics*, 36(2):129–140, 1972.

[18] G. R. Price. Selection and covariance. *Nature*, 227:520–521, Aug. 1970.

[19] N. J. Radcliffe. Forma analysis and random respectful recombination. In R. K. Belew et al., editors, *Proc. of the Fourth Int. Conf. on Genetic Algorithms*, pages 222–229, San Mateo, CA, 1991. Morgan Kaufmann.

[20] S. H. Rice. *Evolutionary Theory: Mathematical and Conceptual Foundations*. Sinauer Assoc., Inc., 2004.

[21] L. Vanneschi, M. Castelli, and S. Silva. Measuring bloat, overfitting and functional complexity in genetic programming. In Branke et al., editors, *Proceeding of GECCO '10*, pages 877–884. ACM, 2010.