Chapter 1

Using Quantitative Genetics and Phenotypic Traits in Genetic Programming

Uday Kamath, Jeffrey K. Bassett and Kenneth A. De Jong

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/50143

1. Introduction

When evolving executable objects, the primary focus is on the behavioral repertoire that objects exhibit. For an evolutionary algorithm (EA) approach to be effective, a fitness function must be devised that provides differential feedback across evolving objects and provides some sort of fitness gradient to guide an EA in useful directions. It is fairly well understood that needle-in-a-haystack fitness landscapes should be avoided (e.g., was the tasked accomplished or not), but much less well understood as to the alternatives.

One approach takes its cue from animal trainers who achieve complex behaviors via some sort of "shaping" methodology in which simpler behaviors are learned first, and then more complex behaviors are built up from these behavior "building blocks". Similar ideas and approaches show up in the educational literature in the form of "scaffolding" techniques. The main concern with such an approach in EC in general and GP in particular is the heavy dependence on a trainer within the evolutionary loop.

As a consequence most EA/GP approaches attempt to capture this kind of information in a single fitness function with the hope of providing the necessary bias to achieve the desired behavior without any explicit intervention along the way. One attempt to achieve this involves identifying important quantifiable behavior traits and including them in the EA/GP fitness function. If one then proceeds with a standard "blackbox" optimization approach in which behavioral fitness feedback is just a single scalar, there are in general a large number of genotypes (executable objects) that can produce identical fitness values and small changes in executable structures can lead to large changes in behavioral fitness. In general, what is needed is a notion of behavioral inheritance.

We believe that there are existing tools and techniques that have been developed in the field of quantitative genetics that can be used to get at this notion of behavioral inheritability. In this chapter we first give a basic tutorial on the quantitative genetics approach and metrics required to analyze evolutionary dynamics, as the first step in understanding how this can be used for GP analysis. We then discuss some higher level issues for obtaining useful behavioral phenotypic traits to be used by the quantitative genetics tools. We give some background of other tools used like the diversity measurements and bloat metrics to analyze and correlate the behavior of a GP problem. Three GP benchmark problems are explained in detail exemplifying how to design the phenotypic traits, the quantitative genetics analyses when using these traits in various configurations and evolutionary behaviors deduced from these analyses.

2. Related work

Prior to the introduction of quantitative genetics to the EC community, research along similar lines was already being conducted. Most notable among these was the discovery that parent-offspring fitness correlation is a good predictor of an algorithm's ability to converge on highly fit solutions [18].

Mühlenbein and Altenberg began to introduce elements of biology theory to EC at roughly the same time. Mühlenbein's work has focused mainly on adapting the equation for the response to selection (also known as the breeder's equation) for use with evolutionary algorithms [19] Initial work involved the development several improved EAs and reproductive operators [21, 23, 24], and progressed to the development of Estimation of Distribution Algorithms (EDAs) [20, 22].

Altenberg's work used Price's Theorem [27] as a foundation for his EC theory. One of his goals was to measure the ability of certain EA reproductive operators to produce high quality individuals, and identify what qualities were important in achieving this [1]. He referred to this as evolvability, and the equations he developed looked similar in some regards to the response to selection equation. In particular he provided a theoretics foundation for why the relationship between parent and offspring fitness (i.e. heritability of fitness) was important.

Another aspect of Altenberg's work involved going beyond a simple aggregation of the relationships between parent and offspring fitness. He focused on the idea that the upper-tail of the distribution was a key element. After all, creating a few offspring that are more fit than their parents can be much more important than creating all offspring with the same fitness as their parents. This is why his equation really became a measure of variance instead of mean, which is what Price's Theorem typically measures. As an indication that his theories were in some sense fundamental to how EAs work, he was able to use them to re-derive the schema theorem [2].

Langdon [14] developed tools based on quantitative genetics for analyzing EA performance. He used both Price's Theorem and Fisher's Fundamental Theorem [26] to model GP gene frequencies, and how they change in the population over time.

Work by Potter et al. [25] also used Price's Theorem as a basis for EA analysis. They also recognized the importance of variance, and developed some approaches to visualizing the distributions during the evolutionary process [5, 6].

The work of Prügel-Bennett & Shapiro [29] [28] is based on statistical mechanics, but it has some important similarities to the methods used in quantitative genetics. Here, populations are also modeled as probability distributions, but the approach taken is more predictive than diagnostic. This means that detailed information about the fitness landscape and reproductive operators is needed in order to analyze an EA. Still, this approach has some interesting capabilities. For example, up to six higher-order cumulants are used to describe the distributions, allowing it to move beyond assumptions of normality, and thus providing much more accurate descriptions of the actual distributions.

Radcliffe [30] developed a theoretical framework that, while not directly related to quantitative genetics, has certain similarities. His formae theory is a more general extension of the schema theorem, and can be applicable at a phenotypic level.

3. Methodology

3.1. Quantitative genetics basics

Quantitative Genetics theory [9, 31]is concerned with tracking quantitative phenotypic traits within an evolving population in order to analyze the evolutionary process. One group that commonly uses the approach are animal breeders for the purpose of estimating what would be involved in accentuating certain traits (such as size, milk production or pelt color) within their populations.

A quantitative trait is essentially any aspect of an individual that can be measured. Since much of the theory was developed before the structure of DNA was known, traits have tended to measure phenotypic qualities like the ones listed in the paragraph above. Traits can measure real values, integer or boolean (threshold) properties, although real valued properties are generally preferred [9].

This approach offers a potential advantage to EC practitioners. Most EC theory is defined in terms of the underlying representation. As a consequence, it becomes difficult to adapt these theories to new types of problems and representations when they are developed. This generally means that the practitioner must modify or re-derive the theoretical equations before they can apply these theories to a new EA that has been customized for a new problem. For the few theories where this is not the case, a detailed understanding of the problem landscape is typically needed instead. Again this presents problems for the practitioner. After all, if they knew this much about their problem, they would not need an EA to solve it in the first place. Quantitative genetics is one of the few theories that does not suffer from these problems.

Populations are modeled as probability distributions of traits by using simple statistical measures like mean, variance and covariance. A set of equations then describe how the distributions change from one generation to the next as a result of certain evolutionary forces like selection and heritability.

An extended version of the theory called multivariate quantitative genetics [13] aims to model the behaviors and interactions of multiple traits within the population simultaneously. This approach represents multiple traits as a vector. As a result, means are also represented as a vector, and variance calculations produce covariance matrices, as do cross-covariance calculations. In other words, a vector and a covariance matrix are needed to describe a joint probability distribution. Other than this change, the equations remain largely the same.

It is difficult to do any long term prediction with this theory [11]. Instead, its value lies in its ability to perform analysis after the fact [11]. In other words, for our purposes the theory is most useful for understanding the forces at work inside an existing algorithm during or after it has been run, rather than predicting how an proposed algorithm might work.

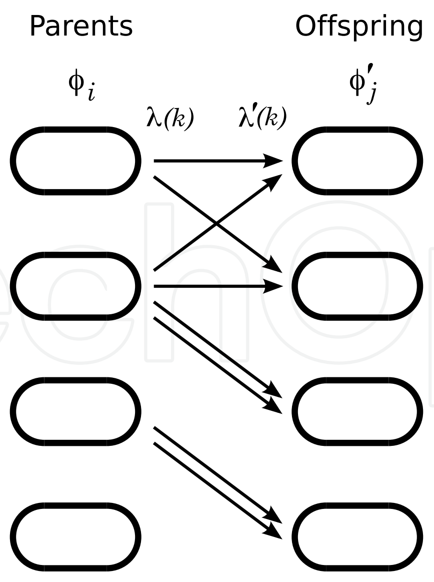

Figure 1. A sample generation showing offspring and parents [3]

In previous work [3], we adapted multivariate quantitative genetics for use with evolutionary algorithms. The goal of that work was to demonstrate how these theories can be used to aid in customizing EA operators for new and unusual problems. Here we will review some important aspects of that model.

To describe the equations, we will refer to Figure 1, which shows a directed graph illustrating 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. Because the quantitative genetics models are built on the idea of a generational evolutionary process, they are most easily applied to to generational EAs like GAs and GP.

It is important that each directed edge represent the same amount of "influence" that a parent has on its offspring. In the figure, each edge represents an influence of 1/2. That is why two edges are drawn between parent and offspring in instances where only cloning or mutation are performed. A vector of quantitative traits ϕ_i is associated with each parent i and another vector of traits ϕ'_j is associated with each offspring j. The two functions $\lambda(k)$ and $\lambda'(k)$ are defined such that they return the index of corresponding parent and offspring, for a given edge k.

We also use the abbreviations ϕ and ϕ' to describe all the traits in the different populations. The symbol ϕ describes all the parent traits, while ϕ' describes all offspring traits. 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.

Several covariance matrices are defined to describe the populations distributions and the forces that cause them to change. $\mathbf{P}$ and $\mathbf{O}$ are covariance matrices that describe the distributions of the *selected* parent and offspring populations respectively. $\mathbf{D}$ describes the amount of trait variation that the operators are adding to the offspring, and $\mathbf{G}'$ can be thought of as quantifying the amount of variation from $\mathbf{P}$ that is retained in $\mathbf{O}$.

$\mathbf{P}$, $\mathbf{O}$, $\mathbf{D}$ are all covariance matrices of the traits defined as $\mathbf{P} = \mathrm{Var}(\phi_\lambda)$, $\mathbf{O} = \mathrm{Var}(\phi')$, and $\mathbf{D} = \mathrm{Var}(\phi'_{\lambda'} - \phi_\lambda)$. $\mathbf{G}'$ is a cross-covariance matrix defined as $\mathbf{G}' = \mathrm{Cov}(\phi'_{\lambda'}, \phi_\lambda)$.

Given these matrices, we can now describe how the population 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 describes how the trait distribution of the selected parents ($\mathbf{P}$) is transformed by the operators into the distribution of the offspring population traits ($\mathbf{O}$).

The factor $\mathbf{G}'\mathbf{P}^{-1}$ is a regression coefficient matrix, and it is very similar to the quantitative genetics notion of narrow-sense heritability (commonly just called heritability). It describes the average similarity between an offspring and *one* of it's parents. The term $\mathbf{D}\mathbf{P}^{-1}$, which we refer to as perturbation, 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, while heritability provides an indication of it's ability to exploit existing information in the population. If heritability is low, that indicates that there is an unexpected bias in the search.

Another relationship that can be drawn from equation 2 is $\mathbf{O}\mathbf{P}^{-1}$. 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.

3.1.1. Scalar metric for matrices and vector operations

Biologists consider the multivariate notion of heritability as the degree of similarity between the two probability distributions that $\mathbf{P}$ and $\mathbf{G}$ describe. These comparisons are often performed using statistical techniques like Common Principle Component Analysis [10, 12].

For simplicity and ease of understanding, it would be ideal to find a metric that expresses terms like heritability and perturbation as a single scalar value. We have chosen to use the following metric,

$$\mathrm{m}(\mathbf{G}', P) = \mathrm{tr}(\mathbf{G}')/\mathrm{tr}(\mathbf{P}) \tag{3}$$

where m is the metric function, and $\mathbf{G}'$ and $\mathbf{P}$ are M by M covariance matrices as described in the previous section.

The result of equation 3 is, of course, our scalar version of heritability from a single parent. Similarly, $\mathrm{tr}(\mathbf{D})/\mathrm{tr}(\mathbf{P})$ would measure perturbation, and $\mathrm{tr}(\mathbf{O})/\mathrm{tr}(\mathbf{P})$ gives us a measure of the overall similarity between the selected parent population and the resulting offspring population.

We chose to use trace because they have an intuitive geometric interpretation. The trace functions is equal to the sum of the diagonal elements in the matrix. It's also equal to the sum of the eigenvalues of the matrix. In geometric representation it shows the sum total of all the variation in the distribution. Determinants are normally used as single measure for matrix operation. It was observed that determinants couldn't be computed for representation like GP, due to generation of individuals that can lead to non-positive semidefinite matrices.

3.2. Phenotypic trait design

Understanding the problem phenotypic landscape along with the search characteristics of the individual (GP program) will be an important step in designing the quantitative phenotypic traits. The key element is that the trait measure defines some search aspect of the individual in the phenotypic landscape. The design of phenotypic trait measures is similar to designing a fitness function for EA - they are problem-specific, and it is more an art and an iterative process to come up with one or more functions that capture the behavior. We have given some broad high level ideas below that can help the designer in more concrete way in coming up with the phenotypic traits for a given problem. Broadly speaking, we can devise the traits thus:

1. At the application domain specific level to see the search behavior measured as quantitative traits.
2. By decomposing an already aggregated fitness function into individual quantitative traits.

1. Application domain specific traits:
   Since most GP programs are used in agent based environments, we will generalize application domain traits to be more for agent based individuals.
   - Agent Based Individuals
     Agent based individuals, can be considered to have some sensors and to execute series of tasks in an environment. One may use several interesting properties as traits such as recognizing the sensors available for the agents , constraints in motion, number of tasks allowed, traps in the environment and way to avoid the traps etc can be interesting set of properties that user might want to use as traits. These properties will vary amongst the individuals and using them as phenotypic traits can give interesting multivariate analyses like the correlation between properties, correlation of these properties with fitness, etc. We can come up with more traits based on exact nature of the agents and tasks they are performing. Some of these may be orthogonal while some may have an overlap with each other. Having an overlap should be avoided as correlated traits can lead to problems likenon-positive semi-definite matrices.
     - Task Oriented Individuals
       In many GP applications, the agent is meant to be working on various sub-tasks. These tasks can be considered decomposable into smaller units. Normally the fitness measures only the end goal or just the higher level tasks performed, sometimes for example the amount of food eaten by the ant agent as the fitness in the ant trail problem. Various behaviors that lead to (or do not lead to) the tasks when quantified, might give good phenotypic behavior of the individuals. Some of the tasks or units can be very important and can be weighted higher as compared to others.

        - Competitive and Co-Competitive Individuals
          Many agent-based systems are competitive in nature, like the predator and prey class of the problems. Effective traits that determine metrics leading to success and failure of competing individuals may be more useful than agent-based traits. For instance, in a predator-prey based agents the fitness is basically how well you are doing against the other. If lower level details like "closeness"ï£¡ to the others, number of moves till attacked, number of changes in directions while moving, etc. can provide interesting metrics that can be used as traits in these domains.
        - Cooperative Individuals
          Another subclass of the agent based problems is the cooperative based agents. These individuals have to be in some kind of team to accomplish the goal. The individual behaviors can be specific decomposable ones or can be evolved during the execution. The performance evaluation of most fitness functions in these domains is measured by weighting individual and team performances. Various cooperative metrics can be measured again at different levels like attempts of cooperative moves, success and failures in the moves, ratio of total attempts to the success or failures, etc.
    - Design based Individuals
      Many GP applications are used mostly in design sub class of problems like circuit design, layout and network design and plan generation problems. Each of these use very high level measures combined in weights like the cost saved, components used, power distribution, etc. Again, using individualized measures and adding as many metrics that are circuit or layout specifics may give more clarity to the search behavior.
    - Regression based Individuals
      Many GP applications are used in curve fitting- finding equations hidden in the data as a category of problems. Various mathematical values ranging from values at different interesting points on the landscape, distances from each point projected to that on the curve, relative errors, etc can form good traits for such individuals to show the phenotypic search behaviors.

2. Aggregated Fitness Functions
   In general there is a certain class of problems where you can use a general notion of decomposing the aggregated fitness function to individualized metrics as traits. In bioinformatics, GP is used in wide range of protein conformation, motif search, feature generations, etc. Most fitness functions are complex aggregated values combining many search metrics. For example, in sequence /structure classification programs many aspects of classification into one value, like true positives, false positives, true negatives, weighted distance and angles etc are combined to give a single score to the individuals. Instead of having such a single aggregated function value, we can use each of them as phenotypic traits.

3.2.1. Issues

After discussing some design principles and methodology, issues related to choice of the traits are discussed in this section.

- Coverage/Completeness
  Ideally we would like to develop as complete a set of traits as possible. By "complete"

we mean that we would like to have a set of traits that describe the whole phenotypic search space as well as possible. Another way of viewing this is to ask "Do the traits that we have uniquely define an individual?" As we mentioned earlier, previous applications of quantitative genetics to EA have used fitness alone, this provided a very limited and incomplete view of the nature of the fitness landscape, especially individuals that are very different can have the same fitness. Similarly an incomplete set of traits can fail to illuminate certain important aspects of a problem.

Domains involving executable objects (like GP), and most machine learning in general, are particularly susceptible to this problem. This is because generalization is a critical part of the learning process. We expect our systems to be able to handle new situations that they never faced during training. One way of addressing this issue is to create traits that are, in a sense, general too. Traits that measure a set of behaviors that all fall into a broad category will be able to achieve the best coverage of the search space.

It is difficult to offer advice as to how one can recognize when they face this situation. Asking the question about uniqueness seems to offer the best general approach. It may be wise to ask oneself this question throughout the design and implementation process. One advantage that quantitative genetics offer though, is that it degrades gracefully in the sense that all the equations are still completely accurate, even with an incomplete set of traits. Ultimately one may only need a subset of traits in order to observe the important behaviors their algorithms, just so long as they are the right subset.

- Unnecessary traits
  Unnecessary traits are either those that are always constant, or those that are so closely correlated with another trait that they essentially are measuring the same thing. These can be more problematic because they can result in matrices that are non-positive definite. In this particular case it would mean that the matrices have one or more eigenvalues that are zero. While this is not actually wrong, with just a small amount of measurement error, or round-off error, the matrices could have negative eigenvalues, which is more problematic. We have devised the metric equation (equation 3) to minimize computational problems related to this situation, but one should try to avoid it if possible.
- Phenotype to Genotype Linking
  If one's goal in using these tools is to identify and fix problem in an algorithm, then one will need to make a connection between the traits, and any aspects of the representation or reproductive operators that are affecting those traits. The more abstract the traits are, the more difficult this becomes, and so very low-level descriptions of behaviors may be more appropriate to achieve this.

  Unfortunately, this can creates a conflict with the issue of trait completeness described above. There we suggested that higher-level traits may be better for getting the best landscape description possible. For example, consider a problem where we are trying to teach an agent to track another agent without being detected. A high-level set of traits might measure thing like: how much distance an agent keeps between itself and the target, the length of time that it is able to maintain surveillance, and the number of times it is detected. These traits may be ideal for covering all the skills that may be necessary for describing the fitness landscape, but they may not be very helpful in identifying what aspect of a representation or reproductive operators are problematic for learn well in this domain. Such connections would be tenuous at best.

At the other end of the scale, a low-level phenotype (conceptually, at least) might be something as simple as the input-output map that exactly describes the actions the agent would take for any given set of inputs. Here we have a much better chance of relating such information to the representational structure, and the effects of the reproductive operators. Unfortunately, it becomes much more difficult to define a complete set of traits. Such a set would have to describe the entire map, and this might mean thousands of traits. The only viable option is to create sample sets of inputs, where each sample would define a single trait. If one can define enough traits to get a reasonable sampling of the input space, or identify important samples that yield particularly valuable information, then this approach could still be useful.

Exactly how to solve this trade-off remains an open issue. Some possible solutions include combining low-level and high-level traits, using different kinds of traits depending on ones goals, or trying to find traits that achieve a middle ground between these two extremes.

3.3. Genetic diversity using lineage

To correlate some important evolutionary behaviors we need to measure genotypic diversity changes in the populations. There are many ways to measure genotypic diversity measurements like tree-edit distances, genetic lineages, entropy etc for understanding the genotypic behavior and correlating it with phenotypic behaviors [7]. Genetic Lineage is the metric more commonly used as it shows significant correlation to fitness [8]. In 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 provides a way to measure distribution of lineage over generations and also the count of unique lineages in the population over generations.

3.4. Bloat measure

Another important factor that we use to correlate the evolutionary behavior changes is with bloat. Bloat, has been described in various researches but very few of them have defined it quantitatively. In our study since we have to measure bloat quantitatively we use the metrics as defined in the recent research [32].

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

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.

4. GP benchmark problems and analyses

In next subsections we will walk through three different GP problems, to discuss the methodology of defining traits, performing experiments with different evolutionary operators and understanding the evolutionary behaviors in context of the given problem. We start with the ant trail problem and perform various experiments by changing the operators, selection mechanisms and pressure to investigate the evolutionary behavior with respect to quantitative genetics metrics. We then move to another agent oriented problem, lawn mower problem showing few experiments involving breeding operators and different selection mechanisms.

Finally we use the symbolic regression problem to describe how traits can be defined and the observations showing generality of our methodology.

All the experiments are performed using ECJ [17] with various standard default parameters like population size of 1024, a crossover depth limit of 17,and the ramped half and half method of generating tree (min/max of 2 and 6) for creating individuals. We will plot average $\mathrm{tr}(\mathbf{D})/\mathrm{tr}(\mathbf{P})$, $\mathrm{tr}(\mathbf{G})/\mathrm{tr}(\mathbf{P})$ and $\mathrm{tr}(\mathbf{O})/\mathrm{tr}(\mathbf{P})$ as quantitative genetics metrics for each generations. We will also plot the average unique ancestors as our genetic lineage diversity measure and bloat metrics from above for some correlations.

4.1. Experiment 1: Santa-Fe Ant trail

Artificial Ant is representative of an agent search problem and also it is considered to be highly deceptive and difficult for genetic programming [16]. The Santa-fe ant problem has a difficult trail and the objective is to devise a program which can successfully navigate an artificial ant to find all pieces of food located on a grid. The total amount of food consumed is used as single point measure of the fitness of the program. The program has three terminal operations forward, left and right for navigation. It has three basic actions like IfFoodAhead, progn2 and progn3 for performing single action and parameter based execution in the sequence. It has three basic actions like IfFoodAhead, progn2 and progn3 for performing single action and parameter based execution in the sequence. IfFoodAhead is a non-terminal function that takes two parameters and executes the first if there is food one step ahead and the second otherwise. Progn2 takes 2 parameters while progn3 takes 3 parameters and executes them in a sequence.

1. **Quantitative Traits for Santa-Fe Ant trail**
   As per our discussions in the phenotypic traits section, various search properties are devised to measure quantitatively behavior of an agent like ant and used for phenotypic traits in the calculations for equation above.

   For all the formulas
   m= moves, d= dimension, trail= point on trail,closest-trail= closest point on trail
   $\delta = distance$

   - **Sum of Distances from Last Trail**: This is the manhattan distance computed for all the moves from where it is to where it was last on the trail. This trait measures the "moving away effect" of the agent to the trail.

   $$\sum_{i=1}^{m}\sum_{j=1}^{d} \|\delta_{i,d} - \delta_{trail,d}\| \tag{5}$$

   - **Sum of Distances to Closest Point on Trail**:This is the manhattan distance computed for all the moves from where it is to point closest on the trail. This trait measures the "closeness" of the agent to the trail.

   $$\sum_{i=1}^{m}\sum_{j=1}^{d} \|\delta_{i,d} - \delta_{closest-trail,d}\| \tag{6}$$

- **Sum of Distances from Last Point**:This is the manhattan distance computed for all the moves from where it is to point last point. This trait measures the "geometric displacement effect" irrespective of trail for the agent.

$$\sum_{i=1}^{m}\sum_{j=1}^{d} \|\delta_{i,d} - \delta_{i-1,d}\| \tag{7}$$

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

$$\sum_{i=1}^{m} \forall d, \{if(\delta_{i,d} - \delta_{i-1,d}) = 0, count = count + 1\} \tag{8}$$

  Most of these quantitative traits show exponential distribution and hence they are transformed to the new set of derived traits by taking the *log* of the originals as insisted by various biologist [9].

2. **Santa-Fe Ant trail GP Experiments**
   To understand the effects of the operator and selection, we will be performing one operator at a time with the selection mechanism mentioned to see the impact.
   - **Subtree Crossover and Tournament Selection size 7**
     Since most GP problems use subtree crossover as the main breeding operator and normally higher selection pressure with tournament size 7 are employed, we use these to plot different metrics explained in the quantitative genetics section as shown in Figure 2.
   - **Subtree Crossover and tournament Selection size 2**
     We change the tournament selection to have lower pressure by changing the tournament size to 2, and observing all the metrics are shown in Figure 3.
   - **Subtree Crossover and Fitness Proportionate Selection**
     Fitness Proportionate Selection generally has lower selection pressure as compared to tournament selection, and by changing the selection mechanism the metrics are shown in the Figure 4.
   - **Homologous Crossover and Tournament Selection size 7**
     Homologous Crossover was designed and successfully employed to control bloat and improve fitness in many GP problems [15]. The impact of using homologous crossover on tournament selection size 2 using the metrics is shown in Figure 5.
3. **Santa-Fe Ant trail Observations**
   - Tournament size 2 gives a weaker selection pressure than tournament size 7. It can be seen that with selection 7 as compared to selection 2, there is rapid convergence in genotypic diversity. This correlates to rapid convergence in the phenotypic trait measurements of O and P. It can be observed that when the genotypic diversity and corresponding phenotypic traits converge, there is rise in the perturbation $\text{tr}(\mathbf{D})/\text{tr}(\mathbf{P})$ curve. The point at which this happens and magnitude of change shifts in generations with selection pressure, i.e with tournament selection size 2 it happens later around generation 50 as compared to around generation 20 with selection 7. Also the increase is

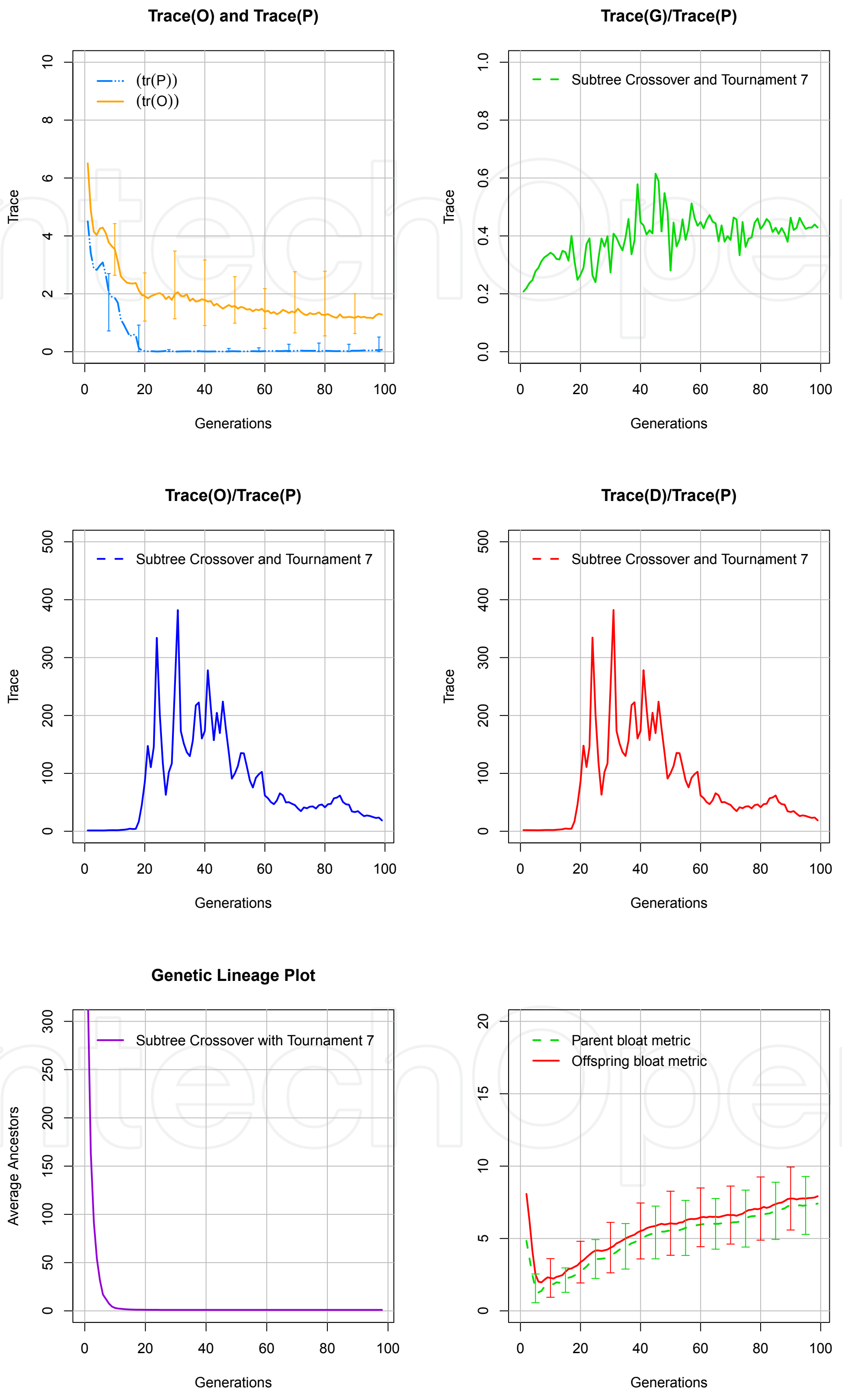

Figure 2. Ant, Subtree crossover, tournament size 7, depth limit 17

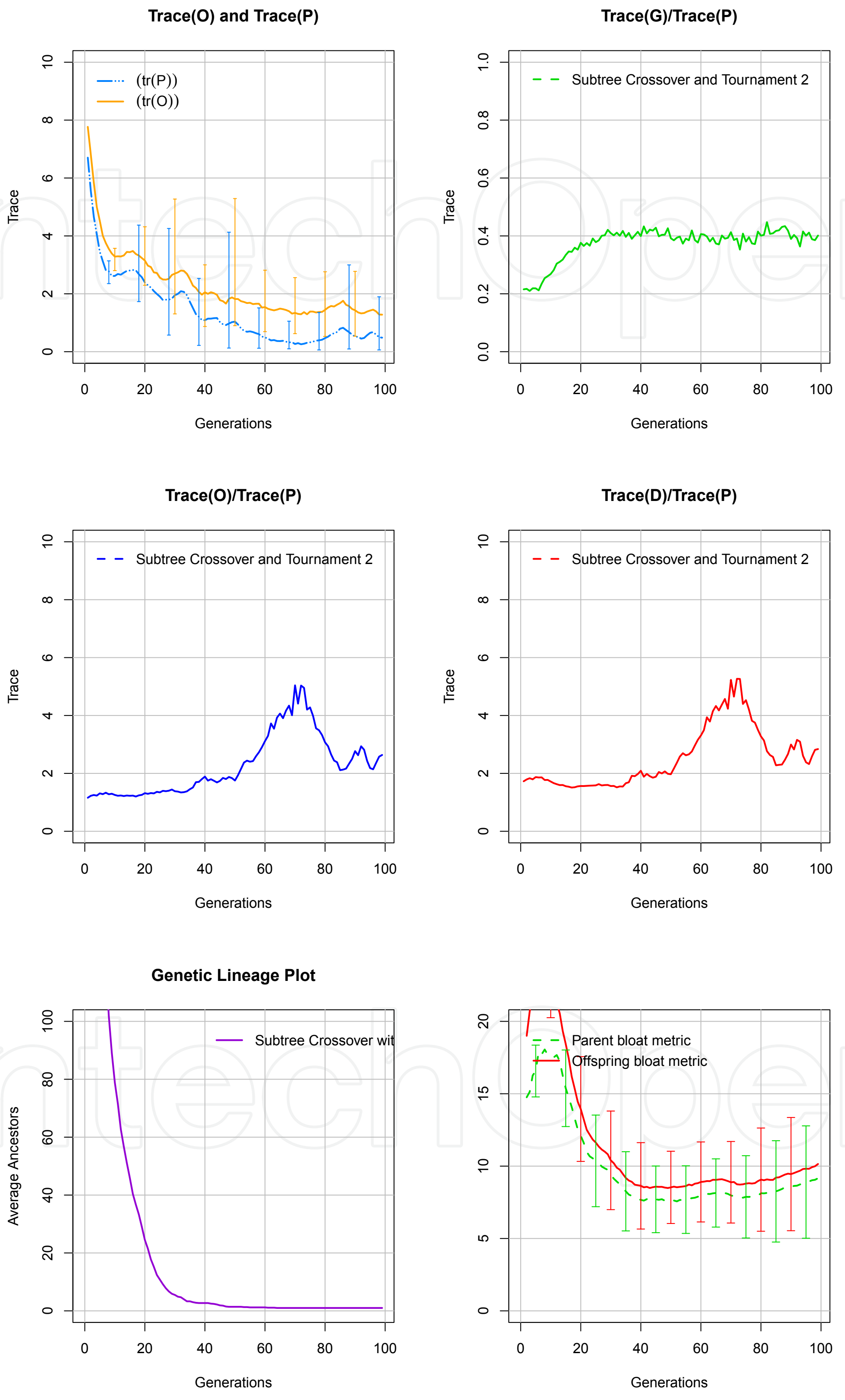

Figure 3. Ant, Subtree crossover, tournament size 2, depth limit 17

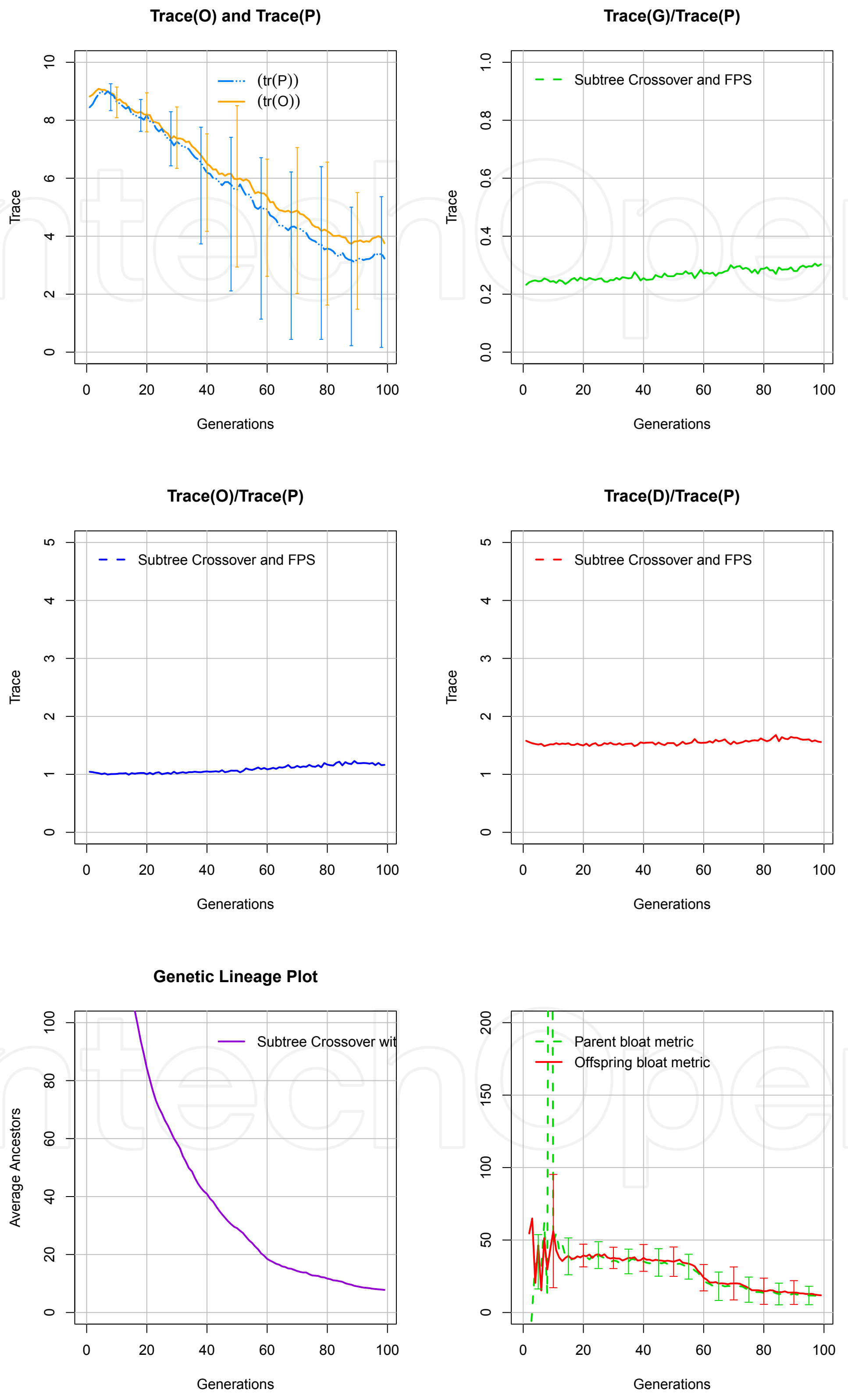

Figure 4. Ant, Subtree Crossover, Fitness Proportionate Selection (FPS), depth limit 17

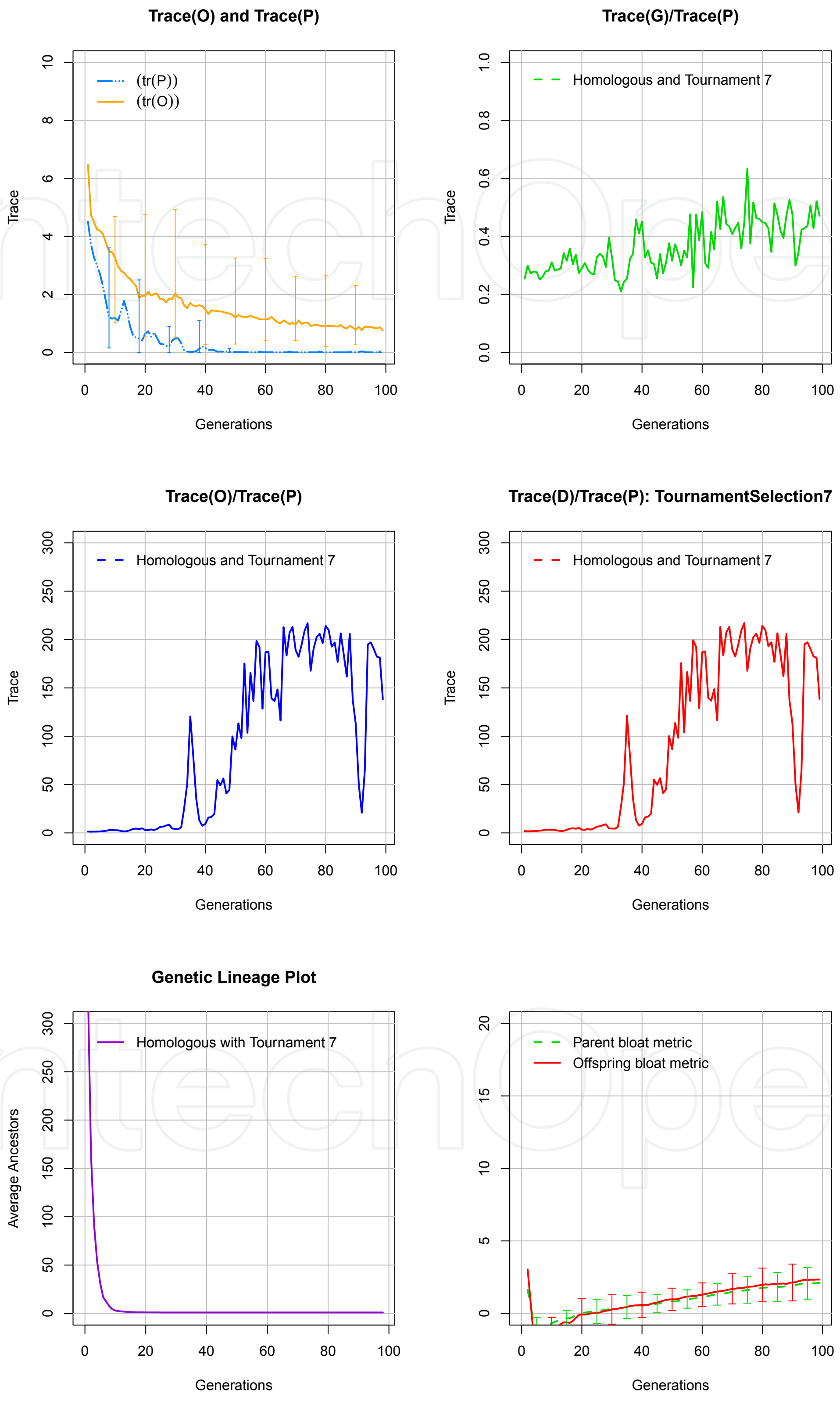

Figure 5. Ant, Homologous crossover, tournament size 7, depth limit 17

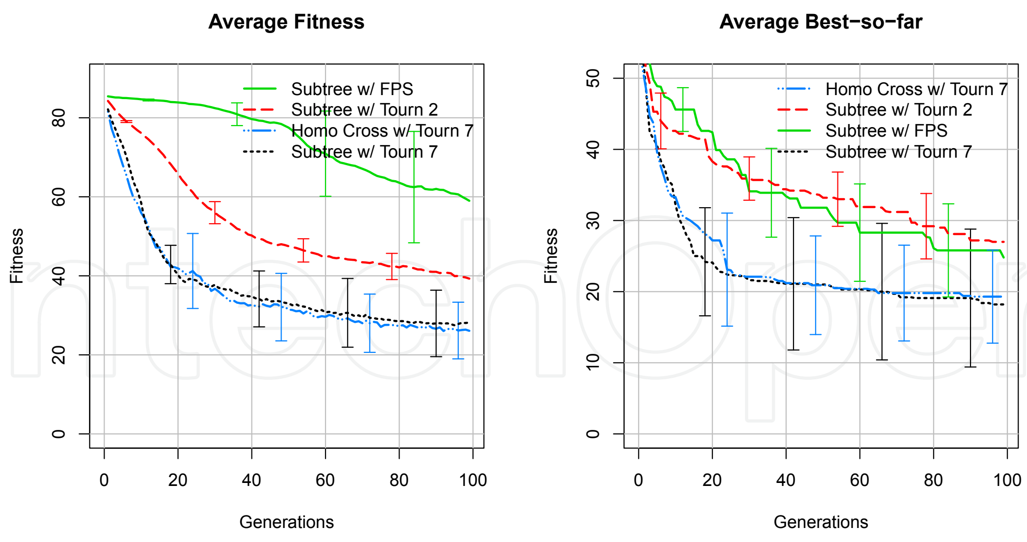

Figure 6. Average and BSF fitness for ant experiments

magnitude lesser (scale of DP, OP with selection 2 as compared to selection 7). Increased selection pressure which may result in lack of diversity may increase perturbation in the system. This increase may be useful in some difficult problems for finding area in the landscape that is not reachable otherwise and may not be effective when more of greedy local search is necessary to reach optimum. In ant trail problem, being a difficult landscape, increased perturbation is helpful to find solution faster as shown in the fitness curves in the Figure 6.

- It can be observed that the increase in perturbation with selection size 7, eventually tapers down and may be attributed to rise in the bloat. As bloat increases beyond a threshold, the effect of changes is reduced and that brings the perturbation down.
- Another important thing to note is with higher selection pressure, when there is premature convergence, it results in statistically significant (95% confidence) difference between the phenotypic behavior of offsprings and parents, while lower selection pressure reduces the difference.
- FPS results in higher genotypic diversity amongst the individuals as observed in the Figure 4, and that results in lower convergence in the population phenotypically and as a result the perturbation effect is constant across all the generations.
- Figure 5 shows that the perturbation increases with reduction in diversity exactly like in subtree crossover, but the perturbation continues to stay higher because of bloat control, however the max-value of perturbation is still lower than in normal crossover. Thus bloat which helped subtree crossover to reduce the impact of perturbation, when controlled by homologous crossover, showed constant value. This is consistent with theory that the bloat is a defensive mechanism against crossover [1].
- Figure 6 show the comparative plots of average and best so far (bsf) with 95% confidence intervals as whiskers. It can be seen that tournament selection with 7 with subtree or homologous are similar. Homologous crossover with reduced perturbations and bloat has real advantage over subtree crossover in this experiment.

4.2. Experiment 2: Lawn mower

The essence of this problem is to find a solution for controlling the movement of a robotic lawn mower so that the lawn mower visits each of the squares on two-dimensional n x m toroidal grid. The toroidal nature of the grid allows the lawnmower to wrap around to the opposite side of the grid when it moves off one of the edges. The lawnmower has state consisting of the squares on which the lawnmower is currently residing and the direction (up,down,left and right) which is facing. The lawnmower has 3 actions that change its state: turning left, moving forward and jumping to specified squares.

1. **Quantitative Traits for Lawn Mower**
   Similar to ant problem, we came up with some quantitative traits to measure the lawn mower behavior in the phenotypic landscape using the design principles. We keep a memory of visited location and have a function *visited(d)* for validating the revisit. We also keep memory of last orientation using omega in for measuring change in orientations in the movements.
   For all the formulas below
   m= moves, d= dimension, δ = *distance and* Ω = *orientation*

   - **Number of Moves**:This measures total number of moves performed by the agent in the execution, which we will refer as m.
   - **Count of Null Movements**:This is the count of zero movements, i.e. no change in displacement for the lawn mower over all its moves. This trait measures the effect of changing code not altering the behavior of the agent.

   $$\sum_{i=1}^{m} \forall d, \{if(\delta_{i,d} - \delta_{i-1,d}) = 0, count = count + 1\} \quad (9)$$

   - **Sum of Distances**:This is the manhattan distance computed for all the moves. This trait measures the "geometric displacement effect" in the movement.

   $$\sum_{i=1}^{m} \sum_{j=1}^{d} \|\delta_{i,d} - \delta_{i-1,d}\| \quad (10)$$

   - **Number of Orientation changes**:This measures number of times the orientation of the lawn mower is changed.

   $$\sum_{i=1}^{m} \forall d, \{if(\Omega_{i,d} \neq \Omega_{i-1,d}), count = count + 1\} \quad (11)$$

   - **Count of Revisits**:This measures number of times the already visited spot is visited.

   $$\sum_{i=1}^{m} \forall d, \{if(visited(d)), count = count + 1\} \quad (12)$$

2. **Lawn Mower GP Experiments**
   We performed subset of experiments from our ant problem on the lawn mower to see differences and similarity in the evolutionary behaviors.

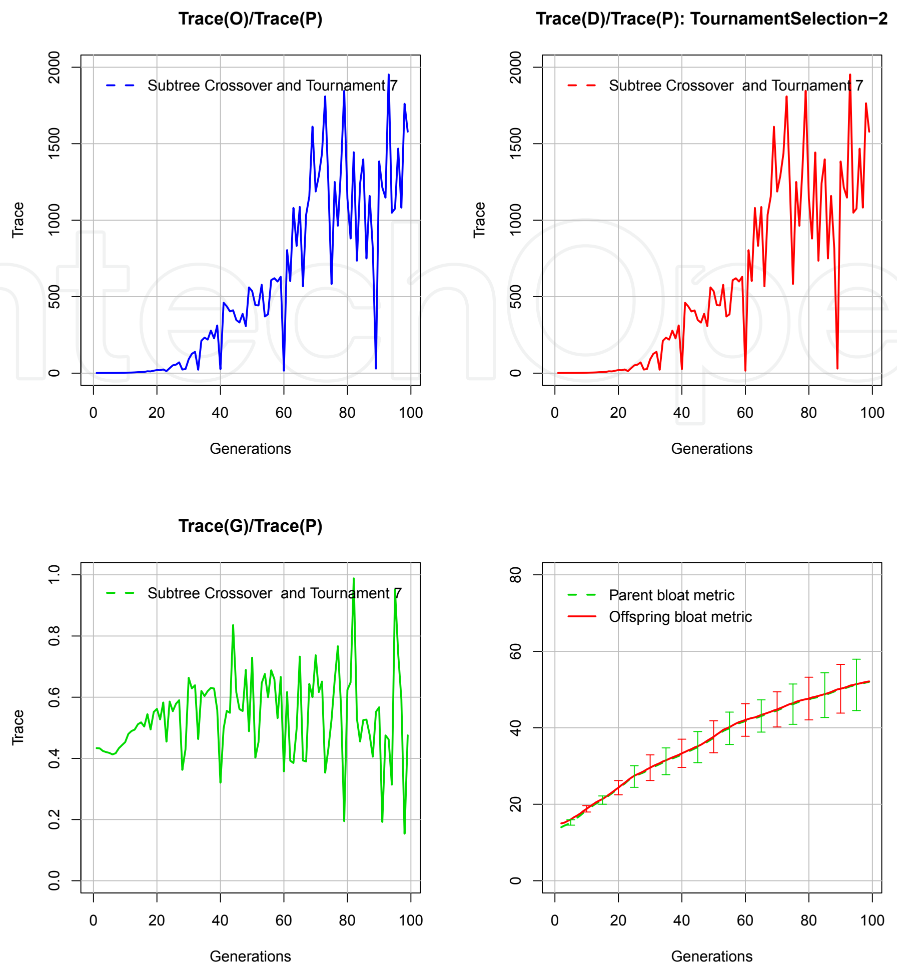

Figure 7. Lawn Mower Subtree crossover, tournament size 2, depth limit 17

- **Subtree Crossover and Tournament Selection size 2**
  We perform comparative subtree crossover with lower selection pressure on our lawn mower problem and show the quantitative genetics metrics plotted in the Figure 7.
- **Subtree Crossover and Fitness Proportionate Selection**
  We change the selection pressure totally by going for FPS instead of tournament selection and plot various metrics in the Figure 8.
- **Homologous Crossover and Tournament Selection size 2**
  Impact of bloat control by using homologous crossover with tournament selection with size 2 with various metrics are shown in the Figure 9.

3. **Lawn Mower Observations**
   - An interesting observation about the perturbation $\mathrm{tr}(\mathbf{D})/\mathrm{tr}(\mathbf{P})$ and $\mathrm{tr}(\mathbf{O})/\mathrm{tr}(\mathbf{P})$ curves can be made from Figures 8 and 9. Both curves tend to increase to a higher level with binary tournament selection as compared to FPS. This is actually a result of the fact that the GP crossover operators have a lower bound on the amount of variation they add to the population [4]. Higher selection pressures will reduce the phenotypic variation in the population more that lower selection pressures. Reproductive operators then return the variation to the operators minimum levels. When selection pressures are higher,

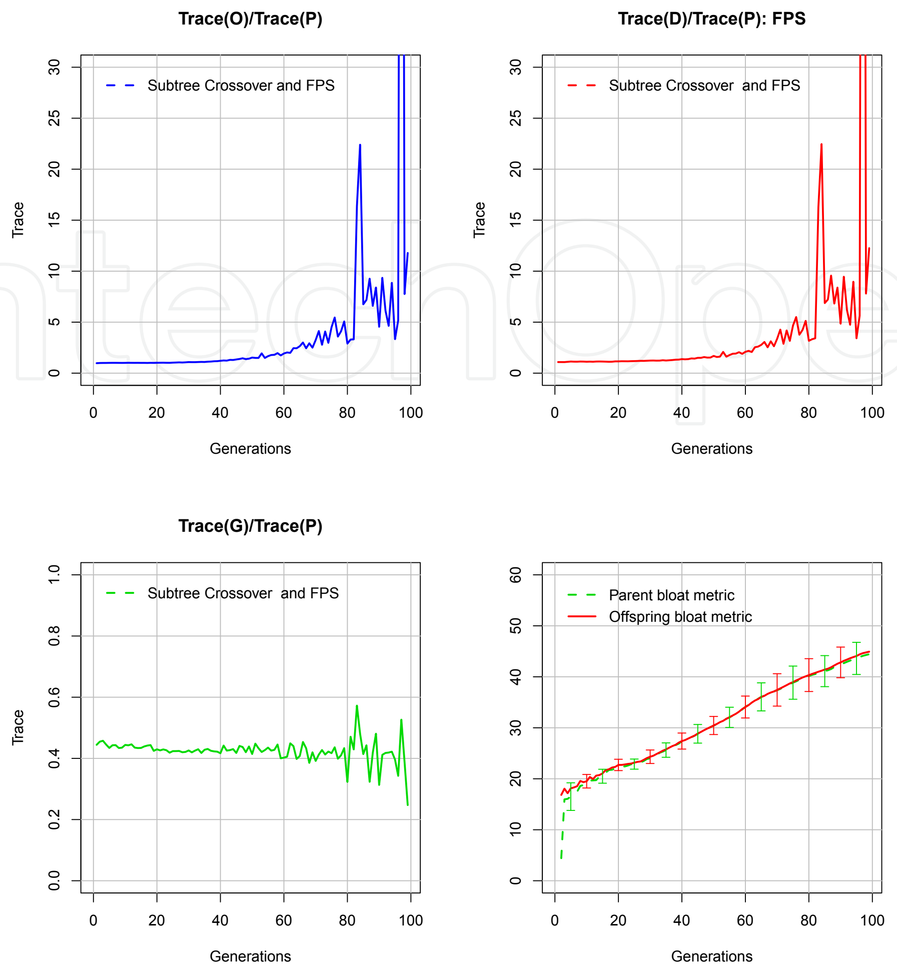

Figure 8. Lawn Mower Subtree crossover, Fitness Proportionate Selection, depth limit 17

the difference between these two amounts will be higher relative to the amount of variation in the selection parent populations. As a result, perturbation and population heritability will appear higher, but this is only because they had further to go to get back to the same place (i.e. the lower bound defined by the operators).

- Homologous crossover shows fairly stable $\mathrm{tr}(\mathbf{D})/\mathrm{tr}(\mathbf{P})$, $\mathrm{tr}(\mathbf{O})/\mathrm{tr}(\mathbf{P})$ and $\mathrm{tr}(\mathbf{G})/\mathrm{tr}(\mathbf{P})$ curves as shown in Figure 9, where the operator on this problem acts similar to the GA based crossover on a simple problem like sphere [3]. As the population converges in phenotype space, crossover is able to adapt and create offspring populations with similar distributions to those of the parent population (as can be seen by the fact that $\mathrm{tr}(\mathbf{G})/\mathrm{tr}(\mathbf{P})$ stays close to 0.5, and even more importantly that O/P stays relatively close to 1). The fact that it is able to do this even at the end of the run is important. It allows the population to truly converge on a very small part of the search space until there is (almost) no variation left. This is often considered to be a weakness of crossover, but in some ways it is really a strength. Without this ability, the algorithm cannot fully exploit the information it gains.
- Figure 10 shows again at the end of the generations there is no significant difference between subtree crossover and homologous crossover, while homologous crossover with better perturbation and heritability may be at advantage.

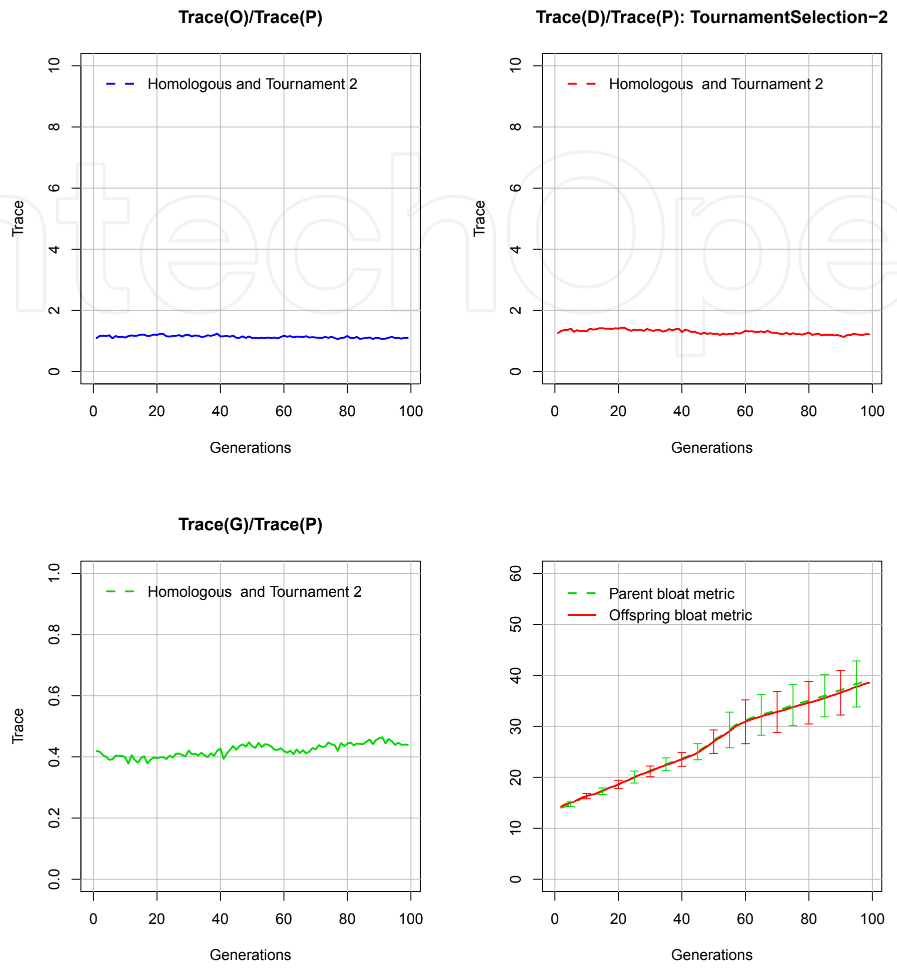

Figure 9. Lawn Mower Homologous crossover, tournament size 2, depth limit 17

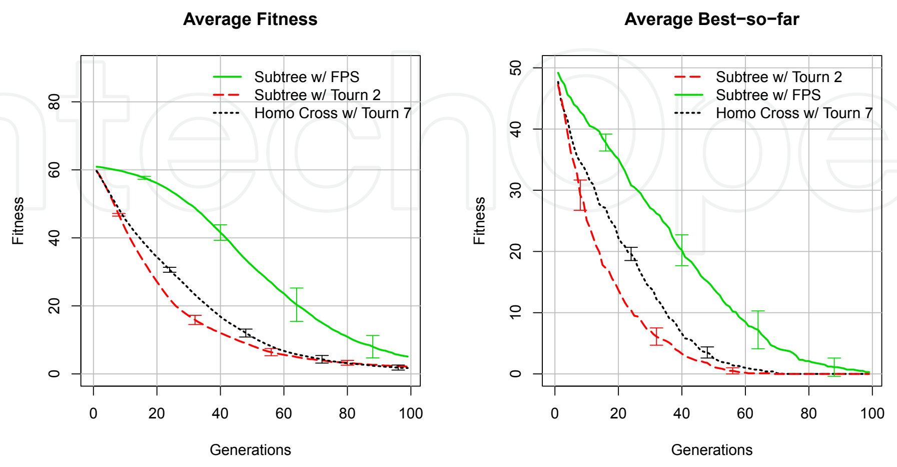

Figure 10. Average and BSF fitness for lawn mower experiments

4.3. Experiment 3: Symbolic regression

Symbolic Regression problem is about finding the equation closest to the given problem, by generating different curves and testing on the sample training points. The absolute error over the training points is used as the fitness function. Terminal would be the variable X and the non-terminals would be mathematical functions like log, sine, cosine, addition, multiplication etc. We used the quintic function for our test. Quintic is given by equation

$$y = x^5 - 2x^3 + x, \quad x = [-1,1] \tag{13}$$

1. **Quantitative Traits for Symbolic Regression** Regression being a mathematical problem in an euclidean space rather than a behavior based agent, we used the values of 10 random points equally distributed on the curve as the trait measurements like [-0.9, -0.7,-0.5...0.5,0.7,0.9]. This is similar to fitness being evaluated over fixed training point, but the difference being here we get individual values rather than aggregated measure. These individual trait values can be important in identifying how the curve changes between parent and offspring during the evolutionary process.

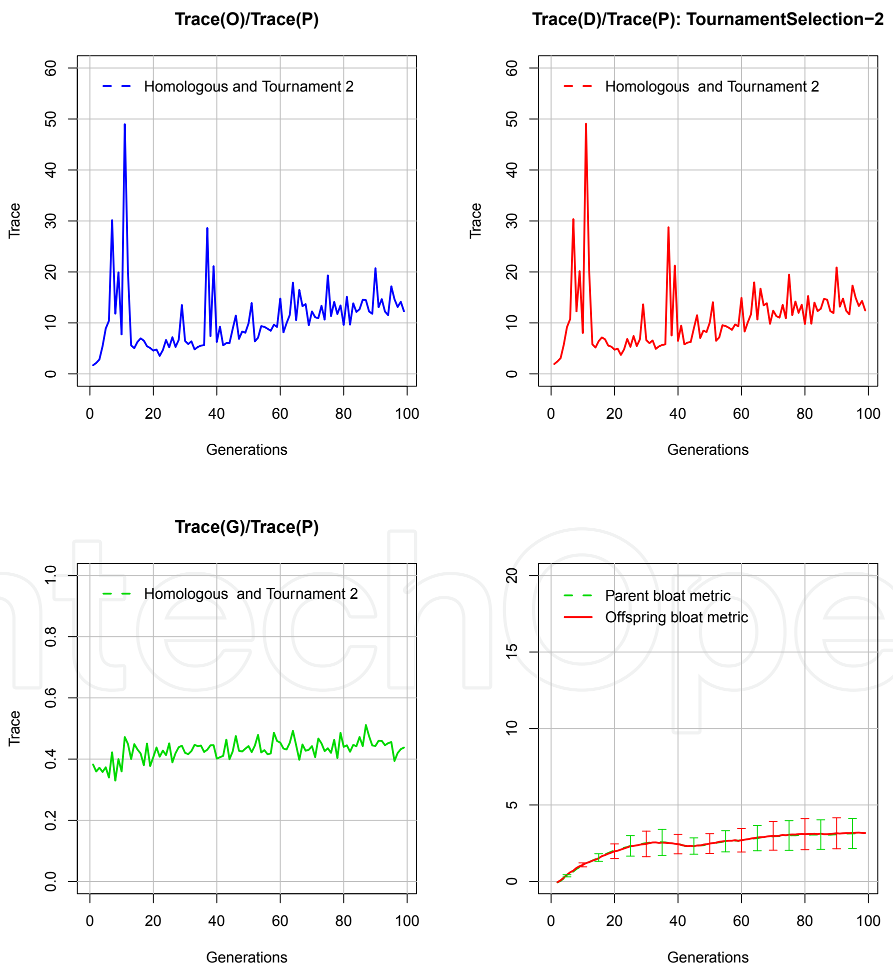

Figure 11. Symbolic Regression, Homologous crossover, tournament size 2, depth limit 17

2. **Regression GP Experiment**
   We will analyze one experiment using Homologous crossover and tournament selection to see generic behavior of GP problems given similar operators and selection pressure.
   - **Homologous Crossover with Tournament Selection size 2**
     Figure 11 shows various quantitative genetic metrics similar to previous experiments for quintic regression problem.
3. **Symbolic Regression Observations**
   - The results of the experiment as seen in Figure 11 is comparative to the results on the ant problem in figure 5. 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 generations.

5. Conclusions and future work

In this chapter we have provided a detailed tutorial on quantitative genetics and some high level design methods to define phenotypic traits needed by quantitative genetics. Using these methods we performed various experiments changing the selection and breeding operator in GP to analyze different evolutionary behaviors of the problem. Evolutionary forces like exploration and exploitation were quantified using quantitative genetics tool set and some interesting correlation with other forces like bloat, diversity, convergence and fitness were made. Many observations and correlations made were generalized across different benchmark GP problems.

In future we would like to perform more experiments to further understand the balance of bloat, selection and breeding operators, as well as designing new operators for resolving issues in a given problem domain.

Author details

Uday Kamath, Jeffrey K. Bassett and Kenneth A. De Jong
Computer Science Department, George Mason University, Fairfax, USA

6. References

[1] Altenberg, L. [1994]. The evolution of evolvability in genetic programming, *in* K. E. Kinnear (ed.), *Advances in Genetic Programming*, MIT Press, Cambridge, MA, pp. 47–74.
[2] Altenberg, L. [1995]. The schema theorem and Price's theorem, *in* L. D. Whitley & M. D. Vose (eds), *Foundations of Genetic Algorithms III*, Morgan Kaufmann, San Francisco, CA, pp. 23–49.
[3] Bassett, J. K. & De Jong, K. [2011]. Using multivariate quantitative genetics theory to assist in ea customization, *Foundations of Genetic Algorithms 7*, Morgan Kaufmann, San Francisco.
[4] Bassett, J. K., Kamath, U. & De Jong, K. A. [2012]. A new methodology for the GP theory toolbox, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2012)*, ACM.

[5] Bassett, J. K., Potter, M. A. & De Jong, K. A. [2004]. Looking under the EA hood with Price's equation, *in* K. Deb, R. Poli, W. Banzhaf, H.-G. Beyer, E. Burke, P. Darwen, D. Dasgupta, D. Floreano, J. Foster, M. Harman, O. Holland, P. L. Lanzi, L. Spector, A. Tettamanzi, D. Thierens & A. Tyrrell (eds), *Genetic and Evolutionary Computation – GECCO-2004, Part I*, Vol. 3102 of *Lecture Notes in Computer Science*, Springer-Verlag, Seattle, WA, USA, pp. 914–922.

[6] Bassett, J. K., Potter, M. A. & De Jong, K. A. [2005]. Applying Price's equation to survival selection, *in* H.-G. Beyer, U.-M. O'Reilly, D. V. Arnold, W. Banzhaf, C. Blum, E. W. Bonabeau, E. Cantu-Paz, D. Dasgupta, K. Deb, J. A. Foster, E. D. de Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G. R. Raidl, T. Soule, A. M. Tyrrell, J.-P. Watson & E. Zitzler (eds), *GECCO 2005: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, Vol. 2, ACM Press, Washington DC, USA, pp. 1371–1378.
URL: *http://www.cs.bham.ac.uk/ wbl/biblio/gecco2005/docs/p1371.pdf*

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

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

[9] Falconer, D. S. & Mackay, T. F. C. [1981]. *Introduction to quantitative genetics*, Longman New York.

[10] Flury, B. [1988]. *Common Principal Components and Related Multivariate Models*, Wiley series in probability and mathematical statistics, Wiley, New York.

[11] Frank, S. A. [1995]. George price's contributions to evolutionary genetics, *Journal of Theoretical Biology* 175(3): 373–388.
URL: *http://www.sciencedirect.com/science/article/B6WMD-45R8FXC-3N/2/01fea9e865de0a05 4158ee82d6237ef7*

[12] Game, E. T. & Caley, M. J. [2006]. The stability of P in coral reef fishes, *Evolution* 60(4): 814–823.
URL: *http://dx.doi.org/10.1111/j.0014-3820.2006.tb01159.x*

[13] Lande, R. & Arnold, S. J. [1983]. The measurement of selection on correlated characters, *Evolution* 37(6): 1210–1226.
URL: *http://www.jstor.org/stable/2408842*

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

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

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

[17] Luke, S., Panait, L., Balan, G., Paus, S., Skolicki, Z., Popovici, E., Sullivan, K., Harrison, J., Bassett, J., Hubley, R., Chircop, A., Compton, J., Haddon, W., Donnelly, S., Jamil, B. & O'Beirne, J. [2010]. ECJ: A java-based evolutionary computation research.
URL: *http://cs.gmu.edu/ eclab/projects/ecj/*

[18] Manderick, B., de Weger, M. & Spiessens, P. [1991]. The genetic algorithm and the structure of the fitness landscape, *in* R. K. Belew & L. B. Booker (eds), *Proc. of the Fourth Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, pp. 143–150.

[19] Mühlenbein, H. [1997]. The equation for response to selection and its use for prediction, *Evolutionary Computation* 5(3): 303–346.

[20] Mühlenbein, H., Bendisch, J. & Voigt, H.-M. [1996]. From recombination of genes to the estimation of distributions: II. continuous parameters, *in* H.-M. Voigt, W. Ebeling, I. Rechenberg & H.-P. Schwefel (eds), *Parallel Problem Solving from Nature – PPSN IV*, Springer, Berlin, pp. 188–197.

[21] Mühlenbein, H. & michael Voigt, H. [1995]. Gene pool recombination in genetic algorithms, *Metaheuristics: Theory and Applications* pp. 53—62.
URL: *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.3488*

[22] Mühlenbein, H. & Paaß, G. [1996]. From recombination of genes to the estimation of distributions: I. Binary parameters, *in* H.-M. Voigt, W. Ebeling, I. Rechenberg & H.-P. Schwefel (eds), *Parallel Problem Solving from Nature – PPSN IV*, Springer, Berlin, pp. 178–187.

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

[24] Mühlenbein, H. & Schlierkamp-Voosen, D. [1994]. The science of breeding and its application to the breeder genetic algorithm (BGA), *Evolutionary Computation* 1(4): 335–360.

[25] Potter, M. A., Bassett, J. K. & De Jong, K. A. [2003]. Visualizing evolvability with Price's equation, *in* R. Sarker, R. Reynolds, H. Abbass, K. C. Tan, B. McKay, D. Essam & T. Gedeon (eds), *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, IEEE Press, Canberra, pp. 2785–2790.

[26] Price, G. [1972]. Fisher's 'fundamental theorem' made clear, *Annals of Human Genetics* 36(2): 129–140.
URL: *http://dx.doi.org/10.1111/j.1469-1809.1972.tb00764.x*

[27] Price, G. R. [1970]. Selection and covariance, *Nature* 227: 520–521.
URL: *http://adsabs.harvard.edu/abs/1970Natur.227..520P*

[28] Prügel-Bennett, A. [1997]. Modelling evolving populations, *Journal of Theoretical Biology* 185(1): 81–95.
URL: *http://www.sciencedirect.com/science/article/B6WMD-45KKVJV-7/2/3ac11d9873754b7db 89bc424fc4919ad*

[29] Prügel-Bennett, A. & Shapiro, J. L. [1994]. Analysis of genetic algorithms using statistical mechanics, *Physical Review Letters* 72(9): 1305–1309.
URL: *http://link.aps.org/abstract/PRL/v72/p1305*

[30] Radcliffe, N. J. [1991]. Forma analysis and random respectful recombination, *in* R. K. Belew & L. B. Booker (eds), *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA'91)*, Morgan Kaufmann Publishers, San Mateo, California, pp. 222–229.

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

[32] Vanneschi, L., Castelli, M. & Silva, S. [2010]. Measuring bloat, overfitting and functional complexity in genetic programming, *in* B. et al.Editors (ed.), *GECCO 10 Proceedings of the 10th annual conference on Genetic and evolutionary computation*, ACM, pp. 877–884.