HYBRID STRUCTURAL AND BEHAVIORAL DIVERSITY TECHNIQUES FOR EFFECTIVE GENETIC PROGRAMMING

By

Armand Rashad Burks

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science — Doctor of Philosophy

2017

ABSTRACT

HYBRID STRUCTURAL AND BEHAVIORAL DIVERSITY TECHNIQUES FOR EFFECTIVE GENETIC PROGRAMMING

By

Armand Rashad Burks

Sustaining the diversity of evolving populations is a fundamental issue in genetic programming. We describe a novel measure of structural diversity for tree-based genetic programming, and we demonstrate its utility compared to other diversity techniques. We demonstrate our technique on the real-world application of tuberculosis screening from X-ray images. We then introduce a new paradigm of genetic programming that involves simultaneously maintaining structural and behavioral diversity in order to further improve the efficiency of genetic programming.

Our results show that simultaneously promoting structural and behavioral diversity improves genetic programming by leveraging the benefits of both aspects of diversity while overcoming the shortcomings of either technique in isolation. The hybridization increases the behavioral diversity of our structural diversity technique, and increases the structural diversity of the behavioral diversity techniques. This increased diversity leads to performance gains compared to either technique in isolation.

We found that in many cases, our structural diversity technique provides significant performance improvement compared to other state-of-the-art techniques. Our results from the experiments comparing the hybrid techniques indicate that the largest performance gain was typically attributed to our structural diversity technique. The incorporation of the behavioral diversity techniques provide additional improvement in many cases.

This dissertation is dedicated to you, Mom.
Thank you for always believing in me.

ACKNOWLEDGMENTS

This material is based in part upon work supported by the National Science Foundation under Cooperative Agreement No. DBI-0939454. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. This work was supported in part by Michigan State University through computational resources provided by the Institute for Cyber-Enabled Research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

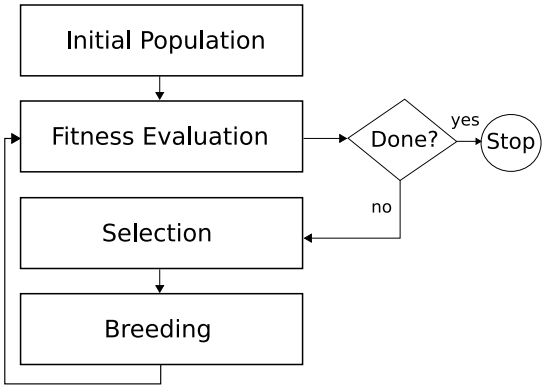
Introduction

1.1 Genetic Programming

Genetic programming (GP) [39] refers to a class of biologically-inspired techniques for the automatic synthesis of computer programs. Mimicking the evolutionary process, GP applies the concepts of natural selection, recombination, inheritance, and variation to a population of computer programs with the goal of automatically assembling a program with a certain desired behavior and characteristics. One of the long-standing and primary goals of GP is as follows. Given a programming task, typically defined by (1) a set of training examples (input cases and desired outputs), and (2) a set of primitives that make up a computer program (i.e. operands that correspond to the input cases and operators that perform some action on the given operands), automatically assemble a computer program that correctly performs the task (i.e. produces the desired outputs for all of the input cases).

Figure 1.1 provides an illustration of the conventional evolutionary cycle in GP. First, a population of programs is randomly generated. Next, each program (referred to as an individual) is evaluated using a problem-specific fitness function to determine

Figure 1.1: **The Conventional Evolutionary Process of GP.**



its fitness, which is typically a measure of its proximity to the optimal solution. After the population has been evaluated, selection pressure is applied by choosing fitter individuals. The selected individuals undergo genetic operators such as crossover and mutation to produce a new population. The process of fitness evaluation, selection and breeding repeats until either a solution is found or a predefined number of cycles (commonly measured in generations or total fitness evaluations) has been reached.

In tree-based GP, which is the form of GP we consider in this work, programs are represented by syntax trees, where the internal (function) nodes are typically operators and the terminal nodes are typically operands. This representation provides great flexibility in expressing and evolving functions. The traditional crossover operator creates new offspring programs by swapping randomly selected subtrees between two parent programs. Similarly, the traditional mutation operators replace a subtree (or node) in a parent program with a randomly generated subtree (or node) using the primitive set.

1.2 Premature Convergence

Premature convergence is a serious problem that is known to hamper the performance of GP, as well as other evolutionary algorithms (EAs). We consider premature

convergence to have occurred during the evolutionary process when the population becomes largely homogeneous before an optimal solution has been discovered. This often occurs during the early stages of the evolutionary process when sub-optimal solutions dominate the population and the typical genetic operators are unable to explore different regions of the search space. As a result, the likelihood of discovering an optimal solution dramatically decreases.

Premature convergence is an even more challenging problem for conventional GP because of the variable-length tree representation of programs. In tree-based GP, populations converge much differently than in EAs that use fixed-length representations such as the binary-encoded genetic algorithm (GA). In a binary-encoded GA, when two genetically identical individuals undergo single-point crossover, the resulting offspring will also be identical to the parents. However, given that GP trees typically encode functions, performing the conventional random subtree crossover operator on two identical trees can still create functionally different offspring when different subtrees are exchanged between the trees. Therefore, even if a GP population is largely genetically homogeneous, it is still possible for the traditional genetic operators to yield different phenotypes.

While GP populations converge differently than in other EAs, it is known that traditional GP populations are still greatly affected by premature convergence. An early analysis of population diversity revealed that GP populations often rapidly lose diversity during the early stages of the evolutionary process [53]. In that study, the authors tracked the propagation of genetic material through the population over time and showed that the population at the final generation tended to inherit all of its genetic material from a very small number of individuals from the initial generation. Inspecting the top four levels of the trees in the population revealed that near the onset of the evolutionary process, a large proportion of the population (over 70%)

became genetically identical to a single individual from the initial generation [53]. Since the rooted portion of a tree defines its functional context, changes made near the root have a much greater impact on its overall functionality. Therefore, prematurely converging to a single rooted structure can limit the variety of phenotypes that can be evolved.

Other studies that examined the properties of the conventional random subtree crossover operator revealed that it also contributes to the loss of diversity [24, 64]. Random subtree crossover operates on two trees and uniformly, randomly selects a node in each tree. The resulting subtrees at the chosen nodes are swapped between the two trees to create two offspring. It was shown that nodes deeper in the trees are more frequently chosen for crossover, while the upper regions of the trees are largely unaltered [24]. This is largely due to the fact that, given the branching factors of trees, the number of nodes lower in the tree is disproportionately larger than the number of nodes at higher levels in the tree. A common method for addressing this is to force the crossover operator to probabilistically select internal nodes more often than leaves. However, Langdon and Poli showed that random subtree crossover tends to exchange little genetic material between trees, which further exacerbates genotypic convergence [64].

In addition to premature convergence in the genotypic space, premature convergence in the phenotypic (behavioral) space is a serious, yet complex, issue in GP [35, 36, 46]. Because many genotypes (trees) can encode the same phenotype (the behavior expressed by a tree when it is executed), the genetic diversity of a GP population does not directly correlate to the phenotypic diversity of the population. Therefore, a genetically diverse population is not guaranteed to also be behaviorally diverse. Likewise, a genetically homogeneous GP population is not guaranteed to be behaviorally homogeneous since it is still possible for new phenotypes to arise, as

we discussed earlier. However, GP populations often experience a loss of behavioral diversity [35, 36]. Due to strong selection pressure, GP populations tend to rapidly converge to a few clusters of individuals that express the same behavior [36, 46].

1.3 Contributions of this Work

There are multiple contributions of this research. We begin with a survey of several preexisting genotypic and phenotypic diversity techniques for GP. Next, we introduce a simple yet powerful novel measure of structural diversity that is free of the notion of genotypic distance. We then present a multi-objective technique that incorporates our structural diversity measure to prevent premature convergence while accelerating search. We validate our technique by demonstrating its utility and performance benefit compared to state-of-the-art and conventional GP approaches, as well as by applying our technique to the real-world problem of tuberculosis screening from X-ray images. We also provide a detailed analysis of the search properties and behavior of our technique.

Next, we introduce hybrid methods for simultaneously sustaining structural and behavioral diversity to achieve more effective GP search than either method in isolation. Empirical analyses reveal that simultaneously maintaining structural and behavioral diversity can improve the search performance of GP by leveraging both aspects of diversity and overcoming some of the challenges faced by either method in isolation. These findings also motivate the design of new GP operators that utilize structural and behavioral diversity to achieve more efficient search.

Chapter 2

Existing Diversity Techniques in Tree-Based GP

Premature convergence has received a lot of attention, and many techniques have been developed in order to prevent it in GP as well as EAs in general. Many such techniques focus on maintaining population diversity as a means of combating premature convergence. This chapter reviews several existing diversity techniques.

2.1 Genotypic Diversity Techniques

The traditional search operators such as crossover and mutation work at the genotypic level, making genetic modifications to the individuals in the evolving population in order to evolve new phenotypes and explore different regions of the search space. Therefore, the genotypes in the evolving population play a vital role in the effectiveness of EAs. As a result, several existing diversity techniques focus on sustaining genotypic diversity in the population in order to prevent premature convergence.

Several genotypic diversity techniques rely on a distance metric to determine the similarity between genotypes in the population. Given a distance metric, a

population with greater genotypic distance between individuals is said to be more genetically diverse. Therefore, maximizing the pairwise distance between individuals of the population has been used as a means of promoting diversity [18].

A type of tree edit distance has been used to promote diversity in GP [18]. To determine the distance between two trees, the trees are first overlaid and then the total nodes that differ in the overlapping region of the two trees is calculated. The distance is normalized by dividing it by the size of the smaller tree. Genotypic diversity is then measured as the average squared distance of each individual to the rest of the population.

Although such genotypic diversity metrics can be used to increase population diversity, there are limitations of such measures [11]. One of the main issues is that depending on the way diversity is defined, high diversity may not necessarily imply good fitness or efficient search [11]. Genotypic diversity is undoubtedly important to the success of EAs because it increases the possible ways of assembling a solution. However, it is less clear how a diversity metric should be defined, how much diversity is needed, and how diversity should be enforced. Moreover, genotypic distance may not accurately reflect phenotypic distance in GP, due to the complicated genotype-to-phenotype mapping. Therefore, many approaches focus on maintaining diversity without the use of an explicit distance metric.

Structure Fitness Sharing (SFS) [33] promotes genotypic diversity based on tree structure. However, instead of explicitly calculating the distance between trees, SFS labels each unique tree structure and tracks the number of individuals and fitness for each structure. The main idea is to promote the search for unique structures and prevent a single structure from dominating the population. This is done by adopting a fitness sharing scheme [25, 26], which reduces the fitness of structures that are over-represented in the population.

Lineage selection [12] was proposed as a means of promoting the search for the "fit and diverse" rather than solely selecting individuals based on fitness. Lineage selection is based on the concept of genetic lineages, where the lineage of an individual can be determined by tracing the path from its root parent (the parent from which it inherited its root node) to the individuals from which it was created through crossover [12, 53]. Lineage selection groups individuals based on genetic lineage, and then modifies the traditional tournament selection operator to hold tournaments among random individuals selected from different lineages.

In a similar fashion to lineage selection, Hereditary Repulsion (HR) [56] uses genetic lineage information to promote diversity. When selecting parents to create offspring, HR selects the first parent randomly and then selects a random subset of candidate individuals for the second parent. Next, the hereditary overlap between the first parent and each of the candidates is determined, where the hereditary overlap of two individuals is the total common ancestors the individuals have in their lineages. In this case, both parents of an individual are considered in the lineage instead of only the root parent. The candidate with the smallest hereditary overlap with the first parent is chosen to undergo crossover.

2.1.1 Structured Populations

Other methods have been introduced for enforcing genotypic diversity by imposing a structure on the population instead of relying on an explicit diversity measure [30, 32]. The main motivation behind such techniques is that fitter genotypes tend to rapidly dominate the population, which accelerates premature convergence. By restricting competition through a structured population, and regularly introducing new genetic material into the population, these methods attempt to prevent premature convergence and promote better exploration of the search space.

The Age-Layered Population Structure (ALPS) uses a hierarchical population to counteract premature convergence [8, 30, 31]. ALPS uses *genotypic age* to segregate the population into several "age layers" in order to prevent older genotypes from dominating the population. Genotypic age is a measure of the amount of time that has passed since an individual's oldest ancestor was created. The age layers restrict competition and breeding to individuals of similar genotypic age. Each layer has an age limit, which can be determined by different numerical aging schemes [30]. As time progresses, the genotypic age of individuals reaches the age limit while the individuals in the layers become fitter. Individuals above the age limit are forced to move out of their layers and compete for membership in the next highest layer by displacing a less-fit individual or an individual that is too old for that layer.

In addition to using age layers to localize competition, ALPS also periodically introduces new randomly generated individuals into the population at the lowest layer. Since individuals only compete against other individuals within a certain range of their genotypic age, new randomly generated individuals don't face the risk of immediately being dominated by older, fitter individuals. Therefore, the lowest layer serves as a source of new genetic material and the discovery of new building blocks that propagate up the hierarchy and help the population escape local optima.

The Hierarchical Fair Competition Framework (HFC) and its variants work on the idea of "fitness gradients," into which the population is organized to make competition between individuals more balanced [32]. HFC organizes the population into several fitness levels, where each level contains one or more subpopulations. By dividing the range of the possible fitness values among the different fitness levels, HFC creates a fitness gradient through which evolutionary search progresses. This effectively decreases local selection pressure at each fitness level because individuals must only compete with other individuals in their own fitness level. This is a ma-

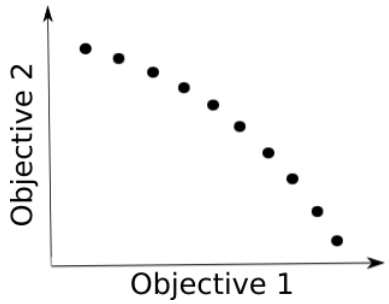
jor advantage over the traditional EA because it protects "stepping stones," which are individuals of intermediate fitness that potentially contain valuable genetic material [32]. This way, stepping stones have more of a chance to spread potentially valuable genetic material through the population without the risk of being dominated by fitter individuals.

To provide a constant source of new genetic material, HFC continuously adds new randomly generated individuals at the lowest fitness level. The new individuals are less likely to be immediately dominated in fitness since competition occurs between individuals of similar fitness. However, HFC still maintains a strong global selection pressure since fitness increases toward the upper levels of the hierarchy [32]. This incorporation of new genetic material is not only a way of increasing genetic diversity, but it can also increase the likelihood of reaching previously unexplored areas of the search space and preventing the population from prematurely converging to local optima.

2.1.2 Multi-objective Diversity Maintenance

As we discussed in Section 1.1, the conventional fitness function is a measure of an individual's proximity to an optimal solution. For many problems, fitness is defined based on a single measure. However, other problems contain multiple criteria that must be optimized, and there are several existing multi-objective techniques (for example, the popular NSGA-II [19]) that have been developed for such problems. Leveraging the power of such multi-objective techniques, diversity maintenance techniques have been designed by incorporating a diversity measure as an additional, artificial objective of the problem alongside fitness [18, 69]. This *multi-objectivization* [38] allows diversity to be maximized while fitness is simultaneously optimized, and allows the use of the many techniques that were originally developed

Figure 2.1: **An Example Set of Pareto-optimal Solutions in a 2D Objective Space.** Both objectives are maximized. No single point dominates any other point in both objective values.



for multi-objective problems.

Such multi-objective methods are based on the Pareto criterion, which considers the trade-off curve of the candidate solutions [68]. The key idea is that given multiple objectives, there may exist many combinations of objective values that are equally optimal (i.e. Pareto-optimal as in Figure 2.1). Therefore, the Pareto criterion compares individuals on all objectives and considers an individual A to dominate individual B if the following holds:

$$\forall i A_i \geq B_i \wedge \exists i \mid A_i > B_i \tag{2.1}$$

where A_i and B_i are the objective values, respectively, of A and B for each objective i. This assumes maximization of all objective values, and the same is true for minimization of any objective, where $\geq$ and $>$ are replaced by $\leq$ and $<$, respectively.

The FOCUS (Find Only and Complete Undominated Sets) method [18] uses a multi-objective scheme to promote genetic diversity while optimizing fitness. Based on the overlapping tree distance metric that we discussed earlier, FOCUS calculates for each individual the average squared distance of that individual to every other individual in the population. This value becomes the diversity objective value for

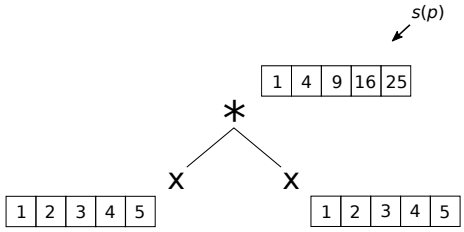
each individual, which is maximized in the multi-objective scheme. Each generation, FOCUS uses the Pareto criterion to remove all individuals that are dominated, considering diversity, fitness, and tree size as objectives. The tree size objective, which is minimized, is used to add parsimony pressure to reduce the problem of bloat, wherein GP trees typically grow dramatically over time. However, it has been shown that parsimony pressure can be detrimental to GP performance [72].

Age-Fitness Pareto Optimization [69] was introduced as a means of using genotypic age as in ALPS to localize competition and maintain a constant source of new genetic material without the overhead of managing a structured population. Age-Fitness Pareto Optimization adopts a multi-objective scheme in which genotypic age is minimized as one objective while fitness is optimized as the second objective. The main goal is to promote the search for newer, fitter genotypes while preventing older genotypes from dominating newer genotypes. Age-Fitness Pareto Optimization adds a new randomly-generated individual to the population each generation as a means of maintaining a constant source of new genetic material. Since the new individual will be non-dominated in the age objective, it has more of a chance to propagate potentially valuable genetic material through the population.

2.2 Behavioral Diversity Techniques

As Section 1.2 discusses, genotypically different trees can express the same behavior due to the complex genotype-to-phenotype mapping in tree-based GP. Therefore, enforcing diversity in the genotypic space does not ensure diversity in the phenotypic (behavioral) space. Fortunately, recent research has focused on maintaining a diverse set of behaviors by operating explicitly in the phenotypic space, avoiding the problem of genotype-to-phenotype mapping.

Figure 2.2: **Semantics of a GP Tree.** The tree p is executed on the input $k = [1, 2, 3, 4, 5]$ and produces a corresponding output $p(k_i)$ for each input case k_i. The semantics $s(p)$ is the vector of outputs $[1, 4, 9, 16, 25]$ produced by executing p on k.



2.2.1 Semantics in GP

Before discussing the behavioral diversity techniques that have been introduced for GP, it is important to first define behavior in GP. Since the goal of GP is to automatically synthesize programs for a given programming task, problems in GP are typically defined with a set of training examples that contain inputs and desired outputs. More formally, a problem can be defined by several input cases k_i, each of which has a corresponding desired output, y_i.

When a program, p is evaluated by the fitness function, the program is typically executed on each input case k_i, for which it produces an output $p(k_i)$. The semantics or behavior, $s(p)$, of p is commonly defined as the vector of outputs that p produces when executed on all the input cases [54, 76]. Figure 2.2 demonstrates how the semantics of a tree is obtained by executing the tree on the input cases.

2.2.2 Semantic-Aware Selection Methods

Several selection methods have been proposed for sustaining behavioral diversity rather than focusing solely on genotypic diversity. The conventional selection methods such as tournament selection and fitness proportionate selection only consider an individual's fitness and are blind to its actual behavior. On the other hand, semantic-aware selection methods consider some aspect of the individual's behavior

when selecting individuals to produce offspring.

Since GP is often applied to problems that consist of multiple training examples (fitness cases), Lexicase Selection [28] considers the performance of an individual on each fitness case. The main goal of Lexicase Selection is to breed individuals that perform well on different sets of fitness cases in order to increase the likelihood that their offspring will inherit complementary traits from the parents. Before selecting each parent from the population, Lexicase Selection chooses a random ordering on the fitness cases and then, in that order, it eliminates parent candidates by retaining only those that have the exact error on each fitness case until it can select only a single individual.

Although Lexicase Selection has been shown to increase behavioral diversity as well as improve search performance in different problem domains, it is known to perform poorly for problems wherein the error for each fitness case is a real-value. Since we used real-valued problems in our test suite, which we describe in detail in Chapter 4, we used Epsilon Lexicase Selection [45], which is a recent enhancement that performs well for real-valued problems. Rather than using the exact current best error value obtained on each fitness case, Epsilon Lexicase Selection eliminates the individuals with error values outside some epsilon of the best error. This alleviates the problem of only one fitness case being used to eliminate candidates due to the nature of the real-valued errors. Epsilon Lexicase Selection can also eliminate the need for a pre-defined epsilon threshold by adapting epsilon based on the current population.

Similar to Lexicase Selection, Comparative Partner Selection [17] considers an individual's performance on the fitness cases. Comparative Partner Selection assigns each individual a binary string fitness characterization (BSFC), which captures the individual's relative performance on each fitness case with respect to its performance

on every other fitness case. Comparative Partner Selection attempts to breed individuals with different BSFCs from each other in order to take advantage of their strengths and improve upon their weaknesses. However, rather than focusing directly on behavioral diversity, Comparative Partner Selection attempts to accelerate convergence to the solution by reducing phenotype variance [17].

Complementary Phenotype Selection [21] is another selection technique that considers the performance of an individual on the fitness cases. Similar to the goal of Lexicase Selection, Complementary Phenotype Selection attempts to breed individuals that perform well on complementary sets of fitness cases in attempt to create offspring that inherit the strengths of the parents and improve upon their weaknesses. To accomplish this, the first parent is selected with a traditional selection method (we used tournament selection in our experiments). Next, for every other individual in the population, an imaginary output vector, $B_{F,M}$, is created by taking the best error between the individual and the first parent on each fitness case. $B_{F,M}$ represents the output vector of an offspring that would perform as well as the better performing parent on each fitness case. The individual that creates the fittest $B_{F,M}$ is selected as the second parent.

Interleaved Sampling (IS) [50] also uses fitness case performance to select individuals for breeding. In contrast to Lexicase Selection, IS alternates between using all of the fitness cases for calculating fitness in certain generations and a single, randomly-selected fitness case in other generations. Similarly, Keep-Worst Interleaved Sampling [50] selects some number of the current most difficult fitness cases rather than selecting a single fitness case randomly in the generations in which the entire set of fitness cases is not used.

Implicit fitness sharing (IFS) [52] incorporates fitness case performance into the fitness calculation. Based on the original fitness sharing concept [25, 26], IFS reduces

the fitness reward given by each fitness case based on how many individuals solve that fitness case. Since individuals are selected based on fitness, IFS effectively reduces the likelihood of selecting individuals that perform similarly.

Rather than directly considering the fitness case performance, Semantic Sharing [58] compares the semantic similarity of individuals in order to reduce the fitness of semantically similar individuals. In order to determine semantic similarity, Sampling Semantics Distance [58] compares the output vectors of individuals and uses a threshold of dissimilarity. Semantic Sharing was shown to improve the performance of GP, while the compared syntactic sharing method did not improve GP performance [58].

Semantics in Selection (SiS) [23] attempts to promote behavioral diversity while still maintaining strong selection pressure for improved fitness. Based on the commonly used tournament selection operator, SiS is used to breed fit individuals that are also behaviorally different from each other. SiS selects the first parent based solely on fitness, i.e. using the original tournament selection method. To select the second parent, SiS holds another tournament to find the fittest individual that is also behaviorally different from the first parent. Unlike other approaches, such as Semantic Sharing, that employ a measure of behavioral similarity [58], SiS considers two individuals to be behaviorally unique if their output vectors differ by at least one element.

2.2.3 Semantic-Aware Operators

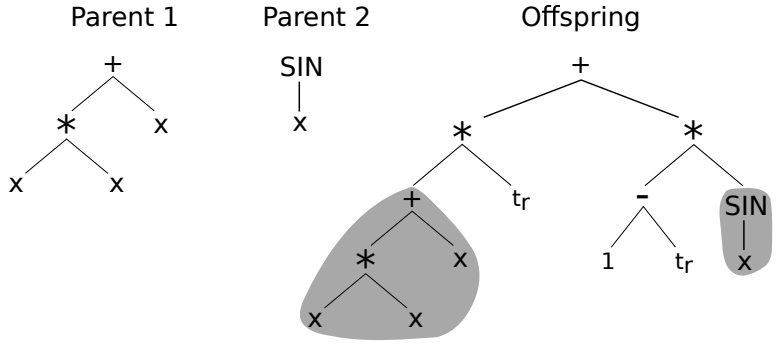
In recent years, several operators that utilize the semantics of individuals during crossover have been introduced [61, 76]. Although some of the methods we discuss in this section do not primarily focus explicitly on diversity, they affect behavioral diversity since they consider the behavior of trees instead of completely randomly

modifying trees. A major limitation of the widely-used random subtree crossover is that it makes no consideration of the context in which the selected subtrees occur or the semantics of the selected subtrees. Therefore, random subtree crossover has been considered to be destructive because swapping subtrees into random contexts, irrespective of the semantic effects, can potentially destroy good solutions [49].

In an effort to overcome the destructive nature of random subtree crossover, context-aware crossover (CAC) [49] exhaustively locates the best possible context at which to insert the subtree from the donating tree into the receiving tree. This is done by randomly selecting a subtree in one parent and swapping the subtree at all possible crossover points in the other parent, excluding the root. Each resulting offspring is evaluated, and the offspring with the highest fitness is chosen. Although this does not directly utilize the semantics of the subtree to be swapped, it considers the semantic effect of each possible crossover by determining the change in fitness. Despite increasing the number of fitness evaluations required per generation, CAC was shown to perform better than random subtree crossover [49].

Other approaches that generate several offspring (referred to as a brood) from two parents have been proposed. Semantically Driven Crossover (SDC) [6] performs random subtree crossover between two parents and only allows offspring to enter the population if they are semantically different from the parents. This is similar to offspring selection [1] in that the process continues until it creates an offspring that satisfies the criteria for acceptance (being semantically different in the case of SDC). The approximately geometric crossover) [40] generates n offspring and selects the offspring whose semantics are as close as possible to the linear combination of the two parents. Other crossover operators have been proposed that consider the semantic similarity of the subtrees exchanged between parents, as well as the semantic equivalence of the children [57, 74].

Figure 2.3: **Semantic Geometric Crossover for Real-valued Expressions.**



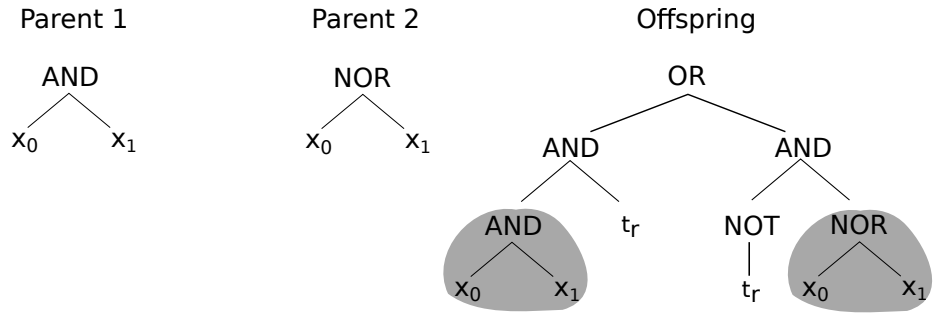
Semantic Geometric Crossover (SGX) [55] was designed to ensure that the semantics of an offspring lie somewhere on the segment connecting the semantics of its parents in the semantic space. SGX creates an offspring tree as a convex geometric combination of the two parent trees (with respect to the semantic space), by combining both trees with either a random constant or a random expression. Figure 2.3 and Figure 2.4 demonstrate how SGX combines two parent trees t_1 and t_2 to create the offspring for real-valued and Boolean domains using Equation 2.2 and Equation 2.3, respectively, where t_r is a randomly-generated expression (or possibly a random constant in the case of the real-valued domain).

$$t_1 * t_r + (1 - t_r) * t_2 \tag{2.2}$$

$$(t_1 \wedge t_r) \vee (\neg t_r \wedge t_2) \tag{2.3}$$

SGX has many appealing properties that improve upon random subtree crossover. First, SGX transforms the fitness landscape traversed by GP into a conical, unimodal landscape since the fitness of an individual is based on its semantic distance to the solution [55, 61]. Furthermore, a geometric crossover guarantees that the fitness of the offspring is no worse that of its worse parent [61].

Figure 2.4: **Semantic Geometric Crossover for Boolean Expressions.**



Although SGX has very attractive properties, the main drawback of its original description [55] is that, as Figure 2.3 and Figure 2.4 show, the offspring will contain the entirety of both parents' trees in addition to a randomly-generated subtree (or constant), which means that tree size grows exponentially over time. However, extensions have been proposed for alleviating this issue [59, 75].

Subtree Semantic Geometric Crossover (SSGX) [59] was recently proposed as an extension to SGX. SSGX attempts to alleviate the issue of exponential tree growth of the original SGX operator. Instead of using the entire parent tree, SSGX selects from the parent the subtree with semantics as close to the semantics of the root of the tree. Furthermore, to prevent selecting a large subtree, SSGX only selects subtrees within a pre-defined size range. This way, SSGX chooses a subtree that is smaller than the parent and approximates the original semantics of the parent. In a set of experiments in the symbolic regression problem domain, SSGX performed similar or better compared to SGX, and significantly reduced tree growth [59].

One issue with current semantic geometric crossovers such as SGX is that although the created offspring are geometric (i.e. the semantics of the offspring lie on the segment connecting the semantics of the parents), crossover may not always produce offspring that are semantically effective (different from the parents). This is because, by definition, the child can be semantically equivalent to one of the

parents and still be geometric [61]. However, this has been addressed to varying extents [42, 43, 44, 62].

Locally Geometric Semantic Crossover (LGX) [42, 44] improves upon random subtree crossover by attempting to find a semantically medial subtree to place into both parents in order to increase the likelihood that the behavior of the resulting offspring will be between that of the parents in the behavioral space. Semantic mediality of offspring was shown to be very valuable in reaching the desired solution behavior [42, 44]. Rather than attempting to directly ensure that the entire offspring program is semantically medial with respect to its parents, LGX attempts to find a subtree that is semantically medial to the subtrees at the selected crossover points in each parent. The intuition behind this is that finding a semantically medial subtree may be easier than creating an offspring that is perfectly semantically medial.

LGX first selects the subtrees to be exchanged between the parents. Next, LGX calculates the midpoint, m, of the semantics of the parents' subtrees, and sets m as the desired semantics for the crossover. A library of subprograms is then searched to find a subprogram with semantics closest to m. LGX also considers the context of the subtrees by only allowing crossover to occur at a homologous locus in each parent (i.e. a locus at which the surrounding structure of the trees is the same).

In addition to the inception of semantic geometric crossover operators, semantic geometric mutation operators were also introduced [55]. In this case, the semantics of the generated offspring are guaranteed to lie within the geometric ball (in the semantic space) centered around the parent's semantics with a radius within some epsilon. Since SGX is incapable of creating offspring that lie outside the convex hull of the population, a solution will not be found if the solution is not contained within the convex hull of the population [61]. Using semantic geometric mutation alongside SGX can help overcome this issue [61].

Chapter 3

The Genetic Marker Diversity Algorithm for GP

We recently introduced the Genetic Marker Diversity Algorithm for GP (GMD-GP) as a structural diversity maintenance technique for tree-based GP [13]. First, we discuss the motivation behind GMD-GP. Next, we present our novel structural diversity measure and discuss how GMD-GP uses this measure for sustaining structural diversity.

3.1 Motivation behind GMD-GP

Two major motivations behind the design of GMD-GP are as follows. First, GP populations have a tendency to rapidly converge to a single rooted structure [53]. Second, certain tree structures (size and shape) are much more difficult for GP to evolve [16, 27]. The rooted structures of the trees are important because they define the overall context of the programs, and thus modifications to a tree nearer the root have more of an impact on the functionality of the tree. This is especially important in early generations because large changes to program behavior can be

seen as exploratory steps in the search space. Although it is generally accepted that less exploration is needed in later generations as good programs are further refined (referred to as exploitation), exploration is an invaluable tool for helping the population escape local optima and avoiding premature convergence.

Furthermore, since certain structures are more difficult for GP populations to reach, if an optimal solution has such a structure, the likelihood of finding the solution is very low, if not impossible. Since upper portions of the trees tend to remain untouched by standard crossover [24], if the population does not converge to the correct rooted structure, it is very unlikely that the structure will be found in later generations (albeit that for problem domains such as symbolic regression, there could be many structures that encode an optimal solution). Maintaining a diverse set of structures in the population may alleviate this issue.

GMD-GP attempts to prevent the population from converging to a single rooted structure by imposing selection pressure to maintain a diverse set of rooted structures in the population throughout the evolutionary process. In order to achieve this, we use the rooted fragments of the trees as *genetic markers* and select against a single genetic marker dominating the population. GMD-GP is fundamentally different from some of the other genotypic diversity techniques that we discussed in Chapter 2 in that GMD-GP only considers a fragment of the tree rather than the entire genotype and GMD-GP does not require an explicit genotypic distance metric.

3.2 Genetic Markers

In order to promote structural diversity, GMD-GP uses genetic markers, which are tree fragments that capture the structure of the trees in the population. Genetic markers are constructed by traversing a tree in depth-first order from a starting

Figure 3.1: **A Rooted Genetic Marker.** A depth-first traversal from the root to level 2 is performed to create the genetic marker, m = (+ (*) (SIN)).

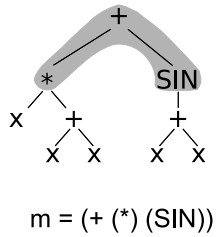
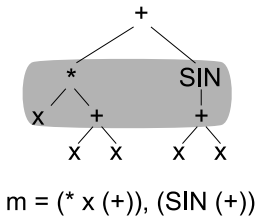


Figure 3.2: **A Non-rooted Genetic Marker.** A depth-first traversal from level 2 to level 3 is performed to create the genetic marker, m = (* x (+)), (SIN (+)).



level, L_i, down to an ending level, L_j. All the nodes in the tree from L_i to L_j will be contained in the genetic marker, which is a partial Lisp-style expression. We chose to use Lisp-style expressions as this is a common way of representing GP trees as character strings [39].

Figure 3.1 and Figure 3.2 illustrate how genetic markers are constructed. In Figure 3.1, the genetic marker is constructed from the root of the tree (level 1) down to level 2. Since the genetic marker does not contain the entire tree, the resulting partial Lisp-style expression simply adds closing parentheses around the function nodes whose children do not belong to the genetic marker. This way, only the fragment of interest is preserved. Figure 3.2 illustrates that a genetic marker can begin at an arbitrary level in the tree. Since the fragment of interest is non-rooted, the resulting genetic marker is a list of partial expressions in left-to-right order. The ordering allows the genetic marker to retain the structure of the tree even though the root does not belong to the genetic marker.

Genetic markers can be used to provide an estimate of the structural diversity

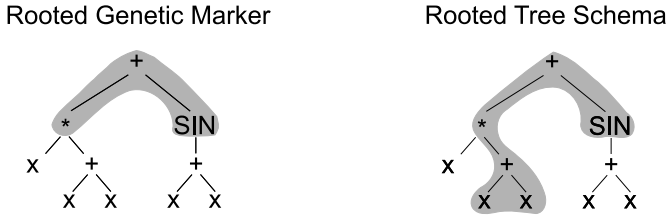
of a given population. To do so, GMD-GP groups members of the population by their genetic markers and considers the density of each genetic marker. This measure allows GMD-GP to determine how widespread a particular structure is in the population. The density, ρ_m, of a genetic marker m in population P is calculated as in Equation 3.1, where $m(p)$ is a function that determines the genetic marker of individual p as we described above.

$$\rho_m = \frac{|P_m|}{|P|}, P_m = \{\forall p \in P \mid m(p) = m\} \tag{3.1}$$

Using Equation 3.1, the densities of all the genetic markers in a population of size n can be calculated in $O(n)$ time, since each individual only needs to be considered once. Furthermore, GMD-GP only uses the top portion of a tree for promoting structural diversity, which means that only a small fragment of the tree needs to be traversed in order to construct the genetic marker. Therefore, a negligible amount of overhead is imposed upon the system in order to construct genetic markers as well as to determine the density of each genetic marker. While it is well-known that the fitness evaluation is usually the most computationally expensive phase of EAs for complex problems, imposing a negligible amount of additional expense is an attractive property of genetic markers. Also, it is possible to further improve efficiency by constructing the genetic marker when the tree is newly generated, or when the nodes are traversed during fitness evaluation.

We can see from the above definition of genetic markers and the corresponding density measure that genetic markers share some properties with some of the schema definitions for GP [63, 65]. Schemata, which are essentially fragments of the genotype, are used to identify common genetic regions shared by individuals in the population. The schema theorem [29] states that short, low-order schemata known as building blocks are combined to form higher-order building blocks which even-

Figure 3.3: **Rooted Genetic Marker vs. Rooted Tree Schema.** A rooted genetic marker containing the top two levels of a tree compared to a rooted tree schema matching a different portion of the same tree.



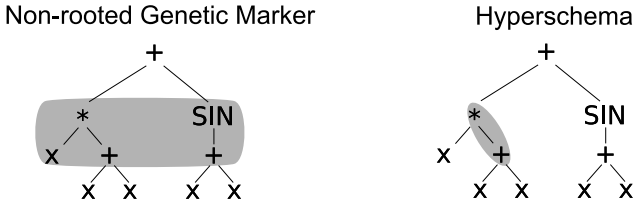
tually lead to solution discovery. The schema theorem allows us to reason about the utility of such building blocks, as well as to predict the number of instances of a particular schema in future generations. Although schemata are easier to define in fixed-length representations such as binary GAs, there have been many schema definitions proposed for GP as well.

The rooted tree schema [65] is perhaps the most closely related schema definition to genetic markers. However, there are some key differences worth discussing. Figure 3.3 shows that, similar to rooted genetic markers, rooted-tree schemata [65] begin at the root of the tree and span across a contiguous fragment of the tree. However, unlike genetic markers, rooted tree schemata are not strictly required to span across an entire level in the tree, as Figure 3.3 illustrates.

Similarly, a non-rooted genetic marker can also be considered as an instance of a hyperschema [63]. However, because hyperschemata employ a "don't-care" symbol in order to provide flexibility and expressiveness, not all hyperschemata can be considered non-rooted genetic markers since genetic markers must span across an entire level in the tree. Figure 3.4 shows the major difference between a hyperschema and a non-rooted genetic marker.

While genetic markers undoubtedly share properties with GP schemata, they are defined more strictly and serve a different purpose. The main purpose of genetic markers is not to provide a new schema definition but to provide an inexpensive

Figure 3.4: **Non-rooted Genetic Marker vs. Hyperschema.** A non-rooted genetic marker spanning the second and third levels of a tree compared to a hyperschema of the same tree.



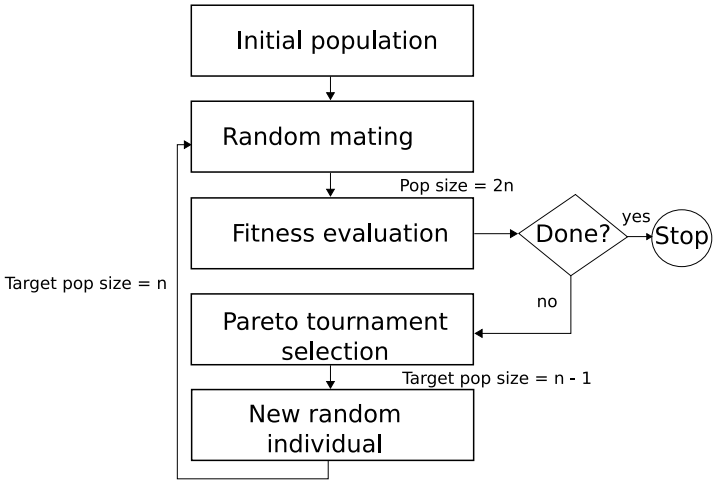
means of estimating structural diversity of the population, namely among the top portions of the trees. In addition to providing an estimate of structural diversity, genetic markers are used in GMD-GP to actively maintain structural diversity.

3.3 GMD-GP Structural Diversity Preservation

GMD-GP uses the density values of the genetic markers in a population in order to prevent a single structure from becoming too widespread. To achieve this, GMD-GP incorporates a multi-objective optimization scheme that simultaneously minimizes genetic marker density while optimizing fitness. This promotes structural diversity by rewarding less-represented structures (and thus preventing a single structure from dominating the population) while still maintaining global selection pressure to improve fitness.

The main evolutionary loop of GMD-GP, which is based on the multi-objective implementation of the Age-Fitness Pareto Optimization scheme [69], is presented in Figure 3.5. Every generation, parents are randomly selected to produce offspring via the typical crossover, reproduction, and mutation operators. In a traditional GP setup, offspring immediately replace their parents and elitism is used to retain the best individuals. Instead, in GMD-GP both the parent population and the offspring are temporarily kept, which increases the population size to $2n$. Next, genetic marker

Figure 3.5: **The Main Evolutionary Loop of GMD-GP.**



density and fitness are used with Pareto tournament selection [69, 71] to remove dominated individuals from the population in order to reduce the population size to $n-1$. The target population size is $n-1$ instead of n because the final step of the evolutionary loop adds a new randomly-generated individual to the population as a means of providing a constant source of new genetic material.

The Pareto tournament selection method with genetic marker density and fitness provides selection pressure for less-represented structures that are also fitter. As its name suggests, Pareto tournament selection holds a tournament among a randomly chosen subset of individuals in the population, and keeps only the individuals that satisfy the Pareto selection criteria. Since multiple objectives are used, only the individuals that are not dominated on all objectives by another individual in the tournament will survive. In GMD-GP, each individual is assigned the density value corresponding to its genetic marker, and then density and fitness are used by Pareto tournament selection to determine domination. In this case, individual A dominates individual B if A is at least equal to B in density and fitness and A is strictly better than B in either density or fitness. If individuals are strictly equal on both objectives, GMD-GP chooses the smaller of the two, or randomly if they are also the same size.

In order to reduce the population size from the temporary size of $2n$ to $n-1$, Pareto tournament selection is repeatedly applied to remove dominated individuals from the population. Therefore, only the current set of non-dominated individuals is guaranteed to survive to the next generation. It is theoretically possible that the set of non-dominated individuals can grow to cover the entire population [69], causing the population to grow beyond n since a new individual is added each generation. However, we have not observed this phenomenon in any of our experiments using genetic marker density and fitness as the Pareto objectives, and it was also not observed in the Age-Fitness Pareto Optimization algorithm [69]. Furthermore, it is possible to restrict the growth of the non-dominated front by modifying the domination criteria [18].

As Figure 3.5 shows, the final step of the evolutionary loop of GMD-GP introduces a new randomly-generated individual into the population. This provides a constant source of new genetic material, which may contain essential building blocks that can later propagate throughout the population. Since one of the goals of GMD-GP is to sustain high diversity, the genetic marker density values tend to be very low. Therefore, the newly generated individual, which would likely contain a unique genetic marker, would likely be less fit than many of the members of the population while it could have the same density value of $\frac{1}{n}$. In order to prevent the new individual from being immediately dominated by a fitter individual with the same density value, we simply add the new individual to the population after the Pareto tournament selection phase, which differs slightly from the Age-Fitness Pareto Optimization algorithm.

Chapter 4

Experimental Validation of GMD-GP

In this chapter, we provide an empirical comparison and analysis of the search performance and behavior of GMD-GP and several state-of-the-art techniques as well as traditional GP. We provide a detailed comparison of the performance and behavior of GMD-GP and the different techniques on a large suite of benchmark problems. These results demonstrate the feasibility of GMD-GP and provide motivation for the work that we will present in the following chapters.

4.1 Benchmark Problem Suite

Before we delve into the experimental results, we first describe the benchmark problem suite that we used to compare the different techniques. In order to provide a thorough comparison and analysis of GMD-GP versus other state-of-the-art techniques, we included a large set of problems of varying difficulty. The problems fall into one of two domains: symbolic regression (a total of 15) and finite algebras (a

total of 10).[1]

The symbolic regression problems, which were chosen from the set of recommended GP benchmark problems in previous work [51], include 8 univariate and 7 bivariate problems of varying difficulty. Symbolic regression is a commonly used problem domain for GP because the tree-based representation provides a flexible means of expressing mathematical functions. Symbolic regression is also a challenging benchmark because GP often spends a lot of the computational budget by only making incremental improvements to sub-optimal solutions, which can severely limit its ability to discover an optimal solution [47].

In the symbolic regression task, GP must evolve a mathematical expression that fits a target function given a set of input points. Fitness is calculated based on how close an individual's output is to that of the target function on all the input points. We used the cumulative absolute error, normalized between 0 and 1 with the following equation.

$$f = \frac{1}{1 + error} \tag{4.1}$$

Table 4.1 lists all the symbolic regression problems, along with the training and testing data that we used. We used the corrected versions[2] of the domains for the training and testing data that were originally given in [51]. The function set {+, -, *, %, SIN, COS, LOG, EXP, $-x$ } was used for all the symbolic regression problems. The terminal set includes a single variable x for all the univariate problems and an x_0 and x_1 for the bivariate problems. We did not include random constants in the terminal set, similar to previous work [41].

The second class of problems in our benchmark suite were selected from [73], in which GP produced human-competitive solutions to finite algebras problems. Each

[1]For the experiments comparing GMD-GP to SSGX, we did not include the finite algebras problems since SSGX was not designed for finite values.

[2]https://cs.gmu.edu/ sean/papers/gecco12benchmarks3.pdf

Table 4.1: **Symbolic Regression Benchmark Problems.** U[a, b, c] means that c points are uniformly randomly selected between a and b, inclusive. E[a, b, c] means that the points are deterministically chosen on an evenly-spaced grid from a to b, inclusive, with an interval of c.

Name	**Target Function**	**Training**	**Testing**
SEXT	$x^6 - 2x^4 + x^2$	U[-1, 1, 20]	E[-1, 1, 0.001]
NGUY3	$x^5 + x^4 + x^3 + x^2 + x$	U[-1, 1, 20]	E[-1, 1, 0.001]
NGUY4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	U[-1, 1, 20]	E[-1, 1, 0.001]
NGUY5	$sin(x^2)cos(x) - 1$	U[-1, 1, 20]	E[-1, 1, 0.001]
NGUY6	$sin(x) + sin(x + x^2)$	U[-1, 1, 20]	E[-1, 1, 0.001]
NGUY7	$ln(x + 1) + ln(x^2 + 1)$	U[-1, 1, 20]	E[-1, 1, 0.001]
KEIJ1	$0.3xsin(2\pi x)$	E[-1, 1, 0.1]	E[-1, 1, 0.001]
KEIJ4	$x^3e^{-x}cos(x)*$ $sin(x)(sin^2(x)cos(x) - 1)$	E[0, 10, 0.05]	E[0.05, 10.05, 0.05]
KEIJ11	$xy + sin((x - 1)(y - 1))$	U[-3, 3, 20]	E[-3, 3, 0.01]
KEIJ12	$x^4 - x^3 + \frac{y^2}{2} - y$	U[-3, 3, 20]	E[-3, 3, 0.01]
KEIJ13	$6sin(x)cos(y)$	U[-3, 3, 20]	E[-3, 3, 0.01]
KEIJ14	$\frac{8}{2+x^2+y^2}$	U[-3, 3, 20]	E[-3, 3, 0.01]
NGUY9	$sin(x) + sin(y^2)$	U[-1,1, 100]	E[-1, 1, 0.001]
NGUY10	$2sin(x)cos(y)$	U[-1,1, 100]	E[-1, 1, 0.001]
NGUY12	$x^4 - x^3 + \frac{y^2}{2} - y$	U[0, 1, 20]	E[0, 1, 0.001]

Table 4.2: **Input-output Mappings for the Operators A1 - A5 Used in the Finite Algebras Problems.**

A1				**A2**				**A3**				**A4**				**A5**			
	0	1	2		0	1	2		0	1	2		0	1	2		0	1	2
0	2	1	2	0	2	0	2	0	1	0	1	0	1	0	1	0	1	0	2
1	1	0	0	1	1	0	2	1	1	2	0	1	0	2	0	1	1	2	0
2	0	0	1	2	1	2	1	2	0	0	0	2	0	1	0	2	0	1	0

of these problems includes a single binary operator that has an underlying algebra which is ternary. The algebras have defined outputs for the finite set of inputs $\{0, 1, 2\}$. The algebras, A1 - A5, which are defined in Table 4.2 were included in the benchmark suite. For each algebra, there are two tasks, which yields a total of 10 finite algebras problems.

The first task is to evolve a *discriminator term*, which must return x if $x \neq y$, and z for all other cases. The fitness cases for this task are simply all combinations of the three inputs $\{0, 1, 2\}$, which results in $3^3 = 27$ total fitness cases. The second task is to evolve a *Mal'cev term* which is defined by the following; $m(x, x, y) = m(y, x, x) = y$. Given the definition, only the specified input combinations have a defined output (the cases where all of the inputs are different from each other are not handled). Therefore, this yields a total of 15 fitness cases.

For both tasks, the function set contained the single operator, A_i for the algebra being used for the problem, and the terminal set, $\{x, y, z\}$ contained a single terminal node corresponding to the inputs. Fitness was calculated as a function of the total error committed on the fitness cases. We also included penalty for single node solutions as in [73].

4.2 Experimental Settings

We conducted two sets of experiments in order to compare and analyze the search behavior and performance of the different techniques we considered. In the first set of experiments, we compared the performance of our approach to several different methods in terms of the speed of convergence to a solution and the percentage of trials in which a solution was found. The second set of experiments was designed to analyze and compare different aspects of population diversity between the different techniques.

Table 4.3 lists the general GP settings that we used for our experiments. For the performance comparisons, we allocated a computational budget of 1 million fitness evaluations for each technique, and terminated the trial when a fitness of 1.0 was reached. All of the results were gathered from 100 independent trials of each algorithm for each problem. For the diversity experiments, we allowed each trial to last for 250,000 fitness evaluations and we collected the different metrics that we discuss below throughout the duration of the trial. For the metrics that are population-based, we collected the data at the end of the generation closest to the nearest thousandth fitness evaluation (since the trial is based on fitness evaluations instead of generations).

The algorithm-specific settings for ALPS were chosen in attempt to balance the number of individuals per age layer, and the frequency of migration and re-initialization of the initial layer given the population size. For SSGX, we used the settings that were used by its authors [59]. We used the top three tree levels for the genetic marker density objective in GMD-GP, as the preliminary experiments in previous work suggested that this performed best on a symbolic regression problem [13]. For the Lexicase Selection extensions that we discussed in Chapter 6, we used the common tournament size of 7 individuals, although we plan to analyze their

Table 4.3: **General GP Settings.**

Parameter	**Value**
Random initialization	Ramped half & half
Maximum evaluations	1 million
Total independent trials	100
Population size	500
Tournament size	2
Crossover probability	0.90
Reproduction probability	0.10
Max. tree depth	17
Max. tree size	512
Elites (GP, Lex, SiS, SSGX)	50 (10%)
ALPS age layers	10
ALPS age gap	20
ALPS Elites	2 per layer
GMD-GP genetic marker start	Level 1 (root)
GMD-GP genetic marker depth	2
SSGX min subtree size	1
SSGX max subtree size	90
SSGX max trials	20

performance using different tournament sizes in the future.

4.3 GMD-GP Performance Comparison

We begin with a comparison of the search performance of GMD-GP versus other GP methods. We used a standard GP setup as a baseline, and compared GMD-GP to the techniques listed in Table 4.4. We refer to a standard GP setup as the traditional generation-based GP using tournament selection, random subtree crossover, reproduction, and no mutation. For all of the metrics we consider, we used a pairwise Mann-Whitney U test to determine statistical significance with $\alpha = 0.05$, except where specified otherwise.

Figure 4.1 shows the mean best fitness over time for each technique for each of the symbolic regression benchmark problems. In order to provide a fair comparison,

Table 4.4: **List of GP Techniques Used in the GMD-GP Performance and Diversity Comparison.**

Technique	**Abbreviation**
Standard GP	GP
Age-Layered Population Structure	ALPS
Age-Fitness Pareto Optimization	A/F
Lexicase Selection	Lex
Semantics in Selection	SiS
Subtree Semantic Geometric Crossover	SSGX

we used fitness evaluations as a measure of time instead of generations since the algorithms consume a different number of fitness evaluations per generation. Figure 4.1 shows that GMD-GP was consistently either the top performing or among the top methods in terms of reaching the higher fitness values significantly faster than the other approaches. This demonstrates that GMD-GP is able to reach high fitness relatively fast while focusing on maintaining diversity. This is often not the case for diversity-promoting techniques, since avoiding premature convergence can come at the cost of requiring longer to converge to a solution.

The plot for the KEIJ13 problem in Figure 4.1 illustrates how GMD-GP tended to not only reach higher fitness values significantly faster, but the best fitness at the end of the trial was also higher than in the other approaches. This is further confirmed by Table 4.5, which shows the mean best overall fitness in each of the symbolic regression problems. In 10 out of 15 problems, GMD-GP reached the highest fitness value at the end of the trial, on average. Furthermore, the largest number of problems in which any other algorithm achieved the highest ending fitness and GMD-GP did not was only 4 out of 15.

We further compared the convergence rate of each technique across all the symbolic regression problems. The convergence rate measures the mean number of fitness evaluations taken to find a perfect fitness solution, in all of the trials in which a solu-

Figure 4.1: **Symbolic Regression Benchmarks - Fitness over Time.** The x-axis shows the total elapsed fitness evaluations, and the y-axis shows the mean best fitness. For clarity, error bars are not shown due to the number of methods plotted.

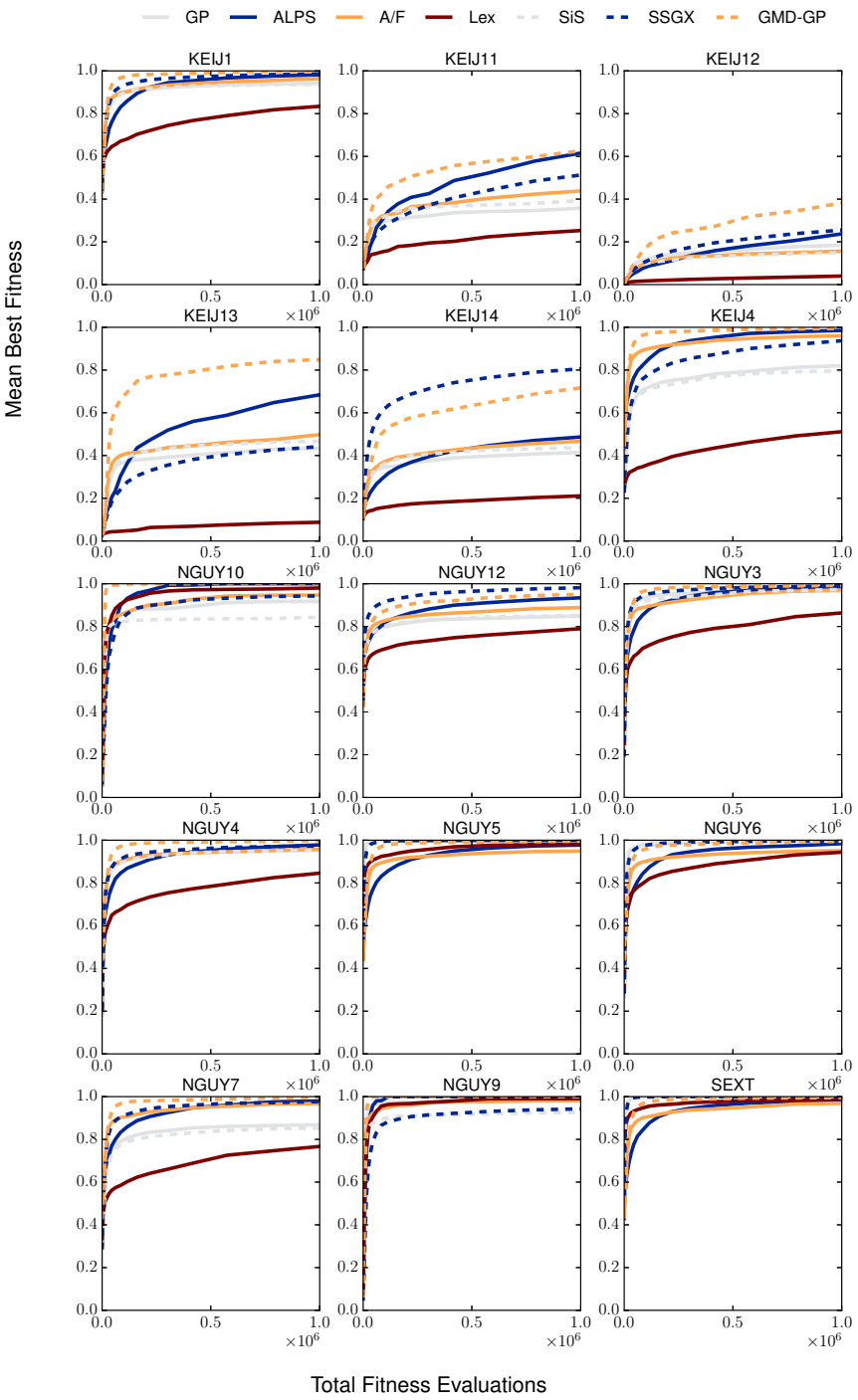


Table 4.5: **Symbolic Regression Benchmarks - Mean Best Ending Fitness.** The highest value is shown in bold font.

	GP	ALPS	A/F	Lex	SiS	SSGX	GMD-GP
SEXT	**1.0**	0.985	0.969	0.989	0.999	**1.0**	0.996
KEIJ1	0.942	0.982	0.964	0.834	0.937	0.986	**0.998**
KEIJ4	0.820	0.985	0.961	0.512	0.796	0.937	**0.996**
KEIJ11	0.357	0.615	0.438	0.253	0.394	0.513	**0.626**
KEIJ12	0.184	0.237	0.157	0.040	0.148	0.255	**0.382**
KEIJ13	0.433	0.684	0.498	0.088	0.467	0.442	**0.849**
KEIJ14	0.414	0.487	0.467	0.211	0.438	**0.806**	0.717
NGUY3	0.967	0.983	0.971	0.864	0.973	**0.992**	**0.992**
NGUY4	0.953	0.978	0.957	0.846	0.954	0.973	**0.998**
NGUY5	**1.0**	0.977	0.949	0.981	0.999	0.999	0.996
NGUY6	**0.997**	0.984	0.951	0.944	0.996	**0.997**	0.994
NGUY7	0.868	0.980	0.968	0.767	0.853	0.977	**0.995**
NGUY9	0.980	**1.0**	0.981	0.990	0.926	0.943	**1.0**
NGUY10	0.918	**1.0**	0.947	0.980	0.844	0.943	**1.0**
NGUY12	0.849	0.934	0.888	0.790	0.852	**0.982**	0.951

tion was found. While this measure is influenced by the computational budget that we set for each trial, it still provides an indication of the speed (in terms of fitness evaluations) at which the different techniques discover a solution within the given computational budget. Furthermore, since the fitness evaluation phase is known to be the most time-consuming step of the evolutionary process for complex problems, being able to find a solution in fewer fitness evaluations is highly desirable.

Figure 4.2 shows a pairwise comparison of the convergence rates of each technique. Each plot shows the percentage of problems in which the technique listed above the plot found a solution in significantly fewer fitness evaluations than the other techniques shown in the plot. GMD-GP faired well against the other approaches, as the highest percentage of problems in which any algorithm found a solution significantly faster was 20% (for GP and SiS), while, most notably, GMD-GP was significantly faster than ALPS and SSGX in 86% and 60% of the problems, respectively.

Next, we compared the success rates of each method, where the success rate is

Figure 4.2: **Pairwise Comparison of Convergence Rate.** Each plot shows the percentage of problems in which the method above the plot found a solution in significantly fewer fitness evaluations, on average, in the successful trials.

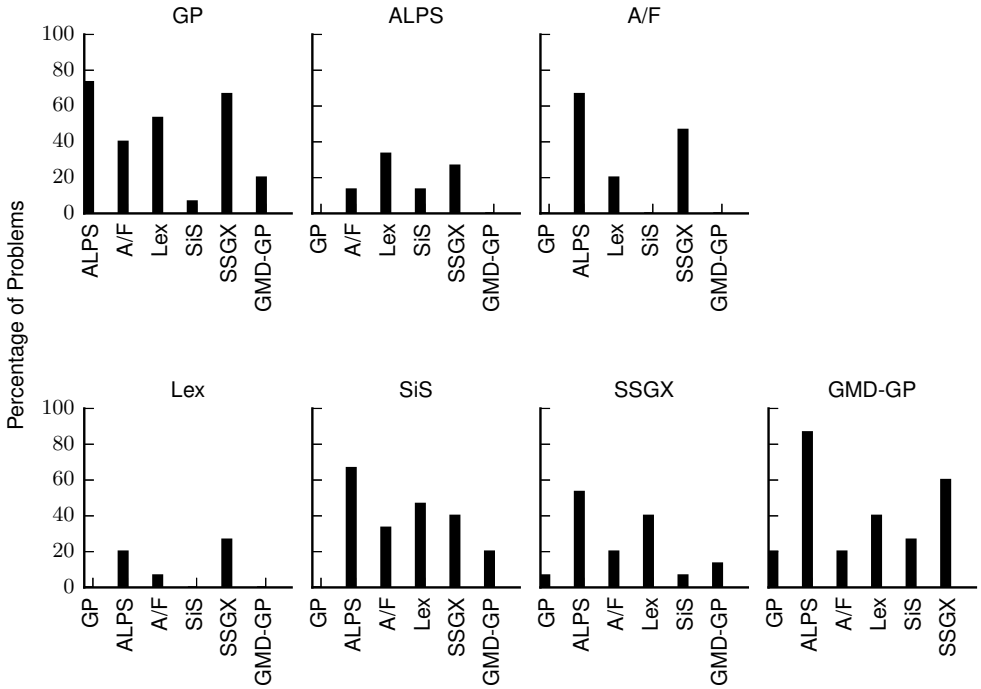
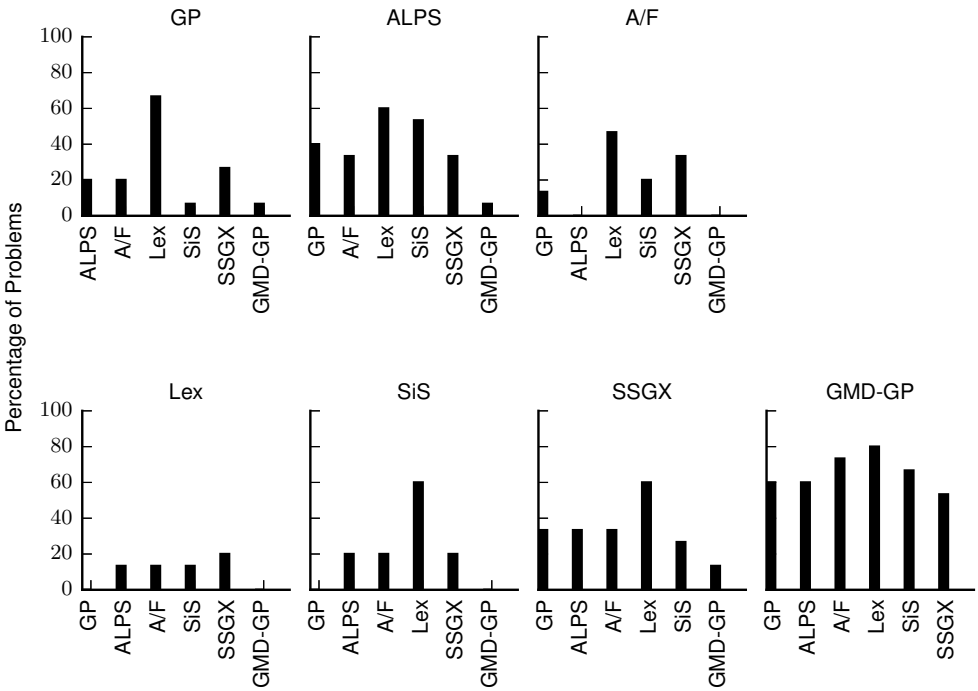


Figure 4.3: **Pairwise Comparison of Success Rate.** Each plot shows the percentage of problems in which the method above the plot had a significantly higher success rate than each other method.



the proportion of trials (out of 100) in which a fitness of 1.0 was reached within the computational budget. We used Fisher's exact test with $\alpha = 0.05$ for determining statistical significance between the differences in success rates. Figure 4.3 clearly demonstrates that GMD-GP outperformed all of the other approaches in terms of success rate. GMD-GP had a significantly higher success rate than every other algorithm in at least 60% of the problems (except for SSGX against which GMD-GP had a significantly higher success rate in 53% of the problems). Combining the fitness and convergence results we discussed earlier, this shows that GMD-GP is capable of often finding solutions relatively faster than the other techniques, as well as finding solutions at a higher rate of success.

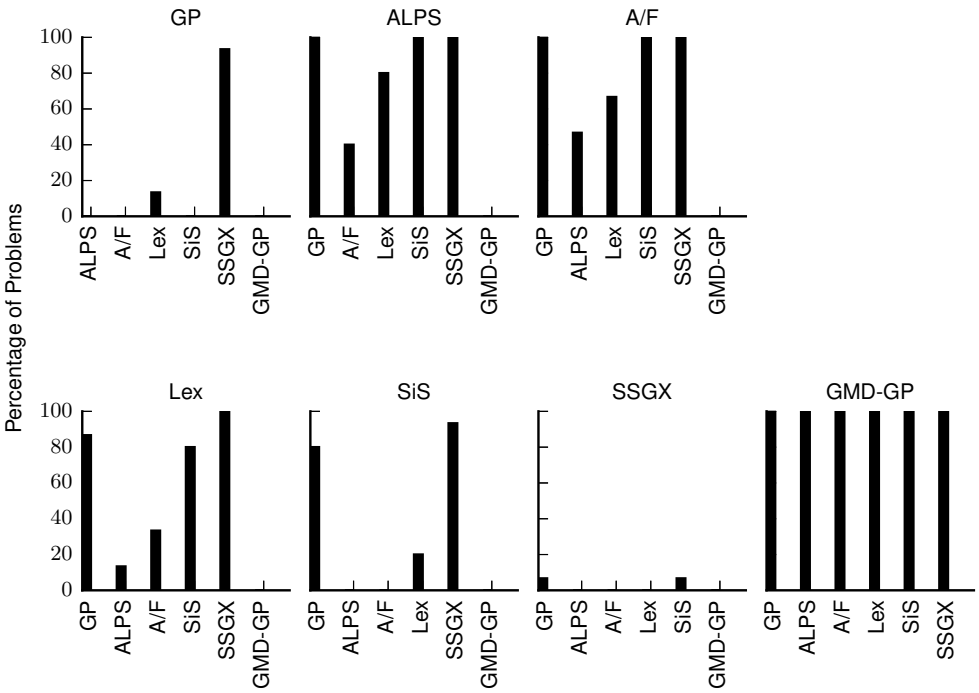
4.4 GMD-GP Diversity Comparison

Next, we conducted a separate set of experiments to analyze the effects GMD-GP has on different aspects of population diversity compared to the other approaches. We conducted another set of 100 independent trials on each problem for each algorithm using the same experimental settings as in Table 4.3 except where we specify differently. We allowed each trial to last for 250,000 fitness evaluations, which is roughly equivalent to 500 generations without elitism and replication, given the population size of 500 individuals. The general trend we observed for each measure we discuss is that the value tended to remain fairly constant after several fitness evaluations. Therefore, we compared the value of each metric at the end of the trial. We used pairwise Mann-Whitney U tests to determine statistical significance between the differences for each measure.

Since GMD-GP focuses on using genetic markers for promoting structural diversity, we first compared the mean maximum genetic marker density of final populations of the different approaches. Similar to our previous study [13], we analyzed the top six tree levels by creating genetic markers composed of two levels at a time. Using multiple genetic markers made of two levels at a time is more informative than a single genetic marker that spans the top six levels. For example, two trees that are identical in the top six levels except for a single node difference would be treated as having different genetic markers although they are otherwise the same.

Figure 4.4 shows the percentage of problems in which each method had significantly lower mean maximum genetic marker density, for genetic markers composed of the top two tree levels at the end of 250,000 fitness evaluations. We report the comparison of the top two tree levels, as the results across the top six levels are generally the same. In all of the problems, compared to all the other algorithms, GMD-GP had significantly lower genetic marker density, which reflects significantly

Figure 4.4: **Pairwise Comparison of Genetic Marker Density.** Each plot shows the percentage of problems in which the method above the plot had significantly lower genetic marker density (of genetic markers composed of the top 2 tree levels) than each other method after 250,000 fitness evaluations.



higher structural diversity. This can be expected since GMD-GP actively focuses on minimizing genetic marker density. However, since this is a major difference between GMD-GP and the other techniques, this suggests that the higher structural diversity contributes to the performance gains that we discussed in Section 4.3.

Next, we compared the behavioral diversity of the populations of the different approaches, where an individual's behavior is defined as the vector of outputs produced on the fitness cases as we described in Section 2.2.1. The behavioral diversity of a population is defined as the proportion of unique behaviors, where two behaviors are considered different if they differ by at least one output [36]. Since the outputs for the symbolic regression problems are real-valued, we used a threshold of 0.01 for considering two outputs to be different.

Figure 4.5: **Pairwise Comparison of Behavioral Diversity.** Each plot shows the percentage of problems in which the method above the plot had significantly higher behavioral diversity than each other method after 250,000 fitness evaluations.

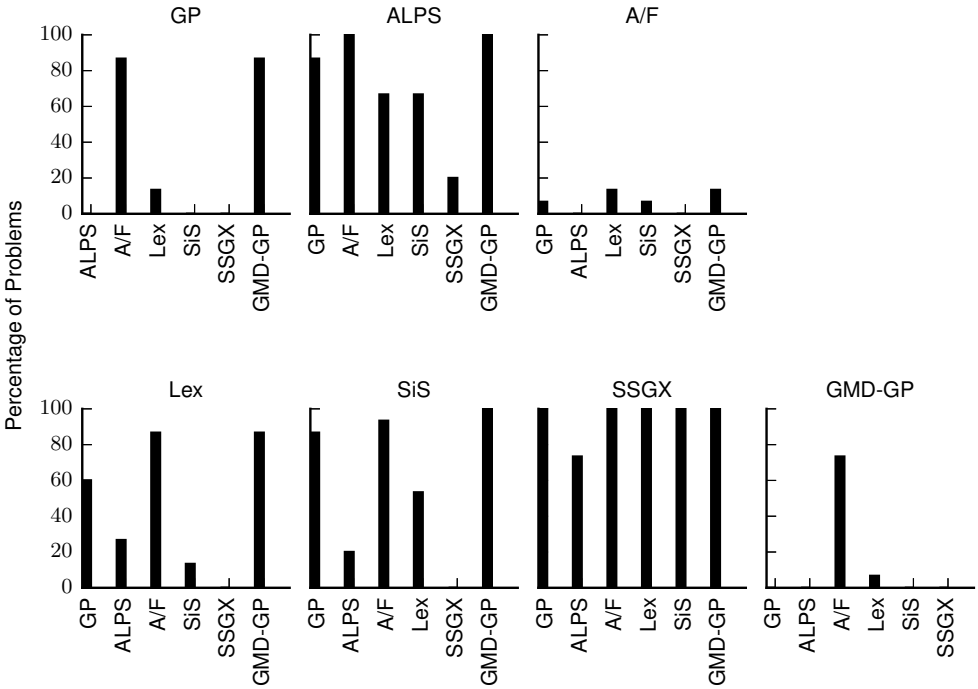
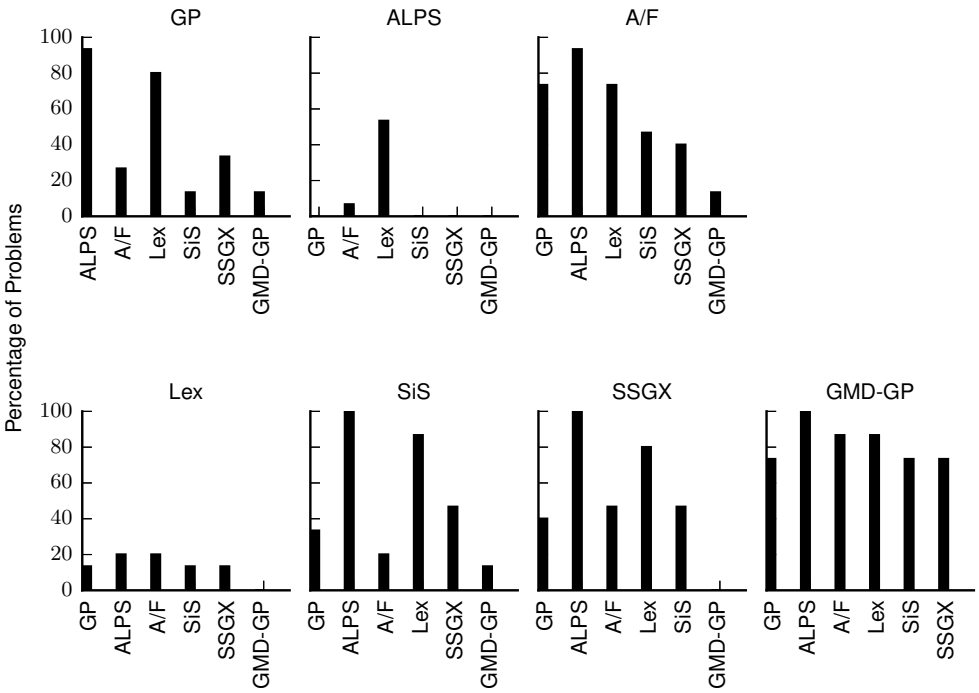


Figure 4.5 shows the pairwise comparison of the behavioral diversity of the populations after 250,000 fitness evaluations. Interestingly, while GMD-GP populations had significantly higher structural diversity than all the other approaches, it had significantly lower behavioral diversity than all of the other approaches except A/F. This clearly demonstrates that high structural (and therefore genotypic) diversity indeed does not correlate directly to high behavioral diversity. However, our results suggest that high structural diversity does still lead to better search performance in GMD-GP.

Finally, we compared GMD-GP and the other approaches in terms of the fitness standard deviation. As we discussed in Section 2.1, it is beneficial to maintain a spread over the range of possible fitness values since intermediate fitness individuals (i.e. "stepping stones" [32]) may contain valuable building blocks that could other-

Figure 4.6: **Pairwise Comparison of Fitness Standard Deviation.** Each plot shows the percentage of problems in which the method above the plot had significantly higher fitness standard deviation than each other method after 250,000 fitness evaluations.



wise be lost. Therefore, high standard deviation of fitness is an indication of how spread out individuals are in the fitness space.

Figure 4.6 shows the pairwise comparison of the fitness standard deviation of the population at the end of the trial. GMD-GP populations had significantly higher fitness standard deviation than all the other approaches in a large percentage of the problems. This shows that while GMD-GP populations typically had significantly less behavioral diversity (and therefore fewer unique fitness values), the difference in the fitness of individuals in GMD-GP populations was significantly higher. This also shows that the genetic marker density objective allows less-represented, and potentially less-fit, structures to survive alongside the superior structures. Furthermore, these results suggest that it is indeed the use of genetic markers that contributes to

this capability, as the fitness standard deviation in GMD-GP was significantly higher than that of A/F (which uses a similar multi-objective scheme with genotypic age instead of genetic marker density).

Chapter 5

GP for Tuberculosis Screening: A Case Study

As we discussed in Chapter 1, GP has been widely used in a number of different application areas to solve real-world problems. In this chapter, we present a new application of GP: using GP to detect the presence of tuberculosis in raw X-ray images. While GP has been applied to the problem of image classification, which we discuss in Section 5.2.1, this is the first application of GP for tuberculosis screening from raw X-ray images.

5.1 The Tuberculosis Epidemic

Tuberculosis (TB) is an infectious disease that typically affects the lungs. TB is caused by the bacterium *Mycobacterium tuberculosis*, and it is spread through the air when an infected person expels bacteria through the air by coughing, for example [60]. TB is a major worldwide epidemic that causes millions of deaths each year. According to the World Health Organization (WHO) 2016 global tuberculosis report, there were around 10.4 million new cases of TB and around 1.4 million TB deaths in 2015,

making TB one of the top 10 causes of death worldwide [60]. The WHO 2016 global report also states that there were 132,120 cases of multidrug-resistant TB in 2015 [60]. Furthermore, TB is a leading cause of death among HIV-positive people [60], and opportunistic infections of HIV-positive people further worsen the TB epidemic [4].

Despite the fact that TB is such a serious worldwide issue, it can be treated and cured if it is detected early enough. Detection methods for TB include:

- sputum smear microscopy, wherein sputum (e.g. saliva and mucus) samples are checked to determine the presence of bacteria. Although this method is over 100 years old and it is a slow method, it is the current gold standard definitive diagnostic [4]

- rapid molecular tests such as the Xpert MTB/RIF® assay, which is the only rapid test recommended by WHO as of the 2016 WHO global tuberculosis report [60]. The cost of such methods is an issue for poverty-stricken countries in which the TB epidemic is a huge problem

- culture methods, in which the bacteria can be observed in a laboratory setting. However, the bacteria are slow-growing, meaning that this process can take months [60]

- skin tests, which can determine whether or not a person has been exposed to TB, but do not indicate active TB and can be affected by vaccinations [4].

In addition to the above detection methods, chest X-ray (CXR) images are also used to aid diagnosis. Although, CXR is a mandatory part of each TB evaluation [4], CXR is not used exclusively as the definitive diagnostic tool. Furthermore, many advances in CXR for computer-aided diagnosis have focused on specific abnormalities such as nodule detection or on sub-problems such as lung region segmentation [4, 37].

5.2 Related Work

Before we discuss the experimental results of using our technique for TB screening, we first discuss some relevant work in image classification. Particularly, we discuss the general problem of image classification and discuss some related work using GP for image classification. We then discuss the basic GP image classification approach that we incorporated into our framework in order to perform TB screening from X-ray images.

5.2.1 GP for Image Classification

The problem of detecting TB using raw X-ray images is an image classification problem. The basic image classification problem is as follows: given an image, determine the category to which it belongs based on information contained in the pixels of the image. GP has previously been used for the purpose of image classification [2, 5, 34, 48, 66]. Particular examples include using GP for breast cancer detection [66], detection of cephalometric landmarks [34], and developing domain-independent raw pixel-based GP classifiers [5].

The image classification problem is typically tackled by first performing feature extraction on the images in order to extract useful features from the images to aid in the classification task. This often involves image processing techniques such as performing segmentation [4] to focus on regions of interest and using image descriptors [4] to serve as features for the image. Then, given a set of features for an image, the typical classification techniques can be applied to determine the class to which the image belongs.

A domain-independent genetic programming technique was previously introduced for image classification using raw image data [5]. Since the approach we use for

tuberculosis screening is based on this work, we discuss this methodology in detail. As we discussed above, the typical approach to image classification is to first perform a feature extraction phase on the images and then perform classification using the features. However the three-tiered GP approach, 3TGP, was designed to evolve programs that are capable of performing all the steps in one GP tree [5]. This is accomplished by using a hierarchical tree structure in which each layer is responsible for performing a different task in order to ultimately classify the image.

The 3TGP technique first stores each training input image as an integer array containing the raw pixel values. Then, given an image to classify, the lowest layer in the tree performs low-level image processing techniques such as filtering the raw input image by performing simple mathematical operations to the pixel values of the image (e.g. add a value to each pixel, etc.). The second layer receives as input the resulting filtered image from the first layer, and then performs aggregation functions such as min, max, median, mean, and standard deviation on the pixels of a given window in the image. Finally, the topmost layer receives the single values produced by the aggregation functions and then performs the typical mathematical functions that are often used in symbolic regression, such as those we described in Chapter 4. The root of the tree then returns a single real value, which is used to determine the class of the image; if the value is less than zero, then image belongs to the negative class, and otherwise it belongs to the positive class.

A two-tiered version (2TGP) was introduced as a more effective and more efficient technique to raw image classification than 3TGP [2]. 2TGP uses a very similar architecture to 3TGP. However, rather than relying on three processing layers in the tree, 2TGP removes the need for the filtering functions of 3TGP by directly feeding the raw image data into the aggregation functions. This removes the extra CPU-intensive image filtering stage while also achieving higher classification accuracy than

3TGP [2].

Along with 2TGP, two additional versions (2TGP-line and 2TGP-mix) were introduced, which modify the aggregation functions to only include the pixels contained in certain types of windows in the image. 2TGP-line only considers a single vertical or horizontal line of pixels rather than a rectangular region as in the original 2TGP version. The 2TGP-mix version is the same as 2TGP-line, except it adds the capability of using circles and rectangles in the aggregation functions. All four versions, 3TGP, 2TGP, 2TGP-line, and 2TGP-mix were compared using different two-class image databases, and 2TGP-line and 2TGP-mix were the best performing methods [2]. Furthermore, the two-tiered versions performed better or at least comparably to a GP method that uses pre-defined image features [2].

5.2.2 Tuberculosis Screening from Chest X-rays

An automated technique for TB screening using chest X-rays was proposed in previous work [37]. This technique relies upon a lung segmentation algorithm to first extract the lung region from the rest of the image. Next, feature extraction is performed on the lung region in order to compute features that can be used to classify the image. Finally, a support vector machine (SVM) classifier is trained on the images using the precomputed features.

Two different sets of features used for classifying the X-rays were compared in their research. The first feature set is inspired by object detection and includes six different shape and texture descriptors, each of which is a histogram containing 32 bins [37]. Therefore, there are a total of 192 features in the first feature set. The second feature set contains low-level features based on content-based image retrieval, containing a total of 594 features [37]. The authors found that the second feature set did not yield significantly better performance than the first (and much smaller)

feature set.

In a later study, three feature sets were used in this approach, including the previous two feature sets that we described above and a third feature set containing six MATLAB shape features [4] . The authors exhaustively searched the three feature sets to find the optimal combination of 18 features that yielded the best performance. An artificial neural network classifier was used in this set of experiments and showed a significant improvement over the SVM classifier.

In that same study, the authors used an artificial neural network classifier to perform classification of the X-rays without the need for lung segmentation [4]. This was done by measuring the changes in the rib profiles within the chest cavity [4]. This technique only requires approximate bounding boxes around the lung region. The feature set is composed of only one descriptor: the pyramid histogram of orientation gradients (PHOG) [9], using 15 bins. This technique was shown to perform nominally better than their previous system in terms of accuracy and the area under the ROC curve, and the authors state that it performs classification 25 times faster [4].

5.3 Experimental Settings

We used the 2TGP-line architecture, which we described in Section 5.2.1, for our tree structure and we use GMD-GP (see Chapter 3) as our GP implementation. Table 5.1 shows the GP settings that we used for our experiments. Since the X-rays can vary in size, we introduced a simple modification to the original 2TGP implementation. Instead of using a predefined width and height and restricting the x and y pixel coordinate nodes to be within that range, we simply changed the X,Y node to a real number in $[0, 1]$ instead of an integer. This way, the X,Y node is treated as a fractional value that refers to a point in the X or Y dimension. For example

Table 5.1: **GP Settings Used for the TB Experiments.**

Parameter	**Value**
Random initialization	Ramped half & half
Maximum generations	50
Total independent trials	30
Population size	1024
Tournament size	7
Crossover probability	0.81
Mutation probability	0.19
Mutation method	Subtree mutation
Max. tree depth	17
Max. tree size	None
GMD-GP genetic marker start	level 1 (root)
GMD-GP genetic marker depth	2

an X node with a value of 0.5 for an image of width 100 would refer to the pixel at index 50. Since the real numbers will not always be an exact pixel, we simply round the number up. While this may not be the optimal implementation, and more sophisticated techniques may perform better, this is a very general way of handling images of varying size without needing to predefine a range of allowable coordinate values.

The primitive set containing the functions and terminals that we used is shown in Table 5.2. The function set includes the 2TGP-line functions from previous work [2]. Each function takes as arguments the input image, the x and y pixel coordinates of the line to consider, the shape (horizontal or vertical), and the size of the line in pixels. The terminal set includes a node that is used to determine the x and y pixel coordinate for the different aggregation functions, an integer that determines the size of the line used in the aggregation functions, a shape node that determines whether the line is horizontal or vertical, a single terminal node to represent the image, and a random double.

Table 5.2: **Image Classification GP Primitive Set.**

Primitive	Description
AggMin	Returns the minimum pixel value in the given line.
AggMax	Returns the maximum pixel value in the given line.
AggMed	Returns the median of the pixel values in the given line.
AggMean	Returns the mean of the pixel values in the given line.
AggStdev	Returns the standard deviation of the pixels in the given line.
RandDoub	Random integer in $[0, 1]$.
Image	Java BufferedImage object representing the X-ray .
Size	Integer representing the size of the line to be used.
X,Y	Integer representing either the x or y pixel coordinate.
Shape	Integer representing the shape of the line, horizontal or vertical.

5.3.1 Tuberculosis Dataset

We used the publicly available Shenzen dataset[1] that was used in previous research [4, 15, 37]. This dataset includes a total of 662 chest X-ray images, consisting of 326 TB-negative examples and 336 TB-positive examples. The images vary in width (from 1130 to 3001 pixels) and height (from 948 to 3001 pixels). We chose to use the Shenzen dataset because of the relatively balanced class distribution compared to other available datasets.

Due to the large size of the images, coupled with the number of images in the training and testing sets, the amount of memory required for storing the images in memory is a technical challenge. However, reading each image every time an individual undergoes fitness evaluation is also very time consuming and inefficient. To overcome this issue, we utilized the high performance computing (HPC) resources at Michigan State University to store all the images in memory during evolution. Although this approach is not practical in the general sense, it serves as a proof of concept that GP can be used on more realistic datasets for raw image classification. Furthermore, although the classifiers are evolved using HPC resources, the classifiers

[1]National Library of Medicine, National Institutes of Health, Bethesda, MD, USA and Shenzhen No.3 People's Hospital, Guangdong Medical College, Shenzhen, China.

themselves can then be used stand-alone on typical computer hardware such as a personal laptop.

In order to determine the accuracy of the evolved classifiers on unseen data, we used a 10-fold cross-validation approach using the entire Shenzen dataset. For each fold, we performed 30 replicate trials. The reported accuracy values in Section 5.4 reflect the mean accuracy of the best-of-run individual from each trial across all 10 folds.

5.4 Results

We begin the discussion of our results with a comparison of the accuracy of our evolved classifiers versus those reported in the previous work that we discussed in Section 5.2.2. Table 5.3 shows the comparison of GP, GMD-GP, and the reported values from [37] and [4] from the Shenzen dataset. We report our mean accuracy on the testing data from the 10-fold cross-validation across all trials, compared to the accuracy values of the other techniques (without exhaustive feature selection).

The best performing technique overall was the artificial neural network approach (ANN) with an accuracy of 85.9%, while GMD-GP had a mean accuracy of 76.35%. However, GMD-GP achieved a maximum testing accuracy of 89.39%. While this is still lower than the 97.05% accuracy achieved by the artificial neural network approach in which the features were selected by exhaustively searching for the optimal combination of 18 features [4], this is important because it demonstrates that the raw image classification approach using GP without feature extraction (2TGP-line, as we described in Section 5.2.1) is capable of performing better than other techniques that require lung segmentation and feature extraction. Also, while the results for GMD-GP and standard GP were similar, this further demonstrates the effectiveness of the

Table 5.3: **Accuracy Comparison of GP and GMD-GP to Previous Work.** We compared against support vector machine, SVM, linear logistic regression, LLR, and artificial neural network, ANN, without lung segmentation from [4, 37].

SVM	**LLR**	**ANN**	**GP**	**GMD-GP**
82.1	84.1	85.9	75.4	76.4

Table 5.4: **Training Accuracy on the Shenzen Dataset.**

	Min	**Max**	**Med**	**Mean**	**Std. Dev.**
GP	0.6879	0.8420	0.7909	0.7901	0.0259
GMD-GP	0.6829	0.8403	0.7995	0.7954	0.0250

2TGP-line approach at evolving classifiers without prior feature extraction or feature selection.

Table 5.4 and Table 5.5 show the accuracy of the best-of-run individuals from each trial on the training and testing data, respectively, across all 10 folds. Table 5.4 shows that GMD-GP performed similar to GP in terms of training accuracy. However, Table 5.5 shows that, while marginal, GMD-GP achieved higher accuracy on the unseen testing data than GP. Most notable is the higher minimum and maximum accuracy that GMD-GP achieved compared to GP. Furthermore, although we used a straightforward classification technique that simply uses a threshold of 0 on the individual's output for determining the class to which an image belongs, more sophisticated techniques such as evolving the class boundaries [22], for example, may increase the accuracy of our approach. These are future directions for this technique.

Table 5.6 and Table 5.7 show the confusion matrix for the best classifier evolved with GP and GMD-GP, respectively, across the entire Shenzen dataset. One of the most important measurements for the TB classifiers is that of sensitivity, which is

Table 5.5: **Testing Accuracy on the Shenzen Dataset.**

	Min	**Max**	**Med**	**Mean**	**Std. Dev.**
GP	0.5606	0.8636	0.7576	0.7538	0.0535
GMD-GP	0.6364	0.8939	0.7670	0.7635	0.0486

Table 5.6: **GP Best Classifier Confusion Matrix.**

	Predicted Positive	**Predicted Negative**
Actual Positive	255	81
Actual Negative	28	298

Table 5.7: **GMD-GP Best Classifier Confusion Matrix.**

	Predicted Positive	**Predicted Negative**
Actual Positive	244	92
Actual Negative	25	301

the portion of TB-positive instances the classifier correctly identifies. The best GP classifier achieved a sensitivity of 0.759 while the best GMD-GP classifier achieved a sensitivity of 0.726. In previous work [4], the radiologists were instructed to error on the side of "over-reading" in order to minimize the chance of misdiagnosing a TB-positive patient. While the GP and GMD-GP classifiers were not highly sensitive, incorporating the confusion matrix into the fitness function could further improve the sensitivity of the evolved classifiers.

Next, we compared the time taken for our technique to classify a given X-ray with the reported results of the ANN technique, which was reported as taking less than 20 seconds [4]. In order to time our evolved classifiers, we performed two experiments. In the first experiment, we took the highest accuracy evolved classifier from all trials and executed it on the entire Shenzen dataset and recorded the time taken to classify each image. Similarly, the second experiment recorded the timing of all best-of-run classifiers in order to identify the general behavior of our evolved classifiers.

Table 5.8 and Table 5.9 show the detailed timing information for the best evolved classifier from all trials for GP and GMD-GP, respectively. This demonstrates that our technique is effective at evolving classifiers that are extremely fast at providing a classification from a raw input image. While the total time taken to load and classify an image is very little (less than half a second, on average), we can see

Table 5.8: **Time Taken (in seconds) for Best Evolved GP Classifier to Classify an X-ray Image.**

	Min	**Max**	**Med**	**Mean**	**Std. Dev.**
Load	0.0418	0.4958	0.3113	0.2954	0.0621
Classify	0.0003	0.0164	0.0005	0.0006	0.0008
Total	0.0422	0.5123	0.3119	0.2960	0.0623

Table 5.9: **Time Taken (in seconds) for Best Evolved GMD-GP Classifier to Classify an X-ray Image.**

	Min	**Max**	**Med**	**Mean**	**Std. Dev.**
Load	0.0418	0.5598	0.3113	0.2964	0.0624
Classify	0.0006	0.0302	0.0014	0.0015	0.0012
Total	0.0425	0.5655	0.3127	0.2979	0.0627

that the actual time the tree takes to classify an image once it has been loaded is even smaller than the time taken to load the image into memory. This can lead to an effective screening method in areas where large numbers of patients need to be screened. This does come at the cost of enormous computing resources required to train a classifier, as we discussed in Section 5.3. However, these solutions could be evolved in an HPC setting and then readily used on inexpensive hardware such as the laptop computer used in the system described in [4]. Further research into reducing the burden of requiring large memory for training is necessary.

Next, we examined the sizes of the best-of-run classifiers from each trial for both traditional GP and GMD-GP. Table 5.10 shows the different metrics we considered for the two techniques. Given the complex nature of the classification task (using raw X-ray images with no preprocessing or feature extraction/selection), we expected the size of the trees to become rather large. While both techniques produced relatively large trees, GMD-GP trees tended to be marginally smaller (with a mean size of 288.45 nodes), on average, than those of GP (with a mean size of 302.15 nodes). The most accurate classifier produced by GMD-GP (89.39% accuracy, as shown in Table 5.5) contained 253 nodes while the most accurate classifier produced by GP

Table 5.10: **Size of the Best-of-run Classifiers.**

	Min	**Max**	**Med**	**Mean**	**Std. Dev.**
GP	44	823	284.50	302.15	136.51
GMD-GP	49	1033	248.50	288.45	150.63

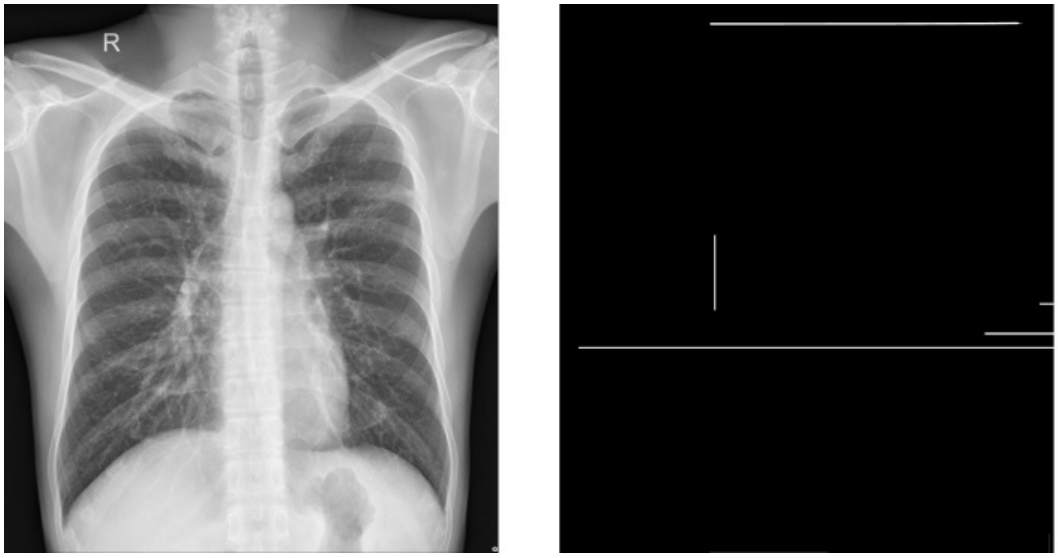
(86.36% accuracy, as shown in Table 5.5) contained 229 nodes.

Finally, we analyzed the behavior of the best evolved classifiers on sample X-ray images from the Shenzen dataset. We first examined all the regions in the images that the classifiers used. To do this, we simply create a new image wherein we set the color to white for every pixel used by the classifier, and then color every unused pixel black. We then pair this new image side-by-side with the original X-ray. We chose this technique rather than attempting to highlight the used pixels on the original X-ray since the classifiers only use single lines of pixels, which are very difficult to visualize on the grayscale X-ray images. Furthermore, we were only interested in providing a general view in order to gain some insight into the behavior of the classifiers, with respect to the regions of the image they use for determining the class of the image.

Figure 5.1 shows the regions of the X-ray used by the best classifier evolved by GP. This shows that the classifier, which achieved an accuracy of 86.36%, only uses a small fraction of the pixels in the image in order to predict tuberculosis. Furthermore, while part of the lung is used in the classifier, not all of the regions that the classifier uses are in the lung area. However, since the images vary in dimensions as well as the actual anatomy of the patients, the exact position used by the classifier will also vary from patient to patient to a certain extent.

We also analyzed the number of nodes in the tree that reference pixel locations, as well as the number of times a single pixel was used by the classifier. Out of 229 nodes, only 42 nodes (i.e. 18.34%) are pixel coordinate nodes. The maximum number of

Figure 5.1: **Regions of the X-ray Used by the Best Evolved GP Classifier.** The original X-ray image is shown on the left, and the image showing the pixels of the X-ray that were used by the classifier is shown on the right. In the image on the right, white pixels are those that were used by the classifier, while black pixels indicate that they were not used by the classifier. Note that the lines have been thickened to aid visualization.

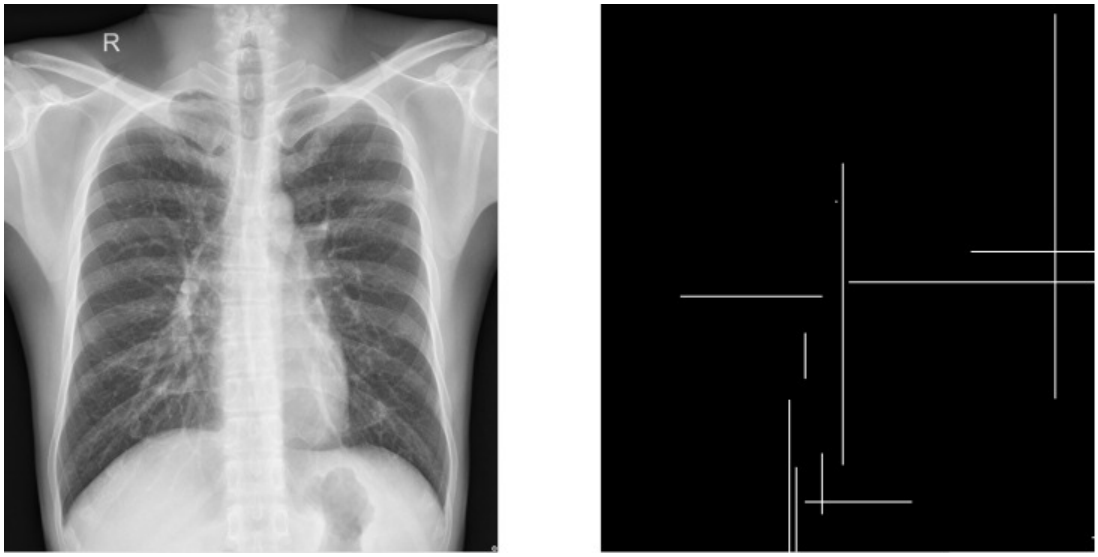


times a particular pixel was referenced by this tree was 8, while the minimum was 1. This shows that some of the lines overlap, which means that the same pixels can be used in different parts of the tree for different calculations.

Figure 5.2 shows the regions of the X-ray used by the best classifier evolved by GMD-GP. This clearly shows that the best classifier evolved by GMD-GP is much more complex than that of GP. This is evident in that there are many more lines used by this classifier, compared to the GP classifier in Figure 5.1. Also, more pixels of the lung region are used by the GMD-GP classifier, although not all of the lines belong to the lung region in this particular X-ray. Further analysis into the nodes in the GMD-GP classifier tree revealed that out of 253 nodes, 68 (i.e. 26.88%) of them are pixel coordinate nodes. In this case, the maximum number of times a particular pixel was referenced by this tree was 8, with a minimum of 1.

Figure 5.1 and Figure 5.2 show an interesting result. We hypothesized that the

Figure 5.2: **Regions of the X-ray Used by the Best Evolved GMD-GP Classifier.** The original X-ray image is shown on the left, and the image showing the pixels of the X-ray that were used by the classifier is shown on the right. In the image on the right, white pixels are those that were used by the classifier, while black pixels indicate that they were not used by the classifier. Note that the lines have been thickened to aid visualization.



best classifiers would typically contain lines mostly within the lung region, since it is the certain abnormalities in the lung that allow the radiologists to diagnose a patient as having tuberculosis. However, this is not the case in these classifiers, since the pixels being used by the classifiers do not always belong to the lung region. Further analysis into the complex behavior of these classifiers is needed to provide better insight into how the classifications are derived.

Chapter 6

Hybridizing GMD-GP with Behavioral Diversity

As we discussed in Chapter 2, several techniques have been introduced to sustain the diversity of GP populations in order to prevent premature convergence and improve the search performance of GP. However, many such techniques focus explicitly on maintaining either structural or behavioral diversity. While structural and behavioral diversity are indeed related and affect each other, we argue that simultaneously focusing directly on both aspects of diversity can be more beneficial than focusing on either in isolation.

There are many potential benefits of simultaneously enforcing structural and behavioral diversity. As we discussed in Chapter 1, the relationship between structural and behavioral diversity is nontrivial because of the complex genotype-to-phenotype mapping in tree-based GP. Thus, techniques that focus solely on genotypic (structural) diversity may be ineffective at promoting and preserving behavioral diversity since many genotypes express the same or similar behavior. Likewise, methods that focus solely on behavioral diversity may still benefit from structural diversity since

it is ultimately the genetic makeup that encodes the behavior expressed by the tree. This is especially true for behavioral diversity methods that use the traditional genetic operators that have been shown to lead to genotypic convergence [24, 53, 64]. We therefore argue, and demonstrate in Section 6.2, that hybrid structural and behavioral diversity techniques may overcome some of these shortcomings that are experienced by either technique in isolation.

We found that while GMD-GP often outperformed other state-of-the art methods in many problems, GMD-GP populations tended to have significantly less behavioral diversity (unique behaviors) while having significantly higher structural diversity [14]. Therefore, incorporating behavioral diversity into GMD-GP may further improve its search performance by more effectively exploring the behavioral space.

6.1 Hybridizing GMD-GP with Behavioral Diversity Selection

In the canonical GP setup, selection pressure is imposed on the population by using fitness to select individuals to produce offspring. However, since GMD-GP adopts a multi-objective scheme that places equal weight on fitness and genetic marker density, no selection pressure is used to create offspring. Instead, GMD-GP randomly selects parents and then uses Pareto tournament selection after the breeding phase to impose selection pressure on the population, as we described in Section 3.3. Therefore, incorporating behavioral diversity instead of random mating during the parent selection phase in GMD-GP is a logical starting point for hybridizing GMD-GP with behavioral diversity techniques.

6.1.1 Lexicase Selection in GMD-GP

We investigated the feasibility of using Lexicase Selection instead of random mating in GMD-GP. Since Lexicase Selection is known to perform poorly in the symbolic regression problem domain [45] (several of the problems in our benchmark suite are symbolic regression problems), we introduced two simple extensions to Lexicase Selection that address this issue. These extensions place bias on the current most difficult fitness cases rather than randomly ordering the fitness cases each time a parent is to be selected. This is motivated by the fact that certain fitness cases tend to be solved by fewer individuals while other fitness cases are solved by a large proportion of the population, and this is dynamic over time. Placing more preference on the current most difficult fitness cases may lead to problem decomposition and thus faster solution discovery.

Before discussing our extensions, we first describe the original Lexicase Selection method in more detail than in Chapter 2. When a parent is to be selected, Lexicase Selection first sets the pool of candidate individuals as the entire population. Next, a random ordering is chosen for the set of fitness cases of the given problem. The first fitness case in the given ordering is then removed from the set, and only the individuals that currently have the exact best error on that fitness case are retained as candidates. This step is repeated until either a single candidate exists or all the fitness cases have been used. If all the fitness cases have been used and more than one candidate exists, one of the remaining candidates is chosen randomly.

Our first extension to Lexicase Selection, which we refer to as Lexicase Selection with Count Bias (Lex-C), is shown in Algorithm 1. Lex-C sorts the fitness cases in ascending order based on how many individuals solve each fitness case. This guarantees that the least-solved fitness case is always used first to filter out parent candidates. While it is possible that the least-solved fitness case is not solved by any

Algorithm 1 Lex-C: Count Bias Sorting of Fitness Cases.

Require: cases {Set of fitness cases for the given problem.}
1: $counts \Leftarrow \emptyset$ {Map of total individuals that solve each fitness case.}
2: **for** $case \in cases$ **do**
3: $counts[case] \Leftarrow$ total individuals that solve $case$
4: **end for**
5: **return** $sortAscending(counts)$ {Return the sorted fitness case indices in ascending order.}

individual in the population, the best-performing individuals with respect to that fitness case will still be selected. Also, since this ordering does not change for a given generation, we use a random subset of the population as parent candidates (instead of the entire population) to avoid selecting the same individual.

Our second extension, which we refer to as Lexicase Selection with error bias (Lex-E), is shown in Algorithm 2. Lex-E is similar to Lex-C. However, Lex-E sorts the fitness cases in descending order based on the average error the population commits on each fitness case. This also places bias towards performing well on the current most difficult fitness cases, as they will have the highest average error. Since this ordering does not change in a given generation, the initial parent candidates are also a random subset of population.

Lex-E slightly differs from Lex-C in that for problems for which error is real-valued, such as symbolic regression, the fitness case with the highest average error may not always be the least solved since whether or not an individual solves a fitness case is often based on some threshold. However Lex-E will perform similar to Lex-C on discrete problems for which there is no error information, such as in the Boolean domain. In this case, if error is 0 for pass and 1 for fail, the average error is equal to the proportion of individuals in the population that fail the fitness case, which yields the same ordering as Lex-C.

Algorithm 2 Lex-E: Error Bias Sorting of Fitness Cases.

Require: cases {Set of fitness cases for the given problem.}
1: $errors \Leftarrow \emptyset$ {Map of average error for each fitness case.}
2: **for** $case \in cases$ **do**
3: $\quad errors[case] \Leftarrow$ average population error on $case$
4: **end for**
5: **return** $sortDescending(errors)$ {Return the sorted fitness case indices in descending order.}

6.1.2 Other Selection Techniques in GMD-GP

In addition to the original Lexicase Selection algorithm and the different extensions that we introduced in Section 6.1.1, we also experimented with several other behavioral diversity selection techniques that we described in Chapter 2. Particularly, we hybridized GMD-GP with Epsilon Lexicase Selection, Complementary Phenotype Selection, and Semantics in Selection.

6.2 Experimental Validation of Hybrid Selection Techniques

In this section, we compare the performance of each of the hybrid selection techniques that we discussed earlier in the chapter with that of their stand-alone counterparts. We demonstrate the performance advantage gained by the hybridization, and we also analyze the effect that the hybridization has on structural and behavioral diversity. We used the same experimental settings as in Chapter 4 for all the experiments in this section.

6.2.1 Performance of Hybrid Selection Techniques

First, we conducted an initial set of experiments to determine the feasibility of hybridizing GMD-GP with the different versions of Lexicase Selection that we discussed in Section 6.1.1. We first compared the performance of the hybrid techniques to GMD-GP and the different variants of Lexicase Selection in isolation. For convenience, Table 6.1 lists the abbreviations for the techniques that we compared in these experiments. In all the following figures and tables, we abbreviate GMD-GP as GMD for space considerations. All experimental settings were the same as in Table 4.3. Since the original Lexicase Selection algorithm is known to often perform poorly for symbolic regression but well in the finite algebras problems [28], we included both the symbolic regression problems as well as the finite algebras problems that we discussed in Section 4.1, which yielded a total of 25 problems for the experiments using original Lexicase Selection.

Given the number of algorithms under comparison, and the number of benchmark problems we used, we discuss the results as follows. We divide our experiments into two sets. First, we present the results of the experiments comparing GMD-GP, the original Lexicase Selection, and the two extensions that we described above, along with their hybrid counterparts. Next, we compare the hybrid versions of GMD-GP using Epsilon Lexicase Selection, Complementary Phenotype Selection, and Semantics in Selection. We chose to include Epsilon Lexicase Selection in the second set of experiments since it was designed specifically for symbolic regression, which is what we consider in the second set of experiments.

Figure 6.1 shows the pairwise comparison of the convergence rates for both the symbolic regression and finite algebras problem domains. In the case of symbolic regression, Lex tended to perform poorly in comparison to the other approaches, which can be expected [45]. However, the plots for the hybrids (especially GMD+Lex-C as

Table 6.1: **List of GP Techniques Used in the Comparison of the Hybrid Selection Techniques.**

Technique	**Abbreviation**
Lexicase Selection	Lex
Epsilon Lexicase Selection	eLex
Lexicase Selection with Error Bias	Lex-E
Lexicase Selection with Count Bias	Lex-C
Complementary Phenotype Selection	CPS
Semantics in Selection	SiS
GMD-GP and Lexicase Selection	GMD+Lex
GMD-GP and Lexicase Selection with Error Bias	GMD+Lex-E
GMD-GP and Lexicase Selection with Count Bias	GMD+Lex-C
GMD-GP and Epsilon Lexicase Selection	GMD+eLex
GMD-GP and Complementary Phenotype Selection	GMD+CPS
GMD-GP and Semantics in Seletion	GMD+SiS

well as GMD+Lex-E) show that hybridization with GMD-GP significantly improved the convergence rate over Lex in isolation. In many cases for the symbolic regression problems, the hybrids did not improve the convergence rate of GMD-GP. This can also be expected since Lex did not perform well in isolation for these problems. However, the hybrid GMD+Lex-C was significantly faster than GMD-GP in a few of the symbolic regression problems while it was not significantly slower in any of the symbolic regression problems.

The results of the convergence rate comparison for the finite algebras problems are nearly the opposite of that of the symbolic regression problems. The plot for Lex reveals that it was significantly faster than all of the other approaches (including GMD-GP) in all of the finite algebras problems, with the exception of the hybrid GMD+Lex against which Lex was still faster in over 60% of the problems. However, the plot for GMD+Lex reveals that hybridizing GMD-GP with Lex significantly improved the convergence rate of GMD-GP in all of the finite algebras problems.

Figure 6.2 further demonstrates the performance benefits of the hybrid techniques in terms of the success rate. The plots for each of the hybrids show that each

Figure 6.1: **Pairwise Comparison of the Convergence Rate of the Hybrids.** Each plot shows the percentage of problems in which the method above the plot found a solution in significantly fewer fitness evaluations, on average, than each other method.

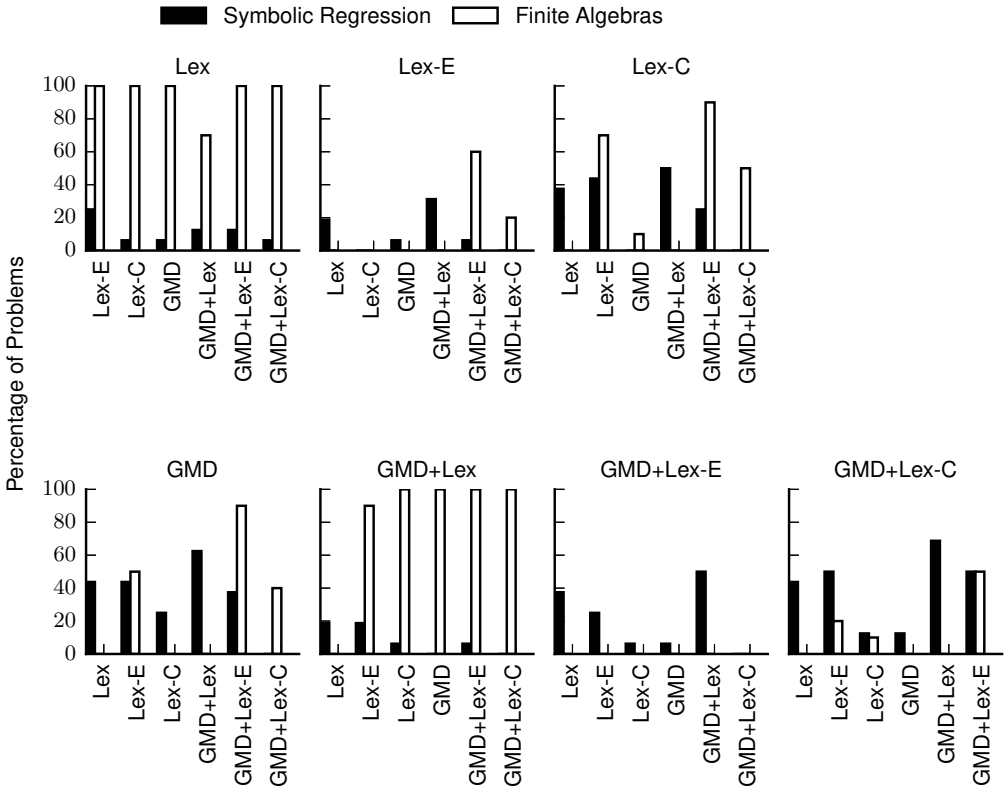
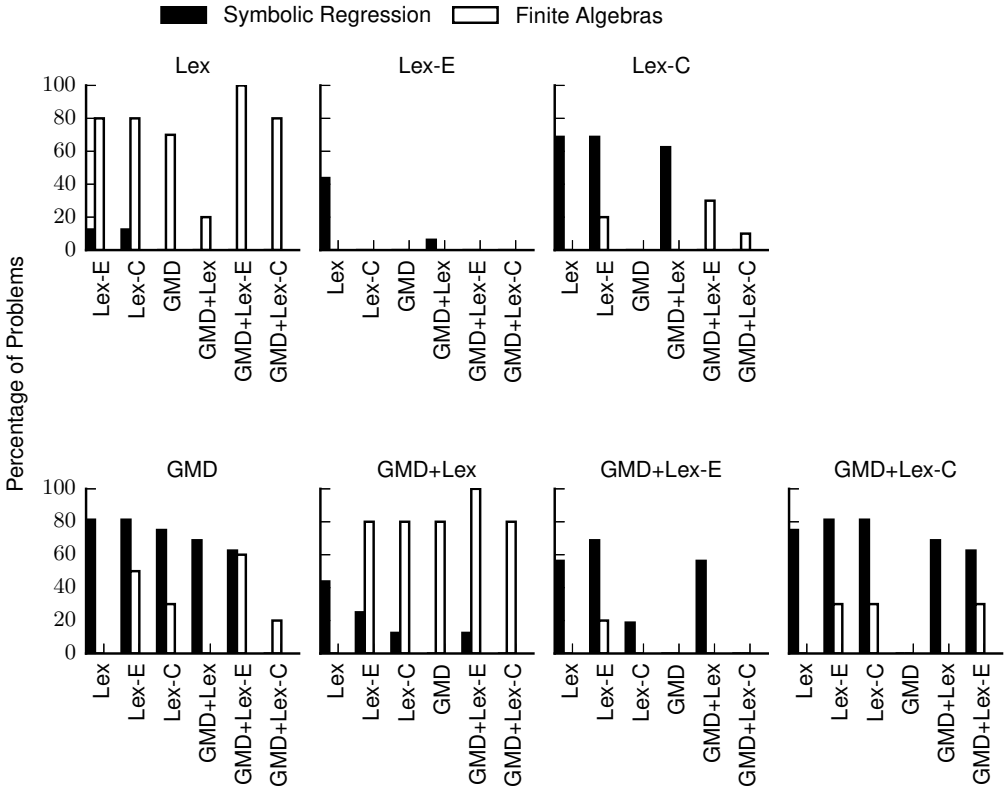


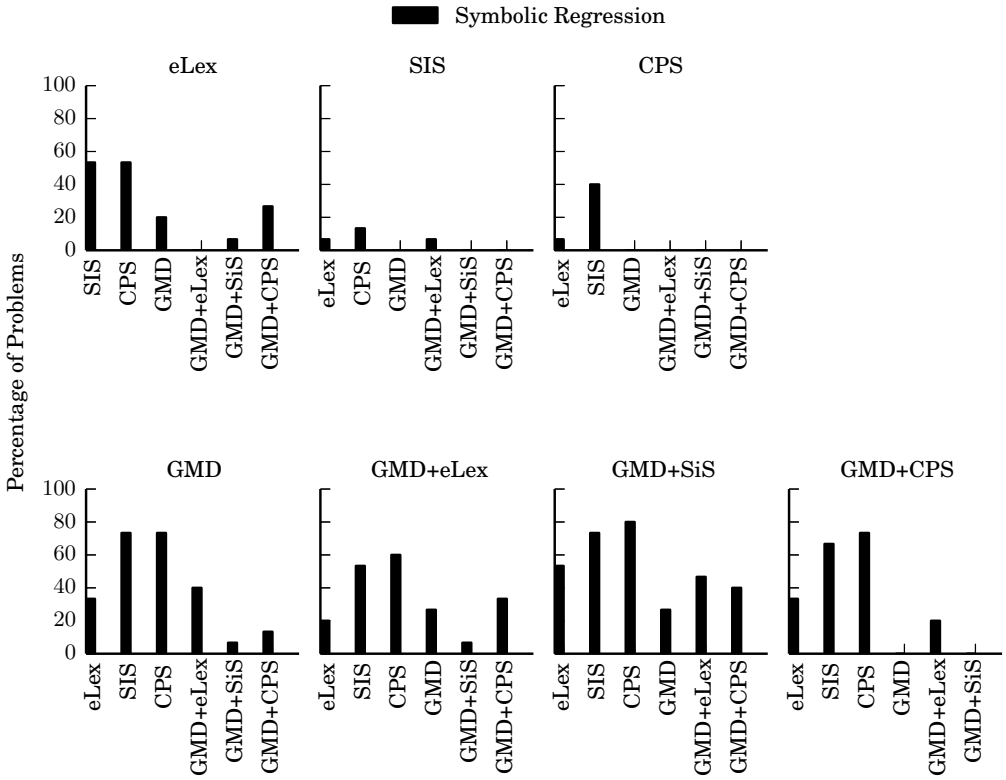
Figure 6.2: **Pairwise Comparison the Success Rate of the Hybrids.** Each plot shows the percentage of problems in which the method above the plot had a significantly higher success rate than each other method.



hybridization significantly improved upon the success rate of Lex in most of the symbolic regression problems. Since the plot for GMD-GP reveals that it already performed very competitively in the symbolic regression problems, the GMD+Lex and GMD+Lex-E hybrids had significantly lower success rates than GMD-GP in 68% and 62% of the symbolic regression problems, respectively. However, the GMD-GP plot also reveals that the hybrid GMD+Lex-C did not negatively impact the success rate of GMD-GP, while, as Figure 6.1 shows, GMD+Lex-C did improve the convergence rate of GMD-GP in some cases.

The success rate comparison for the finite algebras problems reveals a similar result. Figure 6.2 shows that Lex outperformed GMD-GP in terms of success rate in

Figure 6.3: **Pairwise Comparison of the Ending Fitness of the Hybrids.** Percentage of problems in which the method shown above each plot had significantly higher mean ending fitness than each method listed.



the finite algebras problems. However, the hybrid GMD+Lex significantly improved the success rate of GMD-GP in around 80% of the problems, and it did not negatively impact the success rate of Lex in any problem. These results demonstrate that using behavioral diversity selection (which is gained through Lex) in GMD-GP can leverage the strengths of both techniques and counteract their weaknesses.

Figure 6.3 shows the pairwise comparison of the other hybrid selection techniques. Overall, GMD-GP outperformed the other stand-alone techniques in the largest percentage of the problems, having significantly higher ending fitness than eLex, CPS, and SiS in 33.33%, 73.33%, 73.33% of the problems, respectively. The plots for the other stand-alone techniques indicate that CPS and SiS performed similar to each

Table 6.2: **Ending Fitness of the Behavioral Diversity Selection Techniques Compared to the Hybrids.** Percentage of problems in which the stand-alone technique had significantly higher ending fitness than its hybrid counterpart and vice versa.

	Stand-alone vs Hybrid	**Hybrid vs Stand-alone**
eLex	0.0	20.0
CPS	0.0	73.33
SiS	6.67	66.67

other in a majority of the problems, while eLex often outperformed CPS and SiS.

Comparing each technique to its hybrid counterpart, Table 6.2 demonstrates the effectiveness of the hybridization compared to the stand-alone behavioral diversity techniques. For every other behavioral diversity selection technique, the hybrid significantly outperformed the stand-alone technique in a majority of the problems. In fact, with the exception of SiS (in only 6.67% of the problems), none of the behavioral diversity selection techniques significantly outperformed its hybrid counterpart in any of the problems.

Table 6.3 shows the comparison of GMD-GP to each of the hybrids. Since GMD-GP was typically the best-performing stand-alone algorithm, the hybridization with the other techniques did not yield superior performance to GMD-GP in most of the cases. However, it is important to note that GMD-GP did not have significantly higher ending fitness than any of its hybrid counterparts in a majority of the problems. The hybrid using SiS was the best performing compared to GMD-GP, with significantly higher ending fitness than GMD-GP in 26.67% of the problems while only worse than GMD-GP in 6.67% of the problems. The worst performing hybrid, GMD-GP with Complementary Phenotype Selection was worse than GMD-GP in only 13.33% of the problems. This shows that the hybridization not only tended to improve the performance of the behavioral diversity techniques, as we discussed

Table 6.3: **Ending Fitness of GMD-GP Compared to the Hybrids.** Percentage of problems in which the hybrid had significantly higher ending fitness than GMD-GP vs. significantly lower fitness than GMD-GP.

	Better than GMD-GP	**Worse than GMD-GP**
GMD+eLex	26.67	40.00
GMD+CPS	0.00	13.33
GMD+SiS	26.67	6.67

above for Table 6.2, but it also improved the performance of the structural diversity technique in some cases.

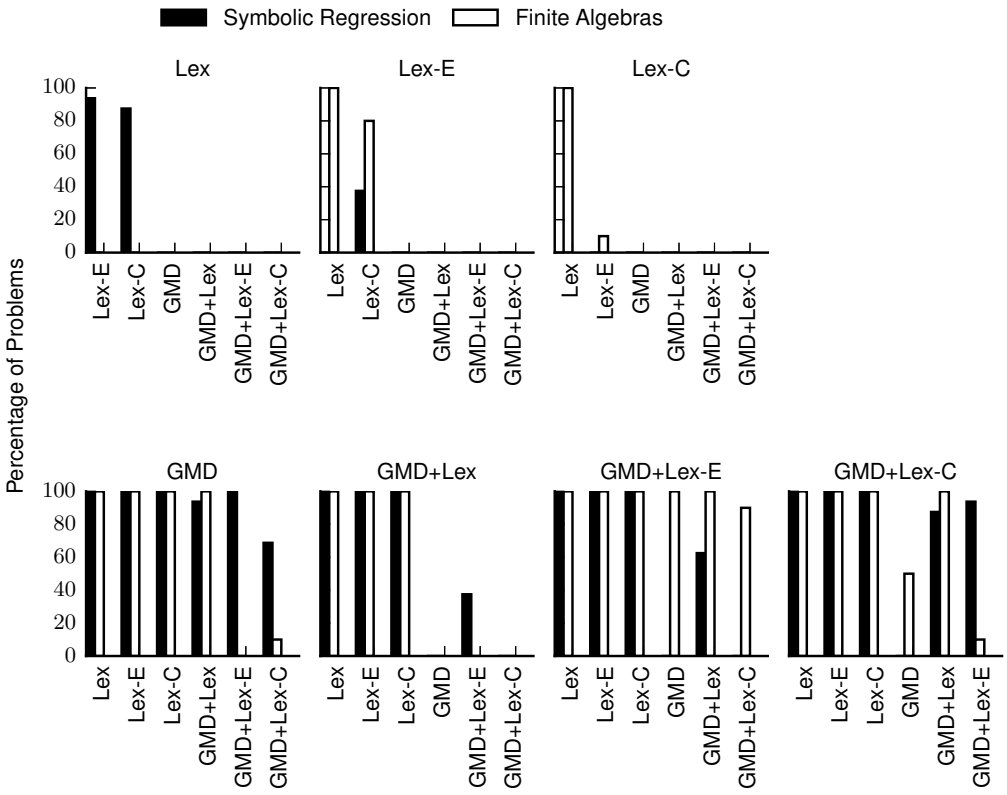
6.2.2 Effect of Hybridization on Diversity

In addition to comparing the performance of the hybrids, we also analyzed the effect that hybridization had on different aspects of diversity. Using the same experimental settings that we described in Section 4.4, we compared the different measures of diversity in the final populations.

Figure 6.4 shows the pairwise comparison of the mean maximum genetic marker density of the final populations, for genetic markers composed of the top two tree levels, in both problem domains. Each of the hybrids had significantly lower ending genetic marker density than Lex in all of the problems in both domains. This shows that the hybridization with GMD-GP significantly and consistently increases the structural diversity over Lex in isolation. Since GMD-GP had significantly lower genetic marker density than Lex, the hybridization with Lex generally increased the genetic marker density of GMD-GP. However, interestingly, GMD+Lex-E and GMD+Lex-C significantly decreased the genetic marker density (and therefore increased structural diversity) of GMD-GP in many of the finite algebras problems.

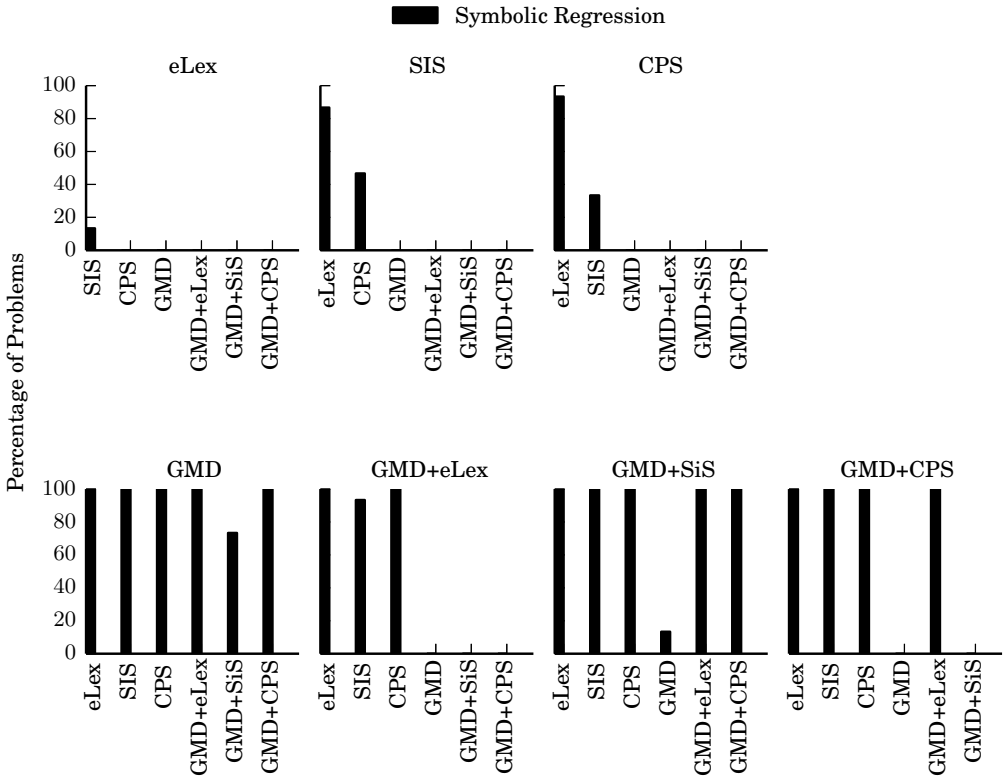
Figure 6.5 shows the pairwise comparison of the mean structural diversity in the

Figure 6.4: **Pairwise Comparison of the Genetic Marker Density of the Hybrids.** Each plot shows the percentage of problems in which the method above the plot had significantly lower genetic marker density (of genetic markers composed of the top 2 tree levels) than each other method after 250,000 fitness evaluations.



top two tree levels for the other hybrid selection techniques. The plot for GMD-GP shows that GMD-GP again had significantly higher structural diversity than the other approaches in nearly all of the problems, which can be expected since structural diversity is the main focus of GMD-GP. Figure 6.5 also shows that each hybrid technique had significantly higher structural diversity than its stand-alone behavioral diversity counterpart in all of the problems. While Figure 6.5 shows the comparison of the structural diversity across only the top two levels, the results were nearly identical across the top six levels. This suggests that the hybridization with GMD-GP indeed significantly increases the structural diversity of the other techniques.

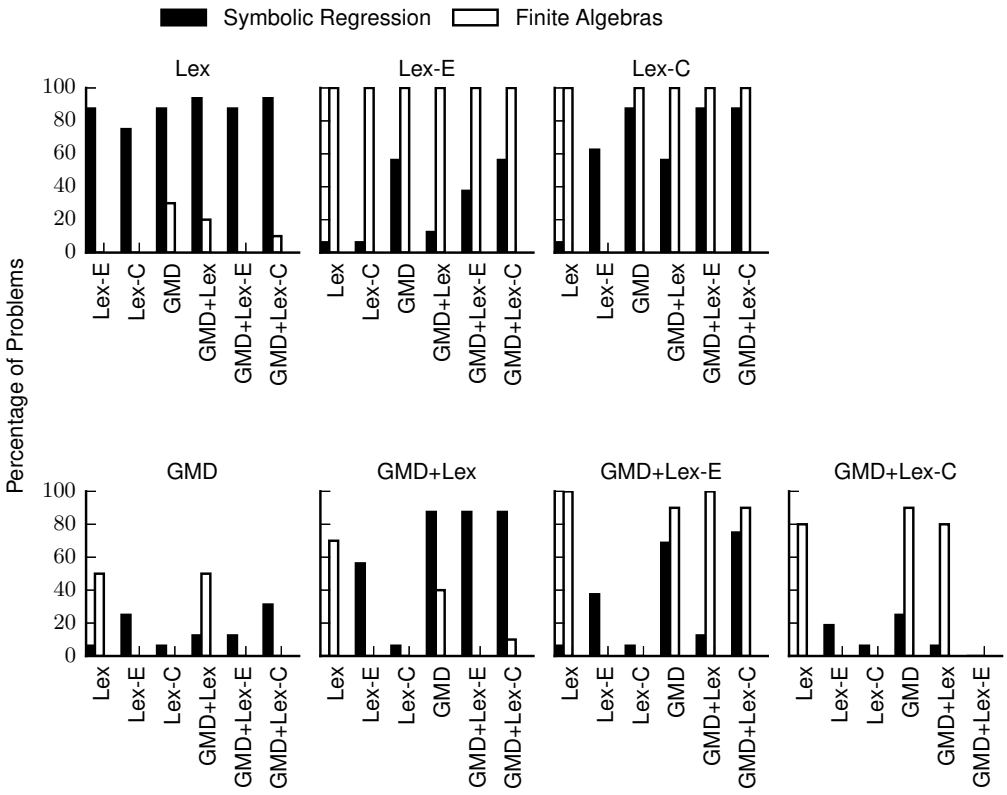
Figure 6.5: **Pairwise Comparison of the Structural Diversity of the Other Hybrids.** Percentage of problems in which the method shown above each plot had significantly higher structural diversity than each method listed.



The pairwise comparison of behavioral diversity in Figure 6.6 shows the reverse of that of structural diversity. We consider two individuals to have different behavior if their output vectors differ by at least one element. Behavioral diversity is then defined as the proportion of unique behaviors in the population. We discussed in Section 4.4 that GMD-GP populations often had significantly less behavioral diversity than the other approaches while having significantly higher structural diversity. The plots for each of the hybrids in Figure 6.6 show that in many cases, the hybrids significantly increased the behavioral diversity of GMD-GP populations. Since Lex maintains a high level of behavioral diversity, the hybridization typically decreased the behavioral diversity of Lex.[1]

[1]In the finite algebras problems, Lex tended to have lower behavioral diversity than GMD-GP

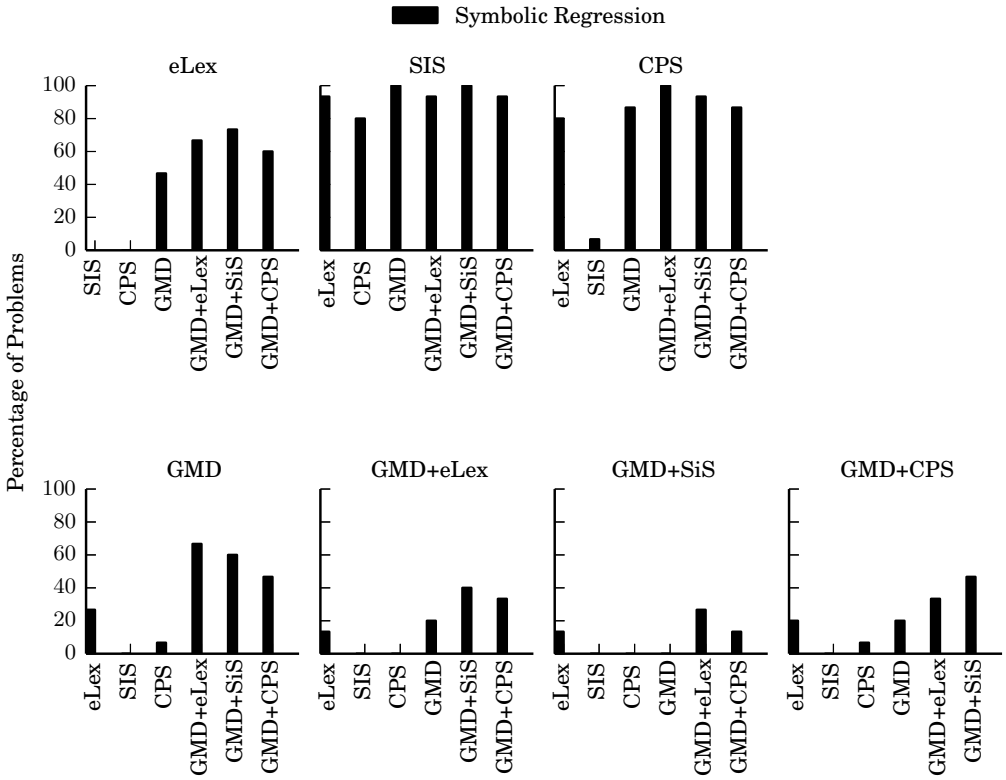
Figure 6.6: **Pairwise Comparison of the Behavioral Diversity of the Hybrids.** Each plot shows the percentage of problems in which the method above the plot had significantly higher behavioral diversity than each other method after 250,000 fitness evaluations.



Next, we compared the behavioral diversity of the other behavioral diversity selection approaches to that of their hybrid counterparts. Figure 6.7 shows the pairwise comparison of ending behavioral diversity across all the problems. In a majority of the problems, GMD-GP had significantly lower behavioral diversity than the behavioral diversity techniques (with the exception of eLex, which still had significantly higher behavioral diversity than GMD-GP in 40% of the problems). This can be expected, since the other techniques focus more directly on behavioral diversity. Interestingly, the plots for the other hybrids show that integrating the behavioral

and the hybrids since it typically converged to the solution quickly although the trial lasted for a fixed 250,000 fitness evaluations.

Figure 6.7: **Pairwise Comparison of the Behavioral Diversity of the Other Hybrids.** Percentage of problems in which the method shown above each plot had significantly higher behavioral diversity than each method listed.

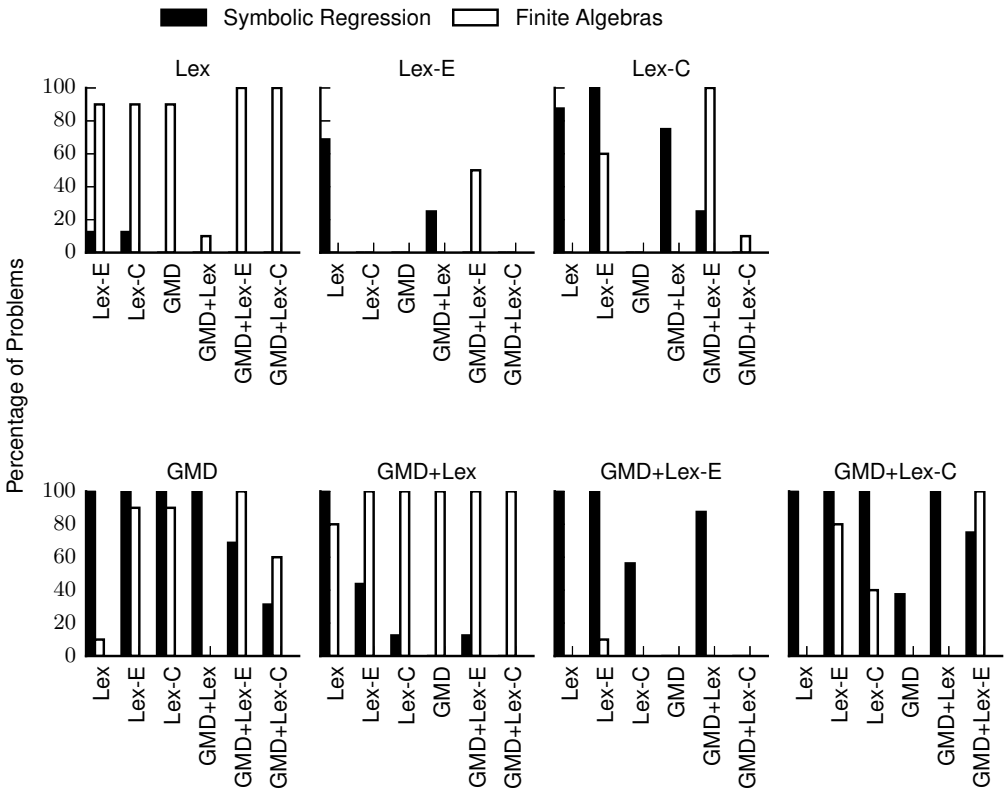


diversity techniques into GMD-GP does not always significantly increase the behavioral diversity of GMD-GP.

We further analyzed the behavioral diversity on a lower level by examining the standard deviation of fitness at the end of the trials. This provides more information than the proportion of unique behaviors alone because fitness standard deviation gives an indication of the difference in individuals' behaviors rather than solely whether or not they are different. Also, since many different behaviors can receive the same fitness value, the standard deviation in fitness provides a more in-depth analysis of the behavioral dynamics of the population.

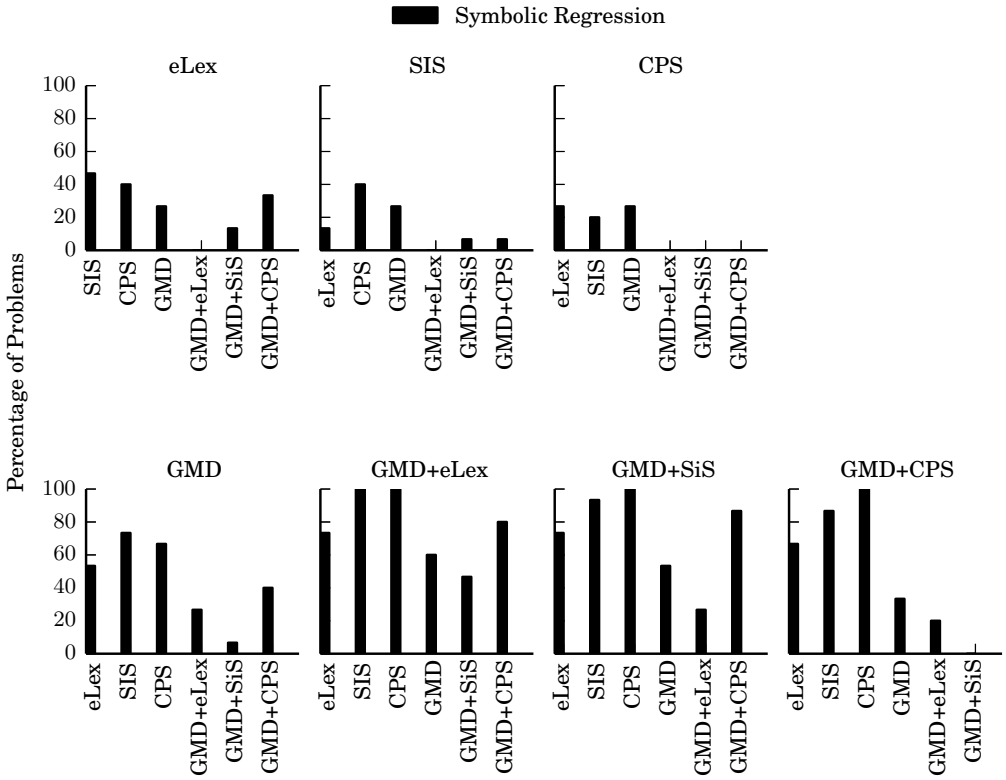
Figure 6.8 shows the pairwise comparison of the fitness standard deviation of

Figure 6.8: **Pairwise Comparison of the Fitness Standard Deviation of the Hybrids.** Each plot shows the percentage of problems in which the method above the plot had significantly higher fitness standard deviation than each other method after 250,000 fitness evaluations.



the hybrid approaches compared to GMD-GP and Lex in isolation. Since GMD-GP populations tended to have significantly higher fitness standard deviation than other approaches, as we discussed in Section 4.4, the hybrids had significantly higher fitness standard deviation than Lex in all of the symbolic regression problems and most of the finite algebras problems. Furthermore, GMD+Lex had significantly higher fitness standard deviation than GMD-GP in all of the finite algebras problems, and GMD+Lex-C had significantly higher fitness standard deviation than GMD-GP in around 40% of the symbolic regression problems. This suggests that the hybridization can improve the behavioral diversity of GMD-GP, while also yielding more widespread exploration of the fitness landscape. This is important because

Figure 6.9: **Pairwise Comparison of the Fitness Standard Deviation of the Other Hybrids.** Percentage of problems in which the method shown above each plot had significantly higher standard deviation in fitness than each method listed.



many behaviors can still map to the same fitness value, and therefore high behavioral diversity does not necessarily guarantee effective search in the fitness space.

Figure 6.9 shows the pairwise comparison of the fitness standard deviation of the populations of the other behavioral diversity selection techniques across the benchmark problems. This shows that while GMD-GP typically had a significantly lower number of unique behaviors than the behavioral diversity techniques (which is shown in the GMD plot of Figure 6.7), GMD-GP populations still tended to have significantly higher standard deviation in fitness in a majority of the problems. While the behavioral diversity techniques did not always provide GMD-GP with significantly higher behavioral diversity, Figure 6.9 shows that the hybridization with eLex, SiS,

and CPS increased the fitness standard deviation of GMD-GP in around 60%, 53%, and 33% of the problems, respectively. Furthermore, Figure 6.9 also shows that the hybrids had significantly higher fitness standard deviation than each of the stand-alone behavioral diversity techniques in a majority of the problems. This suggests that the hybrid techniques improve upon the stand-alone techniques by causing the individuals to be more spread out in the fitness landscape, yielding broader search and perhaps more efficient exploration of the landscape.

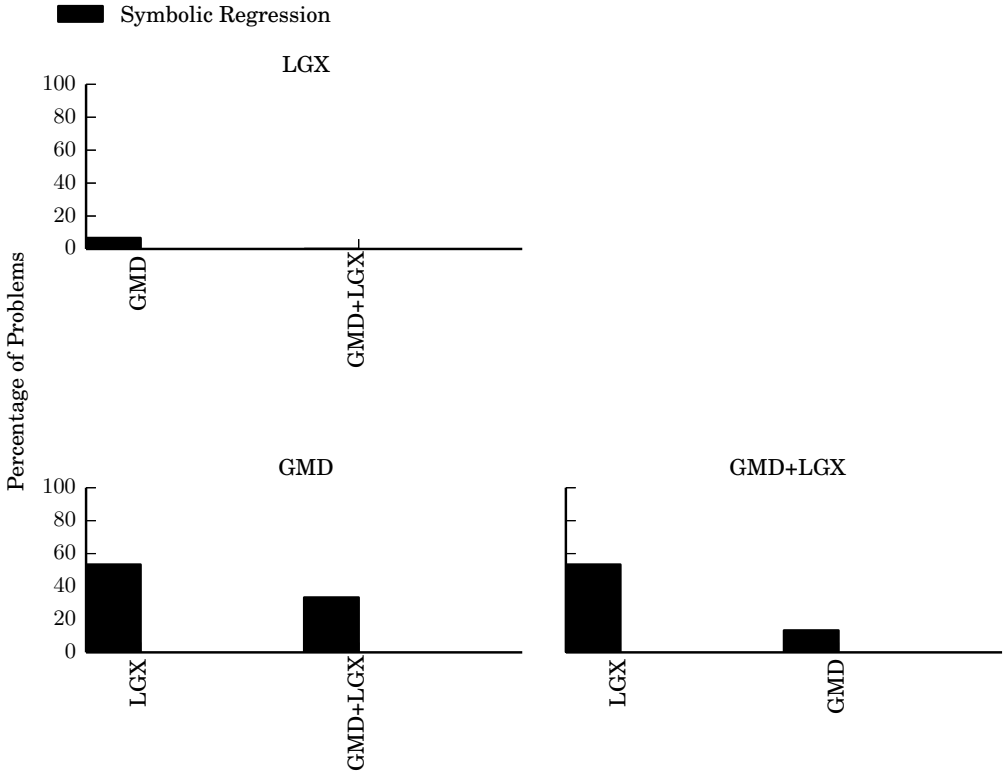
6.3 Hybridizing GMD-GP with Behavior-Aware Crossover

Since the original implementation of GMD-GP uses the traditional genetic operators such as subtree crossover, another natural place to incorporate behavioral diversity into GMD-GP is during crossover. It is known that subtree crossover is highly disruptive [49]. This is because subtree crossover is irrespective of individuals' behavior, using no behavioral information to aid in its selection of crossover points. By incorporating behavior-aware crossover into GMD-GP, we aim to improve its performance by increasing behavioral diversity. To do this, we use the LGX operator that we described in Section 2.2.3 because it was shown to perform well across different problem domains [61].

6.3.1 Performance of GMD-GP and LGX Hybrid

We now discuss the performance effect of hybridizing GMD-GP with the LGX crossover operator. Again, we used the same experimental settings and the symbolic regression benchmark problem suite that we discussed in Section 4.2. Figure 6.10 shows the pairwise comparison of GMD-GP and LGX to their hybrid counterparts in

Figure 6.10: **Pairwise Comparison of the Ending Fitness of the GMD-GP, LGX Hybrid.** Percentage of problems in which the method shown above each plot had significantly higher mean ending fitness than each method listed.

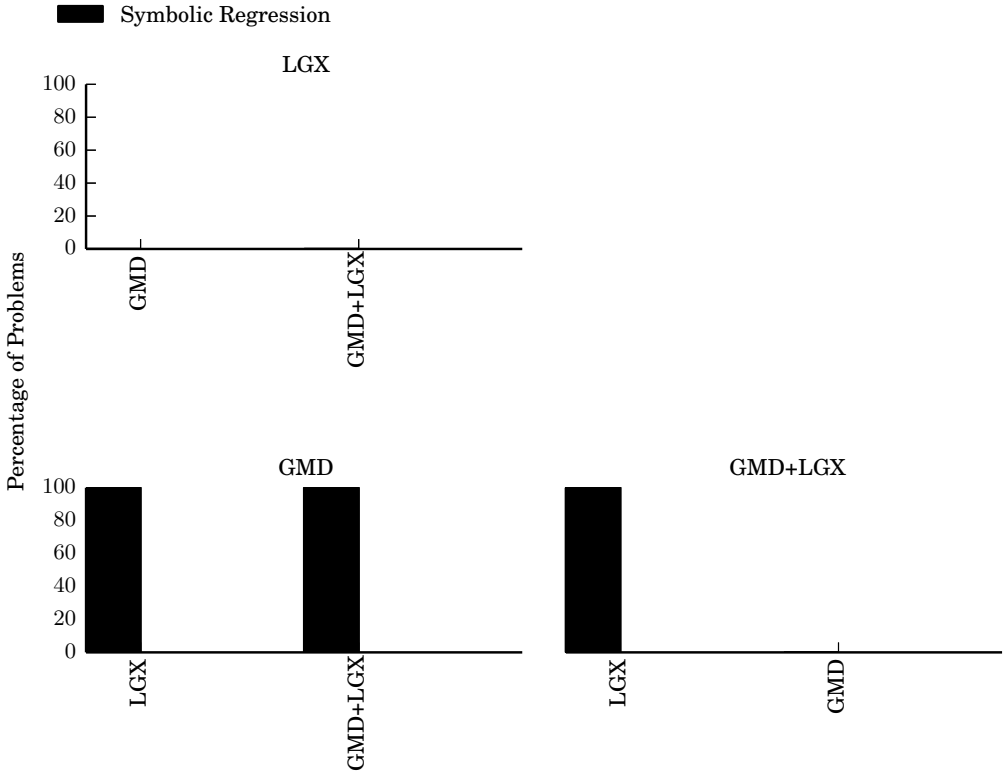


terms of the mean fitness at the end of the trial. This shows a similar result to what we observed for the other hybrid selection techniques. GMD-GP outperformed LGX in 53.33% of the problems, while LGX only outperformed GMD-GP in 6.67% of the problems. Since GMD-GP tended to be the best performing stand-alone technique, the hybridization with LGX did not tend to improve upon GMD-GP in most cases, while it significantly improved upon LGX in 53.33% of the problems.

6.3.2 Effect of LGX Hybridization on Diversity

Next, we analyzed the effect that the hybridization with LGX has on the same measures of diversity that we considered in Section 6.2.2. Figure 6.11 shows the

Figure 6.11: **Pairwise Comparison of the Structural Diversity of the GMD-GP, LGX Hybrid.** Percentage of problems in which the method shown above each plot had significantly higher structural diversity than each method listed.



pairwise comparison of mean structural diversity in the top two tree levels at the end of the trial. Again, since GMD-GP focuses directly on maintaining a high level of structural diversity, it is no surprise that GMD-GP has significantly higher structural diversity than LGX (which we can see in the plot for GMD in Figure 6.11). In fact, GMD-GP has significantly higher structural diversity than LGX in all (100%) of the problems. This is also because LGX only performs crossover in homologous regions of the tree, from the root, downwards, which further encourages the population to become more structurally homologous. Therefore, as we can see in the plot for the hybrid GMD+LGX in Figure 6.11, the hybridization with GMD-GP caused the structural diversity of LGX to significantly increase in all of the problems.

Figure 6.12: **Pairwise Comparison of the Behavioral Diversity of the GMD-GP, LGX Hybrid.** Percentage of problems in which the method shown above each plot had significantly higher behavioral diversity than each method listed.

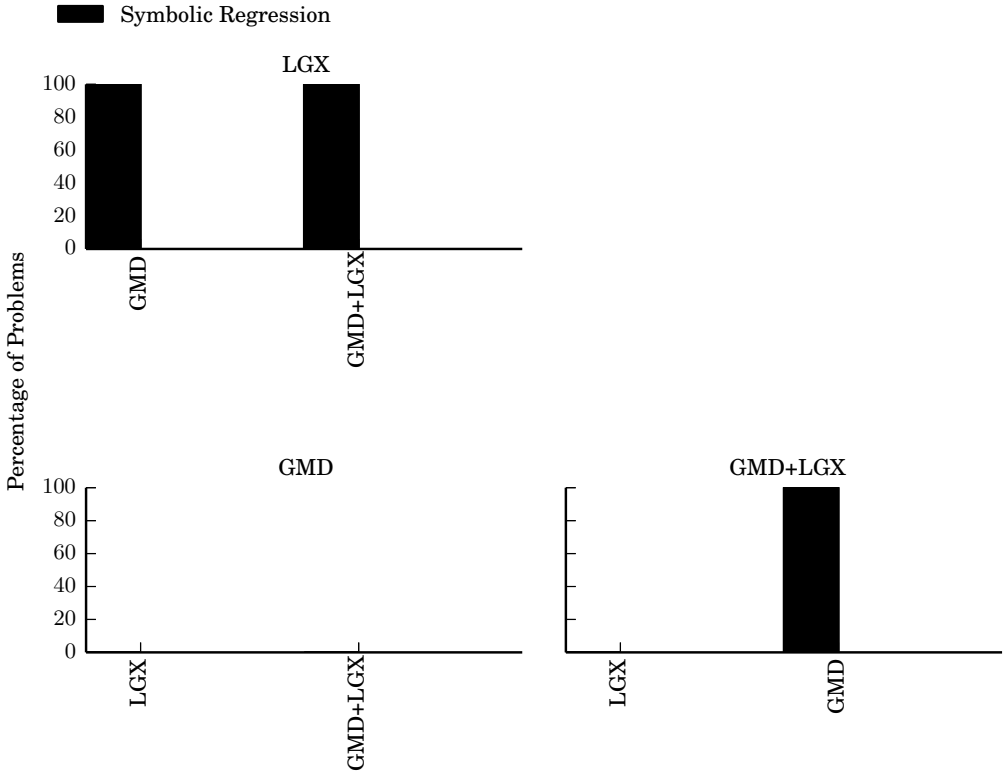
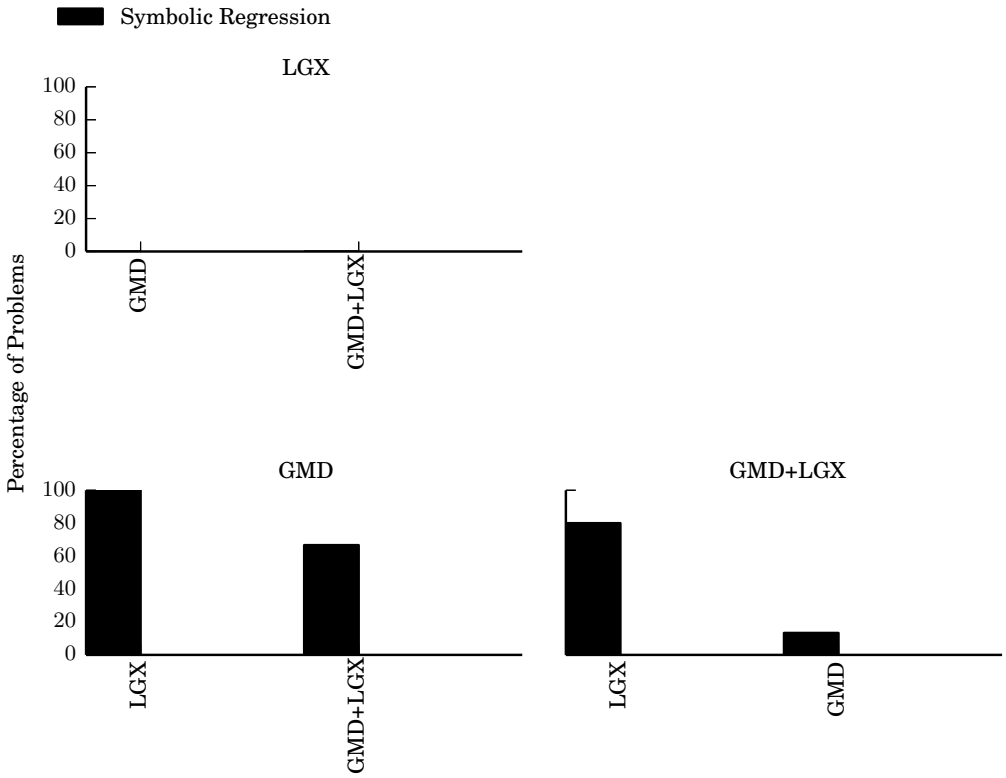


Figure 6.12 shows the pairwise comparison of the mean behavioral diversity at the end of the trial. This shows the opposite result to that of structural diversity. LGX has significantly higher behavioral diversity than GMD-GP in all (100%) of the problems, which can be expected since LGX, by definition, promotes behavioral diversity. Figure 6.12 also shows that the hybrid incorporating LGX into GMD-GP significantly increased the behavioral diversity of GMD-GP populations in all of the problems.

Figure 6.13 shows the pairwise comparison of the standard deviation of fitness at the end of the trial. Analyzing the standard deviation in fitness shows that even though GMD-GP had fewer unique behaviors, GMD-GP populations were more

Figure 6.13: **Pairwise Comparison of the Fitness Standard Deviation of the GMD-GP, LGX Hybrid.** Percentage of problems in which the method shown above each plot had significantly higher standard deviation in fitness than each method listed.



spread out in the fitness landscape than those of LGX in all of the problems. Furthermore the hybridization with GMD-GP significantly increased the fitness standard deviation compared to LGX in 80% of the problems. These results are consistent with the results for the hybrid selection techniques that we discussed in Section 6.2.2.

Chapter 7

Conclusions

This dissertation has presented several contributions to the field of evolutionary computation. We have presented a novel measure of structural diversity for tree-based GP that is simple, efficient, and effective. We have demonstrated the utility of our structural diversity measure by incorporating it into a simple yet powerful GP technique for structural diversity preservation. We have demonstrated the effectiveness of our technique by comparing it to other state-of-the-art diversity techniques on a suite of benchmark problems, and by applying our technique to the real-world problem of tuberculosis screening from raw X-ray images. We then introduced several hybrid techniques that focus simultaneously on structural and behavioral diversity for more effective GP. We demonstrated the effectiveness of these techniques by comparing them to their standalone counterparts on a suite of benchmark problems.

7.1 Conclusions on Structural Diversity

In Chapter 3, we presented GMD-GP, a novel structural diversity technique for tree-based GP, which uses tree fragments as genetic markers to preserve structural diversity. We showed that by only considering a portion of the tree, we can prevent

premature convergence while accelerating search, thereby improving the efficiency of GP. This is important because one major cause of premature convergence in GP is structural convergence, wherein the trees in the population rapidly converge to the same rooted structure in a top-down fashion [53]. By working directly to prevent this phenomenon, GMD-GP is capable of dramatically improving the search performance of GP. Furthermore, our measure of structural diversity is trivial to compute and therefore much more efficient than other genotypic distance measures that use the entire genotype and perform more complex comparisons in the calculation of distance.

We discussed in Section 4.3 that GMD-GP tended to find a solution significantly faster than other techniques in many of the benchmark problems. In particular, GMD-GP was able to improve upon the performance of the Age-Fitness Pareto Optimization algorithm (A/F), upon which it was based. Since GMD-GP uses the same underlying multi-objective evolutionary scheme as A/F, the only difference being the use of genetic marker density as the second objective instead of genotypic age, this further strengthens our argument for the utility of genetic markers. Since the rooted portions of the trees define the overall context of the trees, and changes to the root are more dramatic than changes to the leaves, maintaining a diverse set of rooted structures, which GMD-GP does by preserving genetic marker diversity in the population, can contribute to more efficient search.

In Section 4.4, we conducted several experimental analyses to determine the differences between GMD-GP and several other diversity techniques. These analyses uncovered several differences in the behavior of GMD-GP compared to the other techniques. First, GMD-GP is capable of maintaining a significantly higher level of structural diversity by only focusing on a fragment of the tree, and that structural diversity also affects the level of behavioral diversity. This is evidenced by the fact that GMD-GP populations tended to have significantly higher standard deviations

in fitness compared to the other techniques across all the problems in our benchmark suite. As the standard deviation in fitness is a measure of how spread out individuals are in the fitness landscape, this shows that even though GMD-GP focuses directly on preventing premature structural convergence, GMD-GP is capable of promoting widespread search of the fitness landscape. This is a beneficial property for exploration of complex fitness landscapes, since less-fit individuals that may contain important low-level building blocks are allowed to exist alongside fitter individuals, thus providing low-level building blocks the opportunity to propagate throughout the population.

Although GMD-GP populations typically had significantly higher standard deviation in fitness, which can be considered a measure of behavioral diversity, GMD-GP populations did tend to have significantly fewer unique behaviors compared to the behavioral diversity techniques, which is an interesting phenomenon. As we discussed in Section 4.4, GMD-GP populations had fewer unique behaviors, using the definition of behavior that we employed in our experiments. Further analysis showed that GMD-GP populations tended to have several groups of individuals (consisting of at least two individuals) that expressed the same behavior even though they were genetically unique. This number of groups tended to be significantly greater than those of the other techniques. This can be a beneficial property for problem domains such as dynamic environments in which the population needs to readily adapt to changes. This is a potential application area for future research with GMD-GP.

7.2 Conclusions on Hybrid Structural and Behavioral Diversity Techniques

In Chapter 6, we investigated hybridizing several existing behavioral diversity techniques with our structural diversity technique. We demonstrated how GMD-GP can easily be extended to incorporate behavioral diversity selection into the parent selection phase, while still imposing structural diversity during its multi-objective optimization phase. We also showed that behavioral diversity can easily be incorporated into GMD-GP by utilizing behavior-aware crossover operators. Our results demonstrate the feasibility and effectiveness of such hybrid techniques and serve as a motivation for further research into new techniques that focus directly on sustaining both structural and behavioral diversity in GP.

Our results demonstrate that, in many cases, simultaneously focusing directly on both structural and behavioral diversity can improve upon the shortcomings of both approaches and leverage their strengths in order to achieve more effective search compared to either approach in isolation. Since GMD-GP populations tended to have fewer unique behaviors than the behavioral diversity techniques, the hybridization typically significantly increased the number of unique behaviors of GMD-GP populations while still maintaining a high standard deviation in fitness. Likewise, since the behavioral diversity techniques tended to still be susceptible to low structural diversity, the hybridization tended to significantly increase the structural diversity compared to the standalone behavioral diversity techniques.

7.3 Conclusions on GP for Tuberculosis Screening

Chapter 5 presented a new application of GP: using GP to evolve classifiers for tuberculosis (TB) screening from raw X-ray images. Our results demonstrated that, with the help of the two-tiered GP framework (2TGP), GMD-GP can evolve classifiers that are competitive in terms of accuracy compared to current methods that require image processing and feature extraction. Furthermore, our evolved classifiers are capable of classifying an X-ray image in only a fraction of a second, which is dramatically faster than the reported times for other techniques.

While our technique performed competitively in terms of accuracy and the time taken to classify an image, the large memory requirement during the training phase is still an issue. Since all the training images are currently loaded into memory at the beginning of evolution, this imposes a large memory overhead on the system due to the size of the images. While the evolved classifiers can operate on single images at a time and therefore can be run on smaller hardware such as a laptop computer, improving the system so that the training phase is less memory intensive will make this approach more feasible.

Another issue, which is very common to GP in general, is that of the interpretability of the evolved classifiers. One very desirable and important quality of a computer program that is to be used in the medical domain is that the program is understandable by the professionals that are using it. We discussed in Chapter 5 that the best evolved solutions are rather large trees, which means that interpreting the behavior of the trees is very difficult. However, in the case of TB screening, we can observe the features of the X-ray images that are being used by the classifier (as we showed in Figure 5.1 and Figure 5.2) in order to gain a better understanding of which regions in the X-ray are important to the classifier. However, more research is needed into visualizing and analyzing the behavior of the classifiers on such regions

to better understand how a classification is derived.

7.4 Future Directions

This research scratches the surface of developing techniques that focus simultaneously on structural and behavioral diversity. We demonstrated the feasibility of such techniques by hybridizing our structural diversity technique with existing behavioral diversity techniques. However, we aim to motivate GP researchers to further explore this promising area and develop novel techniques that accomplish the goal of sustaining both types of diversity. Specific research questions in this area are as follows. Can we design novel operators that directly promote both types of diversity rather than using two separate operators? Does one type of operator (e.g. crossover versus selection) provide a different advantage than the other in terms of these hybrid techniques?

Another area in which GMD-GP can readily be extended to incorporate behavioral diversity is during its exploratory phase that occurs after Pareto tournament selection. As we discussed in Chapter 3, after GMD-GP performs Pareto tournament selection, it adds a new, randomly generated individual to the population in order to maintain a constant source of new genetic material in the population. A possible extension is to use semantic geometric mutation to explore the semantic neighborhood of certain individuals. This could be done using either the fittest individuals, the most unique individuals (structurally or behaviorally), or randomly. One potential benefit of this would be that this would allow us to explore targeted areas in the behavioral space while still maintaining a constant source of new genetic material in the population.

Another area for future work is to demonstrate the effectiveness of GMD-GP and

the hybrid techniques on other real-world applications. One area to which GP has been applied is the prediction of time series data such as stock market prediction [70]. A related area of great potential is that of physiological signals. For example, GP has previously been used to classify electroencephalogram (EEG) signals to detect epileptic seizures [7], as well as for brain computer interfaces [3]. Taking this a step further, one question for future research is whether or not GP can be used to evolve EEG classifiers that can be used in a brain computer interface for controlling prosthetic limbs to restore mobility to paralyzed patients. A machine learning technique was successfully applied to a similar task using a cortical implant to restore limb movement to a quadriplegic patent [10]. However, it would be invaluable if the same or better result could be achieved with non-invasive methods, for which GP could potentially be used.

Another area where GP could be used on physiological signals is that of continuous blood pressure monitoring. For example, some techniques have been proposed using signals such as pulse transit time for continuous, cuff-less blood pressure monitoring [20]. Furthermore, one study presented a wearable approach using GP for blood pressure monitoring [67]. Further research into designing affordable and more comfortable systems that can be used by everyday people would be beneficial.

Finally, our results in Chapter 5 only scratch the surface of using GP for tuberculosis screening from raw X-ray images. While our initial results show great promise for the accuracy of the evolved solutions, further research into whether GP can be used for detecting general abnormalities in the lung from X-ray images would be valuable. Evolving classifiers without the large memory overhead required for the training images is another important goal for future research. Also, improving the true positive rate of the evolved classifiers is an essential goal for future work.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Michael Affenzeller and Stefan Wagner. Offspring selection: A new self-adaptive selection scheme for genetic algorithms. In *Adaptive and Natural Computing Algorithms*, pages 218–221. Springer, 2005.

[2] Harith Al-Sahaf, Andy Song, Kourosh Neshatian, and Mengjie Zhang. Two-tier genetic programming: towards raw pixel-based image classification. *Expert Systems with Applications*, 39(16):12291–12301, 2012.

[3] Eva Alfaro-Cid, Anna Esparcia-Alcázar, and Ken Sharman. Using distributed genetic programming to evolve classifiers for a brain computer interface. In *ESANN*, pages 59–66, 2006.

[4] SK Antani. Automated detection of lung diseases in chest x-rays. *Technical Report to the LHNCBC Board of Scientific*, 2015.

[5] Daniel Atkins, Kourosh Neshatian, and Mengjie Zhang. A domain independent genetic programming approach to automatic feature extraction for image classification. In Alice E. Smith, editor, *Proceedings of the 2011 IEEE Congress on Evolutionary Computation*, pages 238–245, New Orleans, USA, 5-8 June 2011. IEEE Computational Intelligence Society, IEEE Press.

[6] Lawrence Beadle and Colin Johnson. Semantically driven crossover in genetic programming. In *Proceedings of the IEEE World Congress on Computational Intelligence*, pages 111–116, Hong Kong, 1-6 June 2008. IEEE Computational Intelligence Society, IEEE Press.

[7] Arpit Bhardwaj, Aruna Tiwari, Ramesh Krishna, and Vishaal Varma. A novel genetic programming approach for epileptic seizure detection. *Computer methods and programs in biomedicine*, 124:2–18, 2016.

[8] Josh C. Bongard and Gregory S. Hornby. Guarding against premature convergence while accelerating evolutionary search. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pages 111–118, New York, NY, USA, 2010. ACM.

[9] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Representing shape with a spatial pyramid kernel. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 401–408. ACM, 2007.

[10] Chad E Bouton, Ammar Shaikhouni, Nicholas V Annetta, Marcia A Bockbrader, David A Friedenberg, Dylan M Nielson, Gaurav Sharma, Per B Sederberg, Bradley C Glenn, W Jerry Mysiw, et al. Restoring cortical control of

functional movement in a human with quadriplegia. *Nature*, 533(7602):247–250, 2016.

[11] Edmund K. Burke, Steven Gustafson, and Graham Kendall. Diversity in genetic programming: An analysis of measures and correlation with fitness. *IEEE Transactions on Evolutionary Computation*, 8(1):47–62, February 2004.

[12] Edmund K. Burke, Steven Gustafson, Graham Kendall, and Natalio Krasnogor. Is increased diversity in genetic programming beneficial? an analysis of the effects on performance. In Ruhul Sarker, Robert Reynolds, Hussein Abbass, Kay Chen Tan, Bob McKay, Daryl Essam, and Tom Gedeon, editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pages 1398–1405, Canberra, 8-12 December 2003. IEEE Press.

[13] Armand R. Burks and William F. Punch. An efficient structural diversity technique for genetic programming. In *GECCO '15: Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, pages 991–998, Madrid, Spain, 11-15 July 2015. ACM.

[14] Armand R. Burks and William F. Punch. An analysis of the genetic marker diversity algorithm for genetic programming. *Genetic Programming and Evolvable Machines*, 18(2):213–245, June 2017.

[15] Sema Candemir, Stefan Jaeger, Kannappan Palaniappan, Jonathan P Musco, Rahul K Singh, Zhiyun Xue, Alexandros Karargyris, Sameer Antani, George Thoma, and Clement J McDonald. Lung segmentation in chest radiographs using anatomical atlases with nonrigid registration. *IEEE transactions on medical imaging*, 33(2):577–590, 2014.

[16] Jason M Daida, Hsiaolei Li, Ricky Tang, and Adam M Hilss. What makes a problem gp-hard? validating a hypothesis of structural causes. In *Genetic and Evolutionary Computation—GECCO 2003*, pages 1665–1677. Springer, 2003.

[17] Peter Day and Asoke K. Nandi. Binary string fitness characterization and comparative partner selection in genetic programming. *IEEE Transactions on Evolutionary Computation*, 12(6):724–735, December 2008.

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

[19] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[20] Xiao-Rong Ding, Yuan-Ting Zhang, Jing Liu, Wen-Xuan Dai, and Hon Ki Tsang. Continuous cuffless blood pressure estimation using pulse transit time and photoplethysmogram intensity ratio. *IEEE Transactions on Biomedical Engineering*, 63(5):964–972, 2016.

[21] Brad Dolin, Maribel Garcia Arenas, and Juan J. Merelo Guervos. Opposites attract: Complementary phenotype selection for crossover in genetic programming. In Juan J. Merelo-Guervos, Panagiotis Adamidis, Hans-Georg Beyer, Jose-Luis Fernandez-Villacanas, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VII*, number 2439 in Lecture Notes in Computer Science, LNCS, pages 142–152, Granada, Spain, 7-11 September 2002. Springer-Verlag.

[22] Jeannie Fitzgerald and Conor Ryan. Exploring boundaries: optimising individual class boundaries for binary classification problem. In *GECCO '12: Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, pages 743–750, Philadelphia, Pennsylvania, USA, 7-11 July 2012. ACM.

[23] Edgar Galvan-Lopez, Brendan Cody-Kenny, Leonardo Trujillo, and Ali Kattan. Using semantics in the selection mechanism in genetic programming: A simple method for promoting semantic diversity. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 2972–2979. IEEE, 2013.

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

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

[26] David E Goldberg and Jon Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Hillsdale, NJ: Lawrence Erlbaum, 1987.

[27] Steven Gustafson, Edmund K Burke, and Graham Kendall. Sampling of unique structures and behaviours in genetic programming. In *Genetic Programming*, pages 279–288. Springer, 2004.

[28] Thomas Helmuth, Lee Spector, and James Matheson. Solving uncompromising problems with lexicase selection. *IEEE Transactions on Evolutionary Computation*, 19(5):630–643, October 2015.

[29] John H Holland. Building blocks, cohort genetic algorithms, and hyperplane-defined functions. *Evolutionary computation*, 8(4):373–391, 2000.

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

[31] Gregory S. Hornby. Steady-state alps for real-valued problems. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 795–802, New York, NY, USA, 2009. ACM.

[32] Jianjun Hu, Erik Goodman, Kisung Seo, Zhun Fan, and Rondal Rosenberg. The hierarchical fair competition framework for sustainable evolutionary algorithms. *Evolutionary Computation*, 13(2):241–277, Summer 2005.

[33] Jianjun Hu, Kisung Seo, Shaobo Li, Zhun Fan, Ronald C. Rosenberg, and Erik D. Goodman. Structure fitness sharing (sfs) for evolutionary design by genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 780–787. Morgan Kaufmann Publishers, 2002.

[34] Andrew Innes. *Genetic Programing for Cephalometric Landmark Detection.* PhD thesis, School of Aerospace, Mechanical and Manufacturing Engineering, RMIT University, Victoria, Australia, 29 August 2007.

[35] David Jackson. Phenotypic diversity in initial genetic programming populations. In *Proceedings of the 13th European Conference on Genetic Programming, EuroGP 2010*, volume 6021 of *LNCS*, pages 98–109, Istanbul, 7-9 April 2010. Springer.

[36] David Jackson. Promoting phenotypic diversity in genetic programming. In *PPSN 2010 11th International Conference on Parallel Problem Solving From Nature*, volume 6239 of *Lecture Notes in Computer Science*, pages 472–481, Krakow, Poland, 11-15 September 2010. Springer.

[37] Stefan Jaeger, Alexandros Karargyris, Sema Candemir, Les Folio, Jenifer Siegelman, Fiona Callaghan, Zhiyun Xue, Kannappan Palaniappan, Rahul K Singh, Sameer Antani, et al. Automatic tuberculosis screening using chest radiographs. *IEEE transactions on medical imaging*, 33(2):233–245, 2014.

[38] Joshua D Knowles, Richard A Watson, and David W Corne. Reducing local optima in single-objective problems by multi-objectivization. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 269–283. Springer, 2001.

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

[40] Krzysztof Krawiec and Pawel Lichocki. Approximating geometric crossover in semantic space. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 987–994, New York, NY, USA, 2009. ACM.

[41] Krzysztof Krawiec and Una-May O'Reilly. Behavioral programming: a broader and more detailed take on semantic GP. In *GECCO '14: Proceedings of the 2014 conference on Genetic and evolutionary computation*, pages 935–942, Vancouver, BC, Canada, 12-16 July 2014. ACM. Best paper.

[42] Krzysztof Krawiec and Tomasz Pawlak. Locally geometric semantic crossover. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '12, pages 1487–1488, New York, NY, USA, 2012. ACM.

[43] Krzysztof Krawiec and Tomasz Pawlak. Approximating geometric crossover by semantic backpropagation. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, pages 941–948, New York, NY, USA, 2013. ACM.

[44] Krzysztof Krawiec and Tomasz Pawlak. Locally geometric semantic crossover: a study on the roles of semantics and homology in recombination operators. *Genetic Programming and Evolvable Machines*, 14(1):31–63, 2013.

[45] William La Cava, Lee Spector, and Kourosh Danai. Epsilon-lexicase selection for regression. In Tobias Friedrich, editor, *GECCO '16: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, page fp376, Denver, USA, 20-24 July 2016. ACM.

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

[47] Sean Luke. When short runs beat long runs. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 74–80, 2001.

[48] M. Maghoumi and B. J. Ross. A comparison of genetic programming feature extraction languages for image classification. In *IEEE Symposium on Computational Intelligence for Multimedia, Signal and Vision Processing (CIMSIVP 2014)*, December 2014.

[49] Hammad Majeed and Conor Ryan. A less destructive, context-aware crossover operator for GP. In Pierre Collet, Marco Tomassini, Marc Ebner, Steven Gustafson, and Anikó Ekárt, editors, *Proceedings of the 9th European Conference on Genetic Programming*, volume 3905 of *Lecture Notes in Computer Science*, pages 36–48, Budapest, Hungary, 10 - 12 2006. Springer.

[50] Yuliana Martinez, Leonardo Trujillo, Enrique Naredo, and Pierrick Legrand. A comparison of fitness-case sampling methods for symbolic regression with genetic programming. In *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V*, volume 288 of *Advances in Intelligent Systems and Computing*, pages 201–212, Peking, 1-4 July 2014. Springer.

[51] James McDermott, David R. White, Sean Luke, Luca Manzoni, Mauro Castelli, Leonardo Vanneschi, Wojciech Jaskowski, Krzysztof Krawiec, Robin Harper, Kenneth De Jong, and Una-May O'Reilly. Genetic programming needs better benchmarks. In *GECCO '12: Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, pages 791–798, Philadelphia, Pennsylvania, USA, 7-11 July 2012. ACM.

[52] R I (Bob) McKay. Fitness sharing in genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 435–442, Las Vegas, Nevada, USA, 10-12 July 2000. Morgan Kaufmann.

[53] Nicholas Freitag McPhee and Nicholas J. Hopper. Analysis of genetic diversity through population history. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1112–1120, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann.

[54] Nicholas Freitag McPhee, Brian Ohs, and Tyler Hutchison. Semantic building blocks in genetic programming. Working Paper Series Volume 3 Number 2, University of Minnesota Morris, 600 East 4th Street, Morris, MN 56267, USA, 12 December 2007.

[55] Alberto Moraglio, Krzysztof Krawiec, and Colin G. Johnson. Geometric semantic genetic programming. In *Parallel Problem Solving from Nature, PPSN XII (part 1)*, volume 7491 of *Lecture Notes in Computer Science*, pages 21–31, Taormina, Italy, September 1-5 2012. Springer.

[56] Gearoid Murphy and Conor Ryan. A simple powerful constraint for genetic programming. In Michael O'Neill, Leonardo Vanneschi, Steven Gustafson, Anna Isabel Esparcia Alcazar, Ivanoe De Falco, Antonio Della Cioppa, and Ernesto Tarantino, editors, *Proceedings of the 11th European Conference on Genetic Programming, EuroGP 2008*, volume 4971 of *Lecture Notes in Computer Science*, pages 146–157, Naples, 26-28 March 2008. Springer.

[57] Quang Uy Nguyen, Xuan Hoai Nguyen, and Michael O'Neill. Semantic aware crossover for genetic programming: The case for real-valued function regression. In Leonardo Vanneschi, Steven Gustafson, Alberto Moraglio, Ivanoe De Falco, and Marc Ebner, editors, *Proceedings of the 12th European Conference on Genetic Programming, EuroGP 2009*, volume 5481 of *LNCS*, pages 292–302, Tuebingen, April 15-17 2009. Springer.

[58] Quang Uy Nguyen, Xuan Hoai Nguyen, Michael O'Neill, and Alexandros Agapitos. An investigation of fitness sharing with semantic and syntactic distance metrics. In *Proceedings of the 15th European Conference on Genetic Programming, EuroGP 2012*, volume 7244 of *LNCS*, pages 109–120, Malaga, Spain, 11-13 April 2012. Springer Verlag.

[59] Quang Uy Nguyen, Tuan Anh Pham, Xuan Hoai Nguyen, and James McDermott. Subtree semantic geometric crossover for genetic programming. *Genetic Programming and Evolvable Machines*, 17(1):25–53, March 2016.

[60] World Health Organization. Global tuberculosis report. 2016.

[61] Tomasz P Pawlak, Bartosz Wieloch, and Krzysztof Krawiec. Review and comparative analysis of geometric semantic crossovers. *Genetic Programming and Evolvable Machines*, 16(3):351–386, 2015.

[62] Tomasz P. Pawlak, Bartosz Wieloch, and Krzysztof Krawiec. Semantic backpropagation for designing search operators in genetic programming. *IEEE Transactions on Evolutionary Computation*, 19(3):326–340, June 2015.

[63] Riccardo Poli. Hyperschema theory for gp with one-point crossover, building blocks, and some new results in ga theory. In *Genetic Programming*, volume 1802 of *Lecture Notes in Computer Science*, pages 163–180. Springer Berlin Heidelberg, 2000.

[64] Riccardo Poli and William B. Langdon. On the search properties of different crossover operators in genetic programming. In *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 293–301, University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998. Morgan Kaufmann.

[65] Justinian P. Rosca and Dana H. Ballard. Rooted-tree schemata in genetic programming. In *Advances in Genetic Programming 3*, chapter 11, pages 243–271. MIT Press, Cambridge, MA, USA, June 1999.

[66] Conor Ryan, Krzysztof Krawiec, Una-May O'Reilly, Jeannie Fitzgerald, and David Medernach. Building a stage 1 computer aided detector for breast cancer using genetic programming. In Miguel Nicolau, Krzysztof Krawiec, Malcolm I. Heywood, Mauro Castelli, Pablo Garcia-Sanchez, Juan J. Merelo, Victor M. Rivas Santos, and Kevin Sim, editors, *17th European Conference on Genetic Programming*, volume 8599 of *LNCS*, pages 162–173, Granada, Spain, 23-25 April 2014. Springer.

[67] G. Sannino, I. De Falco, and G. De Pietro. Genetic programming for a wearable approach to estimate blood pressure embedded in a mobile-based health system. In *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 775–783, Nov 2015.

[68] J David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st international Conference on Genetic Algorithms*, pages 93–100. L. Erlbaum Associates Inc., 1985.

[69] Michael Schmidt and Hod Lipson. Age-fitness pareto optimization. In *Genetic Programming Theory and Practice VIII*, volume 8 of *Genetic and Evolutionary Computation*, chapter 8, pages 129–146. Springer, Ann Arbor, USA, 20-22 May 2010.

[70] Alaa Sheta, Hossam Faris, and Mouhammd Alkasassbeh. A genetic programming model for S&P 500 stock market prediction. *International Journal of Control and Automation*, 6(5):303–314, 2013.

[71] Guido Smits and Mark Kotanchek. Pareto-front exploitation in symbolic regression. In Una-May O'Reilly, Tina Yu, Rick L. Riolo, and Bill Worzel, editors, *Genetic Programming Theory and Practice II*, chapter 17, pages 283–299. Springer, Ann Arbor, 13-15 May 2004.

[72] Terence Soule and James A Foster. Effects of code growth and parsimony pressure on populations in genetic programming. *Evolutionary computation*, 6(4):293–309, 1998.

[73] Lee Spector, David M. Clark, Ian Lindsay, Bradford Barr, and Jon Klein. Genetic programming for finite algebras. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1291–1298, Atlanta, GA, USA, 12-16 July 2008. ACM.

[74] Nguyen Quang Uy, Nguyen Xuan Hoai, Michael O'Neill, R. I. McKay, and Edgar Galvan-Lopez. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119, June 2011.

[75] Leonardo Vanneschi, Mauro Castelli, Luca Manzoni, and Sara Silva. *Genetic Programming: 16th European Conference, EuroGP 2013, Vienna, Austria, April 3-5, 2013. Proceedings*, chapter A New Implementation of Geometric Semantic GP and Its Application to Problems in Pharmacokinetics, pages 205–216. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[76] Leonardo Vanneschi, Mauro Castelli, and Sara Silva. A survey of semantic methods in genetic programming. *Genetic Programming and Evolvable Machines*, 15(2):195–214, 2014.