Genetic Programming with a Norm-referenced Fitness Function

Geng Li
School of Computer Science
University of Manchester,
Oxford Road
Manchester, M14 9PL
lig@cs.man.ac.uk

Xiao-Jun Zeng
School of Computer Science
University of Manchester,
Oxford Road
Manchester, M14 9PL
x.zeng@cs.man.ac.uk

ABSTRACT

In this paper, we develop a new fitness function based on adjustment of the original fitness function using population performance. We call this new fitness function norm-referenced fitness function since it is motivated by the idea of norm-referenced test. Experiments performed in two benchmark problems show that, the norm-referenced fitness function developed is capable of improving the overall performance of GP system. Further analysis of the fitness function reveals that the original fitness function suffers from an implicit bias we named as implicit bias towards exploitation in later generations. This implicit bias pushes the population towards convergence. The norm-referenced fitness developed however does not inherit this bias, and we think this is the main reason why the norm-referenced fitness function is able to outperform the original fitness function. We further study the selection of the newly introduced parameter λ in norm-referenced fitness function and give a number of advices to select the optimal value of the parameter.

Categories and Subject Descriptors

I.2.2 [**Automatic Programming**]: Program synthesis; F.2 [**Analysis of Algorithms and Problem Complexity**]: General

General Terms

Algorithms

Keywords

Fitness Function Adjustment, Population Performance, Internal Fitness Measure

1. INTRODUCTION

In educational assessment, there are mainly two kinds of tests widely used: criterion-referenced test or norm-referenced test [3]. In criterion-referenced test, as the name implies, candidates are measured against predefined objective criteria [3]. A good example of criterion-referenced test is driving test. In norm-referenced test, candidates are measured against a predefined group of exam takers, to give an estimation of their relative positions in that population [3]. Examples of norm-referenced test include the IQ test, and many entrance exams such as GRE. The main difference between norm- and criterion-referenced test is that, the former tries to show the relative ranking of test subjects while the later assesses mastery of certain material [3]. As a result, entrance exams are generally norm-referenced since institutions are more interested in exam takers' relative ranking, while most of the end of term diagnostic exams are criterion-referenced tests because teachers are more interested in whether students have mastered course material or not. One big problem of criterion-referenced test is the possibility that test takers are judged by exam questions which are not appropriate to their level. To overcome this problem, question setters must make sure their expectation matches exam takers' level. On the other hand, norm-referenced test does not have this problem because it does not seek to enforce any expectation of what exam takers should be able to do.

In genetic programming (GP), the role of fitness function is very similar to an exam. Individuals within population are exam takers. The fitness value obtained is the exam result. GP system then uses the exam result to guide the selection of parents in breading phase. The exact form of fitness function may vary from problem to problem. Typically, a fitness function consists of a number of test cases. Each test case consists of a number of inputs and a desired output. It acts as a "question" in exam. An individual's fitness value for a test case is the distance from individual's actual output using given input to the desired output. Formally, let x be an individual, we use $f_i(x)$ to represent the fitness value of individual x for test case i, then, for a fitness function which consists of n test cases, denoted as $F(x)$, we have:

$$F(x) = \sum_{i=1}^{n} f_i(x). \quad (1)$$

The fitness function is a criterion-referenced test. This is because an individual's "absolute" performance is used to judge the quality of that individual. As a result, the fitness function does have the miss-match problem we discussed previously. The fundamental problem of the mismatch between the expectation and the actual performance is that

GECCO'11, July 12–16, 2011, Dublin, Ireland.

it reduces the ability to differentiate between better exam takers and worse exam takers. In another word, when using criterion-referenced test result to establish the relative ranking of test takers, one needs to ensure that test questions match the exam takers' average level, such that those questions are able to differentiate those exam takers' level.

On the other hand, a norm-referenced test would be a better fit in this scenario because, firstly norm-referenced test does not enforce any expectation of test subjects' level, and secondly the primary purpose of the norm-referenced test is to give ranking information telling which exam taker performed at an average level, which exam taker did better, and which exam taker did worse. In this paper, we develop a new norm-referenced fitness function as an alternative to the original fitness function (1) in GP. We firstly develop the concept of internal fitness measure which represents the population's performance. Then the norm-referenced fitness function is developed through adjustment of the original fitness function using internal fitness measures. We perform experiments on two GP domains, one continuous and one discrete. Experiments on both domains show that the norm-referenced fitness function is capable of improving the overall GP performance. The rest of this paper is organised as follows. In the next section, we develop two internal fitness measures. Then, we use those two measures to build the norm-referenced fitness function. In section 3, we give a concrete description of the algorithm and a discussion of a number of implementation issues. Then, in section 4, we test the performance of the new fitness function using two GP problems. Finally, we conclude this paper with a discussion of further works.

2. INTERNAL FITNESS MEASURE

As we discussed in Section 1, the fundamental problem of the criterion-referenced fitness function used in GP is that, it only subjectively judges the quality of candidate solution without taking into account other individuals' performance within the same population. To overcome this problem, in this section, we build a more comprehensive fitness function which not only considers individual's subjective raw fitness, but also takes into account present population's performance. We call the former "external" fitness measure because it is defined by the user and is provided as input into the GP system. We call the later "internal" fitness measure because it is derived from the population and it is independent from the problem the GP system is solving.

In this paper, we consider two ways to build the "internal" fitness measure. Fitness function is a summation of errors from every single test case. Usually, we consider all test cases as a fitness value. The objective of GP system would be to minimize $F(x)$. Now, if we consider each test case independently, for a problem with n test cases, given a population P which consists of m individuals, for each test case, we can calculate the population P's performance:

$$e_i = \frac{\sum_{j=1}^{m} S(f_i(x_j))}{m} \tag{2}$$

where $S(.)$ is a scaling function and $S(x) \in [0,1]$ and $f(.)$ is the same as in (1). Let $\boldsymbol{e} = (e_1, e_2, ..., e_n)$, then $\boldsymbol{e}$ represents the population's performance for all test cases. Normalizing this error vector $\boldsymbol{e}$, we get a weight vector $\boldsymbol{w}^-$, in which:

$$w_i^- = \frac{e_i}{\sum_{i=1}^{n} e_i} \tag{3}$$

This weight vector $\boldsymbol{w}^-$ represents the relative importance of test cases using error as the criterion.

Similar to the calculation above, if we use accuracy rather than error as the criterion, we can calculate the population P's performance, $\boldsymbol{a} = (a_1, a_2, ..., a_n)$, where:

$$a_i = 1 - e_i \tag{4}$$

Normalizing $\boldsymbol{a}$, we get the weight vector $\boldsymbol{w}^+$, where:

$$w_i^+ = \frac{a_i}{\sum_{i=1}^{n} a_i} \tag{5}$$

This weight vector $\boldsymbol{w}^+$ represents the relative importance of test cases using accuracy as the criterion.

We can use $\boldsymbol{w}^-$ and $\boldsymbol{w}^+$ as internal fitness measures to create adjustments of the original fitness function. Let $\boldsymbol{f} = (f_1(x), f_2(x), ..., f_n(x))^T$, if only $\boldsymbol{w}^-$ is used, we get:

$$F_{adj}^-(x) = \boldsymbol{w}^- \cdot \boldsymbol{f}$$

If only $\boldsymbol{w}^+$ is used, we get:

$$F_{adj}^+(x) = \boldsymbol{w}^+ \cdot \boldsymbol{f}.$$

In the next, we give a brief example calculating $\boldsymbol{w}^-$, $\boldsymbol{w}^+$, $F_{adj}^-(x)$ and $F_{adj}^+(x)$, using even parity 3 problem domain which has 8 test cases as an example. Considering a population of 4 individuals: $P = \{p_1, p_2, p_3, p_4\}$. The fitness of each individual is listed in Table 1. For this population P,

Individual	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	F
p_1	0	0	0	0	0	1	1	0	2
p_2	1	0	1	0	1	1	1	0	5
p_3	1	0	1	0	1	0	1	0	4
p_4	0	0	0	0	1	1	1	0	3

Table 1: Fitness Values of 4 Individuals in P

using (2), we get:

$$\begin{aligned} \boldsymbol{e} &= (\frac{2}{4}, \frac{0}{4}, \frac{2}{4}, \frac{0}{4}, \frac{3}{4}, \frac{3}{4}, \frac{4}{4}, \frac{0}{4}) \\ &= (0.5, 0, 0.5, 0, 0.75, 0.75, 1, 0) \end{aligned}$$

Where $S(x) = x$ is used as the scaling function. Normalizing $\boldsymbol{e}$ using (3), we get:

$$\begin{aligned} \boldsymbol{w}^- &= (\frac{0.5}{3.5}, \frac{0}{3.5}, \frac{0.5}{3.5}, \frac{0}{3.5}, \frac{0.75}{3.5}, \frac{0.75}{3.5}, \frac{1}{3.5}, \frac{0}{1.75}) \\ &= (0.143, 0, 0.143, 0, 0.214, 0.214, 0.286, 0) \end{aligned}$$

Similarly, we can also calculate $\boldsymbol{a}$ using (4):

$$\begin{aligned} \boldsymbol{a} &= (1-\frac{2}{4}, 1-\frac{0}{4}, 1-\frac{2}{4}, 1-\frac{0}{4}, 1-\frac{3}{4}, 1-\frac{3}{4}, 1-\frac{4}{4}, 1-\frac{0}{4}) \\ &= (0.5, 1, 0.5, 1, 0.25, 0.25, 0, 1) \end{aligned}$$

Normalizing $\boldsymbol{a}$ using (5), we get:

$$\begin{aligned} \boldsymbol{w}^+ &= (\frac{0.5}{4.5}, \frac{1}{4.5}, \frac{0.5}{4.5}, \frac{1}{4.5}, \frac{0.25}{4.5}, \frac{0.25}{4.5}, \frac{0}{4.5}, \frac{1}{4.5}) \\ &= (0.11, 0.22, 0.11, 0.22, 0.06, 0.06, 0, 0.22) \end{aligned}$$

Using $\boldsymbol{w}^-$ and $\boldsymbol{w}^+$, we can calculate F_{adj}^- and F_{adj}^+. For example, for p_1:

$$\begin{aligned} F_{adj}^-(p_1) &= (0.143, 0, 0.143, 0, 0.214, 0.214, 0.286, 0) \cdot \\ &\quad (0, 0, 0, 0, 0, 1, 1, 0)^T \\ &= 0.5 \\ F_{adj}^+(p_1) &= (0.11, 0.22, 0.11, 0.22, 0.06, 0.06, 0, 0.22) \cdot \\ &\quad (0, 0, 0, 0, 0, 1, 1, 0)^T \\ &= 0.06. \end{aligned}$$

F^-_{adj} and F^+_{adj} consider the present population's performance from two different perspectives of view. They explicitly address the population's performance differences among different test cases, similar to historically assessed hardness developed in [5]. In our case, using the examination analogy, test cases are exam questions. In the original fitness function, all "questions" worth the same mark. In F^-_{adj}, in which the original fitness function is adjusted using $\boldsymbol{w}^-$, "hard" questions which most of students cannot answer are worth more marks. This encourages students to try to solve those hard questions. In F^+_{adj}, where the original fitness function is adjusted using $\boldsymbol{w}^+$, "easy" questions which most of students answer correctly are worth more marks. This promotes students to concentrate on easy questions while ignoring hard ones. The accumulated effect of F^-_{adj} drifts the population towards unexplored region of searching space. This is because a "hard" test case attracts the population by having a bigger weight. But once the population performs better on that "hard" test case, its weight becomes smaller. Then the population moves its "interest" to other "hard" test cases. In another word, F^-_{adj} destabilizes the evolution by keeping changing the searching direction. On the other hand, the accumulated effect of F^+_{adj} drives the population to converge to the current state of evolution. This is because "easy" test cases, which have bigger weights, have even bigger weights as the population evolves. In another word, F^+_{adj} stabilizes the evolution by enforcing the search direction.

In fact, F^-_{adj} and F^+_{adj} represent two different but equally important aspects of the evolution system: the exploration and exploitation [1]. On one hand, F^-_{adj} constantly changes the searching direction to explore the whole searching space. on the other hand, F^+_{adj} drifts the population into convergence, maintaining the stability of the system. The balance between exploration and exploitation is critical to the behavior of GP system [1]. As a result, we can combine F^-_{adj} and F^+_{adj} to build a more comprehensive fitness function based on population's performance:

$$\begin{aligned} F_{adj}(x) &= \lambda \cdot F^-_{adj}(x) + (1-\lambda) \cdot F^+_{adj}(x) \\ &= (\lambda \boldsymbol{w}^- + (1-\lambda)\boldsymbol{w}^+) \cdot \boldsymbol{f} \end{aligned} \quad (6)$$

where λ is a parameter and $\lambda \in [0,1]$. We call this new fitness function (6) the norm-referenced fitness function because it not only considers the raw fitness value $\boldsymbol{f}$, but also takes into account population's performance. Similar to the norm-referenced test, which gives relative rank of test taker, F_{adj} gives an individual's fitness value relative to the present population's performance.

3. ALGORITHM DESCRIPTION

The norm-referenced fitness function developed can be ported into existing GP implementations using adjusted fitness. The concept of adjusted fitness is originally developed by Koza to "exaggerate the importance of small differences in the value of the standardized fitness as the standardized fitness approaches 0" [6]. It was mainly used for fitness proportionate selection. Later on, because of the development of various bloating control methods, the usage of adjusted fitness has been extended. Nowadays, most of GP implementations use adjusted fitness calculation as a custom point where users can alter raw fitness value, applying linear parametric parsimony pressure for example, and underlying selection methods are based on adjusted fitness rather than raw fitness values [9].

In our case, the adjustment of fitness can be implemented in two steps. After every individual's fitness in the present population is calculated, we firstly calculate $\boldsymbol{w}^-$ and $\boldsymbol{w}^+$ using algorithm 1.

Algorithm 1 Algorithm to Build $\boldsymbol{w}^-$ and $\boldsymbol{w}^+$

function build-metric(population p) : $\boldsymbol{w}^-$, $\boldsymbol{w}^+$
$\boldsymbol{e} \leftarrow (e_1 = 0, \ldots, e_n = 0), i = 1 \ldots n$
$\boldsymbol{a} \leftarrow (a_i = 0, \ldots, a_n = 0), i = 1 \ldots n$
for testcase $i = 1 \ldots n$ **do**
  for all individual $x \in p$ **do**
    $e_i = e_i + S(f_i(x))$
    $a_i = a_i + (1 - S(f_i(x)))$
  end for
end for
$\boldsymbol{w}^- \leftarrow (w^-_1 = \frac{e_1}{\sum_{i=1}^n e_i}, \ldots, w^-_n = \frac{e_n}{\sum_{i=1}^n e_i}), i = 1 \ldots n$
$\boldsymbol{w}^+ \leftarrow (w^+_1 = \frac{a_1}{\sum_{i=1}^n a_i}, \ldots, w^+_n = \frac{a_n}{\sum_{i=1}^n a_i}), i = 1 \ldots n$
return $\boldsymbol{w}^-$ and $\boldsymbol{w}^+$
end function

Then, with $\boldsymbol{w}^-$ and $\boldsymbol{w}^+$, we can calculate fitness adjustments for every individual in the population using Algorithm 2.

Algorithm 2 Algorithm to Calculate Fitness Adjustment

function calc-adjusted-fitness(individual x, $\boldsymbol{w}^-$, $\boldsymbol{w}^+$, λ) : f_{adj}
$f_{adj} \leftarrow 0$
for testcase $i = 1 \ldots n$ **do**
  $f_{adj} = f_{adj} + (\lambda \cdot w^-_i + (1-\lambda) \cdot w^+_i) \cdot f_i(x)$
end for
return f_{adj}
end function

Unlike common fitness adjustment calculations such as linear parametric parsimony pressure, in which the adjusted fitness is calculated immediately after the calculation of raw fitness, our adjustment can only be calculated after all individuals within the population are evaluated. This is because the calculation of $\boldsymbol{w}^-$ and $\boldsymbol{w}^+$ require every individual's raw fitness information. In addition, to avoid repeated evaluation, every individual's fitness needs to be stored as a vector indexed by the test case number rather than a single aggregated value. This is because individual test case fitness information i.e. $f(x)$, is required in both Algorithm 1 and Algorithm 2. Without re-evaluation, the runtime overheads to implement the adjustment can be neglected.

4. EXPERIMENTS

In this section, we empirically study the performance of the norm-referenced fitness function developed. In the first experiment, we test the performance using the even parity problem domain. Even parity domain is selected for two reasons. Firstly, it is a discrete problem, i.e. the fitness value for each test case can only be either 0 or 1. This simplifies the choice of scaling function S in (2). Secondly, the parity problem is a representative problem in digital circuit generation, and according to Koza, even parity problem appears

to be the most difficult boolean functions to be detected via a blind random search [6]. The complexity of even parity problem has been thoroughly studied. As a result, parity problem is selected and studied in this paper.

4.1 Initial Experiments

In the first experiment, we test the performance of norm-referenced fitness function using the even parity 5 problem domain. We firstly randomly generate 50 initial populations. Then, for each population generated, we perform GP runs firstly using the original fitness function, and then using the norm-referenced fitness function developed. We test the parameter λ ranging from 0 to 1 in steps of 0.1 for norm-referenced fitness function. So, for each population initialization, we have 11 GP runs, one using original fitness function, and the other 10 using norm-referenced fitness function with different λ values. Since the output of GP system is usually the best individual in the last generation, to compare the performance, we use *best individual in the current generation* as the criterion. The rest experiment parameters are as follows. The population size is 500. Tournament selection is used and tournament size is set to 5. In breeding process, only crossover and reproduction are used with probability 90% and 10% respectively. GP runs for 50 generations. We use $S(x) = x$ as the scaling function in (2). We use GPLab [9] as the testing platform.

Fitness Function	λ	Best Fitness	Std	T-Test
Original		7.46	1.459	
Norm-referenced	1.0	7.70	1.170	$\rightarrow$
	0.9	7.60	1.095	$\rightarrow$
	0.8	7.50	1.063	$\rightarrow$
	0.7	6.92	1.197	$\uparrow$
	0.6	6.32	1.618	$\uparrow$
	0.5	**6.26**	**1.180**	$\uparrow$
	0.4	6.84	2.043	$\rightarrow$
	0.3	9.92	2.252	$\downarrow$
	0.2	11.46	1.846	$\downarrow$
	0.1	11.52	1.526	$\downarrow$
	0.0	11.44	1.627	$\downarrow$

Table 2: Best Fitness at Gen 50 in Even Parity 5 Problem

Table 2 summaries the experiment results we get. The best fitness column is the average best fitness achieved at generation 50 over 50 initializations. In order to illustrate the statistical significance, we perform T-Test with 95% confidence between the original statistics (in the first row) and each λ value. The test results are given in column T-Test in Table 2, where $\uparrow$ represents mean value which is statistically superior to original fitness function, $\downarrow$ represents mean value which is statistically inferior, and $\rightarrow$ represents no statistically significant difference. From Table 2, we find that norm-referenced fitness function developed outperforms the original fitness function when parameter λ is around 0.5. λ equals to 0.5 gives best performance improvement (16.08%). But, when λ is small, the performance is worse compared to the original fitness function. Because we are using the same 50 initializations across experiments of different parameters, in addition to the average performance comparison above, it is also possible to compare performance in a one-to-one basis, as in Table 3. From Table 3, we can find that, when

λ	Num of Better	Num of Equal	Num of Worse
1.0	15	15	20
0.9	17	7	26
0.8	18	11	21
0.7	24	8	18
0.6	26	13	11
0.5	**29**	**13**	**8**
0.4	25	10	15
0.3	6	6	38
0.2	2	2	46
0.1	0	2	48
0.0	0	3	47

Table 3: Number of GP Runs when Norm-referenced Fitness Function Performs Better, Equal, or Worse Compared to the According Original Fitness function with Same Initialization

$\lambda = 0.4, 0.5, 0.6, 0.7$, norm-referenced fitness function performs better in around 50% initializations.

4.2 Analysis of Selection Intensity

Norm-referenced fitness function adjusts the original fitness function using the population's performance. This adjustment changes individuals' fitness ranks and ultimately affects the selection of parents in the breeding phase. Thus, it is possible to analyze the effect of norm-referenced fitness function by studying how it affects the selection of parents. Here, we use selection intensity to analyze the effect of the norm-referenced fitness function. Selection intensity is developed by Blickle and Thiele in [2]. The selection intensity I of a selection method is:

$$I = \frac{\bar{M}^* - \bar{M}}{\bar{\sigma}}$$

where $\bar{M}^*$ is the expected mean fitness after selection, $\bar{M}$ is the expected mean fitness before selection, and $\bar{\sigma}$ is the mean fitness variance before selection. The selection intensity depends on the distribution of fitness of the population before the selection process. As a result, in [2], Blickle and Thiele restricted the fitness distribution to normalized Gaussian distribution in order to derive mathematical formulas, such that different selection methods can be compared. In our case, however, the goal is to analyze how norm-referenced fitness function affects the selection method throughout the evolution process. Thus, we need to consider different distributions of fitness.

The simulation of selection intensity is designed as follows. We select a single GP run from previous experiments in which the original fitness function is used. For each generation, we simulate tournament selections with tournament size 5 firstly using original fitness function, then using norm-referenced fitness function with λ ranging from 0 to 1 in steps of 0.1. We then calculate the selection intensity for each case. In order to reduce the randomness in tournament selection, we use the same random number generator for each simulation, i.e. the same 5 random individuals are selected, only the selection of the best may be altered based on different fitness functions and parameter λ settings. The simulation result is in Figure 1.

From this simulation, we get two direct observations. Firstly, the selection intensity of tournament selection with the orig-

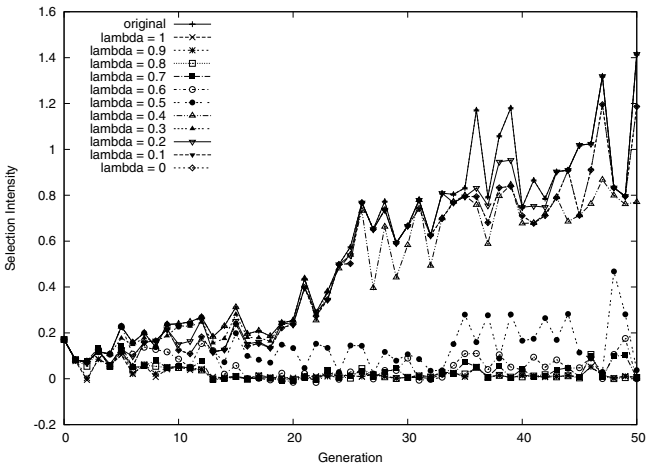

Figure 1: The Selection Intensity of Tournament Selection for the Original Fitness Function and Norm-referenced Fitness Functions with Different λ Settings

inal fitness function increases in later generations. The average selection intensity for the original fitness function from generation 0 to 20 is 0.192, while this average increases to 0.810 for generation 21 to 50. Secondly, for tournament selection with norm-referenced fitness function, the selection intensity behaviors can be divided into three categories. For λ values 0.6 to 1, where $\boldsymbol{w}^-$ dominates F_{adj}, the selection intensity fluctuates around 0. For λ values 0 to 0.4, where $\boldsymbol{w}^+$ dominates F_{adj}, the selection intensity follows the same trend as the original fitness function. For λ value 0.5, the selection intensity seems to be independent from generation. The mean selection intensity is 0.149, which is very close to the initial selection intensity in generation 0, which is 0.171. This observation confirms our discussion in Section 2. $\boldsymbol{w}^-$ promotes exploration by reducing the selection intensity, while $\boldsymbol{w}^+$ promotes exploitation by increasing the selection intensity. λ value 0.5 gives the best balance between exploration and exploitation. As a result, it gives the best performance in the initial experiment.

To further study the selection intensity, Let's review F_{adj}. From (6), we have:

$$\begin{aligned} F_{adj}(x) &= (\lambda \boldsymbol{w}^- + (1-\lambda)\boldsymbol{w}^+) \cdot \boldsymbol{f} \\ &= \textstyle\sum_{i=1}^{n} (\lambda w_i^- + (1-\lambda) w_i^+) \cdot f_i(x) \end{aligned}$$

Substitute w_i^- and w_i^+ using (3) and (5), we get:

$$F_{adj}(x) = \textstyle\sum_{i=1}^{n} (\lambda \frac{e_i}{\sum_{i=1}^{n} e_i} + (1-\lambda)\frac{a_i}{\sum_{i=1}^{n} a_i}) \cdot f_i(x)$$

Substitute a_i using (4), we have:

$$\begin{aligned} F_{adj}(x) &= \textstyle\sum_{i=1}^{n} (\lambda \frac{e_i}{\sum_{i=1}^{n} e_i} + (1-\lambda)\frac{1-e_i}{\sum_{i=1}^{n}(1-e_i)}) \cdot f_i(x) \\ &= \textstyle\sum_{i=1}^{n} \frac{e_i(\lambda n - \sum_{i=1}^{n} e_i) + (1-\lambda)\sum_{i=1}^{n} e_i}{(n - \sum_{i=1}^{n} e_i)\sum_{i=1}^{n} e_i} \cdot f_i(x) \end{aligned}$$

Let $\lambda = \frac{\sum_{i=1}^{n} e_i}{n}$, then:

$$\begin{aligned} F_{adj}(x) &= \textstyle\sum_{i=1}^{n} \frac{n \cdot e_i \cdot 0 + (n - \sum_{i=1}^{n} e_i)\sum_{i=1}^{n} e_i}{n \cdot (n - \sum_{i=1}^{n} e_i)\sum_{i=1}^{n} e_i} \cdot f_i(x) \\ &= \frac{1}{n} \cdot \textstyle\sum_{i=1}^{n} f_i(x) \\ &= \frac{F(x)}{n} \end{aligned}$$

which is irrelevant to e_i. On one hand, we could say that norm-referenced fitness function is useless when parameter $\lambda = \frac{\sum_{i=1}^{n} e_i}{n}$. On the other hand, we can also say that the original fitness function is a "special" kind of norm-referenced fitness function, in which the parameter λ is dynamically adjusted to $\frac{\sum_{i=1}^{n} e_i}{n}$ for each generation. We call this value $\lambda_{original}$, i.e.

$$\lambda_{original} = \frac{\sum_{i=1}^{n} e_i}{n}$$

Since e_i is the population's average error for a single test case i, $\frac{\sum_{i=1}^{n} e_i}{n}$ then is the population's average error for the problem currently solving. During the evolution process, the population's fitness generally improves. So, this value $\frac{\sum_{i=1}^{n} e_i}{n}$ becomes smaller as GP system evolves. In another word, in the original fitness function, the $\lambda_{original}$ value decreases through generations. Since smaller λ means stronger effect of $\boldsymbol{w}^+$, thus, there is an *implicit bias* within the original fitness function increasing the selection intensity as GP system evolves, "pushing" the system to converge.

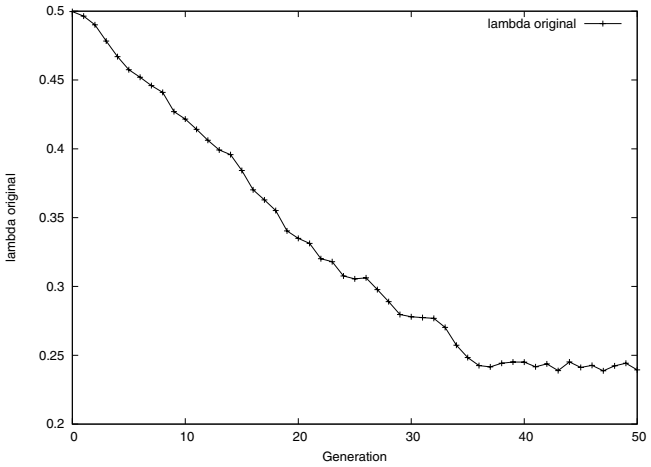

Figure 2: $\lambda_{original}$ for a GP Run in Even Parity 5 Problem using Original Fitness Function

Figure 2 gives an example of how $\lambda_{original}$ changes when the original fitness function is used. In the initial population at generation 0, the $\lambda_{original}$ is 0.49975, which is very close to 0.5. This is not an accident. Parity problem is a binary problem. The output can only be either 1 or 0. Because the initial population is randomly generated, as a result, the expected value of $\lambda_{original}$ in generation 0 would be 0.5. With $\lambda_{original}$ equal to 0.5 initially, the original fitness function achieves a good balance between exploration and exploitation. But, as the population's fitness gets better,

the $\lambda_{original}$ gets smaller, the original fitness function then puts more weight on exploitation, breaking the balance between exploration and exploitation. As a result, after around 35 generations, the GP system's evolution speed has been greatly reduced. So, in conclusion, the main problem of original fitness function is the implicit bias towards exploitation in later generations. On the other hand, because of the fixed λ parameter setting, norm-referenced fitness function does not have this implicit bias.

Using this theory of implicit bias, we can explain why different λ values achieve different performances in the initial experiment. In the initial experiment, λ values ranging from 0 to 0.3 failed to perform well. This is because those λ values are too small such that the GP system does not have enough ability to explore the searching space. As a result, in the selection of parameter λ, we generally should avoid small numbers. This is because small λ values not only cannot reduce the implicit bias towards exploitation in the original fitness function, but also enforce the bias. So, smaller λ values may lead to early convergence of the system, which is not desired.

On the other hand, when λ is too big, the selection intensity would be very small. This does remove the implicit bias in later generations, but it put too much emphasis to exploration in early generations, such that the evolution will be slowed down. This is why λ values 1, 0.9 and 0.8 haven't achieved better performance compared to original fitness function in the initial experiment. In even parity 5 problem domain, λ equals to 0.5 gives the best balance between the exploration and exploitation. In early generations, it mimics the behavior of original fitness function to give a good balance between the exploration and exploitation. In later generations, when the implicit bias starts to affect the original fitness function breaking the balance, the norm-referenced fitness function with λ equals to 0.5 still maintains the balance. We think because the behavior of $\lambda_{original}$ as in Figure 2 would be quite similar in other binary problem domains such as multiplexer, parameter λ equals to 0.5 would be the optimal parameter setting for norm-referenced fitness function in all binary problem domains.

The theory of the implicit bias towards exploitation in the original fitness function not only can be used to explain why norm-referenced fitness function works as we discussed above, the significance of this theory is much more profound. In fact, this theory offers a new perspective while tuning selection pressure in GP. Selection pressure is the key feature of selection methods [8] and it plays a critical role in GP system [11]. Furthermore, selection pressure plays an important role in bloating [4]. But, current research about selection pressure control are mainly concentrated on modifications of selection methods. For example, in [10], a modification of standard tournament selection using population clustering is developed. The theory of implicit bias shows that, in addition to selection methods, *fitness function also plays an important role in the formation of selection pressure*. Adjusting parameter λ in norm-referenced fitness function provides a completely new and effective approach to tune the selection pressure.

4.3 Further Experiments in Even Parity 5 Domain

In the previous simulation, we find that bigger λ values result in smaller selection intensity and conclude that this slows down the convergence speed of the algorithm. In the next experiment, we study whether smaller selection intensity leads to better convergence of algorithm despite the speed. In this experiment, we select 10 initializations created in the first experiment and run them for 300 generations instead of 50. Given enough time for GP algorithm to evolve, we then check if the norm-referenced fitness function leads to better convergence for big λ values. The rest of the parameters are the same as in initial experiment. Here, we test norm-referenced fitness with λ values ranging from 0.5 to 1 in steps of 0.1.

The experiment result is summarized in Table 4 and Figure 3. In Table 4, each cell gives the best raw fitness achieved. The number in brackets is the generation when the best raw fitness is achieved. GP runs in which the optimal solution is found (raw fitness 0) are highlighted. From Figure 3, we can find that, original fitness function performs very well in the first 100 generations. The best fitness of generations has been improved from 13.7 at generation 0 to 5.7 at generation 100. But after that, it failed to continuously improve the best fitness of generation. It takes 200 generations to improve the best fitness of generation to 4.6 at generation 300. On the other hand, norm-referenced fitness function is able to continuously improve the best fitness of generation. Bigger λ value results in slower improvements in early generations. When λ equals to 1, it takes 62 generations for norm-referenced fitness function to outperform original fitness function. It takes 72 generations when λ equals to 0.9. This number is reduced to 61 when λ equals to 0.8, 45 when λ equals to 0.7, 50 when λ equals to 0.6, and 10 when λ equals to 0.5. But, given 300 generations to evolve, norm-referenced fitness function outperforms original fitness function in almost all cases for all λ values experimented. In addition, there is no clear difference in performance for different λ values. This suggests that even though bigger λ value results in smaller selection intensity and it slows down the evolution, given enough time to evolve, big λ value can achieve the same level of performance as appropriate λ, and greatly outperforms the original fitness function. Thus, when selecting λ value, we can increase the number of generations GP runs to reduce the sensitivity of the parameter λ.

4.4 Regression Problem

All previous experiments deal with discrete domain, in which $f(x)$ is binary. This makes the selection of scaling function S in (2) quite simple. In regression problems, however, $f(x)$ usually is not bounded, i.e. $f(x) \in [0, \infty)$. In this case, for certain unfit individuals, $f(x)$ could be extremely large. These outliers affect the calculation of e_i and ultimately affect the calculations of both $\boldsymbol{w}^-$ and $\boldsymbol{w}^+$. For example, using GP to solve quartic symbolic regression problem ($x^4+x^3+x^2+x$ with 20 points generated from [-1, 1] as test cases), individual $2x^4 + x^3 + 3x^2$'s error for test case $x = 1$ is $6 - 4 = 2$. Another individual $e^{e^{e^x}}$'s error for the same test case is 3814275.1. If we add those two errors together directly in (2), the effect of the first error value will be neglected. Moreover, from pure implementation point of view, adding very big numbers together in (2) may cause floating point number overflow. Then e_i will be ∞ and the whole calculation of $\boldsymbol{w}^-$ and $\boldsymbol{w}^+$ will be wrong.

To solve this problem, we need an effective scaling function in (2) to control those outliers. In this paper, we develop a

Initialization	Original	Norm-referenced						Best Param
		$\lambda = 1.0$	$\lambda = 0.9$	$\lambda = 0.8$	$\lambda = 0.7$	$\lambda = 0.6$	$\lambda = 0.5$	
1	3 (300)	2 (300)	**0 (110)**	1 (300)	**0 (140)**	**0 (232)**	**0 (193)**	$\lambda = 0.9$
2	3 (300)	**0 (240)**	1 (300)	**0 (213)**	**0 (151)**	**0 (192)**	**0 (100)**	$\lambda = 0.5$
3	5 (300)	**0 (219)**	3 (300)	1 (300)	1 (300)	2 (300)	1 (300)	$\lambda = 1.0$
4	5 (300)	3 (300)	1 (300)	1 (300)	1 (300)	3 (300)	**0 (156)**	$\lambda = 0.5$
5	8 (300)	**0 (290)**	1 (300)	**0 (204)**	2 (300)	3 (300)	**0 (190)**	$\lambda = 0.5$
6	5 (300)	4 (300)	**0 (244)**	**0 (213)**	3 (300)	1 (300)	**0 (209)**	$\lambda = 0.8$
7	3 (300)	**0 (215)**	**0 (252)**	**0 (188)**	1 (300)	**0 (215)**	3 (300)	$\lambda = 1.0, 0.6$
8	4 (300)	2 (300)	5 (300)	1 (300)	**0 (281)**	1 (300)	**0 (271)**	$\lambda = 1.0$
9	4 (300)	3 (300)	3 (300)	**0 (227)**	1 (300)	1 (300)	1 (300)	$\lambda = 0.8$
10	6 (300)	**0 (198)**	**0 (240)**	3 (300)	5 (300)	**0 (207)**	3 (300)	$\lambda = 1.0$
Avg. Fitness	4.6	1.4	1.4	0.7	1.4	1.1	0.8	

Table 4: Experiment Results for 10 Initializations Running 300 Generations using Even Parity 5 Problem

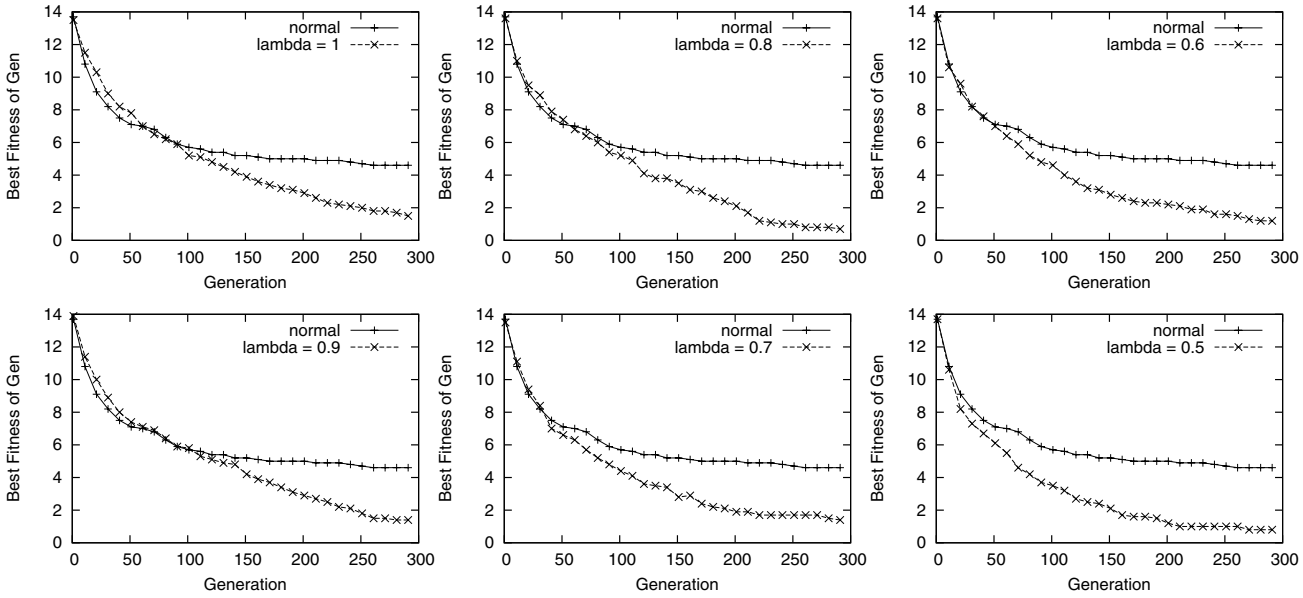

Figure 3: Comparison of Best Fitness of Generation Changes over Generations Between Original Fitness Function and Norm-referenced Fitness Functions with Different λ Settings using Even Parity 5 Problem

simple linear scaling function as follows. Given a population $P = \{x_i | 1 \leq i \leq m\}$, for a problem which contains n test cases, let

$$E = \{f_i(x_j) | 1 \leq i \leq m, 1 \leq j \leq n\}$$

Let $a = min(E)$ i.e. the minimal value in E, and $b = trimmean(E)$, i.e. the trimmed mean value of E, then the scaling function S is:

$$S(x) = \begin{cases} 0, & x = a \\ \dfrac{1}{2(b-a)}x - \dfrac{a}{2(b-a)}, & a < x \leq 2b - a \\ 1, & x > 2b - a \end{cases} \quad (7)$$

This scaling function uses a linear function to map the minimal error value (a) to 0, and the trimmed mean value (b) to 0.5. All values bigger than $2(b-a)$ are mapped to 1. For every generation, we rebuild E and recalculate the value of a and b before calculating e_i and a_i. One thing to note is that, $S(f(x))$ here is an adjustment of $f(x)$ relative to present population's performance. Comparing the value of $S(f(x))$ from different generations would be meaningless.

We test the performance of norm-referenced fitness function with the linear scaling function using sextic regression problem introduced in [7]. In sextic problem, the equation to be regressed is $y = x^6 - 2x^4 + x^2$ where $x \in [-1, 1]$. The experiment is designed as follows. Similar to the initial experiment, we firstly generate 50 initializations. Then for each initialization, we firstly run GP with original fitness function, and then run with norm-referenced fitness function. In the later, the parameter λ ranges from 0.5 to 1 in steps of 0.1. We also use best individual in the last generation as comparison criteria. The population size is 500. Tournament selection is used and the tournament size is 5. crossover and reproduction are used for selection of parent in breading with probability 90% and 10% respectively. GP runs for 50 generations. In the scaling function (7), when calculating trimmed mean, we discard 10% values at both high and low ends.

Similar to initial experiment, the experiment results are

summarized in Table 5. In sextic problem, we find that λ equals to 1.0 gives the best performance improvement (40.35%). The optimal λ value in sextic problem is much

Fitness Function	λ	Best Fitness	Std	T-Test
Original		0.347	0.318	
Norm-referenced	**1.0**	**0.229**	**0.207**	$\uparrow$
	0.9	0.288	0.278	$\uparrow$
	0.8	0.313	0.339	$\rightarrow$
	0.7	0.306	0.319	$\rightarrow$
	0.6	0.364	0.342	$\rightarrow$
	0.5	0.329	0.301	$\rightarrow$

Table 5: Best Fitness at Gen 50 in Sextic Problem

bigger compared to parity problem. We think this is mainly because of the usage of scaling function (7), rather than the change of problem domain. In previous discussion, we find that the original fitness function is a "special" kind of norm-referenced fitness function, in which the parameter $\lambda_{original}$ equals to $\frac{\sum_{i=1}^{n} e_i}{n}$ for each generation. In parity problem, this value goes down as GP evolves. But in sextic problem, because of the linear scaling function (7), in which the trimmed mean is always mapped to 0.5, the value of $\lambda_{original}$ does not change through generations and

$$\lambda_{original} \approx 0.5$$

for all generations. In early generations, when individuals are generally unfit, this value is reduced by the scaling function. In later generations, when individuals are generally fit, the scaling function amplifies this value. As we discussed previously, big λ value prevents GP system from converging in later generations and this leads to better performance. Since the λ has been amplified by the scaling function in later generations, as a result, we need even bigger λ values to effectively control the convergence speed. In our case, the scaling function (7) amplifies λ in sextic problem. So, bigger value ($\lambda = 1$) performs best. In conclusion, the selection of scaling function has much bigger implication to the norm-referenced fitness function. It may affect the value of optimal parameter λ. For the linear scaling function (7) developed, we can generally choose big λ values like 0.9 or 1.

5. CONCLUSION

In this paper, we develop an adjustment of original fitness function using population performance. The motivation of the newly created norm-referenced fitness function comes from comparison of norm- and criterion- referenced tests in real world examinations. We develop this new fitness function in the context of GP and use two well known benchmarks: even parity 5 and sextic, to show the effectiveness of the adjustment we made. We find that, instead of using absolute fitness value, using fitness value relative to present population's performance can greatly improve the overall performance of GP in terms of best individual generated. Further analysis of the norm-referenced fitness function shows that it improves the performance through reducing selection intensity in parent selection, and thus preventing premature convergence of the algorithm. We further study the issue of selecting parameter λ in norm-referenced fitness function and give a number of heuristics to find the optimal λ value.

Although the new fitness function is developed and tested in the context of GP, this algorithm can be easily applied to other population based optimization algorithms. Thus, one further work following this study is to apply norm-referenced fitness function to a broader range of algorithms such as genetic algorithm. In addition, we would also like to further study how parameter λ affects the performance of algorithm, and ultimately leads to automatic adjustment of the value of λ. This then frees users from defining λ.

6. REFERENCES

[1] T. Blickle and L. Thiele. A mathematical analysis of tournament selection. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 9–16, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[2] T. Blickle and L. Thiele. A comparison of selection schemes used in evolutionary algorithms. *Evol. Comput.*, 4:361–394, December 1996.

[3] L. A. Bond. Norm- and criterion-referenced testing. *Practical Assessment, Research & Evaluation*, 5(2), 1996.

[4] S. Gustafson, A. Ekárt, E. Burke, and G. Kendall. Problem difficulty and code growth in genetic programming. *Genetic Programming and Evolvable Machines*, 5:271–290, September 2004.

[5] J. Klein and L. Spector. Genetic programming with historically assessed hardness. In R. L. Riolo, T. Soule, and B. Worzel, editors, *Genetic Programming Theory and Practice VI*, Genetic and Evolutionary Computation, chapter 5, pages 61–75. Springer, Ann Arbor, 15-17May 2008.

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

[7] J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge Massachusetts, May 1994.

[8] R. Poli, W. B. Langdon, and N. F. McPhee. *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008. (With contributions by J. R. Koza).

[9] S. Silva and J. Almeida. Gplab-a genetic programming toolbox for matlab. In *In Proc. of the Nordic MATLAB Conference (NMC-2003*, pages 273–278, 2003.

[10] H. Xie, M. Zhang, and P. Andreae. Automatic selection pressure control in genetic programming. *Intelligent Systems Design and Applications, International Conference on*, 1:435–440, 2006.

[11] H. Xie, M. Zhang, and P. Andreae. An analysis of constructive crossover and selection pressure in genetic programming. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 1739–1748, New York, NY, USA, 2007. ACM.